Visual words for pose computation

Srikrishna Bhat

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Département de formation doctorale en informatique

Mots visuels pour le calcul de pose

(Visual words for pose computation)

THÈSE

pour l'obtention du

Doctorat de l'Université de Lorraine

(spécialité informatique)

présentée par

Srikrishna Bhat

Composition du jury

Rapporteurs :	David FILLIAT	PR, ENSTA - ParisTech
	Simon LACROIX	DR, CNRS, LAAS
Examinateurs :	Vincent LEPETIT	chercheur EPFL - Suisse
	Bernard GIRAU	PR, Université de Lorraine
	Marie-Odile BERGER	CR HDR, INRIA Nancy Grand-Est
	Frédéric SUR	MCF, Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Mis en page avec la classe thloria.

Dedicated to Mankuthimmana Kagga
and the quote
"O naive one, just be one among all"

Contents

Introduction

Les progrès récents dans le domaine de la vision par ordinateur ont accru la capacité des machines à inférer la structure 3D de l'environnement à partir d'images. La diminution du coût et l'augmentation des capacités des caméras et des ordinateurs ont conduit à l'émergence de nouvelles applications qui offrent une expérience visuelle plus riche à l'utilisateur final. Dans cette thèse, nous traitons le problème de l'estimation de la pose de la caméra (position et orientation) à partir d'une image de l'environnement capturé dans cette pose. L'estimation de la pose de la caméra est souvent une brique de base des applications de la Réalité Augmentée. La Réalité Augmentée a de nombreuses applications [10] dans les domaines de la médecine, de la défense, du divertissement, de la navigation, etc.

Pour calculer la pose de la caméra dans une image donnée (que nous appelons *image de test*), il est nécessaire d'identifier dans les images l'emplacement de caractéristiques 2D (des points par exemple, des lignes, des courbes, etc) correspondant aux objets de l'environnement 3D dont la position est connue par rapport à un système de coordonnées 3D fixe. La méthode Perspective-n-Point (PnP) [63, 45] permet d'estimer la pose, en utilisant l'association entre les coordonnées 3D de quelques-uns des points dans l'environnement et leur localisation 2D dans l'image. Afin d'appliquer PnP nous avons besoin de connaître une information géométrique (les coordonnées 3D) et des propriétés photométriques de quelques-uns des points dans l'environnement. Les propriétés photométriques d'un point 3D sont sa *représentation visuelle*, qui permet l'appariement dans une image de test donnée afin d'identifier son emplacement dans l'image 2D. L'information géométrique, c'est-à-dire les coordonnées 3D des points qui sont identifiés dans une image de test, est utilisée pour obtenir les contraintes géométriques nécessaires dans PnP.

Sur la base des besoins de l'application visée et de la technologie disponible, différentes techniques d'estimation de la pose peuvent adopter différentes approches pour obtenir des informations géométriques de l'environnement. Dans certaines applications [112], un modèle 3D de l'environnement cible est supposé connu. Dans certains cas, le modèle 3D est construit en utilisant un ensemble d'images d'apprentissage au cours d'une étape de formation « hors-ligne » [93]. Les méthodes basées sur SLAM (Simultaneous Localization And Mapping [11]) calculent le modèle 3D de l'environnement « en ligne » [27], lors de l'exécution de l'application. La construction du modèle 3D (à la fois en ligne et hors ligne) à partir d'images nécessite d'identifier les emplacements 2D de certains points de l'environnement dans plusieurs images, sur la base de la similarité photométrique. Nous appelons l'association entre les positions 2D d'un seul point 3D dans deux images une *correspondance 2D-2D*, et l'association entre un point 2D dans une image et ses coordonnées 3D dans l'environnement une *correspondance 2D-3D*. Les correspondances de points 2D-3D sont utilisées pour effectuer l'estimation de la pose pour une image donnée. Les correspondances de points 2D-2D sont utilisées pour construire le modèle 3D de l'environnement. En raison du bruit dans le processus de formation d'image et la quantification

des propriétés photométriques, quelques correspondances incorrectes sont susceptibles de se produire dans la plupart des situations. Pour traiter ces cas, les calculs sont effectués à travers le processus RANSAC [32] qui peut éliminer les correspondances aberrantes.

Défis et approches possibles

Pour les raisons suivantes, il est difficile d'identifier un nombre suffisant de points 3D dans une image de test pour estimer la pose de manière robuste en utilisant leurs propriétés photométriques. Les variations de la pose de la caméra donnent des changements significatifs dans les intensités des pixels au voisinage d'un point 3D. Elles peuvent aussi entraîner des occultations partielles ou totales des régions autour de certains des points dans l'environnement. Plusieurs points ayant les mêmes propriétés visuelles peuvent aussi entraîner des ambiguïtés. En outre, la recherche des correspondances dans toute l'image de test peut être extrêmement coûteux en temps pour les applications qui nécessitent une exécution temps-réel sur des images vidéo. Par conséquent, l'une des tâches les plus difficiles dans l'estimation de la pose est d'obtenir une représentation visuelle qui est efficace et fiable pour identifier un nombre suffisant de points d'une image de test sous différentes poses de la caméra. Une approche simple consiste à ajouter des marqueurs physiques qui produisent des motifs d'image facile à détecter dans l'environnement [47, 57]. Mais placer de tels marqueurs dans des positions appropriées est difficile, surtout lorsque la taille de de l'environnement croît. Par conséquent, de nombreuses approches ont été développées pour réduire le besoin d'interventions manuelles.

Suivi de caractéristiques lors de variations régulières de la pose

Afin d'éviter l'utilisation de marqueurs physiques, certaines techniques d'estimation de la pose utilisent des points 3D dans l'environnement qui produisent des caractéristiques génériques dans les images (comme les bords ou les points d'intérêt [26]), qui peuvent être détectés de manière fiable. Le modèle 3D, c'est-à-dire les coordonnées des points 3D correspondant aux caractéristiques de l'image, doit être fourni ou calculé en-ligne au moyen de méthodes comme SLAM. Les caractéristiques initialisées dans la première image sont suivies dans les images successives en supposant que la variation de pose est régulière dans une vidéo. Les bords peuvent être utilisés pour obtenir des candidats pour les projections de points 3D qui produisent des gradients bien marqués. En raison du *problème de l'ouverture*, les bords ne peuvent pas être utilisés pour identifier la position exacte d'un point dans une image. Des contraintes géométriques différentes de celles utilisées dans PnP doivent être utilisées afin de calculer la pose [42]. En plus du problème de l'ouverture, les méthodes basées sur le gradient sont sensibles au fouillis créé par l'arrière-plan. Au lieu d'utiliser simplement le gradient de l'intensité des pixels, les techniques de points d'intérêt choisissent un sous-ensemble des emplacements des images qui remplissent certaines conditions comme des extrema de certains filtres. Restreindre l'appariement à quelques points réduit la complexité de calcul et les contraintes utilisées pour les choisir augmentent la fiabilité. Les points d'intérêt peuvent être appariés dans des images successives à l'aide de procédés simples comme la corrélation croisée [26].

Inconvénients

Les méthodes basées sur le suivi de caractéristiques qui utilisent SLAM pour la construction du modèle 3D souffrent d'une dérive. Elles ont besoin de raffinements réguliers de la carte 3D [86] ou des équipements supplémentaires comme une caméra stéréo [113]. Même pour des

environnements de petite taille, un procédé parallèle de construction d'une carte 3D précise est recommandé [55]. Afin d'éviter d'utiliser SLAM, le modèle 3D de l'environnement devrait être fourni préalablement. Il est difficile d'avoir des modèles 3D pour les environnements de grande taille. Même si le modèle 3D est disponible, les caractéristiques correspondant aux points 3D doivent être initialisées dans la première image. Les techniques de suivi qui utilisent une simple corrélation entre pixels dans des trames successives ne sont pas assez robustes pour traiter le cas de grandes variations de pose. Par conséquent, dans ces procédés il est difficile de récupérer les caractéristiques qui réapparaissent après une occultation intermédiaire. Des méthodes d'apprentissage en-ligne en temps réel ont été proposées [90, 39, 46] pour l'obtention de détecteurs robustes. Mais les résultats sont principalement obtenus sur de petits objets planaires. Une phase d'apprentissage hors-ligne peut être utilisée pour surmonter ces limitations en construisant un modèle 3D et en calculant une représentation visuelle robuste des points 3D à partir d'un ensemble d'images d'apprentissage.

Apprentissage de la représentation de points 3D à partir d'images d'apprentissage

Le but de la reconnaissance de points 3D est de calculer une représentation visuelle qui peut être utilisée pour identifier les points 3D dans une image donnée sans utiliser aucune information préalable sur leur emplacement dans l'image, et sans exiger une image avec une vue très proche des points au moment de l'exécution. La représentation visuelle d'un point 3D se compose d'un ensemble de valeurs calculées à partir d'une ou plusieurs images du point 3D. La représentation doit permettre d'identifier le point 3D correspondant dans les images à partir de points de vue très changeants. De grandes variations du point de vue entraînent un changement énorme dans l'aspect des objets. Ceci est plus difficile par rapport au suivi de caractéristiques 2D, dans lequel l'aspect de l'image autour d'un point varie régulièrement dans les trames successives. Le coût de calcul augmente pour deux raisons. Afin d'assurer la robustesse du processus, l'obtention des descripteurs d'apparence peut impliquer des calculs qui sont plus coûteux que la simple corrélation ou la détection de contours utilisée dans le suivi des caractéristiques 2D. Deuxièmement, le processus d'appariement doit être effectuée sur l'image complète, à la différence du suivi de caractéristiques 2D dans lequel la connaissance de la pose de l'image précédente réduit de manière significative la zone de recherche.

Des vecteurs de descripteurs de points d'intérêt [69] qui quantifient les caractéristiques visuelles d'un point d'intérêt en utilisant les pixels dans son voisinage, peuvent fournir des correspondances 2D-2D plus fiables que les approches simples comme la corrélation entre les pixels. Ces descripteurs ont des propriétés invariantes sous certaines transformations qui sont susceptibles de se produire en raison de la variation de pose. Cependant, dans la pratique, la plupart des descripteurs intègrent seulement des modèles approximatifs des transformations réelles. Plusieurs descripteurs d'un point 3D peuvent être extraits à partir d'images sous différentes poses afin d'améliorer encore les performances de la représentation visuelle [108].

Approche proposée

Les descripteurs de points d'intérêt extraits d'un ensemble d'images d'apprentissage ont été utilisés dans le passé pour construire un modèle 3D et obtenir sa représentation visuelle [93, 38, 51]. Nous suivons une approche similaire et employons un cadre à deux étapes constitué d'une phase d'apprentissage et d'une phase de test. Dans l'étape d'apprentissage, nous apparions les descripteurs SIFT [69] extraits d'images d'apprentissage et calculons la structure à partir du mouvement

(SfM, Structure from Motion [107]) sur les appariements 2D-2D ainsi calculés pour obtenir un ensemble de points 3D dans l'environnement. Après l'obtention de points 3D, nous utilisons les descripteurs SIFT associés aux positions 2D dans les images d'apprentissage d'un point 3D pour représenter ses propriétés photométriques. Pendant la phase de test, nous utilisons la représentation visuelle dérivée des descripteurs SIFT associés aux points 3D afin de les détecter dans une image test et effectuer l'estimation de la pose.

Notre étude met l'accent sur l'expérimentation de différentes manières d'apparier les descripteurs SIFT pour les correspondances 2D-2D au cours de l'apprentissage et de différentes manières d'obtenir une description visuelle des points 3D pour calculer les appariements 2D-3D pendant la phase de test. Nous utilisons des outils logiciels accessibles publiquement pour effectuer l'extraction de caractéristiques (c'est-à-dire SIFT) et les calculs géométriques (c'est-à-dire SfM et PnP). L'ensemble du processus peut se résumer comme suit:

- **Étape d'apprentissage**:

  1. Extraction des descripteurs SIFT (vecteurs d'apprentissage) à partir d'images et appariement pour obtenir des correspondances 2D-2D.
  2. SfM pour calculer les points 3D correspondant à certains des appariements 2D-2D.
  3. Association de chaque point 3D aux descripteurs SIFT correspondant aux positions 2D dans les images utilisées pour les calculer.

  À ce stade, chaque point 3D est associé de manière unique à un ensemble de vecteurs d'apprentissage. Cet ensemble permet d'obtenir une représentation visuelle du point 3D qui définit la règle pour faire correspondre un vecteur descripteur SIFT avec le point 3D.

- **Étape de test**:

  1. Extraction des descripteurs SIFT (vecteurs de test) de l'image de test donnée.
  2. Mise en correspondance des vecteurs de test avec des points 3D en utilisant la représentation visuelle obtenue à partir des vecteurs d'apprentissage associés. La position 2D de chaque vecteur de test qui correspond à un point 3D fournit une correspondance 2D-3D.
  3. Estimation de la pose à partir de l'ensemble des correspondances 2D-3D en utilisant PnP.

Dans les deux étapes, nous pouvons rencontrer des correspondances entre points incorrects (2D-2D ou 2D-3D). Pour traiter ces cas, RANSAC [32] est utilisé et peut, jusqu'à un certain point, rejeter des mauvaises correspondances.

Contribution

Notre contribution dans cette approche basée sur deux étapes est la suivante:

1 : Établissement de correspondances 2D-2D pendant l'apprentissage

Comme mentionné précédemment, nous utilisons des descripteurs SIFT extraits des images d'apprentissage (vecteurs d'apprentissage) pour obtenir des correspondance 2D-2D. Les auteurs de la méthode SIFT ont présenté une technique standard de mise en correspondance de points-clés [38] Par la suite, la popularité des descripteurs SIFT a conduit à l'application de différentes techniques de clustering pour regrouper les descripteurs SIFT à partir de plusieurs images et pas seulement deux. Ces approches de clustering, désignées comme *mots visuels* [105, 87], traitent une image comme un document de mots dans laquelle chaque mot est représenté par les descripteurs d'apprentissage appartenant à un seul cluster. Un mot visuel peut être considéré comme la caractérisation de régions d'images à partir desquelles les descripteurs sont extraits. Nous utilisons le cadre des mots visuels pour représenter le modèle 3D de telle sorte que chaque point 3D est représenté par un mot visuel unique. Certains des groupes obtenus à partir des descripteurs SIFT d'apprentissage peuvent contenir plusieurs vecteurs de descripteurs SIFT provenant d'une unique image d'apprentissage, par exemple les groupes contenant des descripteurs correspondant à des motifs répétés dans l'environnement. Nous les éliminons, car ils ne sont pas adaptés pour représenter de manière unique un point 3D. Chaque groupe retenu peut être traité comme un ensemble de descripteurs extraits des différentes projections 2D d'un point 3D unique et peut donc être utilisé pour obtenir des correspondances 2D-2D pour SfM.

La plupart des mots visuels développés dans le passé utilisent le clustering par k-moyennes ou par mean-shift. Nous construisons des cartes 3D en utilisant les appariements 2D-2D obtenus par ces deux types de clustering et la mise en correspondance de points SIFT standard. Nous proposons également une nouvelle façon de former des mots visuels (que nous appelons aussi la clôture transitive adaptative - ATC) qui effectue l'appariement de descripteurs SIFT basé sur la recherche par intervalle (c'est-à-dire deux descripteurs SIFT sont appariés si leur distance est inférieure à un seuil). La valeur de seuil utilisée pour l'appariement est adaptée en fonction de la distribution des vecteurs d'apprentissage dans l'espace des descripteurs SIFT. Nous commençons avec une grande valeur du seuil, et apparions les vecteurs d'apprentissage en utilisant cette valeur. Ces appariements établissent une relation sur l'ensemble des vecteurs d'apprentissage qui est réflexive et symétrique. Nous appliquons une clôture transitive sur ces appariements pour obtenir une relation d'équivalence dans lequel chaque partition est traitée comme un cluster. Les clusters qui ont de multiples instances dans la même image (et donc ne conviennent pas pour représenter un point 3D) sont l'objet d'un nouveau clustering, de la même manière mais avec un seuil de distance réduit. Nous répétons ce processus pour un ensemble fixe de seuils en ordre décroissant. Il en résulte un ensemble de clusters, dans lequel les vecteurs de descripteurs dans un cluster sont appariés les uns aux autre avec une valeur de seuil qui est adaptée à la zone de l'espace des descripteurs dans lequel le cluster est situé.

Nous évaluons la performance des différentes techniques d'appariement à l'aide de certaines mesures de la précision des appariements 2D-2D et de la qualité du modèle 3D construit à partir de ces appariements. Ces mesures sont énumérées ci-dessous:

1. **Proportion de correspondances 2D-2D conformes aux contraintes épipolaires**: La précision d'une correspondance 2D-2D est mesurée à l'aide des contraintes épipolaires entre paires d'images. Les contraintes épipolaires sont obtenues à partir de la position des caméras connues d'après la « vérité terrain » disponible pour certaines images d'apprentissage.

2. **Proportion des vecteurs d'apprentissage associés aux points 3D**: Les appariements qui ne respectent pas les contraintes géométriques sont supprimés en cours de SfM. Par conséquent, une grande proportion de vecteurs d'apprentissage passant SfM sans se faire rejeter indique une meilleure qualité d'appariement.

3. **Nombre moyen de vecteurs d'apprentissage associés à un point 3D**: Une bonne proportion de points 3D calculés à partir d'un grand nombre de points 2D indique la capacité de la méthode d'appariement à grouper en un même cluster les descripteurs de points 3D obtenus à partir de points de vue différents.

4. **Planarité des points 3D appartenant à un plan**: Nous marquons manuellement les parties des images d'apprentissage contenant une seule surface plane de l'environnement. Tous les points 3D calculés à partir des descripteurs SIFT appartenant à ces parties doivent se situer dans le plan. Nous mesurons à quel point ces points 3D s'adaptent à un plan.

5. **Efficacité algorithmique**: Le processus d'apprentissage hors-ligne n'est pas prévu pour fonctionner à la cadence vidéo. Néanmoins, les méthodes de clustering impliquent des calculs intensifs qui peuvent devenir prohibitifs lorsque le nombre de vecteurs d'apprentissage augmente. Nous comparons le temps CPU nécessaire pour différentes approches utilisées dans nos expériences.

2 : Représentation visuelle pour les correspondances 2D-3D pendant l'étape de test

À la fin de l'étape d'apprentissage, nous obtenons des points 3D dans l'environnement et un ensemble de descripteurs SIFT appartenant au cluster (ou mot visuel) associé à chaque point 3D. Étant donné un ensemble de descripteurs SIFT d'une image de test (vecteurs de test), nous avons besoin de faire correspondre chaque vecteur de test avec ces clusters. Il s'agit d'un problème de classification supervisée [30]. On peut donc utiliser diverses techniques d'apprentissage statistique pour reconnaître les points 3D dans une image test. Apparier un vecteur de test avec l'ensemble des vecteurs de formation peut être coûteux algorithmiquement. Nous essayons de réduire le coût de la mise en correspondance en testant deux autres ensembles de vecteurs d'apprentissage provenant de l'ensemble des vecteurs extraits à partir des images d'apprentissage. Le premier est obtenu en ne retenant que les vecteurs d'apprentissage qui sont associés à des points 3D, c'est-à-dire ceux qui sont utilisés pour obtenir des correspondances 2D-2D pour SfM et au moins une de ces correspondances est conservée et utilisée pour générer un point 3D par le processus de SfM. Le second est obtenu en utilisant uniquement les centres des clusters associés aux points 3D, c'est-à-dire chaque mot visuel associé à un point 3D est représenté par un vecteur unique.

Nous pouvons évaluer les résultats de la phase de test à deux niveaux: (i) la précision de la reconnaissance de points 3D et (ii) la précision de l'estimation de la pose. La précision de l'estimation de la pose dépend de la précision de la classification lors de la reconnaissance de points 3D. Mais, en raison de l'étape RANSAC qui peut tolérer quelques correspondances aberrantes, il est possible de ne pas constater beaucoup de différence dans l'estimation de la pose, même si certains des descripteurs SIFT test sont mal classés. En outre, la qualité des points 3D appariés peut influer la précision de l'estimation de la pose. Dans les deux cas (reconnaissance et estimation de la pose), nous avons besoin d'une vérité terrain pour évaluer la performance. Il est relativement plus facile d'obtenir des positions de caméra « vérité terrain » pour évaluer

l'exactitude de la pose que de connaître les points 3D réellement associés aux vecteurs de test. Dans nos expériences, nous avons surtout évalué l'exactitude de l'estimation de la pose. Quand nous avons besoin d'évaluer la précision de la reconnaissance de points 3D, on obtient le label de la classe pour les descripteurs des images de test en exécutant l'étape d'apprentissage dans laquelle les vecteurs de test sont inclus avec les vecteurs d'apprentissage tout en effectuant le clustering et SfM.

Nous avons d'abord expérimenté deux stratégies différentes de reconnaissance sur la base de la classification au plus proche voisin (i) seuil sur la distance avec le plus proche voisin, (ii) seuil sur le rapport des distances entre les deux vecteurs d'apprentissage les plus proches appartenant à deux différents points 3D. Dans les deux cas, l'appariement est effectué en utilisant différentes valeurs de seuil séparément, avec l'ensemble d'apprentissage en entier et les deux autres ensembles d'apprentissage mentionnés plus haut. Pour des modèles 3D basés sur MS (mean-shift) et ATC, nous effectuons une technique de reconnaissance supplémentaire. Pour MS, nous effectuons une itération de mean-shift (avec les mêmes paramètres que ceux utilisés lors de l'apprentissage) pour chaque vecteur de test. Pour ATC nous apparions un vecteur de test avec différents groupes en utilisant le seuil adaptatif des groupes avec lequel les clusters respectifs sont formés au cours de l'apprentissage. Nous faisons l'estimation de la pose sur les correspondances 2D-3D obtenues grâce à ces différents schémas d'appariement au plus proche voisin sur différents ensembles d'apprentissage appartenant à différents modèles 3D. La pose estimée est comparée avec la vérité-terrain pour l'évaluation. Chaque évaluation de la pose estimée des images de test mesure la performance de la combinaison de l'appariement 2D-2D et des schémas de reconnaissance de points 3D utilisés respectivement lors des étapes d'apprentissage et de test.

Nous menons d'autres expériences avec des classificateurs linéaires et non linéaires qui appliquent une transformation optimale à l'espace des descripteurs avant de procéder à la classification. Pour les techniques basées sur une transformation linéaire nous procédons à la reconnaissance de points 3D en utilisant des stratégies de plus proches voisins dans le domaine transformé, comme décrit dans le paragraphe précédent. Pour la classification non-linéaire, nous utilisons des SVM avec noyau gaussien. Nous faisons l'apprentissage d'un classificateur SVM pour chaque point 3D. Nous expérimentons avec des valeurs différentes pour le paramètre du noyau σ. Nous essayons aussi la technique décrite dans [64] pour calculer automatiquement le paramètre de noyau pour chaque SVM à partir des échantillons de d'apprentissage disponibles.

3 : Stratégies d'accélération pour le clustering mean-shift

Le clustering mean-shift déplace itérativement chaque vecteur d'apprentissage vers les maxima locaux de la fonction de densité de probabilité estimée à partir de l'ensemble d'apprentissage [23]. Le coût de calcul de MSC devient prohibitif grand que la dimension de l'espace des caractéristiques augmente. La principale source de ces coûts est le processus de recherche par intervalle [36] qui implique le calcul, à chaque itération, de l'ensemble des vecteurs d'apprentissage dans un intervalle de largeur de w autour du vecteur moyen courant. Nous avons conçu une stratégie pour accélérer le calcul en divisant l'ensemble des vecteurs d'apprentissage en groupes tels que les vecteurs dans un groupe n'influeront jamais le calcul des clusters dans un autre groupe. Cette stratégie effectue des calculs exacts contrairement aux méthodes qui utilisent des sous-échantillonages [34] ou des approximations [20]. Néanmoins, nous obtenons une accélération modérée par rapport à ces méthodes inexactes, pour une gamme spécifique de valeurs de

paramètres utilisés pour effectuer le mean-shift. Notre technique est applicable à tous les types de noyaux qui sont à support compact et de largeur de bande fixe.

Notre méthode d'accélération est basée sur l'observation que, pendant le mean-shift, si un vecteur d'apprentissage x est au-delà de $\sqrt{2}$ w de tous les vecteurs qui sont à une distance de w du vecteur moyen courant, alors x sera au-delà de la distance de w du vecteur moyen suivant. C'est pourquoi nous n'avons pas besoin de comparer le vecteur suivant avec de tels vecteurs d'apprentissage. Si nous divisons les vecteurs d'apprentissage dans des groupes en appliquant la clôture transitive de la relation obtenue en les appariant avec un seuil de distance $\sqrt{2}w$, alors chacun de ces groupes n'interférera jamais avec aucun autre groupe au cours du calcul du mean-shift. Nous ne savons pas si cette propriété sera valable pour toute valeur de seuil inférieure à $\sqrt{2}w$. Néanmoins nous prouvons que le seuil ne peut pas être inférieur à $\frac{\pi}{\sqrt{6}}w$.

La stratégie d'accélération n'est utile que lorsque la taille de chaque groupe obtenu par l'opération de clôture transitive est nettement inférieure à la taille de l'ensemble d'apprentissage entier. Elle exige que les vecteurs d'apprentissage sont bien séparés les uns des autres en groupes tels que l'opération de clôture transitive sur l'appariement basé sur la distance peut fournir un nombre raisonnable de partitions. Lorsque les données sont bien séparées, les techniques de recherche rapides peuvent également augmenter l'efficacité de mean-shift en liant les régions séparées par une étape de pré-traitement et d'effectuer la recherche uniquement dans des régions délimitées au voisinage du vecteur requête. Dans nos expériences, nous aimerions voir si les partitions données par la clôture transitive peuvent accélérer le mean-shift, qui utilise déjà une technique de recherche rapide. Nous utilisons [79] pour la recherche rapide. Nous effectuons la recherche exacte sans utiliser l'option d'approximation disponible dans [79].

4 : Caractéristiques provenant de vues simulées par ASIFT

Si l'environnement peut être supposé localement plan, alors de nouvelles vues de l'environnement peuvent être générées en appliquant des transformations affines à une de ses vues [78, 89]. Nous utilisons ASIFT [78], qui calcule des descripteurs SIFT supplémentaires à partir de vues simulées de l'image originale. Ces descripteurs supplémentaires peuvent être utilisés pour obtenir des correspondances lorsque la variation de pose entre les images d'apprentissage et de test est grande.

Nous explorons différentes façons d'incorporer dans la carte 3D les descripteurs SIFT obtenus à partir de vues simulées. La première approche consiste à utiliser des descripteurs extraits à l'aide d'ASIFT pour réaliser le SfM. Nous constatons que cela se traduit par la génération de nouveaux points 3D plutôt que par l'ajout de nouveaux descripteurs aux points 3D calculés à partir des descripteurs de l'image originale. C'est pourquoi nous concevons des méthodes qui ajoutent des descripteurs à une carte 3D à partir de vues nouvelles. Nous essayons d'ajouter les descripteurs ASIFT proches dans l'image à ceux qui sont associés à la carte 3D. Nous avons aussi essayé d'utiliser seulement la partie de calcul du descripteur dans l'algorithme SIFT pour obtenir des descripteurs à partir d'emplacements d'image dans l'image simulée qui correspondent aux emplacements des descripteurs de l'image originale associés à la carte en 3D.

Nous concevons un cadre expérimental pour comparer la qualité des appariements 2D-3D obtenus par ces différents schémas. En utilisant les informations vérité-terrain disponibles pour l'un des jeux de données utilisés dans nos expériences, nous réduisons l'influence des facteurs sans rapports (par exemple: la qualité de la carte 3D estimée par l'intermédiaire d'un processus RANSAC

qui n'est pas lié à la simulation de vues), qui affecte la mesure de performance.

Organisation

Notre travail consiste principalement à établir des correspondances de points (2D-2D et 2D-3D). Les contraintes géométriques découlant de ces correspondances sont utilisées pour résoudre le problème du calcul de pose. Dans le chapitre 1, nous décrivons brièvement les outils que nous utilisons pour effectuer des calculs géométriques sur les correspondances de points. Il s'agit principalement de concepts de la géométrie projective et RANSAC.

Dans le chapitre 2, nous présentons différentes stratégies dans la littérature utilisées pour établir des correspondances ponctuelles pour l'estimation de la pose. Nous donnons un bref résumé de la méthode d'extraction des descripteurs SIFT qui est largement utilisé dans nos expériences (ainsi que dans les travaux connexes). Nous organisons la littérature à l'instar de [60] et nous motivons le choix de notre méthode à deux étapes décrite ci-dessus sur la base des travaux antérieurs. A la fin du chapitre, nous présentons le cadre des mots visuels utilisé dans notre travail pour représenter les points 3D.

Dans le chapitre 3, nous présentons l'étape d'apprentissage de notre méthode d'estimation de la pose. Il s'agit principalement de la description des schémas d'appariement utilisés pour établir les correspondances 2D-2D avec les paramètres que nous utilisons dans nos expériences pour construire la carte 3D. Nous décrivons les bases de donnée et la méthode d'évaluation que nous utilisons pour mesurer la qualité des différents schémas d'appariement. Nous présentons les résultats expérimentaux et concluons avec les observations que nous tirons des résultats.

Dans le chapitre 4, nous présentons l'étape de test dans laquelle nous présentons des expériences avec différentes règles de mise en correspondance pour reconnaître les points 3D dans une image test. L'évaluation est effectuée sur différentes combinaisons de schémas d'appariement 2D-2D utilisés dans l'étape d'apprentissage et des stratégies de reconnaissance des points 3D utilisée dans l'étape de test.

Dans le chapitre 5, nous décrivons notre technique d'accélération de mean-shift et la preuve mathématique soutenant que notre modification n'a aucune incidence sur le résultat final de l'algorithme. Nous présentons les résultats que nous obtenons sur les descripteurs SIFT extraits des images que nous avons utilisées dans nos expériences.

Dans le chapitre 6, nous présentons les expériences conduites avec les vues simulées. Nous justifions le choix d'utiliser ASIFT et décrivons les différentes approches adoptées pour améliorer la robustesse du calcul des appariements 2D-3D.

Enfin, nous concluons la thèse dans le chapitre 7 en résumant les différents résultats obtenus. Nous mentionnons aussi quelques perspectives possibles.

Introduction

Recent advances in the area of computer vision have increased the ability of machines to infer 3D structure of the surrounding environment from images. Decrease in the cost and increase in the capabilities of the cameras and computers have led to emergence of new applications which provide richer visual experience to the end user. In this thesis we deal with the problem of estimating camera pose (position and orientation) from an image of the surrounding environment captured in that pose. Camera pose estimation is an integral part of Augmented Reality. It has many applications [10] in the areas of medicine, military, entertainment, navigation etc.

In order to compute pose of the camera in a given image (which we refer to as *test image*) we need to identify the 2D location of image features (e.g. points, lines, curves etc) corresponding to the objects in the environment whose 3D position is known with respect to a fixed 3D coordinates system. Perspective-n-Point (PnP) algorithm [63, 45] which is the most popular class of methods for pose estimation, uses association between 3D coordinates of some of the points in the environment and their 2D location in the image to obtain the necessary geometric constraints to determine pose. In order to apply PnP we need to know geometric information (i.e. 3D coordinates) and photometric properties of some of the points in the environment. Photometric property of a 3D point is used to obtain its quantified *visual representation* which can be matched with a given test image to identify its 2D location in the image. Geometric information i.e. 3D coordinates of the points which are identified in a test image are used to obtain the geometric constraints needed in PnP.

Based on the requirement of the target application and available technology, different pose estimation techniques adopt different approaches to obtain geometric information of the environment. In some applications[112] 3D model of the target environment is assumed to be known. In some cases, the 3D model of the environment is built using a set of training images during an offline training step[93]. SLAM (Simultaneous Localization and Mapping [11]) based methods compute the 3D model of the environment online [27] while running the application. Building the 3D model (both offline and online) from images, involves the task of identifying 2D locations of some of the points in the environment in multiple images based on photometric similarity. We refer to the association between 2D locations of a single 3D point in two images as 2D-to-2D point match and the association between a 2D point in an image and its 3D coordinates in the environment as 2D-to-3D point match. 2D-to-3D point matches are used to perform pose estimation for a given image. 2D-to-2D point matches are used to build the 3D model of the environment. Due to the noise in the process of image formation and quantification of photometric properties, few incorrect point matches are likely to occur in most of the situations. To handle such cases computations are performed through RANSAC process [32] which can discard outliers while fitting a hypothesis on a set of samples.

Challenges and Approaches

Identifying sufficient number of 3D points in a test image for robust pose estimation using their photometric properties reflected in the images is a difficult task due to the following reasons. Variation in camera pose imparts significant change in pixel intensity pattern in the image of a 3D point. It also may result in partial or complete occlusion of regions surrounding some of the points in the environment. Multiple points with similar visual properties in the environment may lead to ambiguity. Moreover, searching the whole test image to match the visual representation of various points may be prohibitively time consuming for applications which require real-time execution on video frames. Hence, one of the most challenging task in pose estimation is to obtain visual representation which can efficiently and robustly identify sufficient number of points in a test image under varying camera pose. One simple approach is to add fiducial markers which produce easy-to-detect image patterns to the environment [47, 57]. But placing distinguishable markers at appropriate positions is difficult especially when the size of environment increases. Hence, many approaches to reduce the need for manual interventions have been developed.

Feature tracking under smooth variation of pose

In order to avoid the need to use fiducial markers, some pose estimation techniques use 3D points in the environment that produce generic *image features* (like edges or interest points[26]) which can be reliably detected. 3D model i.e. the coordinates of the 3D points corresponding to the image features either should be provided or computed online through methods like SLAM. The features once initialized in the first frame are tracked in the successive frames assuming smooth variation of pose in a video. Edge features can be used to obtain candidate image locations for 3D points which produce sharp image gradients. Due to aperture problem, edges cannot be used to identify the exact location of a point in an image. Different geometric constraints than those in PnP need to be used in order to compute pose[42]. In addition to aperture problem, gradient based methods are sensitive to background clutter. Instead of using gradient of pixel intensity, interest point techniques choose a subset of image locations which satisfy certain conditions like extrema of output of some filter operation. Matching only at few selected locations reduces the computational complexity and the conditions used to choose them increase the reliability. Interest points can be matched across successive images using simple process like cross correlation [26].

Drawbacks

Feature tracking based methods which use SLAM for 3D model construction suffer from drift. They need regular refinement of the 3D map[86] or additional equipments like stereo camera[113]. Even for small sized environments a parallel batch method to build accurate 3D map[55] is recommended. In order to avoid using SLAM, 3D model of the environment should be provided. It is difficult to have 3D models for large environments. Even if the 3D model is available, the features corresponding to the 3D points should be initialized in the first frame. The tracking techniques which use simple correlation between pixels in successive frames for matching are not robust enough for handling large pose variations. Hence, it is difficult to recover the features which reappear after intermediate occlusion in such methods. Online realtime learning methods [90, 39, 46] have been proposed for obtaining robust detectors. But the results are mainly obtained on small planar objects. An offline learning stage can be used to overcome these limitations by building a 3D model and computing robust visual representation for 3D points from a set of training images.

Learning 3D point representation from training images

The aim of 3D point recognition technique is to compute visual representation which can be used to identify 3D points in a given image without using any prior information about their image location and without requiring an image with closely related view of the points at run time. Visual representation of a 3D point consists of a set of values computed from one or more images of the 3D point. The representation should enable identification of the corresponding 3D point in the images from widely varying views. Wide variation in viewpoint causes huge change in the image of the objects. This is more challenging when compared to 2D feature tracking in which the image pattern around a point vary smoothly in successive frames. Computationally the cost increases due to two reasons. In order to achieve robustness the process of obtaining appearance descriptors may involve computations which are costlier than simple correlation or edge detection used in 2D feature tracking. Secondly, the matching process should be performed over the full image, unlike 2D feature tracking in which the knowledge of pose from the previous image reduces the search region significantly.

Interest point descriptor vectors[69] which quantify the visual characteristics of an interest point using the pixels in its neighborhood, can provide more reliable 2D-to-2D matches between images than simple approaches like correlation between pixels. These descriptors have invariant properties under some of the transformations which are likely to occur due to pose variation. However, in practice, most descriptors incorporate only approximate model of the real transformations. Multiple descriptors of a 3D point extracted from training images containing its view under different pose can be used to further improve the performance of the visual representation[108].

Our approach

Interest point descriptors extracted from a set of training images have been used in the past to build 3D model and obtain its visual representation [93, 38, 51]. We also follow a similar approach and employ a 2-stage framework consisting of a training stage and a test stage. In the training stage we match SIFT [69] descriptors extracted from training images and perform SfM (Structure from Motion [107]) on the 2D-to-2D matches thus computed to obtain a sparse set of 3D points in the environment. After obtaining 3D points, we use the SIFT descriptors associated with the 2D locations in the training images of a 3D point to represent its photometric properties. During test stage we use the visual representation derived from SIFT features associated with the 3D points in order to detect them in a test image and perform pose estimation.

Our investigation focuses on experimenting with different ways of matching SIFT descriptors for establishing 2D-to-2D matches during training and various ways of obtaining visual description for the 3D points to compute 2D-to-3D matches during test stage. We use publicly available software tools to perform feature extraction (i.e. SIFT) and geometric computations (i.e. SfM and PnP). The whole process can be summarized as follows:

- **Training Stage**:
    1. Extract SIFT descriptors (training vectors) from training images and match them to obtain 2D-to-2D matches.
    2. Perform SfM to compute 3D points corresponding to some of the 2D-to-2D matches.

3. Associate each 3D point with the SIFT descriptors corresponding to the 2D image locations used to compute it.

At this point each 3D point is uniquely associated with a set of training vectors. This set is used to obtain visual representation of the 3D point which defines the rule for matching a SIFT descriptor vector with the 3D point.

- **Test Stage**:

1. Extract SIFT descriptors (test vectors) from the given test image.
2. Match the test vectors with 3D points using the visual representation obtained from the associated training vectors. The 2D location of each test vector which matches with a 3D point provides a 2D-to-3D correspondence.
3. Estimate pose from the set of 2D-to-3D correspondences through PnP.

In both stages we may encounter incorrect point correspondences (2D-to-2D or 2D-to-3D). To handle such cases RANSAC [32] is employed which up to some extent can discard the outliers.

Contribution

Our contribution during the course of our investigation while following the above mentioned 2-stage framework is as follows:

1 : Establishing 2D-to-2D matches during training

As mentioned earlier, we use SIFT descriptors extracted from training images (training vectors) to obtain 2D-to-2D matches. The author of the SIFT method presented a standard keypoint matching technique [38] to match the SIFT descriptors extracted from two images. Subsequently, the popularity of the SIFT descriptors has lead to application of different clustering techniques to group SIFT descriptors from several images instead of just two. These clustering approaches, collectively named as visual word framework[105, 87], treat an image as a document of words in which each word is represented by the training descriptors belonging to a single cluster. Visual word can be considered as characterization of the image pattern common to the image regions from which its member descriptors are extracted. We use visual word framework to represent 3D model in such a way that each 3D point is uniquely represented by a visual word. Some of the clusters obtained from the training SIFT descriptors may contain multiple SIFT descriptor vectors from a single training image, for example clusters containing descriptors corresponding to repeated patterns in the environment. We discard them, as they are not suitable to uniquely represent a 3D point. Each retained cluster can be treated as set of descriptors extracted from different 2D projections of a single 3D point and hence can be used to obtain 2D-to-2D matches for SfM.

Most of the visual words developed in the past use either k-means or mean-shift clustering. We build 3D map using 2D-to-2D matches obtained through these two types of clustering and standard SIFT keypoint matching. We also propose a novel way of forming visual words (which we call as Adaptive Transitive Closure - ATC) which performs range based matching on SIFT descriptors (i.e. two SIFT descriptors are matched if they are within a distance threshold). The

threshold value used for matching is adapted based on the distribution of training vectors in SIFT descriptor space. We start with a large distance threshold, match the training vectors based on this threshold. These matches establish a relation on the set of training vectors which is reflexive and symmetric. We apply transitive closure on these matches to obtain an equivalence relation in which each partition is treated as a cluster. The clusters which have multiple instances in a single training image (and hence not suitable for representing a 3D point) are clustered again in a similar way with a reduced distance threshold. We repeat this process for a fixed set of thresholds in the decreasing order. This results in a set of clusters in which the descriptor vectors in a cluster are matched to each to each other with a threshold value which is adapted to the region of the descriptor space in which the cluster is situated.

We evaluate the performance of various matching techniques using some aspects of the accuracy of 2D-to-2D matches and the quality of 3D model built from those matches. These aspects are listed below:

1. **Portion of 2D-to-2D matches complying with epipolar constraints**: Accuracy of a 2D-to-2D match is measured using epipolar constraints between the pair of matches images. Epipolar constraints are obtained from the ground truth camera positions available for some of the training images used in our experiments.

2. **Portion of the training vectors getting associated with the 3D points**: The matches which do not comply with the geometric constraints are discarded during SfM. Hence, large portion of training vectors successfully passing through SfM without getting rejected, indicates a better quality of matches.

3. **Average number of training vectors associated with a 3D point**: A good portion of 3D points computed from large number of 2D points indicates the ability of the matching method to group descriptors of the 3D point from different views into a single cluster.

4. **Planarity of 3D points belonging to a plane**: We manually mark the portions in the training images containing a single planar surface of the environment. All the 3D points computed from the SIFT descriptors belonging to these portions should lie in plane. We measure how well these 3D points fit to a plane.

5. **Computational efficiency**: The offline training process is not expected to run at video frame rate. But, clustering methods involving computationally intensive operations may become prohibitively expensive when the number of training vectors increase. We compare the CPU time required to complete various matching schemes applied in our experiments.

2 : Visual representation for 2D-to-3D matching during test stage

At the end of training stage we obtain 3D points in the environment and a set of SIFT descriptors belonging to the cluster (or visual word) associated with each 3D point. Given a set of SIFT descriptors from a test image (test vectors), we need to match each test vector with these clusters. This is a supervised pattern classification problem [30]. Hence we can use various statistical learning techniques to recognize 3D points in a test image. Matching a test vector with the whole set of training vectors may be computationally expensive. We try to reduce the cost of matching by experimenting with two additional sets of training vectors derived from the whole set of vectors extracted from training images. The first one is derived by retaining only those training vectors which are associated with 3D points i.e. those which are used to establish

2D-to-2D matches for SfM and at least one of such matches is retained and used to generate a 3D point by the SfM process. The second one is derived using only the centers of the clusters associated with the 3D points i.e. each visual word associated with a 3D point is represented by a single vector.

We can assess the outcome of the test stage at two levels, (i)accuracy of 3D point recognition and (ii)accuracy of pose estimation. Accuracy of pose estimation is dependent on the classification accuracy during 3D point recognition. But, due to the RANSAC step which can tolerate few outliers, we may not find much difference in the estimated pose even if some of the test SIFT descriptors are misclassified. In addition, the quality of the matched 3D points may influence the accuracy of estimated pose. In both cases (i.e. recognition and pose estimation) we need ground truth information in order to evaluate the performance. It is relatively easier to obtain the ground truth camera positions for evaluating the accuracy of pose estimation than to know the actual 3D points associated with the test vectors. In our experiments we mainly evaluate the accuracy of pose estimation. When we need to evaluate accuracy of 3D point recognition, we obtain the class label for the descriptors in test images by running the training stage in which test vectors are included along with the train vectors while performing clustering and SfM.

First we experiment with two different recognition strategies based on nearest neighbor classification (i) threshold on the distance with nearest neighbor, (ii)threshold on the ratio of distances to the two nearest training vectors belonging to two different 3D points. In both cases the matching is performed using different threshold values separately with the whole training set and the two additional derived training sets. For MS (mean-shift) and ATC based 3D models we perform an additional recognition technique. For MS, we perform mean-shift iteration (with the same parameters used during training) for each test vector. For ATC we match a test vector with different clusters using the adaptive threshold with which the respective clusters are formed during training. We run pose estimation on 2D-to-3D matches obtained through these various nearest neighbor based matching schemes on different training sets belonging to different 3D models. The estimated pose is compared with the ground truth for evaluation. Each evaluation of the estimated pose of the test images measures the performance of the combination of 2D-to-2D matching and 3D point recognition schemes employed during training and test stages respectively.

We further experiment with linear and non-linear classifiers which apply an optimal transformation on descriptor space before performing classification. For linear transformation based techniques we perform 3D point recognition using nearest neighbor strategies in the transformed domain as described in the previous paragraph. For non-linear classification, we use SVM with gaussian kernel. We learn one SVM classifier for each 3D point. We experiment with different values for kernel parameter σ. We also try the technique described in [64] to automatically compute the kernel parameter for each SVM from the available training samples.

3 : Acceleration strategy for mean-shift clustering

Mean-shift clustering iteratively moves each training vector towards the local maxima of the probability density function estimated from the training set[23]. The computational cost of MSC becomes prohibitively huge as the dimension of the feature space increases. The major source of this cost is the range search process [36] which involves computing, in each iteration, the set of training vectors within a range w from the current mean vector. While experimenting with

mean-shift clustering we conceived a strategy to accelerate its computation by dividing the set of training vectors into groups such that vectors in one group will never influence the computation of clusters in another group. The strategy performs exact computations unlike methods which use subsampling [34] or approximation [20]. But, we obtain moderate acceleration compared to these inexact methods, for a specific range of parameter values used to perform mean-shift. Our technique is applicable to all type of kernels which have compact support and fixed bandwidth.

Our acceleration method is based on the observation that, during mean-shift, if a training vector x is beyond $\sqrt{2}w$ from all the vectors which are within the range w from the current mean vector, then x will be beyond the range w from the next mean vector. Hence we need not compare the next mean vector with any of such training vectors. If we divide the training vectors into groups by applying transitive closure on the relation obtained by matching them with a distance threshold $\sqrt{2}w$, then each such group will never interfere with any other group during mean-shift computation. We do not know whether this property will hold for any threshold value less than $\sqrt{2}w$. But we prove that the threshold cannot be less than $\frac{\pi}{\sqrt{6}}w$.

Our acceleration strategy is useful only when the size of each group obtained through transitive closure operation is significantly less than the size of the whole training set. It requires that the training vectors are well separated from each other in groups so that the transitive closure operation on range based matching can provide reasonable number of partitions. When the data is well separated, fast range searching techniques can also increase the efficiency of mean-shift operation by arranging the data vectors beloging to separate regions in a tree structure through a pre-processing step. When a query vector is given, the tree structure is used to confine the range search to a limited region around the query vector. In our experiments we would like to see whether TC partitions can accelerate mean-shift operation which already utilizes a fast range searching technique. We use [79] for fast range search. We perform exact range search without using approximation option available in [79].

4 : Features from simulated views through ASIFT

If the 3D shape of the environment can be assumed to be locally planar, then new views of the environment can be generated by applying affine transformations to one of its views[78, 89]. We use ASIFT [78] technique, which computes additional SIFT descriptors from simulated views of the original image. These additional descriptors can be used to obtain matches when pose variation between training and test images is large.

We explore different ways of incorporating SIFT features obtained from simulated views into the 3D map. The first approach is to use descriptors extracted using ASIFT for performing SfM. We observe that this results in generation of new 3D points rather than addition of new features to the 3D points computed from descriptors from the original image. Hence we design methods which add descriptors to a 3D map from new views. We try adding the ASIFT descriptors closely located in the image to those associated with the 3D map. We also try using only the descriptor computation part of the SIFT technique in order to obtain descriptors from image locations in the simulated image which correspond to the locations of the descriptors in the original image which are associated with 3D map.

We design an experimental framework to compare the quality of 2D-to-3D matches obtained through these various schemes. Using the ground truth information available for one of the

datasets used in our experiments, we reduce the influence of unrelated factors (eg: quality of estimated 3D map through a RANSAC process which is not related to view simulation) which effect the performance measure.

Organization

Our work mainly involves establishing point correspondences (2D-to-2D and 2D-to-3D). The geometric constraints arising from these correspondences are used to solve the pose computation problem. In chapter 1, we briefly describe the tools we use to perform geometric calculations on point correspondences. It mainly involves concepts of projective geometry and RANSAC.

In chapter 2, we present different strategies in the available literature used for establishing point correspondences for pose estimation. We give a short summary of SIFT descriptor extraction method which is extensively used in our experiments (as well as in the past related works). We organize the literature along the lines of [60] and motivate the choice of our two stage framework described above based on the past works. At the end of the chapter we present the visual word framework used in our work to represent 3D points.

In chapter 3, we present the training stage of our pose estimation framework. It mainly involves the description of various matching schemes used to establish 2D-to-2D matches along with the parameters we use in our experiments to build 3D map. We describe the datasets and evaluation method we employ to measure the quality of different matching schemes. We present the experimental results and conclude with our observations based on the results we obtain.

In chapter 4, we present the test stage in which we experiment with various matching rules for recognizing the 3D points in a test image. The evaluation is performed on different combinations of 2D-to-2D matching schemes used in the training stage and the 3D point recognition strategy used in the test stage.

In chapter 5, we describe our mean-shift acceleration technique and the mathematical proof supporting the claim that our modification does not effect the final outcome of the algorithm. We present the moderate results we obtain on the SIFT descriptors extracted from the images we used in our experiments.

In chapter 6, we present our experiments with using features from simulated views. We justify the choice of using ASIFT and describe various approaches adopted for improving the robustness of 2D-to-3D match computation.

Finally, we conclude the thesis in chapter 7 by summarizing the various results we obtain in our work. We also mention different directions in which we would like to carry forward the investigation in future.

Chapter 1

Mathematical tools and softwares for solving geometry problems

1.1 Overview

As mentioned in the introduction of this thesis, in our work we mainly deal with establishing two types of *point matches* (aka *point correspondences*) which we refer as 2D-to-2D and 2D-to-3D respectively. Each 2D-to-2D point match (or simply 2D match) consists of a pair of 2D image coordinates of a single 3D point of the environment in two different images. A 2D-to-3D point match consists of a 2D coordinate in an image and the 3D coordinate of the corresponding 3D point in the environment. We use the geometric constraints arising from point matches to solve two problems in 3D geometry, namely Structure from Motion and Pose Estimation. In Structure from Motion (SfM) problem, given a set of 2D-to-2D matches in a set of images, we need to compute the pose of the cameras in the images and 3D coordinates of the points corresponding to the 2D-to-2D matches. In pose estimation problem, given a set of 2D-to-3D correspondences in an image we need to compute the pose of the camera in that image. This chapter describes the publicly available softwares we use to solve the two geometric problems from point matches. For SfM, we use Bundler[106] which is based on the widely referred work by Snavely et. al[107] (we use 2D-to-2D matches which we compute instead of the output of image matching process integrated with the Bundler package). For pose estimation we use EPnP[63] and DLSPnP[45] methods. EPnP provides fast solution to the problem through a non-iterative optimization process, but the currently available implementation cannot be used for 2D-to-3D matches in which all the 3D points lie in a plane. For planar 3D points, we use DLSPnP which provides solution for general case.

First we introduce in section 1.2, few notions of homogeneous coordinate system which are used to express the geometric relations in projective geometry. More details about the mathematical and computational concepts of projective geometry can be found in the excellent book by Hartley and Zisserman[43]. We follow the pinhole camera model to establish the relationship between the 3D points in the environment and their projection in an image. This projection model, introduced in section 1.3, sets the basis for formulating the geometrical framework of pose estimation problem. In the subsequent sections we describe how point correspondences can be used to recover geometric information from images obtained through pinhole camera. As already discussed above the first task (SfM) is to recover scene geometry using 2D-to-2D correspondence between a set of images. The second problem is to compute the pose of a camera

using 2D-to-3D correspondences. The methods to perform these tasks are presented in sections 1.5 and 1.6 respectively. Before that in section 1.4 we present RANSAC algorithm which is used to robustly perform the computations related to these two main tasks in the presence of incorrect point correspondences.

1.2 Homogeneous coordinates

In multiple view geometry we often encounter operations on vectors belonging to $\mathbb{R}^n$ (usually $n \in \{2, 3\}$) in which it is convenient to represent the vectors in *homogeneous coordinates*. Homogeneous coordinate system enables us to express many mathematical formulae in multiple view geometry in linear form. The homogeneous coordinates of a vector $\mathbf{x} = (a_1, a_2, ..., a_n)^T \in \mathbb{R}^n$, denoted by $\tilde{\mathbf{x}}$, is represented by $(sa_1, sa_2, ..., sa_n, s)^T \in \mathbb{R}^{n+1}$ where $s \in \mathbb{R}\backslash\{0\}$. Any two vectors $\tilde{\mathbf{x}}$, $\tilde{\mathbf{y}}$ in this homogeneous coordinates are said to be equivalent i.e. $\tilde{\mathbf{x}} \sim \tilde{\mathbf{y}}$ if $\tilde{\mathbf{x}} = s\tilde{\mathbf{y}}$ for some $s \in \mathbb{R}\backslash\{0\}$. The set $\mathbb{R}^{n+1}\backslash\{\mathbf{0}\}$ with such an equivalence relation $\sim$ is called a projective space and denoted by $\mathbb{P}^n$.

In this chapter we mainly use two notions in order to facilitate the description of the geometric methods used in our work. The first one is the function π which maps a vector $\tilde{\mathbf{x}} \in \mathbb{P}^n$ with non-zero value for the last coordinate to the inhomogeneous coordinates $\mathbf{x} \in \mathbb{R}^n$ as follows:

$$\pi(a_1, a_2, ..., a_n, a_{n+1})^T = (a_1/a_{n+1}, a_2/a_{n+1}, ..., a_n/a_{n+1})^T \tag{1.1}$$

If $\tilde{\mathbf{x}} \sim \tilde{\mathbf{y}}$ then $\pi(\tilde{\mathbf{x}}) = \pi(\tilde{\mathbf{y}})$.

The second one is related to the representation of points and lines in $\mathbb{P}^2$. A point $\mathbf{q} = (q_x, q_y)^T$ in 2D plane lies on the line $\mathbf{l} = (a, b, c)^T$ if and only if $aq_x + bq_y + c = 0$. This expression can be represented in $\mathbb{P}^2$ using innerproduct between $\mathbf{l}$ and $\tilde{\mathbf{q}} \sim (q_x, q_y, 1)^T$ as follows:

$$\mathbf{l}^T\tilde{\mathbf{q}} = 0 \tag{1.2}$$

Since the representation of $\mathbf{l}$ is also homogeneous, the points are lines are equivalent to each other in many aspects in $\mathbb{P}^2$.

1.3 Pose of a pinhole camera

Image is formed when the light rays reflected from the environment intersect the image plane of the camera. In our work we assume that all the images are captured through a pinhole camera in which the light rays forming the image converge to a single point called camera center as illustrated in figure 1.1. In addition we assume that the two adjacent sides of the image plane are orthogonal i.e. the image plane is a perfect rectangle without any *skew*. The line from camera center perpendicular to the image plane is called principal axis. The point where the principal axis meets the image plane is called principal point.

In practice, the real cameras may have radial distortion due to which the camera center, the image point $\mathbf{q}$ and the world point Q in the figure 1.1 cannot be assumed to be collinear. Hence, we estimate the distortion parameters of the camera using a calibration target in a separate process before conducting the experiments. Using these parameters we perform undistortion in order to fit each image into pinhole camera model.

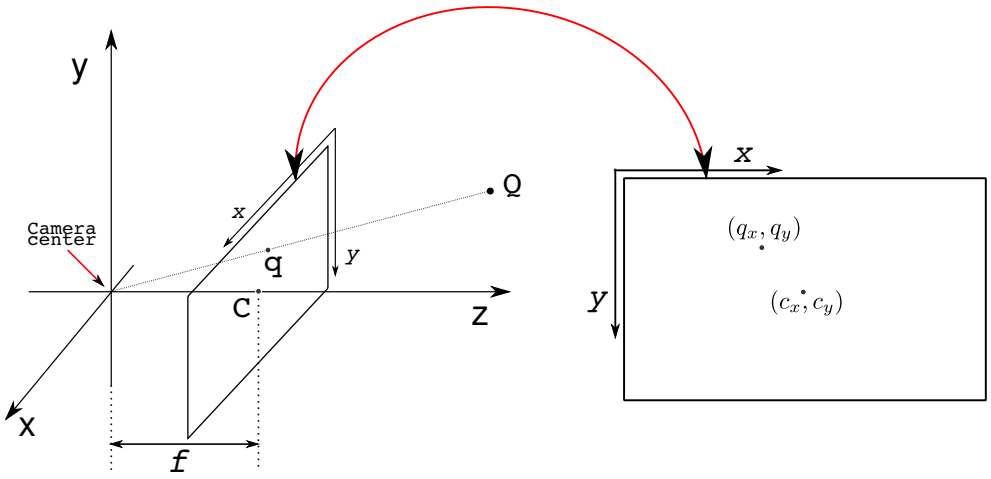


Figure 1.1: Pinhole camera model : For the sake of readability we show image plane between the camera center and the 3D point Q

1.3.1 Camera coordinate system

The 3D camera coordinate system for a given camera pose is defined by treating camera center as origin and principal axis as Z-axis. The X and Y axes are aligned respectively with the horizontal and vertical directions of the rectangular image plane. Similarly the 2D image coordinate system also defined by treating the horizontal and vertical lines aligned with the rectangular image plane as its x and y axes. Let (c_x, c_y) be the image coordinates of the principal point. If the distance f between the camera center and the image plane is expressed as f_1 and f_2 in terms of the pixel units of x and y axes of the image, then the image coordinates $q = (q_x, q_y)^T$ of the projection of a 3D point with camera coordinates $Q^c = (Q^c_x, Q^c_y, Q^c_z)^T$ can be expressed as:

$$\begin{bmatrix} q_x \\ q_y \end{bmatrix} = \pi \left(\begin{bmatrix} f_1 Q^c_x + c_x \\ f_2 Q^c_y + c_y \\ Q^c_z \end{bmatrix} \right) = \begin{bmatrix} (f_1 Q^c_x + c_x)/Q^c_z \\ (f_2 Q^c_y + c_y)/Q^c_z \end{bmatrix} \tag{1.3}$$

We define *intrinsic parameter* matrix K as

$$K = \begin{bmatrix} f_1 & 0 & c_x \\ 0 & f_2 & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

Then the equation 1.3 can be expressed as

$$q = \pi \left(K Q^c \right) \quad i.e. \quad \tilde{q} \sim K Q^c \tag{1.5}$$

The *normalized image coordinates* $\hat{q}$ of an image point q is defined as:

$$\hat{q} = K^{-1} \tilde{q} \tag{1.6}$$

1.3.2 Projection in world coordinates

In practice the 3D points are represented in the world coordinate system which is different from camera coordinate system. Hence, the coordinates of a 3D point should be transformed

to camera coordinates before applying equation 1.5. Let us assume that the camera center is situated at C and its orientation is represented by a 3×3 rotation matrix R in the world coordinate system. Then the transformation from world coordinates $Q = (Q_x, Q_y, Q_z)^T$ to camera coordinates $Q^c = (Q^c_x, Q^c_y, Q^c_z)^T$ can be obtained by $Q^c = R(Q - C)$. Substituting this value in the equation 1.5 we get $q = \pi(KR(Q - C))$. If we use homogeneous coordinates (i.e. $\tilde{q}$ instead of q and $\tilde{Q}$ instead of Q) and define $P = K[R \mid -RC]$, then, we obtain a linear expression :

$$\tilde{q} \sim P\tilde{Q} \quad i.e. \quad q = \pi\left(P\tilde{Q}\right) \tag{1.7}$$

We can see in equation 1.7 that the camera matrix P is a homogeneous matrix i.e. only ratio of the matrix elements is significant. If we use normalized image coordinates, then, equation 1.7 can be expressed in terms of $\hat{P} = [R \mid -RC]$ which transforms world coordinates to view coordinates as follows:

$$\hat{q} \sim \hat{P}\tilde{Q} \tag{1.8}$$

If we know K, R and C then for any image point q we can determine the set of 3D points in the environment that map to q under this camera projection. All such points fall on a ray centered at C with a direction $R^{-1}K^{-1}\tilde{q}$. Any point on this *back-projected* ray corresponding to q can be expressed as

$$C + a.R^{-1}K^{-1}\tilde{q} \quad \text{or} \quad C + a.R^{-1}\hat{q} \tag{1.9}$$

where a is a scalar.

The position C and orientation R, also referred as *extrinsic parameters*, represent the pose of a camera. Pose computation, which is the main goal of our work, is achieved by relating the image coordinates with the world coordinates. Rest of the chapter is dedicated to describing the mathematical and computational techniques used in Bundler[106] to establish world coordinates (with 3D points in it) and EPnP[63] to compute pose from point-to-point correspondences in images.

1.4 RANSAC

RANdom Sample And Consensus (RANSAC) algorithm [32] is a process used when we want to fit a mathematical model on a dataset containing gross errors (Eg: Incorrectly matched image points in a set of point correspondences). Suppose the mathematical model we want to fit is $\mathcal{F}_\theta$ which is determined by the set of parameters θ whose value can be estimated from a set of points S by minimizing the error:

$$\theta_S = \underset{\theta}{\operatorname{argmin}} \sum_{x \in S} Error(\mathcal{F}_\theta(x)) \tag{1.10}$$

Let n_r be the least number of data points needed in a set S in order to compute θ_S and the given dataset $\mathcal{D}$ contains $n > n_r$ number of points in which some of the points may be outliers. Outlying points are expected to have large deviation from the model that fits the rest of the points in $\mathcal{D}$. Due to the presence of such outliers, the parameter estimation process for equation 1.10 may produce incorrect result while trying to minimize the overall error by fitting the model onto grossly erroneous points. RANSAC is an iterative process which can be used to remove outliers from $\mathcal{D}$ before estimating θ.

In j^{th} iteration of RANSAC, a subset $S_j \subset \mathcal{D}$ is chosen randomly and θ_{S_j} (as defined in equation

1.10) is computed. Let $S_j^* \subset \mathcal{D}$ be the set of *inliers* i.e. points which are in agreement with the model $\mathcal{F}_{\theta_{S_j}}$ corresponding to the value of θ_{S_j}. The loop is terminated after a fixed number of iterations (other criteria for termination are possible [32]). Let S_{best}^* be the S_j^* with highest number of elements computed over the different iterations. Finally, θ is computed using S_{best}^*.

Formally, the RANSAC algorithm we use takes two arguments: (1)the error tolerance E_T used to determine whether a point is inlier or outlier w.r.t. a model and, (2) number of RANSAC iterations N_R. Initially $S_{best}^* = \varnothing$. In each iteration, the following operations are performed:

1. Select a random subset $S_j \subset \mathcal{D}$ with n_r elements.
2. Compute θ_{S_j} using S_j.
3. Compute the consensus set $S_j^* = \{x \in \mathcal{D} : Error(\mathcal{F}_{\theta_{S_j}}(x)) \leq E_T\}$.
4. If cardinality of S_j^* is greater than S_{best}^* then assign $S_{best}^* \leftarrow S_j^*$.

In the context of our work $\mathcal{D}$ is the set of point matches. The methods which perform geometric computation from point matches are used in step (2). S_{best}^* is the retained set of inlying point matches which is used to compute the final output at the end of RANSAC.

1.5 Geometry from 2D-to-2D point matches

The task of extracting geometric information from multiple views is initiated from two views. Section 1.5.1 provides brief description about fundamental and essential matrices which are used to represent the epipolar constraints in two views. Section 1.5.2 presents triangulation method used to compute the coordinates of a 3D point from its projection in images with known poses. It also describes homography based technique to verify whether a pair of images are well conditioned for triangulation. In section 1.5.3 we describe how these geometric concepts (presented in section 1.5.1 and 1.5.2) in conjunction with a technique called bundle adjustment can be used to perform Structure from Motion i.e. to compute the 3D points and camera positions from 2D matches on a set of images $\mathcal{I}$.

1.5.1 Epipolar geometry

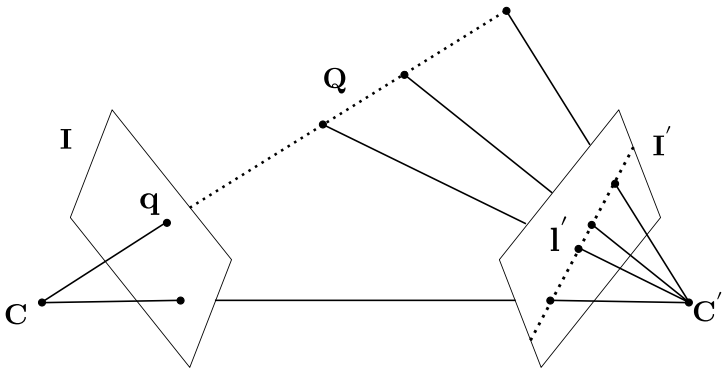


Figure 1.2: Epipolar constraints through 2D-to-2D match

The geometric information based on 2D-to-2D correspondences between two images is described through epipolar constraints. Figure 1.2 shows two images I and $I^{'}$ captured from two different poses. Given an image coordinate q in I the corresponding 3D point Q should lie on the line joining C and q. Hence the corresponding point $q^{'}$ in $I^{'}$, if visible, should lie on the projection of this line $\mathbf{l}^{'}$ in $I^{'}$. If the 2D points are expressed in homogeneous coordinates then this constraint can be expressed using a 3×3 *fundamental matrix* F as follows:

$$\mathbf{l}^{'} \sim F\tilde{q} \tag{1.11}$$

Since $q^{'}$ lies on this line we have

$$\tilde{q}^{'T} F \tilde{q} = 0 \tag{1.12}$$

If we know the intrinsic parameters K and $K^{'}$ then we can obtain a similar relation as in equation 1.12 for normalized image coordinates using *essential matrix* $E = K^{'T} F K$ expressed as follows:

$$\hat{q}^{'T} E \hat{q} = 0 \tag{1.13}$$

1.5.1.1 Computing F and E

Both F and E can be computed from 2D-to-2D correspondences. F is a rank 2 matrix with 7 degrees of freedom which can be computed without knowing the intrinsic parameter matrices K and $K^{'}$ (defined in equation 1.4) of images I and $I^{'}$. It usually requires 8 2D-to-2D correspondences. The computation can be based on algebraic error or geometric error. Algebraic error based computation minimizes the sum of the left hand side of equation 1.12 over the given 2D-to-2D correspondences[44]. Minimizing algebraic error for equation 1.12 mainly involves solving linear equations, hence, it is quick and easy to implement. Geometric error based methods try to minimize the geometric image distances, like distance between a 2D point and the epipolar line on which it should lie according to the computed F. These methods are computationally expensive but provide more accurate estimation.

Essential matrix E has only 5 degrees of freedom. Once we compute E for two images I and $I^{'}$ we can determine the relative rotation and direction of translation between the cameras in the images(section 9.6.2 of [43]). Hence, using E we can determine the relative pose of two images up to a scale. E can be computed either from F in which case we need eight 2D-to-2D correspondences or using 5-point algorithm[85] on normalized coordinates which needs only five 2D-to-2D correspondences but involves solving various non-linear equations. Hence, when there are many pairs of images, 8-point algorithm[44] to compute fundamental matrix is used for a RANSAC based process to discard incorrect 2D-to-2D correspondences (i.e. outliers). RANSAC based 5-point algorithm is used to compute E from the set of inlier matches obtained through RANSAC. As we see in section 1.5.3, the relative pose (up to a scale) computed from E is used to initialize the poses of two images chosen from $\mathcal{I}$. Once we know the poses of some of the images, we can compute the coordinates of the 3D points whose projection is known in at least two such images through triangulation as explained in the next subsection.

1.5.2 Well conditioned two-view triangulation

Suppose we have computed the relative pose of I and $I^{'}$ in figure 1.2 through E. For a given 2D match $q \leftrightarrow q^{'}$, we can obtain its 3D coordinates Q by computing the intersection of back-projected rays from q and $q^{'}$ as in equation 1.9. This process is known as triangulation. It

can be extended to multiple views in which case we have to find the 3D point close to multiple back-projected lines.

In practice the values of q and $q^{'}$ contain noise. When camera centers in two images (C and $C^{'}$ in figure 1.2) are very close i.e. when they have short baseline, the back-projected rays lie at a very short angle from each other. Under such a condition the 3D coordinates of Q obtained through triangulation is highly sensitive to noise in the values of q and $q^{'}$. Hence it is preferable to avoid initializing SfM with such image pairs. It turns out that the 2D matches $q \leftrightarrow q^{'}$ in image pairs with identical camera centers can be mapped through a 3×3 *homography* matrix H such that:

$$\tilde{q^{'}} \sim H\tilde{q} \tag{1.14}$$

If a significant portion of 2D matches between an image pair are outliers with respect to the homography estimated through a RANSAC process, then we can say that the baseline between the image pair is significantly wide for reliable triangulation.

1.5.3 Structure from Motion (SfM) through Bundler

In this section we briefly describe the Bundler package[106] which we use for the task of computing 3D points and camera poses for a given set of 2D-Matches $\mathcal{M}$ in a set of images $\mathcal{I} = \{I_1, I_2, ..., I_M\}$ of an environment. RANSAC process is incorporated in Bundler and we use the same parameters for RANSAC as in its code without any modification. Each element of $\mathcal{M}$ is a pair of 2D coordinates $(q, q^{'})$ belonging to two different images in $\mathcal{I}$. SfM process in Bundler has three stages. In the first stage the outliers in 2D-to-2D correspondences between each image pair are discarded based on geometric constraints obtained through RANSAC based fundamental matrix. In the second stage world coordinates are established by computing the relative pose through essential matrix corresponding to an image pair chosen from $\mathcal{I}$. Initial set of 3D points are obtained from this image pair through triangulation. In the third stage, additional images from $\mathcal{I}$ are added successively in an iterative process by computing their pose through the projection of 3D points computed till previous iteration. After adding new images bundle adjustment[67] process is applied to refine the values of camera pose and the coordinates of the 3D points. Additional 3D points are computed based on the 2D matches in the newly added images through triangulation. This iterative process in the third stage is continued till all the images in $\mathcal{I}$ are added. These three stages are explained in the following subsections.

1.5.3.1 RANSAC based epipolar constraints

In this stage the Bundler algorithm prunes the set of 2D-to-2D matches $\{(q_i, q_i^{'})\}$ between each pair of images (I_k, I_l). As mentioned in the beginning of the chapter, we mainly deal with establishing these matches (presented in the next chapter) and use the matches we compute instead of using the matching process integrated with the Bundler package. RANSAC based linear 8-point algorithm[44] is used to compute an initial set of inliers in $\{(q_i, q_i^{'})\}$. In each RANSAC iteration j, fundamental matrix F_j is computed using 8 randomly chosen elements in $\{(q_i, q_i^{'})\}$. Following error function is used to compute the inliers:

$$dist(q_i^{'}, F_j\tilde{q}_i)^2 + dist(q_i, F_j^T\tilde{q}_i^{'})^2 \tag{1.15}$$

This error function is the sum of the squared distances of the points in a 2D-to-2D match from the corresponding epipolar line determined by F_j. At the end of RANSAC, the fundamental

matrix F is recomputed through a non-linear process which minimizes the sum of the function in equation 1.15 over the set of inliers returned by the RANSAC process. Using the F obtained through non-linear minimization, inliers are recomputed. The outliers are discarded from $\mathcal{M}$. The retained inlier 2D-matches in $\mathcal{M}$ are *chained* to obtain 2D-tracks. Essentially, a 2D-track is a set of 2D points in different images which are connected to each other through the retained 2D-to-2D matches in $\mathcal{M}$. Each 2D-track is treated as 2D projections of a single 3D point.

1.5.3.2 Setting up initial image pair

SfM process is initialized on a pair of images in $\mathcal{I}$. This first pair of images are chosen so that we can compute its relative pose (through essential matrix E as explained in section 1.5.1.1) and an initial set of 3D points (through triangulation as explained in section 1.5.2). Hence the chosen pair should have sufficient number of 2D matches and should be well conditioned for triangulation. This initial pair is chosen as follows. For each image pair RANSAC based homography is estimated using the set of 2D matches which are inliers with respect to the corresponding fundamental matrix. The ratio of outliers w.r.t. this homography is computed. For an image pair (I_k, I_l), let a_{kl} and b_{kl} be the number of inliers w.r.t. fundamental matrix and the ratio of outliers w.r.t. homography matrix respectively. If there are image pairs with b_{kl} greater than a fixed threshold, then the image pair having highest a_{kl} among those is chosen. If no such image pair exists, then the image pair with highest b_{kl} among those having a_{kl} above a fixed count is chosen. If the second condition also fails (we never faced such a situation in our experiments) then SfM process is aborted. In the next subsection we describe how to grow this initial reconstruction from two images to include all the images and the 2D tracks in $\mathcal{I}$.

1.5.3.3 Bundle Adjustment

Bundle adjustment algorithm[67] uses the 2D projections of a set of 3D points in a set of images to refine the values of camera pose of the images and coordinates of the 3D points from an initial estimate by reducing the reprojection error through a non-linear least squares optimization process. Bundler starts with the estimate of camera pose and 3D coordinates for two images in $\mathcal{I}$ (as described above) and adds new images from $\mathcal{I}$ iteratively. In each iteration it refines the initial estimate by invoking bundle adjustment. The initial estimate of camera pose for the images and coordinates of 3D points to be added are obtained as follows. In each iteration the images in $\mathcal{I}$ which contain sufficient number of 2D-tracks of the already estimated 3D points are added to the bundle adjustment process. Their camera poses are initialized by Direct Linear Transformation (section 7.1 of [43]) using the 2D-to-3D correspondences obtained from those tracks. New coordinates of the 3D points corresponding to the additional 2D-tracks introduced by the newly added images are computed using multiple view triangulation. Finally, the process is terminated when all the images in $\mathcal{I}$ are processed or the number of 2D-tracks of the already computed 3D points in any unprocessed image is insufficient to estimate its initial pose.

1.6 Pose from 2D-to-3D point matches

In this section we briefly describe two algorithms we use to solve perspective-n-point (PnP) camera pose computation. RANSAC is not incorporated in the available implementaions of the two algorithms. Hence, we implement the RANSAC wrapper for both. Given a set of n 2D-to-3D correspondences $\{q_i \leftrightarrow Q_i\}_{i=1}^{n}$ in an image I we need to determine its position and orientation (C, R) where C is a 3×1 position vector and R is a 3×3 rotation matrix.

1.6.1 Efficient PnP (EPnP)

EPnP algorithm computes the camera coordinates $\{Q_i^c\}_{i=1}^n$ of the given 3D points. The values of (C, R) can be obtained by using the closed form solution[49] which computes the Euclidean transformation between $\{Q_i\}_{i=1}^n$ and $\{Q_i^c\}_{i=1}^n$.

The core of EPnP is a non-iterative method for computing the camera coordinates of four 3D points $\{Q_i^{'}\}_{i=1}^4$ called as control points. In the publicly available implementation of EPnP, which we use, these control points are obtained from $\{q_i \leftrightarrow Q_i\}_{i=1}^n$ as follows. $Q_1^{'}$ is the centroid of the 3D points $\{Q_i\}_{i=1}^n$. The rest of the three control points are chosen so that the set $\{Q_2^{'}, Q_3^{'}, Q_4^{'}\}$ forms a basis aligned with the principal directions of $\{Q_i\}_{i=1}^n$. Now, each Q_i can be uniquely expressed as follows:

$$Q_i = \sum_{j=1}^{4} \alpha_{ij} Q_j^{'} \quad \text{such that} \quad \sum_{j=1}^{4} \alpha_{ij} = 1 \tag{1.16}$$

Let $\{Q_i^{'c}\}_{i=1}^4$ be the camera coordinates of the control points. The same combination in the above equation can be used to express camera coordinates as $Q_i^c = \sum_{j=1}^4 \alpha_{ij} Q_j^{'c}$. Hence, to compute (C, R), all one needs to obtain is $\{Q_i^{'c}\}_{i=1}^4$.

From equation 1.5 we obtain $\tilde{q}_i \sim K Q_i^c = K \sum_{j=1}^4 \alpha_{ij} Q_j^{'c}$. Expressing this relation with x, y and z subscripts indicating the coordinates in the respective axes we get

$$\begin{bmatrix} q_{ix} \\ q_{iy} \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_1 & 0 & c_x \\ 0 & f_2 & c_y \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} Q_{jx}^{'c} \\ Q_{jy}^{'c} \\ Q_{jz}^{'c} \end{bmatrix} \tag{1.17}$$

With a little algebraic manipulation on equation 1.17 we can obtain the following:

$$\begin{aligned} \textstyle\sum_{j=1}^4 \alpha_{ij} f_1 Q_{jx}^{'c} + \alpha_{ij}(c_x - q_{ix}) Q_{jz}^{'c} = 0 \\ \textstyle\sum_{j=1}^4 \alpha_{ij} f_2 Q_{jy}^{'c} + \alpha_{ij}(c_y - q_{iy}) Q_{jz}^{'c} = 0 \end{aligned} \tag{1.18}$$

Writing the above equations in a matrix form with $Q_i^{'c}$s as unknowns we get

$$MY = \mathbf{0} \tag{1.19}$$

where M is $2n \times 12$ sized matrix and $Y = [Q_{1x}^{'c}, Q_{1y}^{'c}, Q_{1z}^{'c}, Q_{2x}^{'c}, ..., Q_{4z}^{'c},]^T$ is 12×1 vector of unknown camera coordinates of the 3D points which we want to compute. From this equation it is clear that Y is a linear combination of the singular vectors $\{Y_i\}$ corresponding to right null space of M.

$$Y = \sum \beta_i Y_i \tag{1.20}$$

These singular vectors are the eigen vectors of 12×12 matrix $M^T M$. Hence, in order to obtain the camera coordinates Q_i^c, all we have to compute is the values for β_i. As explained in [63], if we enforce the constraint that the distance between any two of the 3D points in the camera coordinates $Q_i^{'c}$ should be same as in world coordinates $Q_i^{'}$, then the task of computing the coefficients β_i of this linear combination can be performed by solving a small constant number of quadratic equations. Thus EPnP method provides a solution without involving iterative minimization of the reprojection error. When all the 3D points lie in a plane, it is not possible to compute the control points $\{Q_i^{'}\}_{i=1}^4$ in the way it is done in the current implementation of

EPnP. For such cases we use the software described in next section which is slower than EPnP but provides solution for general configuration of 3D points.

1.6.2 Direct Least-Squares PnP (DLSPnP)

The unit vector from camera center towards a 3D point Q_i in camera coordinates can be expressed as

$$\bar{q}_i = \frac{\hat{q}_i}{\|\hat{q}_i\|} \tag{1.21}$$

where $\hat{q}_i$ is the normalized image coordinates (equation 1.6) of the projection q_i of Q_i in the image. For a camera with pose (C, R) this value in equation 1.21 can be expressed as

$$\bar{r}_i = \frac{RQ_i - RC}{\|RQ_i - RC\|} = \frac{RQ_i + t}{\alpha_i} \tag{1.22}$$

where $t = -RC$ and $\alpha_i = \|RQ_i + t\|$.

DLSEPnP[45] computes (C, R) by minimizing $\sum_{i=1}^{n} \|\bar{q}_i - \bar{r}_i\|^2$. Formally the optimization process can be expressed as follows:

$$\begin{aligned} \underset{\alpha_i, R, t}{\text{minimize}} \quad & \sum_{i=1}^{n} \|\bar{q}_i - \frac{RQ_i + t}{\alpha_i}\|^2 \\ \text{subject to} \quad & R^T R = I_3,\ det(R) = 1,\ \alpha_i = \|RQ_i + t\| \end{aligned} \tag{1.23}$$

In [45] C and α_i's are expressed in terms of R and R is expressed using 3 Cayley-Gibbs-Rodrigues (CGR) parameters. This removes all the constraints in equation 1.23 and transforms the whole optimization process into minimization of a polynomial function with respect to CGR parameters. Different roots of the optimality condition of this polynomial function provide different solutions. Out of these multiple solutions we choose the one having minimum reprojection error on the given set of n 2D-to-3D correspondences $\{q_i \leftrightarrow Q_i\}_{i=1}^{n}$.

Chapter 2

Feature correspondence for pose estimation: Literature survey

In the previous chapter we described the available software tools we use to obtain geometrical information from point correspondences. The main topic of our investigation is to establish point correspondences in images. In this chapter we discuss various methods available in literature related to establishing feature correspondences in images for pose estimation. A feature correspondence is the 2D location of an image feature (eg: points, lines, curves or any image region with a specified pixel pattern) associated with an object in the environment. Similar to the notion of point correspondence (defined in the previous chapter) we refer to 2D-to-2D and 2D-to-3D feature correspondences as follows. A 2D-to-2D feature correspondence associates the image features of an object in two images. A 2D-to-3D feature correspondence associates an image feature with the 3D coordinates of the corresponding object. Feature correspondences are established in images by detecting image features of the environmental objects in images by using their *visual characteristics* i.e. measurements based on the pixel patterns in their images. We classify the methods for establishing feature correspondences in video frames into two categories:

1. **Feature tracking without learning**
   Smooth variation of pose in consecutive frames of the video is assumed. Feature locations, once initialized in a frame, are searched locally in a neighborhood of their previous position.

2. **Feature recognition through offline learning stage**
   Visual characteristics of some of the objects in the environment is learned from their example images during an offline training stage to obtain their *visual representation*. The visual representation of an object is used to *recognize* it in the current video frame without any prior assumptions on its location in the image.

In section 2.1, we present one of the popular techniques used to identify reliably detectable points in images which are suitable for matching. In sections 2.2 and 2.3 we present existing techniques in tracking and recognition respectively (more detailed survey is available in [60] which is the inspiration for the categorization in these two sections). Methods which use a combination of these two techniques are presented in the subsection 2.3.3. In section 2.4 we describe visual word framework which can be used to incorporate some aspects of the text matching techniques for matching images. Finally, in section 2.5, we conclude by motivating our course of investigation based on the past works described in this chapter.

2.1 SIFT : Interest point detection and description

Interest point detection techniques locate points in an image with properties which are stable[33] under some of the transformations which are likely to occur in the conditions in which they are used. They are distinctive from their immediate neighbors and surrounded by textured regions. Interest point techniques choose a subset of image locations which satisfy certain conditions like extrema of output of some filter operation. Matching only at few selected locations reduces the computational complexity and the conditions used to choose them increase the reliability. Interest point descriptors are used to quantify the visual characteristics of an image point using the local pixel patterns around it. A combination of interest point detector and descriptor can be used to automatically detect suitable locations in images and match them by comparing their visual characteristics represented by the descriptors. However, in practice, most descriptors incorporate only approximate model of the real transformations. The robustness of matching can be further improved by using descriptors from training images[108]. In this section we briefly describe SIFT feature computation [69] method which we use extensively in our experiments. This 128 dimensional descriptor vector has invariance properties w.r.t. scale and in-plane rotation of the camera.

2.1.1 SIFT descriptor computation

Gaussian and DoG (Difference of Gaussian) pyramids of the original image are constructed through scale-space computation. The image locations corresponding to extrema in different scales (or levels) of DoG pyramid are used as keypoints. The extrema locations are computed through interpolated DoG function to obtain subpixel resolution. Descriptor for a keypoint is derived from pixels in its fixed size neighborhood in the level of the gaussian pyramid corresponding to the level of DoG in which the keypoint is detected. This selection of neighborhood pixels from gaussian pyramid based on the scale achieves scale invariance. Magnitude and orientation of the local image gradient is computed for each pixel in the neighborhood. Dominant direction of image gradient in the neighborhood is assigned as orientation of the keypoint. Histograms of the image gradients from sub-patches of the neighborhood (typically 8 bins from each of 16 different 4×4 patches of 16×16 neighborhood) are concatenated to obtain 128 dimensional descriptor. In order to achieve orientation invariance, the entries in the descriptor and the gradient orientations are rotated relative to the keypoint orientation before computing the histogram.

2.1.2 Standard SIFT keypoint matching

In [69] a technique for matching two sets of SIFT descriptors A and B extracted from images I_A and I_B is mentioned. We refer to it as Standard SIFT keypoint matching. For each element x_A in A, its first and second nearest neighbors x_{B1}, x_{B2} are computed in B. Let d_1 and d_2 denote the distance of x_{B1} and x_{B2} from x_A. We say that x_A matches its first nearest neighbor x_{B1} if the second nearest neighbor x_{B2} is significantly farther from x_A when compared to x_{B1} i.e. $\frac{d_1}{d_2}$ is below a fixed threshold.

2.1.3 Other keypoint detectors

In the past decade many interest point methods have been proposed [92, 13, 89]. SIFT method is one of the best candidate for computing descriptors around interest points in terms of matching accuracy [74, 25]. But it is computationally expensive for real-time applications. Many alternative faster methods have been proposed for this purpose. FAST keypoint detector[92] learns the

rules for detecting interest points (corners) from a set of training images. Random Ferns[89], BRIEF[19] use simple binary tests for feature matching to achieve faster computations. But experimental evaluation [65] suggests that SIFT performs better than Random Ferns in terms of matching accuracy. FAST and BRIEF need additional modifications [95] to achieve robustness against scale and in-plane rotation. Hence, we chose SIFT method for our experiments despite the computational cost of comparing its high dimensional descriptors. In order to achieve real-time processing we may have to explore, in future, parallelizing the computations or mapping descriptors to metric space in which computations can be faster[108].

2.2 Feature tracking without a learning stage

Methods which establish feature correspondences in video frames without a learning stage rely on the objects of the environment which produce automatically detectable generic features (like edges or interest points) in their images or use the pixel patterns around the detected object features in the adjacent previous frames in order to detect them in the current frame. In both cases it is desirable to have a prior guess about the pose of the camera in the current frame. These features can only provide candidate positions for some of the objects of the environment in an image. A prior guess about their 2D-projection in the current image is needed to identify which is which. Similarly, it is difficult to perform a robust real-time exhaustive search in the entire current image for the object features detected in the previous frames. Hence, in feature tracking, features of the objects of the environment (once initialized in one of the frames) are searched locally in the subsequent frames around their previous image positions.

Some of the pose estimation methods based on feature tracking do not need the 3D model (i.e. the 3D coordinates of the objects corresponding to the image features being detected in images) of the environment. For example in [103] pose is obtained from the homography between a planar structure of the scene and the current image. For the cases in which the 3D model is required, it is assumed to be known or computed online[27] using SLAM (Simultaneous Localization and Mapping)[11] based techniques.

2.2.1 Tracking using edge-based features

Gradient and edge information can be computed efficiently from images using simple linear filters. Due to their dependence on variation of image intensity rather than actual value of intensity, edge-based methods are inherently less sensitive to changes in illumination. But, due to the aperture problem it is not possible to compute the exact location of a point within the edge contour.

The strategy of using a robust technique to initialize the pose (and thereby the projection of the 3D model) in an image and then using a fast 2D tracking technique in the successive frames of a smoothly varying video is employed in one of the earliest experiments by Bray[18]. Line based 3D model is initialized in the first image by using an improved version of the discretized *view-sphere* search [37] around the model. To cope with change of scale (or distance from the camera), the author of [18] uses *angle constraints* and *direction constraints* [40] on the reprojected lines while searching the view-sphere. The pose is refined using the line correspondences based on the method in[68]. To side-step the problem of aperture and partial occlusion, the reprojected 3D lines are extended over the entire image in the successive frames while tracking through *disparity analysis*[12]. But the iterative nature of the disparity analysis process prevents it from achieving

realtime speed.

In [42](RAPiD) 3D object points that fall on high contrast edges are chosen as control points. The perpendicular distance between the positions of the control points in the previous frame and the corresponding high contrast edges in the current frame is expressed as a linear function of the parameters representing the change of pose. Use of perpendicular distance from the edge feature avoids the aperture problem. The linear relation reduces the pose computation process to finding the pseudo-inverse of a matrix to achieve realtime speed.

[101] uses piecewise curved contour tracker[14] in which the contour is represented as a chain of 3D points. This permits the use of features other than just points and lines for tracking. The set of contours S is manually initialized in the first frame. Subsequently S is updated to reflect disappearance or re-appearance of features in the current frame based on reprojection error and the presence of edges along their projection. In [56], multiple edge correspondence hypotheses are maintained using *condensation* algorithm[53] to robustly track the edges of complex 3D objects with self-occlusions under abrupt motions.

Edge detection process needs carefully chosen threshold values. It is difficult to come up with a universally applicable set of threshold values. Aligning the reprojected edges of the 3D model to the image regions with high gradient norms can obtain better results in such cases. In [58], a 2D line sketch of the polyhedral 3D model (used for representing a vehicle like car) is generated by projecting the visible segments on to a blank image based on the previous pose. This line image is smoothened by a Gaussian filter to obtain gradient of the view sketch. 3D pose of the model is estimated by reducing the difference between this synthetic gradient and the gradients of the current frame. [72] exploits additional information that the expected direction of the projected model contour, ideally, should be perpendicular to the direction of the gradient. Here, the gradient fitting algorithm tries to minimize the sum of the dot products of the gradient and the contour direction over the whole reprojected contour of the model.

Apart from aperture problem, edge-based tracking has two additional drawbacks. It restricts the set of 3D models to those having sharp edge features. It is sensitive to background clutter which may create confusion in associating the 3D objects with image features. To overcome these drawbacks the features should exploit the local texture information in the images.

2.2.2 Using texture information for tracking

Optical flow [50] provides dense correspondences between two images by matching the pixels with similar intensity values under smoothness condition on the variation of pixel velocity in the neighborhood. But it quickly results in drift. Additional verifications are needed to prevent it. In [82], after computing optical flow, different regions in the image are warped using affine deformation models estimated from the motion vectors in the respective regions. Normalized SSD difference of a region with the corresponding warped image is used as confidence measure for the motion vectors in the region. Only piece-wise planar regions with correctly estimated motion vectors will pass the confidence measure. In [77] graph-cut based cost constraint is applied when the pixels enclosed inside an edge contour in the previous frame are mapped to opposite sides of the edge through motion vector.

Interest points combine the advantage of edge and texture based methods. They are detected at

stable image locations of textured regions. They can be matched across different images using simple process like cross correlation of the image patches around them. In [26], interest points from the saliency operator of Shi and Tomasi [98] are matched in successive frames using normalised sum-of-squared difference correlation to compute pose through a SLAM based 3D model. In [102] planar structures of the world scene are tracked (after manual initialization in the first frame) using minimum intensity change detector[111]. Pose is computed from frame to frame homographies.

2.2.3 Need for offline learning

Feature tracking based methods which use SLAM for 3D model construction suffer from drift. They need regular refinement of the 3D map[86] or additional equipments like stereo camera[113]. Even for small sized environments a parallel batch method to build accurate 3D map is recommended[55]. As we see in section 2.3.3, even for the task of robot navigation in which a robot for most of the time moves in a fixed orientation, it is preferable to build the 3D model of the environment through an offline process and apply strategies of recognition or keyframe selection methods to obtain robust matching[94].

In order to avoid using SLAM, 3D model of the environment should be provided. It is difficult to have 3D models for large environments. Even if the 3D model is available, the features corresponding to the 3D points should be initialized in the first frame. The tracking methods which use simple correlation between pixels in successive frames for matching are not robust to recover the features which reappear after intermediate occlusion. Online realtime learning methods [90, 39, 46] have been proposed for obtaining robust detectors. But the results are mainly obtained on small planar objects. An offline learning stage can be used to overcome these limitations by building a 3D model and computing robust visual representation for 3D points from a set of training images.

2.3 Recognition based methods

Recognition based methods use visual representation to establish 3D feature correspondences in a given image. In our work we focus on visual representation of 3D points. Visual representation of a 3D point consists of a set of values computed from one or more images of the 3D point. The representation should enable identification of the corresponding 3D point in the images from widely varying views. Wide variation in viewpoint causes huge change in the image of the objects. This is more challenging when compared to 2D feature tracking in which the image pattern around a point vary smoothly in successive frames. Computationally the cost increases due to two reasons. In order to achieve robustness the process of obtaining appearance descriptors may involve computations which are costlier than simple correlation or edge detection used in 2D feature tracking. Secondly, the matching process should be performed over the full image, unlike 2D feature tracking in which the pose of the previous image reduces the search region significantly.

We classify the types of appearance descriptors into two categories, *global* and *local*, which we present in next two subsections. In our work we use local descriptors which, over the time, have become immensely popular compared to the former for the reasons we mention at the end of section 2.3.1.

2.3.1 Global appearance based representation

The challenge of identifying 3D points from arbitrary pose is similar to those tackled by object recognition techniques which use supervised pattern classification[30] to obtain a robust representation of a given object from training images captured from different viewpoints. These techniques[80, 116] learn *global appearance* of a specific homogeneous *class* of objects (here the class can be human faces or a particular kind of fruit, furniture etc) from a set of normalized training images in which each image contains one instance of the whole object with a little or no scale variation.

In [59] global appearance of the objects is used to obtain initial approximation of the pose. Training images of the object to be tracked are captured by evenly varying viewpoint around it while simultaneously recording the camera pose in each image. Training images are normalized by resizing the sub-image containing the segmented object to a fixed size. Discrete Cosine Transform (DCT) based encoding after truncating higher frequencies of each image is used to represent global appearance. While tracking, DCT co-efficients of the object (which is segmented and resized to the size of training images) are directly compared with those in the training set to obtain initial estimate of the pose which is refined further by matches obtained through edge based local features.

It is difficult to adapt global appearance based methods to scale changes and occlusion. Training process involves meticulous task in which images need to be captured and normalized for each object we want to detect. Moreover, the discrete valued labelling cannot be used directly to infer continuous values like position and orientation. Hence it is more appropriate to use interest points based methods which automatically identify reliably detectable points in an image using local image characteristics. Subpixel level information provided by interest points can help to compute accurate pose. Even during partial occlusion, the local features from the visible portion of the environment can provide necessary information for identification. Next we present pose estimation techniques based on local appearance.

2.3.2 Local appearance based representation

Attempts to perform recognition based pose estimation through local interest points goes back to late 90's. Even though interest points based on simple computational steps (termed as *characteristic points*) have been used to detect stable landmarks of the environment [48] even earlier, they are rather used for deciding navigation instructions (like turn left or right etc) for the robot than exact localization. In [99, 100], a set of training images were captured from known locations within a planar grid of 2.0m by 2.0m. Interest points (termed as *landmarks*) detected at local maxima of edge density in training images are grouped based on the Euclidean distance between the PCA (Principal Component Analysis) representation of the image subwindow around the points. For online-localization, similar landmarks are detected in the image captured from current unknown location and matched with the landmarks that are grouped during training. Linear variation of the landmark characteristics w.r.t. camera pose is assumed. Pose is computed iteratively as a linear combination of the poses of the training images corresponding to the matched tracks. Here the pose is computed without building the 3D map of the environment.

2.3.2.1 Recognition using known 3D models

In [62] the rules of point matching are learned from training images in order to compute the pose of an object in a given image. The object is either assumed to be planar or the 3D model and the pose of the object in the training images are assumed to be known. New views of the object are synthesized from the available views in the training images. If the object is locally planar then new views are generated by applying simple affine transformations to the training images. Otherwise, standard computer graphics texture mapping techniques are used to generate new views of the 3D model. Harris interest points [41] are detected in the original training images. Each interest point defines a *point class*. For each detected interest point 100 synthetic views are generated. Image patches of size 32×32 are selected around each interest point in the training images and the simulated views. The patch is half-sized and 20 dimensional PCA transform is learned from the set of patches. The set of patches are transformed to this PCA space to obtain a set of training vectors with class labels. 20 *mean vectors* are computed for each class using k-means clustering. Nearest Neighbor classifier with the set of means is used to classify a new vector. To reduce incorrect matches, the training vectors which get wrongly classified are identified during the training stage. A point class having more than 10% of misclassified training vectors is discarded. In [61, 89] the run time complexity is further reduced by using *randomized trees* and *random ferns* respectively instead of Nearest Neighbor classifier. In these methods different views of a 3D point are obtained during training either by planar assumption or by mapping the texture of the training images captured from known viewpoints onto the known 3D model. They cannot learn the different views of 3D points from training images without these assumptions.

2.3.2.2 Recognition without a given 3D model

Interest point descriptors [69, 13] can be used to obtain 2D matches when no prior guess about their location in the training images is available. These 2D matches can serve two purposes. First, they can be used to obtain 3D points in the environment and camera positions in the training images by imposing geometric constraints (section 1.5). Secondly, the descriptors associated with a 3D point can be used to obtain its visual representation.

In [93], 3D information extracted from training images is used to perform object recognition in heavily cluttered pictures taken from arbitrary viewpoints. A set of training images of each individual target object is separately captured (without any clutter) by systematically selecting viewing angles in order to cover its various views. 2D matches are computed between manually selected pairs of training images. For each pair of images 2D matches are established in multiple stages. Affine invariant regions [75] based on Harris interest point[41] and DoG (Difference of Gaussian) operator are computed. A SIFT[69] descriptor and a parallelogram shaped image patch based on dominant gradient orientation are used to describe each interest point. Under locally planar assumption on the object surface, a *rectifying transformation* which transforms the parallelogram to a unit square is associated with each interest point. The initial list of potential 2D matches between each manually selected training image pair is found by choosing for each patch in the first image a fixed number of top patches in the second image as ranked by SIFT descriptor. The potential matches which satisfy neighborhood consistency in the respective images and normalized correlation criteria between the corresponding rectified image patches subjected to RANSAC based geometric consistency test. 3D model of the object in the training images is computed from the 2D matches through bundle adjustment. During the test stage

similar matching strategy is adopted to match the patches from a test image with those associated with 3D point. The over all matching process for this technique is computationally intensive.

The need for manual selection of image pairs and computationally complex representation of interest points for matching limits the applicability of [93]. In [38], the 3D map of the target environment is computed from a set of training images for pose computation. 2D matches are established between each pair of training images by matching the SIFT descriptors extracted from respective images and subjected to epipolar constraints in order to remove outliers. In order to avoid computing epipolar constraints between each pairs a selective approach is followed. Image pairs are sorted in the decreasing order of the number of 2D matches. Epipolar constraints are computed between the pairs in this order while skipping the pairs which do not add any new member to the set of images already participated in this process. This process imposes epipolar constraints in a linear time over the number of images. The set of image pairs with geometrically consistent matches is a tree structure free of cycles. Multi-view 2D correspondences (2D tracks) are computed by traversing the tree and stitching together pairwise matches. 3D points corresponding to the 2D tracks are computed through bundle adjustment. This establishes the association between a computed 3D point and the SIFT descriptors corresponding to its 2D track. At run time SIFT descriptors extracted from the current frame are matched with the descriptors associated with 3D points. The 2D-to-3D correspondences thus established is used to compute pose of the camera through a RANSAC based process which minimizes the reprojection error of the 3D points identified in the current frame. The main drawback here is that the number of descriptors associated with the 3D map grows linearly with the number of training images. Hence it is difficult to obtain real-time performance for large environment. In [29] 3D map is built in a similar way. But the SIFT descriptors from the test image are matched only with selected training images. For each test image a small set of training images are selected using loop closure detection technique[8]. Loop closure detection is accelerated by arranging the training SIFT descriptors in a hierarchical data structure[87].

Recently, clustering methods are increasingly becoming popular to group descriptors computed from all the training images instead of pairwise image matching. Clustering techniques provide scope to use the information in the overall distribution of the descriptors. In [120] object recognition involving multiple objects is performed by building 3D model of each object (as in [93]) from a training set of its images. SIFT descriptors extracted from the training images taken around an object are clustered using k-means clustering. The SIFT descriptors in different images that belong to the same cluster are associated with each other to obtain 2D matches across the training images. The 3D points obtained through structure from motion on these 2D matches are associated with the corresponding clusters. When a test image is given, the SIFT descriptors extracted from it are matched with each 3D model using threshold on the distance to the nearest k-means cluster centers associated with 3D points of the model. Over-segmentation is performed on the test image to obtain different image regions, hoping that no region thus obtained is part of more than one object. Pose hypothesis is obtained for each pair of region and 3D model by using regionwise 2D-to-3D matching. The 3D points in different models are reprojected on to the test image using each hypotheses. Voting based on the reprojected points on different segmented regions is used to select the right hypothesis.

In [51] mean-shift clustering[22, 23] with multiple *bandwidth* parameter is used instead of k-means. Mean-shift method automatically decides the number of clusters to be formed based on the bandwidth parameter which is a rough indication of intra-cluster variation. SIFT descrip-

tors from a test image are matched with clusters at each bandwidth level separately. A test descriptor is matched based on the ratio of the first and second nearest neighbor cluster center. Matching at multiple bandwidth levels provides more matches since a test descriptor rejected at one level may get accepted in another level. But it also significantly increases outliers which may drastically increase the number of RANSAC (presented in section 1.4) iterations needed to guarantee a consistent set of inliers corresponding to correct matching. Hence the authors present *view-constrained RANSAC* which exploits constraints based co-visibility of the model features to reduce the required number of RANSAC trials.

In [31] mean-shift clustering is applied on the image locations of the features extracted from the test image to spatially isolate different objects. Multiple instances of the same object within a spatial cluster are identified by the number of inlier sets of matches between the descriptor vectors of the points within the spatial cluster. 3D models of the different objects in the database are matched with the features of each isolated spatial cluster. Pose of the object is computed through a RANSAC process in which the sample selection (i.e. choosing elements of S_j in the description provided in section 1.4) is prioritized based on the visibility, co-visibility of the 3D points.

In [17], training stage is used for improving 2D matches during test stage without building any 3D model. Visual word vocabulary tree [87] is constructed during offline training stage from training images. During test stage, this visual dictionary is used to establish 2D matches for the current frame with the previous frames. Using the 2D matches, the pose of the camera in the current frame is computed relative to the previous frames.

2.3.3 Combining different 2D tracking features and recognition

While the robustness of recognition methods can solve the problem of gradual drift and need for manual initialization, they introduce jitter due to the independently computed pose in successive frames. They are computationally expensive when compared to simple SSD based correlation techniques used for matching points in consecutive frames of a video. In this section we describe the methods which combine tracking and recognition techniques in order to achieve robustness and speed without jitter.

In [94], 3D map built from a set of training images through the 2D-to-2D matches obtained by zero normalized cross correlation around Harris corners[41] is used for autonomous robot navigation. At run time, 2D-to-3D correspondence for a test image is obtained by matching the Harris corners in the test image with those associated with 3D points in a selected keyframe from the set of training images. The keyframe for the first test image is chosen by selecting the training image with highest number of inlying matches. For each subsequent test image, the keyframe closest to the camera center of the previous image is selected for matching. This simple strategy for keyframe selection and matching is possible since the path followed by the robot during test and train are very close.

2D-to-2D matches with an image of known pose can provide geometric constraints which can be used in combination with the 2D-to-3D geometric constraints to improve robustness when the available 2D-to-3D correspondences are insufficient or concentrated in a particular portion of the current image. A basic framework for such a combination is presented in [104]. Pose is estimated by minimizing the weighted sum of reprojection error corresponding to 2D-to-3D matches and

the average distance of 2D-to-2D matches with the previous frame from their epipolar lines. In [112] 2D-to-2D matches with the previous frame is used to reduce jitter and improve robustness of pose estimation. Its formulation involves computation of a transfer function which uses the CAD model of the environment to compute the error of the 2D-to-2D matches without explicitly computing their 3D coordinates. Interframe jitter can also be reduced by adding a cost function which penalizes large change in pose between successive frames[90]. In [109] the 3D structure obtained through an offline SfM process is used to obtain additional geometric constraints on the matches compute during online. It does not use any photometric information from the offline stage.

2.4 Visual word framework

In the visual word framework[105, 87], a dictionary of visual words $\{D_1, D_2, ..., D_k\}$ is built in such a way that a feature descriptor x extracted from 2D location of an image can be matched with any D_i to decide whether x belongs to the word D_i or not. Thus, an image can be considered as a document containing a set of visual words obtained by matching each descriptor extracted from the image with the dictionary. This enables us to apply the text search techniques to perform image retrieval[105], location recognition[97], loop-closure detection[8] etc.

In practice visual words are formed from the set of training vectors $D = \{x_1, x_2 \dots x_n\}$ (i.e. the set of descriptor vectors extracted from training images) in two steps:

1. **Visual word formation**: Perform clustering on D to obtain a set of clusters $D^c = \{D_1, D_2, ..., D_k\}$ such that $D_i \subset D$ and $D_i \cap D_j = \varnothing$ for $i \neq j$. Each cluster D_i forms a visual word.

2. **Visual word recognition**: Using the elements of visual word D_i, obtain the rule for matching a test descriptor vector x with it.

There are multiple options in both steps. In step (1), different clustering techniques will lead to different D^c. After clustering, we can use various rules in step (2) for matching a test vector with the clusters in D^c. Each matching rule defines a visual word region v_i for cluster D_i in the descriptor space F, which consists of all the vectors in F that match with D_i. That is, descriptor vector x is said be matching with D_i if $x \in v_i$. We use visual word framework to represent 3D model in such a way that each 3D point is represented by a visual word. Ideally, we would like each 3D point in the 3D map of the environment to be associated with one visual word region v such that the local descriptor extracted from any view of the 3D point belongs to v and a local descriptor of any other 3D point in the environment lies outside v.

2.4.1 Types of visual words

In this subsection we classify the visual word computation techniques based on the way in which the training vectors are clustered. Each clustering technique is characterized by the parameters need to be provided, the flexibility of shape of clusters it can form and the computational cost. In our experiments, we have applied visual words from each category on SIFT[69] features extracted from training images.

2.4.1.1 Type1 : k-means based clustering

In the training step, the set of training vectors is divided into a pre-determined k number of groups by clustering [105, 87]. Training vectors within the Voronoi region corresponding to the cluster

center of a group form a visual word. K-means algorithm may turn out to be computationally slow during training due to range search operation in each iteration till convergence. Another drawback of this method in this context is the need to fix the value for k i.e. the number of visual words. Since we use visual words for representing 3D points in the environment, this condition requires a rough estimation of the number of different 3D points associated with the training vectors.

2.4.1.2 Type2 : Range based matching

The training vectors within a distance range r from each other are grouped into the same visual word. In this framework we need not provide the parameter k i.e. the number of clusters. The cluster formation is faster when compared to k-means because it does not invove any iterative computation of means. In [8], online visual dictionary is built while performing loop closure detection. The region associated with a visual word is defined as the spherical region of radius r around a SIFT descriptor. Visual dictionary which is initally empty is built incrementally as follows. For each SIFT descriptor x extracted from the current image, the dictionary is searched for a visual word centered within a distance of r from x. If x does not match any of the existing words, then a new visual word centered at x is added to the dictionary. If a match is found then no update is performed to the dictionary. The set of SIFT descriptors which match with the exisiting words in the dictionary provide necessary information to detect loop closure.

2.4.1.3 Type3 : Mean-shift (MS) based clustering

Mean-shift algorithm provides a method to form free shape clusters based on the distribution of the training vectors in D. It automatically decides the number of clusters to be formed based on the bandwidth parameter h of the kernel function K involved in the computation. We have to provide the value of h, which is a rough indication of intra-cluster variation. Given a set of training vectors we can fit a probability function on it[30]. Usually these functions are multi-modal in nature which attain peak values in the regions where the distribution of the training vectors is dense. A gradient-ascent algorithm initialized at a training vector will iteratively converge to the associated peak. Mean-shift clustering technique computes this peak associated with each training vector and groups all the training vectors converging to the same peak into a single cluster[22, 23]. For each training vector $x_j \in D$, mean-shift algorithm iteratively searches for peaks of the density function. Starting with the initial estimate $\mu_j^0 = x_j$, the mean vector is updated in each iteration as follows:

$$\mu_j^{l+1} = \frac{\sum_{x_i \in D} K(x_i - \mu_j^l) x_i}{\sum_{x_i \in D} K(x_i - \mu_j^l)} \tag{2.1}$$

As we can see from equation (2.1), for each vector $x_j \in D$, in each iteration l, we need to compute the distance of μ_j^l with all the vectors of D. Hence this algorithm has high computational complexity.

2.4.1.4 Type4 : Visual words from SfM

Instead of clustering the whole set of training features, in this method, initial matches between each pair of training images is established by matching the local descriptors extracted from respective images. In [38] standard SIFT keypoint matching (presented in section 2.1.2) is used to obtain 2D-to-2D matches between images. In [52], mutual nearest neighbor matching is used.

These matches are provided to a Bundle Adjustment process (explained in section 1.5) to obtain 3D point corresponding to these 2D matches. The features in the 2D track of a 3D point form the visual word. In this framework the number of clusters is automatically decided based on the number of 3D points. The clusters can be of free shape and computation time depends upon the way 2D-to-2D matches are established. The matching techniques in [38, 52] are fast when compared to k-means and mean-shift because the distance comparison need to be performed only once.

2.4.2 Recognition

The method used to form the visual word inherently provides a way for classifying a test vector x. For k-means based visual word x can be classified to the class of the nearest cluster center. For range based clusters (Type2), x can be matched based on the threshold on the distance of the nearest training vector of the cluster and so on. But a strategy different from the one that is used for clustering can be used as in [51] where x is matched with mean-shift clusters through ratio of distance with the first and second nearest cluster center. Various supervised methods like metric learning on whole descriptor space[73], learning individual metric for each visual word[84], learning a supervised classifier[24], fuzzy classification[114] have been employed for visual word recognition. The choice of recognition rule influences the accuracy and efficiency of the recognition process. For example, if we chose to represent the visual word D_i using the cluster center (as in k-means), then we need to compare a test descriptor only with one vector (i.e. the cluster center) per visual word. But a single vector per visual word cannot represent clusters with complex boundaries. In our experiments we use threshold on the distance to nearest neighbor, threshold on the ratio of distances to the two nearest neighbors, matching based on mean-shift operation and some of the popular representations based on linear and non-linear statistical learning techniques.

2.5 Conclusion

In this chapter we presented different techniques for establishing feature correspondence to compute pose of the camera in an image. For the reasons mentioned in section 2.2.3, the recognition based methods having an offline learning stage for 3D point detection are well suited for robust pose estimation. For wider applicability it is desirable to use a framework which can automatically build the necessary 3D information of the environment from the training images. As we see in section 2.3.2.2, such a framework should be able to establish 2D matches across training images during the training stage and obtain a robust representation for the 3D points in the environment. These requirements motivate our work presented in the next two chapters in which we experiment and evaluate different visual word formation and recognition techniques to compute and represent 3D points of the target environment for pose estimation.

Chapter 3

Building sparse 3D map from training images

This chapter presents the training stage of our pose estimation framework. We experiment with various ways of establishing 2D correspondences using training vectors (SIFT descriptors extracted from training images). After SfM, some of these SIFT descriptors get associated with 3D points. We assess some aspects of the quality of the matches and the resulting 3D map.

3.1 Overview

We use different clustering schemes and standard SIFT keypoint matching to obtain 2D-to-2D matches across training images. In the next two sections (section 3.2 and 3.3) we describe the way in which we build 2D-to-2D correspondences using the clusters in the set of training vectors. We design a novel way of clustering SIFT descriptors which can overcome some of the drawbacks in range based matching scheme for SIFT descriptors. This is one of our main contributions in this thesis. Section 3.4 motivates and presents this method. We build sparse 3D map from each set of 2D-to-2D matches obtained through different methods by running SfM. In section 3.5 we describe different data sets on which we run these experiments. In section 3.6 we describe framework for experiments presented in this chapter. It mainly deals with the implementation details along with the range of various parameters used to compute each type of clustering scheme. We also mention the different aspects we use to measure the quality of matches provided by the various matching schemes. The actual results of these assessments and their analysis are presented in section 3.7. Finally, in section 3.8, we end this chapter with the conclusions we derive from analysis of the results.

3.2 2D-to-2D correspondences for SfM from clusters

Let $D = \{x_1.....x_N\}$ be the set of SIFT descriptors extracted from the training images $\{I_1, I_2, ..., I_M\}$. There are two different ways in which we can obtain 2D-to-2D matches across training images, (i) we can directly match the SIFT descriptors between each pair of training images (for example, using standard SIFT keypoint matching) or (ii) we can perform clustering on D and use the SIFT descriptors belonging to same cluster to establish matches across images. These matches are based on just photometric similarity on which geometric constraints are applied during SfM. In the first case, we obtain 2D-to-2D matches which can be directly used to perform SfM. In the second case we use the 2D locations of each pair of training vectors in a cluster to establish

a 2D-to-2D correspondence. While doing so, the 2D locations of the training vectors belonging to a cluster is treated as 2D-track of a single 3D point. In the next subsection, we present our pruning step to discard clusters not suitable to represent a single 3D point.

3.3 Pruning clusters for SfM

A clustering technique partitions D into several clusters. Some of the clusters obtained from D (for e.g. clusters containing descriptors corresponding to repeated patterns in the environment) may contain multiple SIFT descriptor vectors from a single training image. They are not suitable to uniquely identify 3D points. In [96], patterns occurring more than 5 times in the same image are a priori discarded, in order to handle burstiness [54] of visual elements. We also apply a similar strategy in a stricter sense. We discard such clusters. Later in section 3.7.1, we show that we get rid of spurious matches due to repeated patterns in the environment by discarding such clusters. After this step, we expect to retain only those clusters having SIFT descriptors of a physical 3D point. The 2D locations corresponding to each descriptor of such a cluster forms the 2D-track of the physical 3D point across respective training images.

Using the clusters retained after pruning, we establish 2D-to-2D matches in the following way. For each training image pair (k, l), we find the set of clusters containing training vectors from both I_k and I_l. Such a cluster is guaranteed to contain exactly one vector in I_k and I_l, since we have already removed, during pruning step, all the clusters containing multiple training vectors in a single image. Each such cluster provides a 2D-to-2D match between I_k and I_l. We then obtain Structure from Motion (SfM) by running Bundler on these 2D matches. Bundler discards the 2D matches which do not satisfy epipolar constraint and rebuilds the 2D-tracks by chaining the retained 2D matches (section 1.5.3.1). This may split some of the original 2D-tracks produced by the clusters into multiple 2D-tracks. Bundler tries to compute 3D points corresponding to the rebuilt 2D-tracks by minimizing the reprojection error. The 2D-tracks with high reprojection error will be discarded by SfM and will not produce any 3D points. At the end of SfM, we obtain camera pose for the training images and a set of 3D points in the environment. Each such 3D point is uniquely associated with the cluster which was used to obtain the corresponding 2D matches. Some clusters may be associated with multiple 3D points due to the splitting of their 2D-tracks. Such clusters cannot be used to describe a unique 3D point. We discard those 3D points and clusters. In our experiments, the number of such 2D-tracks is negligible. At this stage, the set of 3D points and the set of retained clusters are in one-to-one correspondence.

3.4 Adaptive transitive closure (ATC) based clustering

We design a novel clustering technique using range based matching. Range based matching is performed using a fixed threshold on the chosen type of distance measure between two descriptors. Our method evolves through two observations in Euclidean distance based matching of SIFT descriptors.

First, the Euclidean distance threshold is not suitable when distribution of vectors in a cluster is heteroscedastic. For illustrating this we use a 2D diagram in figure 3.1. It shows the distribution of vectors belonging to 3 clusters in three colors. The distance threshold r needed to match x and $x^{'}$, both belonging to red cluster, will match x with some of the vectors belonging to green and blue clusters. In section 3.4.1, we show this phenomena in SIFT vectors and propose

a *transitive closure* based approach to improve descriptor matching in video images.

Second, the threshold cannot be uniform throughout the descriptor space i.e. in some re-

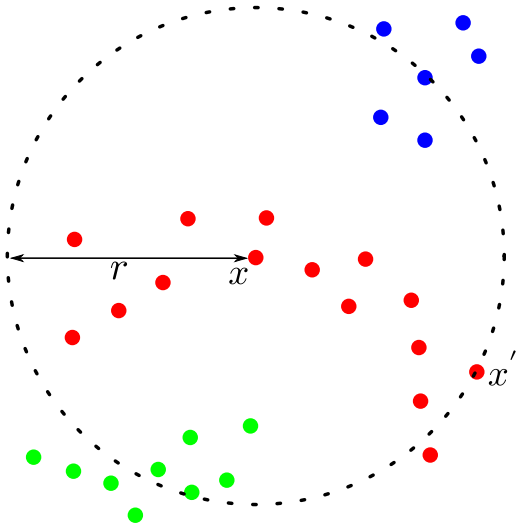


Figure 3.1: Heteroscedastic distribution vectors within clusters

gions we may have to use relatively larger thresholds to establish matches which in other regions may lead to wrong matches. Figure 3.2 illustrates this problem. There are three clusters whose vectors are marked in three different colors. The nearest neighbor distance for some of the vectors in blue cluster is large enough to generate incorrect matches between vectors belonging to green and red clusters. In section 3.4.2 we show this phenomenon on an example pair of images and propose a technique to adaptively choose different thresholds in section 3.4.3. These two proposed modifications outline our ATC clustering method which are published in [15] and [16] respectively. The process of computing ATC clusters is formally presented in procedure 1.

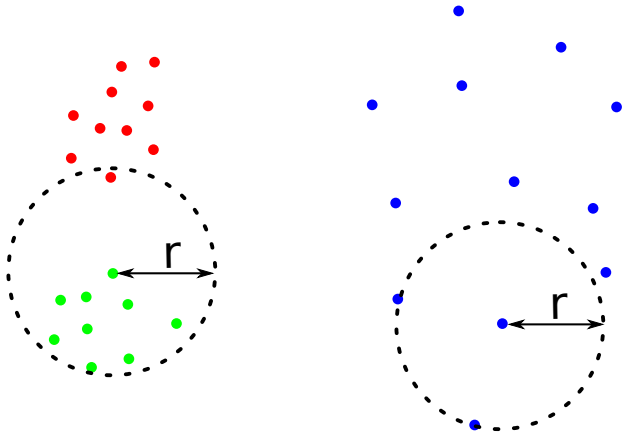


Figure 3.2: Non-uniform intra-class distance

(a)

(b)

(c)

Figure 3.3: Matching with fixed threshold 150 under different pose variation (a)19 matches between 1 and 45 (b)12 matches between 45 and 90 (c)2 matches between 1 and 90

3.4.1 Transitive closure (TC) under smooth variation of pose

To demonstrate the problem of heteroscedasticity, we capture a short video of 90 frames numbered 1 to 90. Figure 3.3 shows the result of matching SIFT features with a threshold value $r = 150$. The figure contains 3 frames numbered 1, 45 and 90 from the video, each of which contain image of an office scene from three different view points. The first row shows image 1 and 45, second row shows image 45 and 90, and finally third row shows image 1 and 90. It can be observed that the pose difference between image pair (1, 90) is more than the pose difference between (1, 45) and (45, 90). In each row, we have marked the location of the SIFT features with '+' sign in the left image which has a matching SIFT feature in the right image, which is also marked similarly. These correspondences are identified by numbers near to the '+' mark. For example, in the first row (figure 3.3(a)), point numbered 7 (located at the top-left portion) in the left image is extracted from the location marked by + near to the number and found to be within a distance $r = 150$ with the SIFT feature extracted from the location numbered 7 in the right image. By visually examining the matches we can see that in the first row all the points out of 19 matches only the point number 9 is matched incorrectly according to the geometry of the scene. In the second row, the point number 2 is an incorrect match out of 12 correspondences. But there are only two matches for (1, 90) in the third row. A closer observation of the first two rows of the figure can reveal that there is at least one common point between (1, 90) which are detected by SIFT algorithm, but the feature descriptors extracted from those points did not match in the third row of the figure 3.3 because the distance between them is more than 150. For example point number 8 in the first and second row belong to the same location of the environment. But they do not match between frame 1 and 90 because the distance between the corresponding SIFT features is more than 150.

To overcome this problem we propose to use transitive closure operation on matches obtained by strict matching radius in a set of images with smoothly varying pose. Let D be the set of SIFT descriptors $\{x_1, x_2 \ldots x_n\}$ extracted from the training images. Let $d(x, y)$ be the Euclidean distance measure between two descriptors x and y. As in range based cluster formation methods, we use a distance threshold r to match SIFT features. Two SIFT features x_1, x_2 are said to be *similar* if $d(x_1, x_2) < r$. This similarity relation is *reflexive* and *symmetric* on D. We perform transitive closure operation on this similarity relation. Each equivalence class in D obtained in this way represents a cluster. Hence our cluster v is represented by a set of feature descriptor $\{x_{v_1}, .., x_{v_n}\}$.

3.4.1.1 Algorithm of transitive closure based clustering

Our algorithm to compute clusters is as follows: Let V be the set of clusters which is initially empty. Eventually it will contain mutually exclusive subsets of D, each one of which represents one cluster. The algorithm will loop over each element of D. In each iteration $i = 1..n$ it will execute following 3 steps :

1. Find the clusters in V which have at least one element x such that $d(x, x_i) < r$.
2. If any such words found in V, then merge all those sets together by union operation to form a single cluster and add vector x_i to it.
3. If no such set is found in V then a new element $\{x_i\}$ is added to V.

In iteration i of the algorithm described above, $i - 1$ vector comparisons are needed in step 1. To complete n iterations it needs $\sum_{i=1}^{n}(i-1) = \frac{n(n-1)}{2}$ vector comparisons. This algorithm,

which is an adaptation of Tarjan's algorithm [88] for connected components in graphs, exactly computes the transitive closure of interest. Our transitive closure based clustering can also be considered as a variation of Single-Link clustering [1] in which nodes are merged using a fixed distance threshold instead of shortest distance between two clusters.

Figure 3.4 shows the matches between image frames 1 and 90 the clusters obtained through

Figure 3.4: 22 points are matched between frame 1 and 90 using transitive closure on a relation established for $r = 75$

transitive closure operation on matches established for $r = 75$. This value for r is much stricter when compared to 150 which is used in figure 3.3. Due to the smoothness of the pose variation in video images, TC based clusters provides 22 matches out of which all the points except points numbered 10 and 13 seem to be correctly matched. It is interesting to note that the matches are spread across the region of the environment which is visible to both the views in image 1 and 90. It turns out that we cannot just relax the threshold value r in order to obtain more 2D-to-2D correspondences while directly matching frames 1 and 90. Figure 3.5 shows the result of matching frames 1 and 90 directly using $r = 250$. If we carefully examine the matches we can see that 7 correspondences (numbered 2, 7, 8, 9, 10, 11 and 12) of the 13 are incorrect.

It is not possible to obtain as many matches as in figure 3.4 even if we use standard SIFT keypoint matching technique (described in section 2.1.2) on image pair (1, 90). The matches with threshold values 0.7, 0.8 and 0.9 on the ratio of the distance of first and second nearest neighbors is shown in figure 3.6(a), (b), (c) respectively. We can see that the number of matches obtained through thresholds 0.7 and 0.8 are less compared to what we obtain through TC clusters. If we relax the threshold to 0.9 (last row of the figure), then we obtain many incorrect matches. The illustrations in this subsection show that under the assumption of smoothly varying pose we can obtain reliable 2D-to-2D matches between images using a strict threshold matching threshold on Euclidean distance between SIFT descriptors. But it is still difficult to determine the right threshold for obtaining TC clusters. As mentioned in the beginning of the section 3.4, a single threshold value cannot be applied on the whole descriptor space. In the next subsection we illustrate this problem and propose a solution.

Figure 3.5: Points matched between frame 1 and 90 for r=250. Total 13 matches. Matches numbered 2, 7, 8, 9, 10, 11 and 12 are incorrect.

3.4.2 Choosing a threshold for TC cluster computation

It is preferable to have all the SIFT descriptors corresponding to a single 3D point in a single cluster. It provides 2D coordinates of a 3D point from more number of views. Wider views provide more accurate coordinate estimation for a 3D point during SfM. Descriptors from more views can also help obtaining better visual representation of the 3D point. As we increase the threshold value r, we obtained more matches due to the relaxed matching condition. This may add incorrect matches. Just a single incorrect match between members of two dissimilar clusters results in a merge.

This problem is illustrated in figure 3.7. It shows portions of two training images and positions of 5 training SIFT features (marked by green '+' and numbered from 1 to 5). By closely observing the local image pattern around these points, we can see that locations 3 and 2 in image (a) match with 4 and 5 in image (b) respectively. Apparently there is no other match among these points. When we compute TC clusters with $r = 100$, it combines 2 and 5 correctly, but 3 and 4 belong to different clusters. If it is increased to $r = 150$, then we obtain both the matches, but the feature vector at location 1 gets wrongly merged with the cluster of the feature vector at location 2. The pruning step (section 3.3) will reject the cluster containing 1 and 2 due to its repetition in the same image. Hence, TC cluster with matching range fixed to one of the above values will definitely loose one of the matches. This can be prevented if we can adaptively compute the clusters for locations 3 and 4 with $r = 150$ and for 2 and 5 with $r = 100$. Essentially, what we do is to compute clusters for both the values of r and perform pruning on clusters with higher range value $r = 150$. If a cluster is rejected, then we look for its sub-words in the set of clusters with $r = 100$ and perform pruning again. This adaptive strategy can retain both the matches shown in the image.

3.4.3 Adaptive transitive closure (ATC)

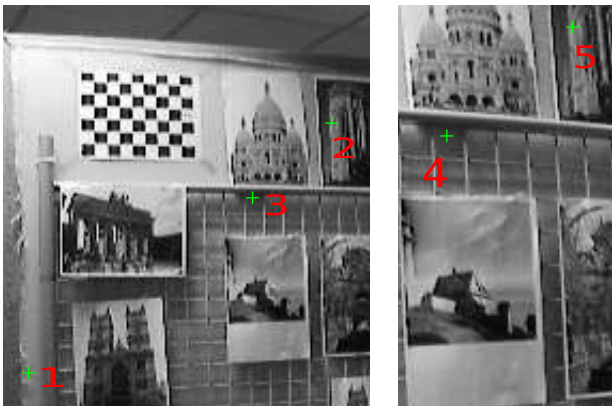
Our strategy relies on the assumption that if a matching range value r is significantly high for a particular kind of visual pattern p in the training set of features, in a sense that it will match p with regions which are dissimilar to p, then the cluster containing p along with wrongly matched

(a)

(b)

(c)

Figure 3.6: Matching frame 1 and 90 with standard SIFT keypoint matching method (a)Threshold of 0.7 provides 7 matches (b)0.8 provides 13 matches (c)A threshold of 0.9 provides 69 matches with many incorrect correspondences



(a) (b)

Figure 3.7: Range selection for matching in TC.

patterns will repeat in at least one of the training images and will be rejected by pruning process. We can detect such clusters and recompute the matches between its members using a more stricter range $r^{'} < r$. TC algorithm will partition this cluster into one or more sub clusters. By applying this strategy recursively we obtain clusters in which the matching threshold is adapted to the characteristics of the visual patterns represented by them.

We implement the above strategy for a fixed set of values of r and design a recursive top down approach for obtaining clusters. We obtain TC clusters for multiple values of $r = \{a_1, a_2, ..., a_k\}$ where $a_1 > a_2 ... > a_k$. This does not add much computational overhead because the range search for the highest value $r = a_1$ retrieves all the required candidates for computing TC clusters with other lesser range values. Let $V_1, V_2, ..., V_k$ be the corresponding set of clusters. Since $a_i > a_j$ for $i < j$, the matches in V_i contains all the matches in V_j. Figure 3.8(a) shows TC clusters for three different radii $a_1 > a_2 > a_3$. For $r = a_3$ four clusters w_1, w_2, w_3, w_4 are formed (plotted in color green, red, blue and black respectively). When the value of r is increased to a_2 it merges w_2, w_3, w_4 into a single TC cluster. When $r = a_1$ all four clusters are merged together. The corresponding hierarchical structure is shown in figure 3.8b. For any $i < j$, that is $a_i > a_j$, we can say that each cluster in V_i is obtained by merging one or more clusters in V_j. Equivalently, each cluster in V_j is a subset of a single cluster in V_i.

The process of obtaining the set of ATC (Adaptive TC) clusters V from hierarchy of set of TC clusters $V_1, V_2, ..., V_k$ is as follows. V is initialized empty. Starting with the set of clusters having highest matching range value, that is V_1, we recursively select candidate clusters which are accepted by the pruning process (presented in section 3.3) and add them to V. If a cluster $v^i \in V_i$ is rejected while pruning due to repeated occurrence in some images, then we look for its sub-clusters in V_{i+1}. Pruning is applied on each sub-cluster. Clusters which successfully pass the pruning step are added to V and will be used for building 2D-tracks. Unsuccessful ones will in turn lead to pruning of their sub-clusters in the next level, till we reach V_k. This process is formally described in procedure 1

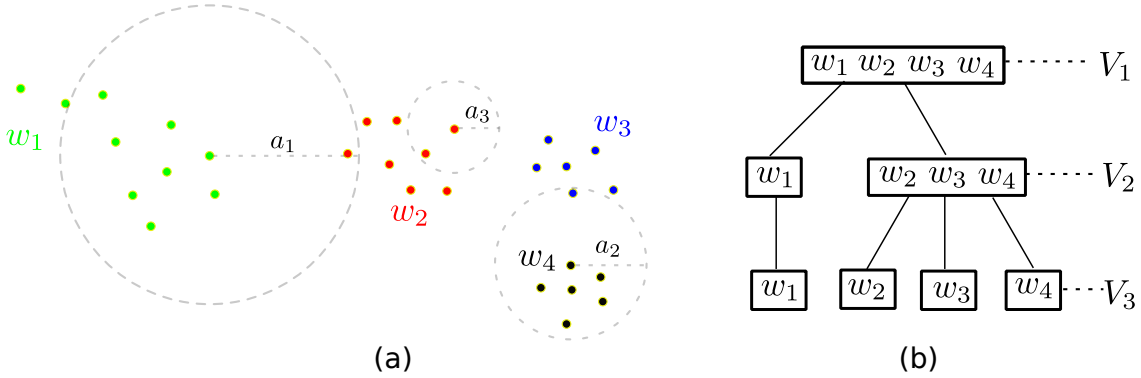


Figure 3.8: Hierarchy of TC clusters

Procedure 1 Computing ATC clusters

Input: TC clusters $V_1, V_2, ..., V_k$
Output: V the set of ATC clusters

$V \leftarrow \varnothing$
Push all the clusters in V_1 into a stack of clusters $ClSt$
while $ClSt$ is not empty **do**
  $v \xleftarrow{\text{pop}} ClSt$ /*Pop out the top element of the stack to v */
  Prune v as described in section 3.3.
  if Pruning accepts v **then**
    $V \leftarrow V \cup \{v\}$ /*Add the cluster to V
  else
    if $v \in V_i$ where $i < k$ **then**
      Push all the sub-clusters of v in V_{i+1}
    end if
  end if
end while

3.5 Data for experiments

Our aim is to test the ability of various cluster formation techniques to provide 2D/3D matches for a test image under different types of pose variation between test and train data. To specifically suit our requirements we capture a single test sequence and multiple train sequences each having a different degree of pose variation with the test sequence. This data set is extensively used in our experiments. We describe this data set in detail in section 3.5.1. In addition we perform few experiments on two of the publicly available data sets which we describe in section 3.5.2.

3.5.1 MAGRIT Pose Gradation Data (MPG)

For our experiments we capture data inside a room using a camera whose intrinsic and distortion parameters are known. Immediately after obtaining the images we undistort them once and for all for our experiments. We have 7 different short video sequences, out of which 6 training sequences (TD1 to TD6) are captured while manually moving the camera and the last one is a *Test* sequence captured by fixing the camera at a particular orientation on a robotic table which is instructed to move in a circular path. The radius of the circular path is 42.5 centimeters. Using the robotic measurements we obtain relative positions of the camera in each test image. A rough sketch of the tracks are shown in figure 3.9. The camera tracks of the sequences are marked with curves in various colors. The track corresponding to each training sequence TDi is tagged with TDi- -N where $i \in \{1, 2, 3, 4, 5, 6\}$ and N is the number of images in the sequence. The arrows indicate camera orientation at subsampled positions on the track. $TD1$ and $TD4$ have maximum pose variation w.r.t the test sequence. $TD5$ encircles the test sequence, but the orientations at closest camera positions are such that there is less than 50% view overlap with the test sequence in each image. There are three planar surfaces (marked as P1, P2 and P3) in the room on which we have pasted pictures which contain texture information. Some of the pictures are repeated in order to introduce ambiguity. The distance between the center of the circular robotic trajectory and the angular meeting point of the planar surfaces P2 and P3 is approximately 275 centimeters.

3.5.1.1 Comparing pose variation of training sets

In this section we illustrate the pose variation between test set and each training set in MPG. For each training set we try to show the extent to which it covers the regions captured by the test images and the difference in the viewing angles at which the training and test images are oriented while capturing a common region of the environment. We also indicate few possible ambiguities while matching features between test and training images due to the repetition of the pictures.

TD1, TD2: Figure 3.10(a) contains four images in two rows. At the top left we have placed one of the training images and the other 3 adjacent images belong to TD1. The yellow shaded portions in the four images show the same region of the environment in the surface P2. TD1 covers only P1 and P2, while test image contains P2 and P3. Hence P2 is the only common region between images in TD1 and most of the images in test set. There is a significant difference in angle and scale at which P2 is viewed by TD1 and test image. The red lines show some of the repeated pictures which may lead to incorrect matches. Figure 3.10(b) shows a test image at left and a training image from TD2 at right. Similar to TD1, the common region between test and the training images is P2, but the difference is the viewing angle is slightly reduced.

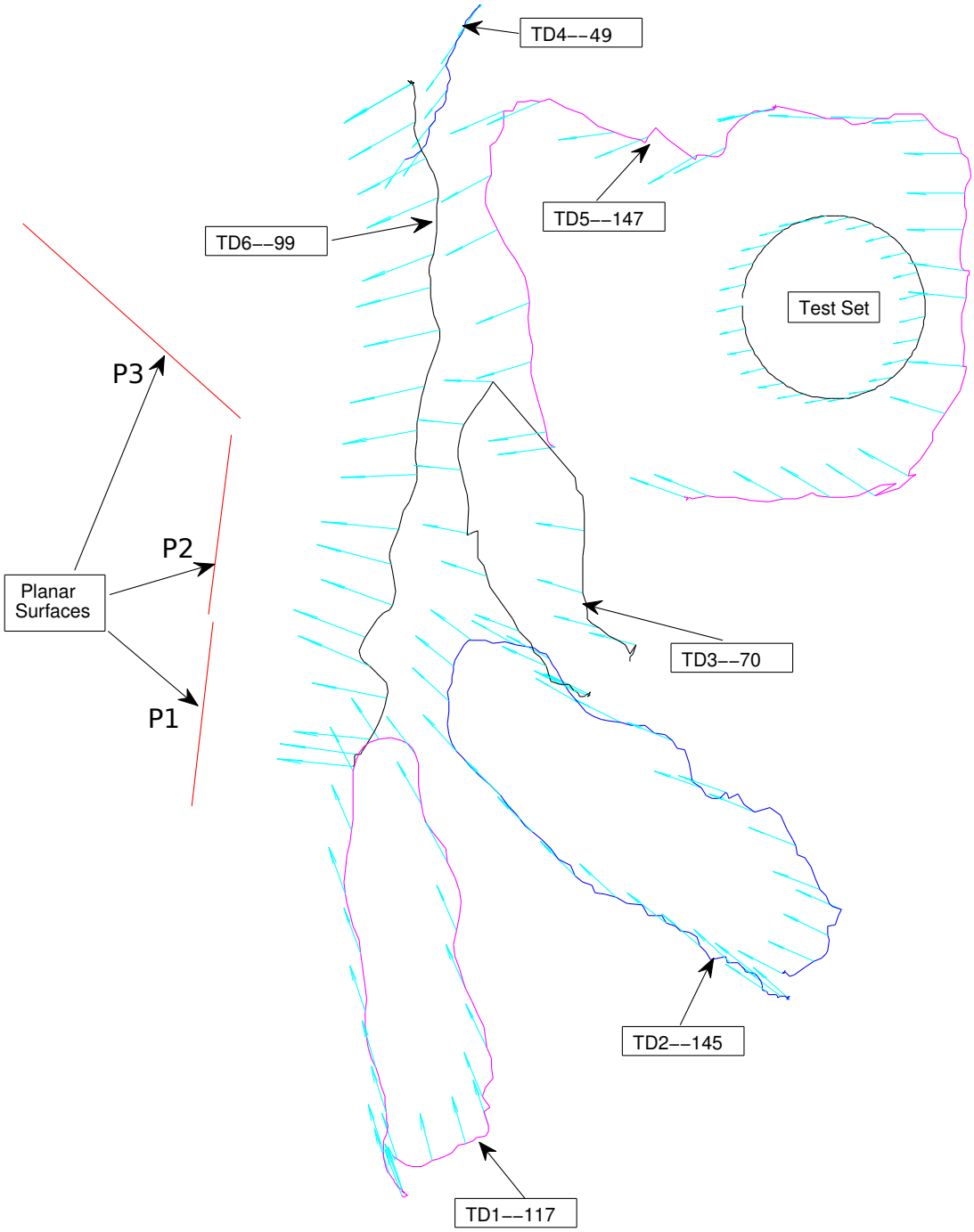


Figure 3.9: Camera tracks in the sequences captured for experiments.

TD3, TD4, TD5: Figure 3.11(a), (b), (c) illustrate the pose variation for TD3, TD4 and TD5 respectively. TD3 contains training images at similar viewing angle as in test set, but there is a significant difference in scale. For TD4 the predominant common visible region is the left portion of P3 with significant scale variation. For TD5, as we can see from figure 3.9, the camera positions in the training set surround those in test set. Figure 3.11.c shows a pair of test and training images in which the camera positions are very close. We can see that there is a significant difference in the viewing angle between the training and test images.
TD6: In TD6 (figure 3.12), training images cover the visible regions of the test image in parts at huge scale variation. The green lines between test and training images connect the matching regions. For Train Image 1 (top right) and Train Image 2 (bottom left), the scale variation can be 4 to 5 times. For plane P3, there are some training images (like Train Image3 at bottom right) in which the scale difference slightly less.

3.5.2 RoboImage and IDOL

RoboImage repository [7] contains 119 images of 60 different scenarios under different lighting conditions. Each set is captured from a fixed set of known camera poses in a plane as shown in figure 3.13. The camera positions are arranged in 3 arches and a line dividing each of them into half. We choose the set corresponding to scenario 2 of the repository. Figure 3.14 shows 6 images of this set corresponding to left most, middle and right most camera positions of the two arcs at the extreme. In this chapter we use these images to assess the accuracy of the 2D-to-2D matches provided by various matching techniques.

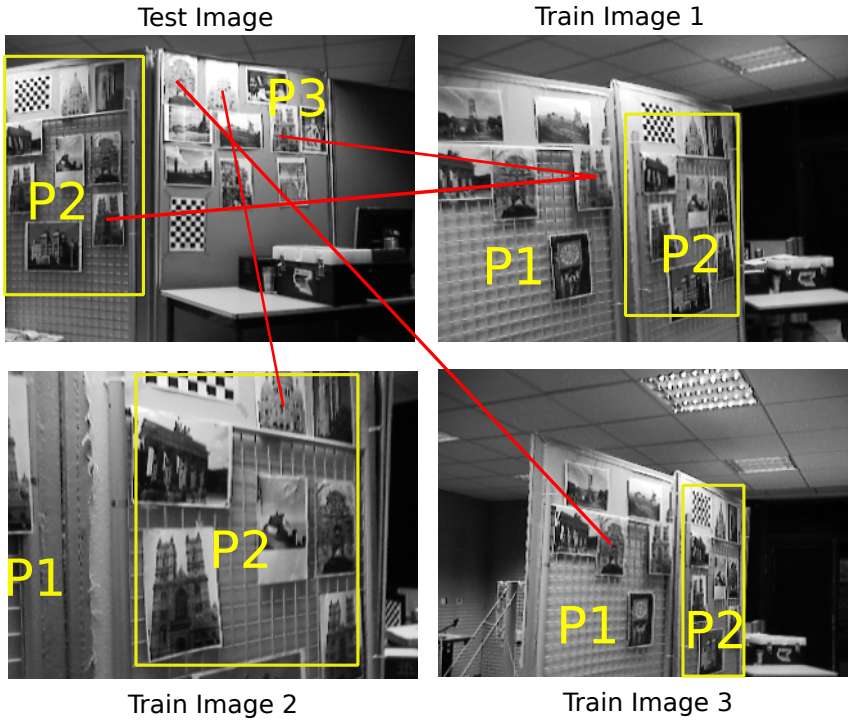
IDOL dataset [71] contains images captured inside a building through a camera mounted on a robot whose position can be measured through sensors. The robot rapidly moves in various rooms and the corridor situated in a single floor of the building while capturing the image. We have chosen 87 images of the kitchen area which contains sufficient texture information for SIFT descriptor matching. The path traced by the camera inside the selected portion is shown in figure 3.19(a). For this set we do not have ground truth information or another set of images which can be used as a test set for pose computation. Hence, we use this set of images only once to assess the quality of SfM on 2D-to-2D matches obtained from various matching techniques.

3.6 Experimental Framework

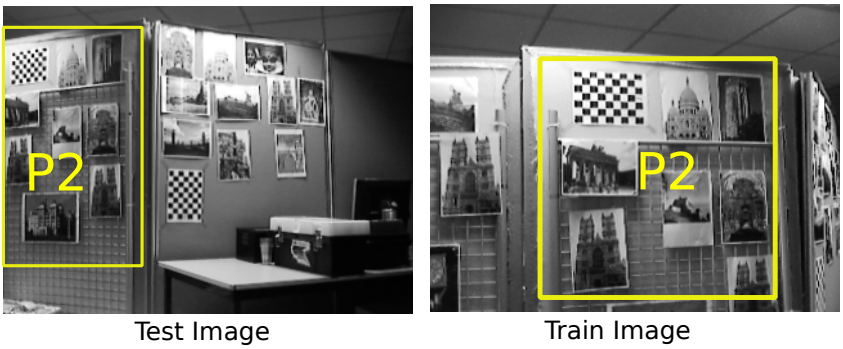
For experiments in this chapter we use 8 different training sets, namely TD1, ..., TD6 and the two selected set of images from RoboImage and IDOL repository described above. From each set of training images we extract SIFT descriptors. We compute different types of clusters (presented in section 2.4.1) from the training vectors. In the next subsection we provide implementation details and range of parameters we use for each type of cluster. Then in section 3.6.2 we present the way we assess the quality of the 2D-to-2D correspondences provided various clusters.

3.6.1 Implementation details

Parameters for Type1 : k-means
The value of k in k-means clustering corresponds to the number of 3D points associated with the 2D locations of the training images at which SIFT descriptors are computed. We choose different values for k between 100 and $\lceil N/4 \rceil$, where N is the number of training vectors extracted from training images. We sample this range of values and run k-means for the values



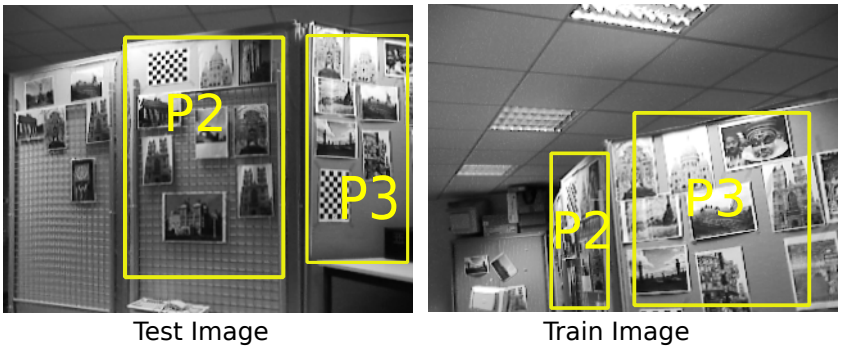
(a)



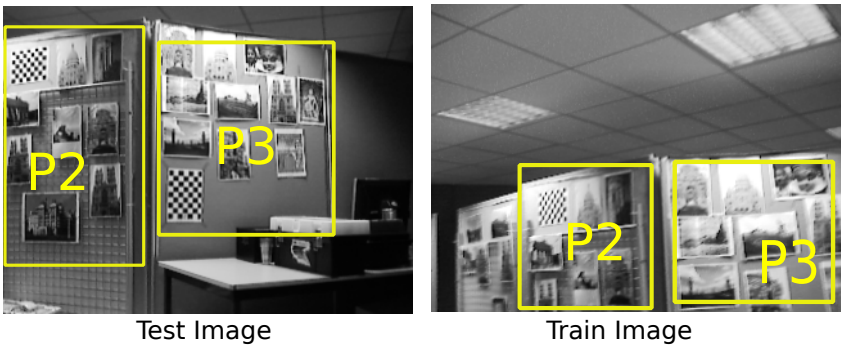
(b)

Figure 3.10: Illustrating pose variation in MPG for TD1 and TD2: The yellow rectangles marked with plane IDs indicate the portion of the environment common to train and test images.
(a)Test vs TD1 : Contains one test image at the left top and three training images. Red lines indicate some of the possible ambiguities in matches due to repeated pictures on different planes P1, P2 and P3.
(b)Test vs TD2 : The difference in viewing angle between test and train images is slightly less when compared to TD1.

(a)



(b)



(c)

Figure 3.11: Illustrating pose variation in MPG for TD3, TD4 and TD5: The yellow rectangles marked with plane IDs indicate the portion of the environment common to train and test images. (a)Test vs TD3 (b)Test vs TD4 (c)Test vs TD5

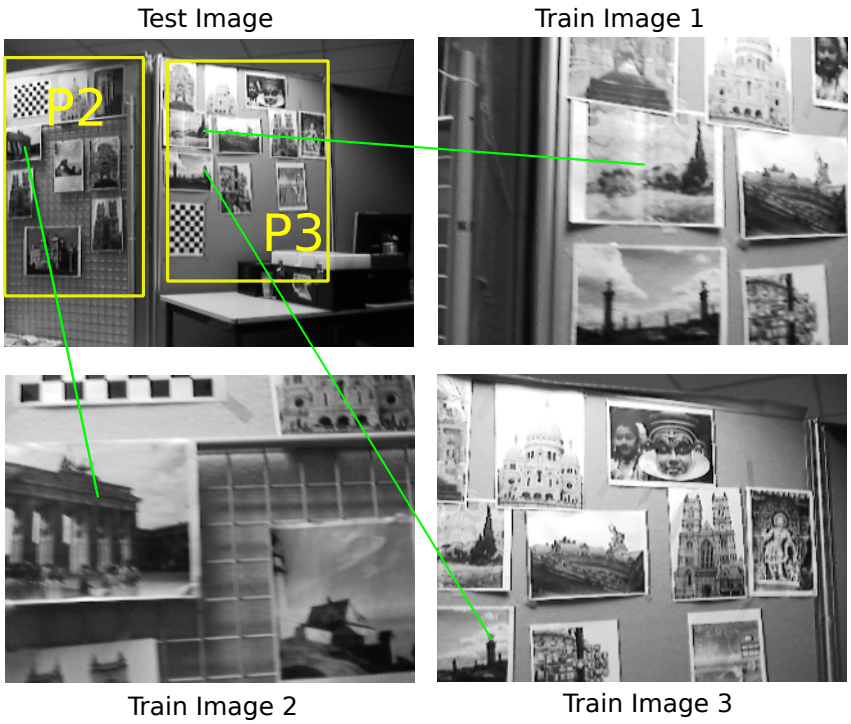


Figure 3.12: Illustrating pose variation in MPG for TD6 : Green lines connect the matching regions of test and training images. We can see that the training images cover the visible regions of the test image in parts at high scale variation. For the planar region P3 scale variation reduces for some of the training images.

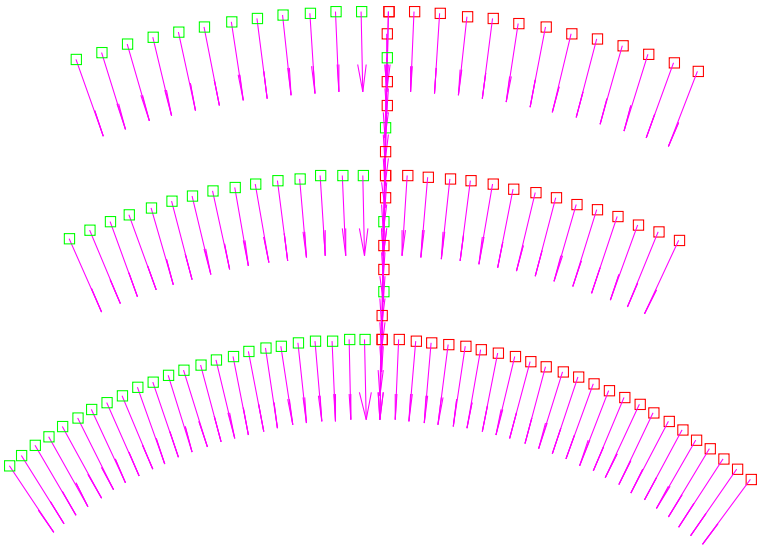
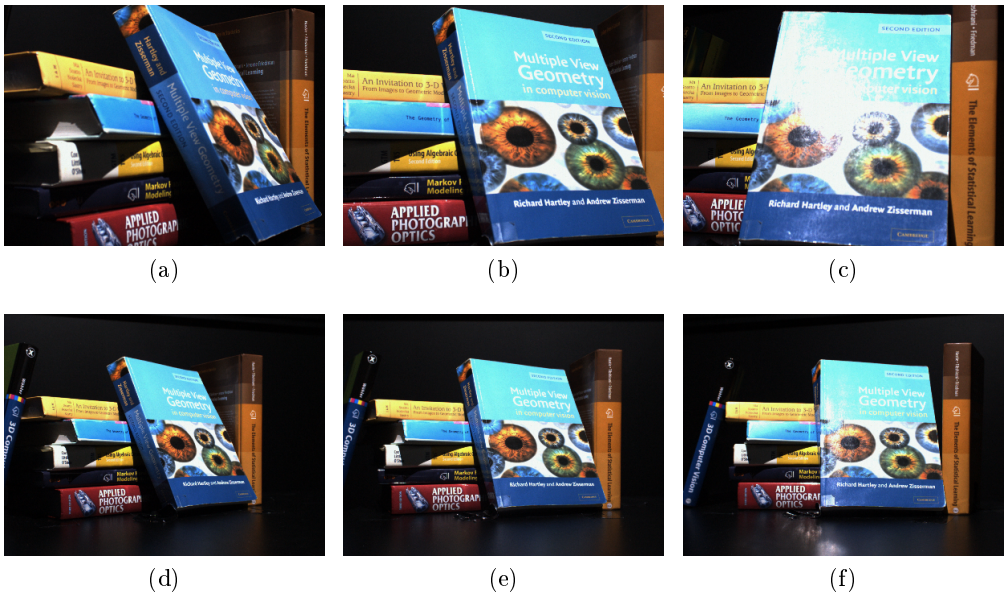


Figure 3.13: Camera positions in RoboImageData



(a) (b) (c)

(d) (e) (f)

Figure 3.14: Sample images from 6 key camera positions of RoboImage data.

$\{100, 200, 300, 500, 1000, 1500, 2000, ..., \lceil N/4 \rceil\}$. We use integer k-means software [115] to perform clustering.

Parameters for Type2 : TC and ATC
The range values we use for TC are $r = \{75, 100, 125, ..., 300\}$. We use the same TC clusters to perform ATC i.e. the values of $\{a_1 > a_2 ... > a_k\}$ for our ATC technique proposed in section 3.4.3 are $\{300, 275, ..., 75\}$ in all experiments.

Parameters for Type3 : gaussian and uniform mean-shift
In mean-shift algorithm (explained in section 2.4.1.3), starting from each training vector new mean is computed iteratively using the weighted combination of neighbors. We stop the mean-shift iteration when the updated vector lies within a distance ϵ_1 from the previous position. After performing mean-shift iterations on all the training vectors, the modes within a distance ϵ_2 are merged and represented by their mean (as in the implementation of [36]). We experiment with two types of kernels: (1)Uniform i.e. $K(x) = 1$ for $\|x\| \leq w$ and $K(x) = 0$ for $\|x\| > w$ and (2)Gaussian i.e. $K(x) = e^{-(\|x\|^2/\sigma^2)}$. Uniform kernel has *compact support* i.e. we have to explicitly provide a bandwidth parameter w, beyond which the kernel has zero value. Gaussian kernel has infinite support, but for practical reasons, we derive w, beyond which the value of the kernel is negligible[23]. For variance σ of the Gaussian kernel, our w is such that, if $\|x\| > w$, then $e^{-(\|x\|^2/\sigma^2)} < \frac{\epsilon_1}{128}$, where 128 is the dimension of SIFT features. We use kd-tree based *exact range search* (i.e. without any approximation for range search) algorithm [79], for speeding up mean-shift iterations. For Gaussian and uniform mean-shift we use $\sigma = \{30, 40, 50, 60, ..., 130\}$ and $w = \{75, 100, ..., 300\}$ respectively. We use values $\epsilon_1 = .1$ and $\epsilon_2 = 1$.

Parameters for Type4 : Standard SIFT keypoint matching
We run standard SIFT keypoint matching with 4 threshold values on the ratio of distances of first and second nearest neighbor. They are $\{0.6, 0.7, 0.8, 0.9\}$. Here also we use fast range search

algorithm [79] without any approximation to match the SIFT descriptors in each pair of training images.

3.6.2 Evaluation of clustering methods for SfM

In this section we outline some of the assessments which measure the quality of the 2D-to-2D matches. Since we do not have the ground truth information of the SIFT descriptor correspondences between the training images, we cannot compute the accuracy of a given set of 2D-to-2D matches. Hence, we evaluate the quality through different aspects listed in the following subsections. The actual results and analysis are presented in section 3.7.

3.6.2.1 Fundamental matrix constraint on 2D-to-2D matches

In RoboImage dataset, we know the ground truth camera positions for all the images. Hence, we can obtain fundamental matrix between any image pair. We use these fundamental matrices assess the quality of 2D matches produced by a matching scheme. On the error function defined in the equation 1.15 we use the same threshold as in Bundler (in which the threshold value is 9) to decide whether a 2D match is acceptable or not. We recall that during SfM, incorrect 2D matches are discarded based on the same error function before performing Bundle Adjustment. In our experiments we assess the total number and percentage of 2D matches produced by a matching scheme on RoboImage data which satisfy this constraint.

3.6.2.2 Descriptor participation and track length in 3D map

The SfM process computes 3D points corresponding to a subset of the 2D matches which satisfy geometric constraints. The rest of the matches are discarded. Hence, the portion of training vectors getting associated with the 3D map indicates the ability of a matching scheme to produce correct 2D matches. The 2D locations of the SIFT descriptors used to generate the 2D matches corresponding to a 3D point forms its 2D-track in the images. The 2D-tracks provide information necessary to compute the coordinates of the corresponding 3D point through triangulation (section 1.5.2). The descriptors associated with the 2D-track of a 3D point is used to obtain its coordinates and visual representation. With a longer 2D-track we are hoping to cover wider view of the corresponding 3D point. Hence, for a 3D point, it is preferable to have longer 2D-track in order to obtain better estimate of its 3D coordinate and visual description. We measure the percentage of descriptors participating to build the 3D map and lengths of 2D-tracks associated with the 3D points to assess the quality of 2D matching scheme.

3.6.2.3 Planarity of planar 3D points

The environment captured by the images in our MPG dataset contains 3 planar surfaces P1, P2 and P3 (figure 3.9). We manually mask the planar regions in all training images. Using these masks we can identify the SIFT features extracted from the 2D locations belonging to these surfaces. We say that a 3D point Q belongs to plane Pi if any one of the SIFT descriptor in its 2D-track is extracted from a 2D location inside the mask of Pi in a training image. All the 3D points associated with features extracted from the image of a plane should lie in a plane. Hence the planarity of those 3D points can be used to measure the quality of the 2D matches. We fit a plane on these 3D points using RANSAC based least square fit. The goodness measure can be the root mean square distance of the 3D points from the plane. But we cannot apply this measure across different 3D models, unless they are represented in the same scale. To

normalize the scale, we manually mark few 2D correspondences in the training images. Since SfM computes the camera positions for the training images, we can compute the 3D positions of these manually marked locations by triangulation. Then we manually measure the ground truth distances between those manually marked points in the environment. Using the ratio of the ground truth distance to the corresponding distance in the 3D model for these points, we can normalize the scale. Using the scale normalized 3D models we can compare the root mean square distance of the 3D points from the best fit plane.

3.6.2.4 Efficiency

The time complexity of a 2D matching scheme is essential to determine its scalability. Even though, we compute 2D matches in an offline training stage, a matching scheme requiring many iterations over the training vectors in D may restrict the size of the environment it can cover. In our experiments we evaluate the time complexity of each matching scheme we use and also compare the actual time taken to execute them on the training data.

3.7 Experimental Results

Eight training sets i.e. TD1 to TD6, RoboImage and IDOL data sets provide 59713, 75707, 41512, 27782, 52811, 50556, 73097 and 27489 number of SIFT descriptors respectively. This amounts to an average of 500, 600 and 300 features per image for MPG, RoboImage and IDOL respectively. This number is significantly low for IDOL when compared to MPG and RoboImage.

Labelling matching schemes:After computing SIFT features from each training set we match them in different ways with parameters mentioned in section 3.6.1. In tables and plots we represent the method used for matching SIFT features and the corresponding parameter as follows. We use labels G, U, T, A, B, K to represent Gaussian mean-shift, Uniform mean-shift, range based TC, ATC, standard SIFT keypoint matching and k-means based clusters respectively. '*L p*' represents matching scheme labelled 'L' with p as its parameter. For eg: *B 0.7* represents standard SIFT keypoint matching with a threshold of .7 on the ratio of distances of first and second nearest neighbors, *K 5500* represents k-means clustering with k=5500, *G 100* represents mean-shift clustering with Gaussian kernel of $\sigma = 100$ and so on. ATC has fixed parameter for all the cases. Hence, the value p corresponding to the label A is left blank.

3.7.1 Pruning

The 2D matches derived through clustering D undergo a pruning process (section 3.3) which discards clusters corresponding to repeated patterns. Figure 3.15 shows the 2D locations of the descriptors belonging to the three largest of T 125 in one of the training images in TD1. We can get rid of many possible ambiguous matches by discarding such clusters. In the next subsection we illustrate the clear advantage of pruning by visually comparing the 2D matches in a training image pair provided by our ATC clustering and the direct image to image matching through standard SIFT keypoint matching.

Figure 3.15: SIFT features belonging to the three largest clusters (marked by '+' in green, blue and red). Clusters are computed by range based transitive closure method on TD1 with T 125.

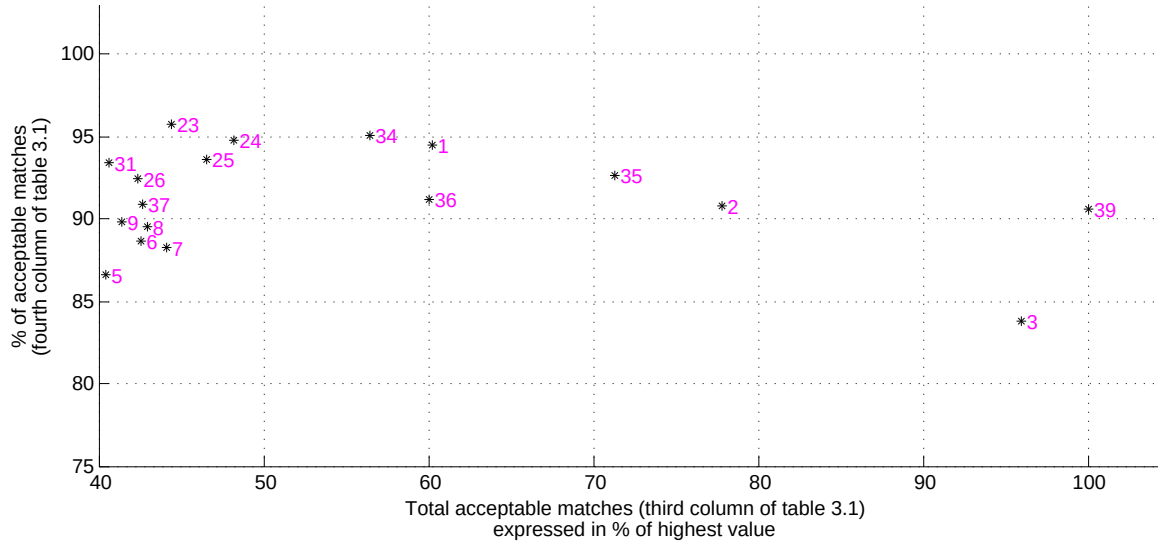
3.7.2 Fundamental matrix constraint on 2D-to-2D matches in RoboImage data

For describing results in this subsection, we refer to image pairs having at least 45° difference in viewing direction in RoboImage dataset as wide pairs. 2D matches between image pairs which comply with the epipolar constraint based on ground truth fundamental matrix are referred as acceptable matches. A wide pair having at least 10 acceptable matches is called acceptable wide pair. The result of assessing the 2D matches using fundamental matrix based constraint on RoboImage data is shown in table 3.1 and figure 3.16. The first column of the table shows the number to identify the matching scheme. This number is used to tag the points of the plots in the figure 3.16. The second column indicates the matching scheme which is labelled as described in the beginning of section 3.7. RoboImage data which consists of 119 images produces 73097 SIFT descriptors. The third column shows the number of total acceptable 2D matches (in thousands) established between these descriptors. We have shown only the rows corresponding to matching schemes which produce at least 300 thousand acceptable 2D matches. The fourth column shows the percentage of acceptable 2D matches out of total number of matches. Fifth and sixth columns assess the 2D matches per image pairs. Fifth column shows the percentage of acceptable wide pairs in total number of wide pairs. Sixth column shows the mean percentage of acceptable 2D matches out of total matches in acceptable wide pairs. The seventh column shows the average number of acceptable 2D matches in an acceptable wide pair. All the measures in the column numbers 3 to 7 in the table 3.1 are such that we expect the quality of the 2D matches with a higher value in these columns to be better. Hence, a plot with any two of those columns in x and y directions will contain the best quality matches at the right top portion. The last column shows the CPU time utilized to perform clustering. This value is used in section 3.7.5.

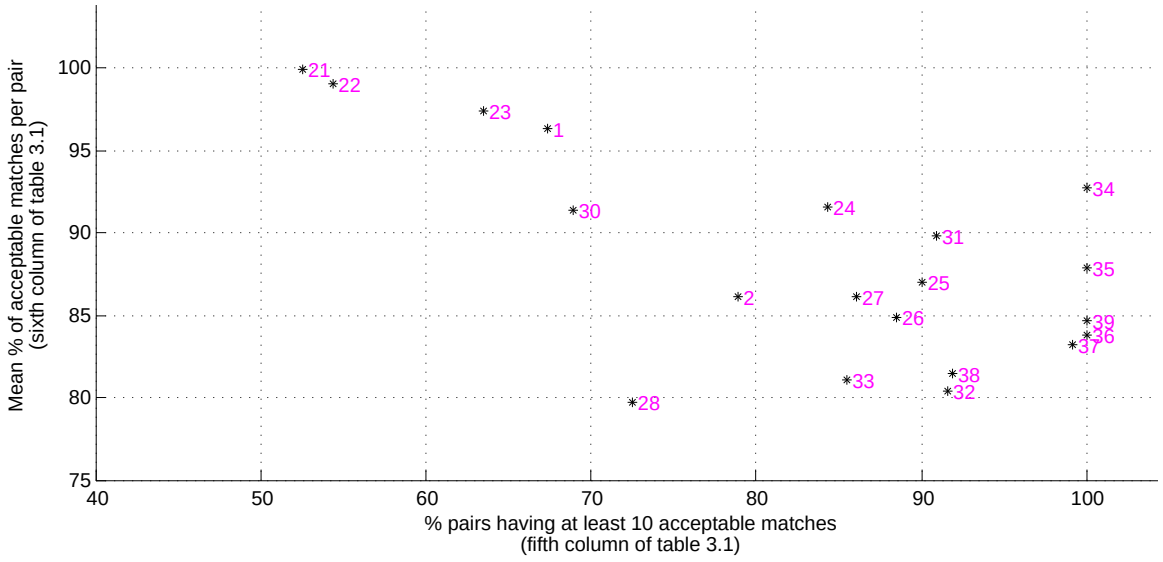
In figure 3.16, the values of the table 3.1 are displayed graphically. Figure 3.16(a) shows the values in third and fourth columnn in x and y axes respectively. Values in the third column are converted to percentage of highest value in that column before plotting. The point numbered 39 (corresponding to ATC) has the highest value in x-axis (hence highest number of 2D matches that comply with epipolar constraint). If we want a better value in y-axis (i.e. the percentage of 2D matches which comply with epipolar constraint) even by 1% from that corresponding to ATC, we have to sacrifice more than 25% (to reach point number 35 corresponding to TC 100) of the value in x-axis. Figure 3.16(b) plots the values in fifth and sixth column in its respective

	Scheme	Total acceptable matches ×1000	% of acceptable matches	% of acceptable wide pairs	Average % of acceptable matches per acceptable wide pair	Average acceptable matches per acceptable wide pair	Time in hours
1	B 0.6	799	94.44	67.37	96.37	22.24	0.15
2	B 0.7	1033	90.77	78.85	86.17	29.11	0.16
3	B 0.8	1274	83.77	99.09	60.20	35.16	0.16
4	K 1500	394	84.22	81.27	62.49	17.93	0.20
5	K 2000	537	86.62	85.20	64.28	24.25	0.19
6	K 2500	565	88.67	77.04	57.46	19.37	0.27
7	K 3000	586	88.28	69.18	59.68	19.79	0.30
8	K 3500	571	89.54	69.49	59.26	19.94	0.34
9	K 4000	550	89.82	57.70	60.69	18.57	0.30
10	K 4500	506	90.44	47.43	59.99	14.97	0.33
11	K 5000	487	92.37	45.32	66.17	15.52	0.34
12	K 5500	463	91.93	41.99	60.40	13.37	0.50
13	K 6000	439	92.54	15.71	67.72	13.33	0.41
14	K 6500	421	92.59	17.52	70.73	11.97	0.43
15	K 7000	389	92.99	8.76	76.91	12.24	0.45
16	K 7500	369	93.49	5.74	79.57	12.95	0.54
17	K 8000	357	94.07	4.23	78.67	12.43	0.63
18	K 8500	337	94.16	0.91	94.97	17.0	0.55
19	K 9000	320	94.49	3.63	83.90	11.75	0.59
20	K 9500	305	94.43	1.81	86.60	12.50	0.72
21	G 50	390	98.17	52.57	99.86	22.86	1.54
22	G 60	500	97.25	54.38	99.0	26.34	3.06
23	G 70	589	95.74	63.44	97.38	28.16	4.81
24	G 80	640	94.72	84.29	91.56	25.23	6.62
25	G 90	618	93.56	90.03	87.0	24.50	8.26
26	G 100	563	92.46	88.52	84.93	24.11	9.88
27	G 110	494	91.22	86.10	86.16	26.54	11.53
28	G 120	384	89.01	72.51	79.76	18.85	15.47
29	U 125	376	96.37	37.46	98.02	14.37	0.38
30	U 150	482	94.71	68.88	91.40	18.38	0.83
31	U 175	540	93.44	90.94	89.82	21.65	1.42
32	U 200	519	90.22	91.54	80.47	21.80	2.35
33	U 225	424	88.88	85.50	81.12	19.98	3.81
34	T 75	750	95.04	100.0	92.75	27.90	0.02
35	T 100	947	92.63	100.0	87.91	36.49	0.03
36	T 125	797	91.22	100.0	83.83	37.08	0.05
37	T 150	567	90.93	99.09	83.23	28.16	0.07
38	T 175	397	89.36	91.84	81.47	20.68	0.10
39	A	1328	90.61	100.0	84.73	67.32	0.34

Table 3.1: Evaluating 2D matches on RoboImage data using fundamental matrix constraint



(a)



(b)

Figure 3.16: Quality of 2D matches on RoboImage data. Both (a) and (b) plot the values in different columns of table 3.1 as indicated in the description of their axes.

axes. We focus on the points at the right top i.e. 24, 25, 31, 34, 35, 36, 37 and 39. 24, 25 and 31 correspond to G 80, G 90 and U 175. 39 belongs to ATC and the rest belong to TC. If we compare the values in the seventh column of the table 3.1 (i.e. average number of 2D matches per wide pair which comply with epipolar constraint) corresponding to these points at right top of figure 3.16(b), we can say that ATC provides nearly twice the number of acceptable 2D matches when compared to the rest. Hence, among the schemes which perform significantly better over others in fifth and sixth columns, ATC has a huge advantage in the seventh column.

We observe similar results when we manually inspect the 2D matches in MPG dataset. Figure 3.17 shows 2D matches obtained for a training image pair in TD1. The first row corresponds to standard SIFT keypoint matching with threshold 0.6 (B 0.6). There are 18 matches in which the 6 matches numbered 1, 2, 3, 4, 8 and 9 are incorrect. The second row shows the matches when we increase the threshold to 0.7. It provides 44 number of matches in which the 13 matches numbered 1, 2, 3, 4, 5, 6, 7, 8, 14, 15, 16, 17, 23 are incorrect. The last row shows the 2D matches provided by ATC clustering method. It has 28 matches in which only 2 matches numbered 2 and 28 appear to be incorrect. Figure 3.18 shows the matches obtained through G 80 and U 175. For G 80 (top row of the figure) there are 15 matches in which the matches numbered 1, 2, 6 are incorrect. For U 175 (bottom row of the figure) there are 18 matches in which the matches numbered 1, 2, 11 are incorrect. These examples illustrate that the ATC clusters provide large number of good 2D matches for SfM.

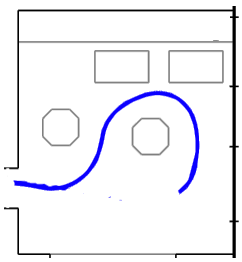
The ability to obtain most of the available 2D-to-2D correspondences with high ratio of inliers is crucial for SfM when the number of available features are less. On IDOL only ATC succeeds to compute all the camera positions through SfM (there are 87 images in the portion of IDOL image set selected for our experiments). Figure 3.19 shows the results. There are 6 plots each showing the track of camera positions. The plot 3.19(a) at the top left shows the ground truth path (in blue color) followed by the robot moving inside the kitchen area. Figure 3.19(b) shows the camera path computed through SfM on ATC. In this case SfM successfully computes all the camera positions and the path looks similar to the ground truth. For other matching schemes, which are labelled as B, K, G, and U, we display the result corresponding to the parameter which succeeds in performing SfM for highest number of camera positions. They are B 0.6, K 3500, G 110 and U 200, for which the camera paths are shown in plots (c) to (f) of figure 3.19. They manage to compute positions of only 71, 56, 30 and 25 cameras respectively. Moreover, the track connecting these positions do not resemble the ground truth.

3.7.3 Descriptor participation, track length on MPG data

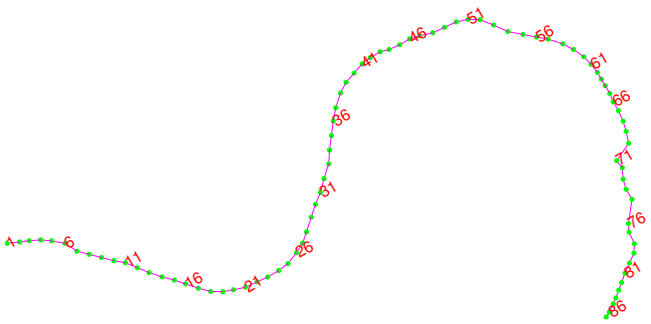
In figure 3.20 we plot the percentage of descriptor participation and average 2D-track length of 3D points. The 2D-track length in y-axis is transformed to percentage of the largest value for plotting. We tag the points with the label of the clustering scheme. Due to the lack of space we do not include the parameter with the label. The subplots (a), (b), (c), (d) of the figure 3.20 show the result on TD1, TD3, TD4, TD6 of MPG respectively. The result on TD2 and TD5 are similar to that of TD1 and TD6 respectively. We can see that ATC lies in the right top portion in each plot indicating high descriptor participation and long 2D-tracks.

Figure 3.17: Comparing 2D-to-2D matches : The first two rows show the matches through standard SIFT keypoint matching with threshold 0.6 and 0.7 respectively. The last row shows the matches produced through ATC.

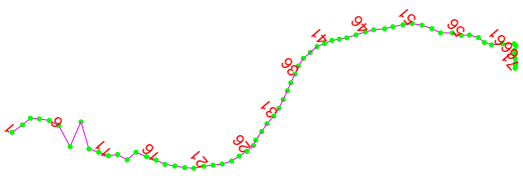
Figure 3.18: Comparing 2D-to-2D matches. Top row : Gaussian mean-shift G 80, Bottom row : Uniform mean shift U 175



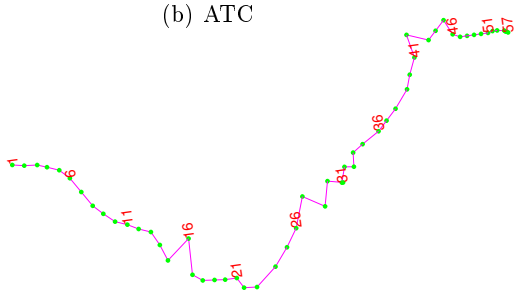
(a) Ground Truth



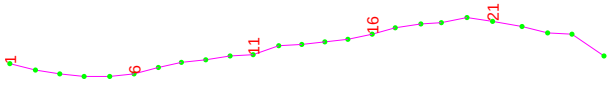
(b) ATC



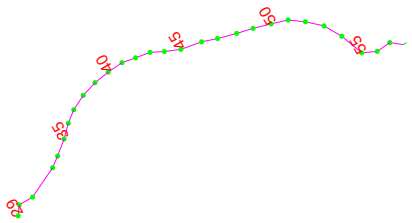
(c) B 0.6



(d) K 3500



(f) U 200



(e) G 110

Figure 3.19: Result of SfM on a selected portion of IDOL video sequence. (a) Ground truth camera path. Each plot from (b) to (f) shows camera path computed through SfM on 2D-to-2D matches obtained through a clustering scheme. (b)ATC, all 87 camera positions are computed and camera path resembles ground truth. (c)Standard SIFT matching with threshold 0.6. Positions of 71 images are computed. (d)K-means with k=3500. Positions of 56 images are computed. (e)Gaussian mean-shift with $\sigma = 110$. Positions of 30 images are computed. (f)Uniform mean-shift with range = 200. Positions of 25 images are computed.

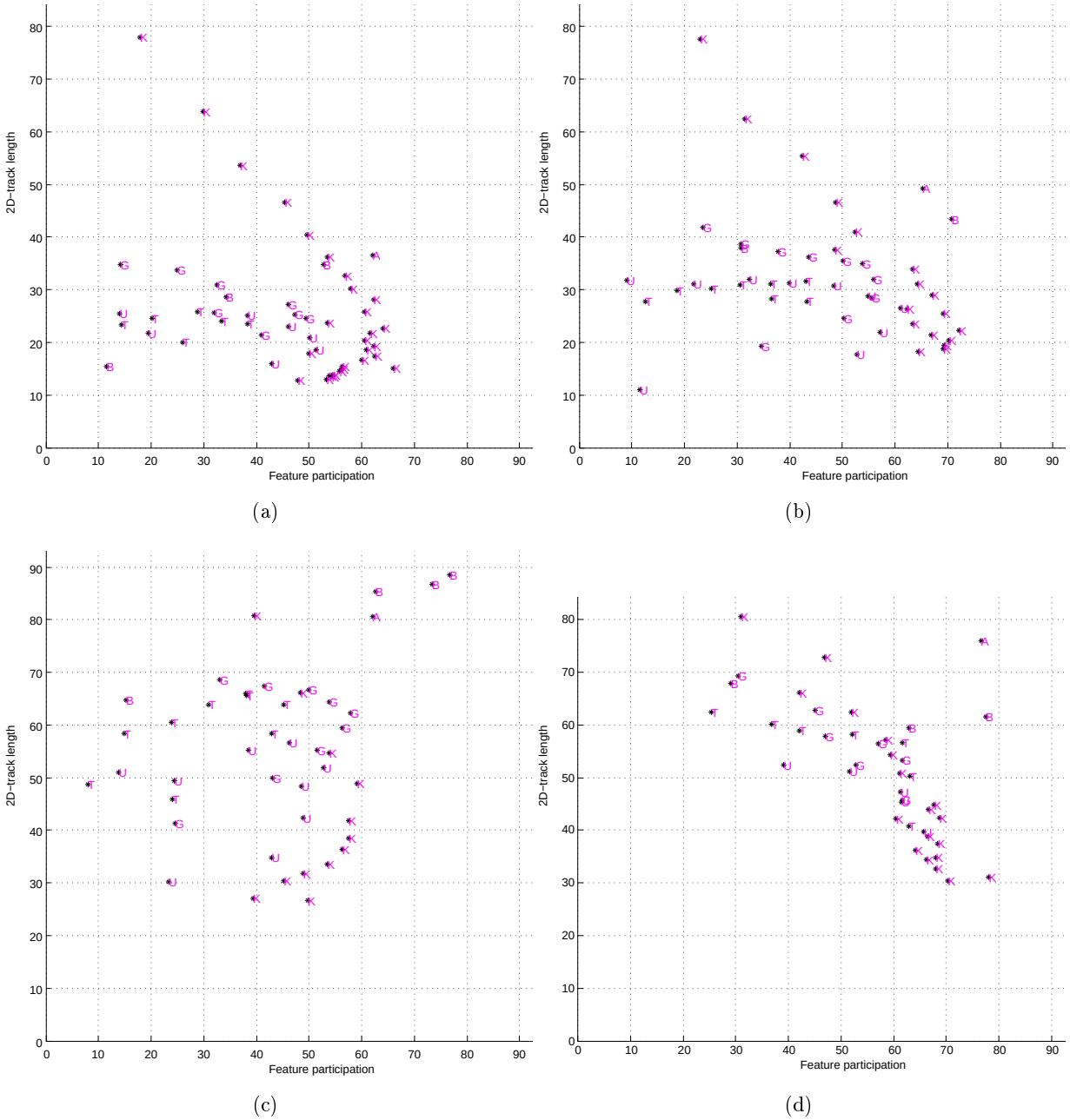


Figure 3.20: Descriptor participation and 2D-track length : (a)TD1 (b)TD3 (c)TD4 (d)TD6

3.7.4 Planarity of planar 3D points

We take multiple length measurements in the environment between the points which are manually marked in the training images to obtain multiple estimates of the scale for each 3D map. We use the average of these scale values to convert the unit of measurement of the 3D maps to centimeters. The result of fitting a plane on planar points is shown in figure 3.21. We treat the planar 3D points within 5 centimeters from the best fit plane as inliers. The result of this assessment for TD1 is shown in table 3.2. For TD1 we have used 6 different length measurements for estimating the scale. In order to fit within a page the table displays only the values corresponding to 3D maps in which the coefficient of variation[2] of the multiple scale estimates (from different length measurements) is less than 10% and the ratio of inliers in the two planes (P1 and P2) is above 85%. Ratio of inliers and root-mean-squared distance (RMSD) of the inlying 3D points in the two planes P1 and P2 are shown in the last four columns respectively. Results indicate that different matching schemes provide 3D map with similar quality. The RMSD of the inlying 3D points is less than 2 centimeters. This value is less than 1% of the distance of the planes from the camera (distance from camera to planes is more than two meters as mentioned in section 3.5.1). In section 3.7.3 we have observed that for ATC, the descriptor participation in the 3D map is high. From table 3.2, we can see that the increased descriptor participation did not decrease the quality of the 3D points of ATC.

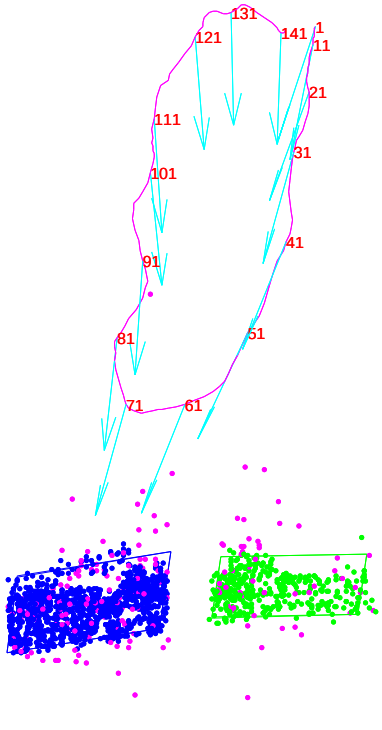


Figure 3.21: Fitting planes on planar 3D points in ATC based SfM on TD2. The blue and green points show inliers in the planes P1 and P2. Points in magenta are outliers.

3.7.5 Computational Efficiency

The last column of the table 3.1 shows the CPU time utilized in hours to compute the clusters for various matching schemes. The values of the column are plotted in figure 3.22. Each bar in the diagram is tagged with the number in the first column of the table. Height of a bar indicates

Scheme	Inlier ratio		RMSD	
	P1	P2	P1	P2
B 0.7	0.91	0.95	1.78	1.50
B 0.8	0.90	0.95	1.89	1.48
K 1500	0.91	0.96	1.79	1.36
K 2000	0.94	0.96	1.69	1.51
K 2500	0.92	0.89	1.81	1.92
K 3000	0.91	0.95	1.74	1.55
K 3500	0.91	0.93	1.76	1.61
K 4000	0.91	0.93	1.75	1.56
K 4500	0.92	0.93	1.71	1.65
K 5000	0.91	0.93	1.80	1.62
K 5500	0.90	0.92	1.81	1.64
K 6000	0.91	0.93	1.85	1.67
K 7000	0.90	0.91	1.94	1.72
K 7500	0.91	0.91	1.82	1.68
K 8000	0.90	0.91	1.89	1.75
K 8500	0.91	0.91	1.81	1.70
K 9000	0.89	0.90	1.92	1.81
K 10000	0.91	0.90	1.95	1.74
K 10500	0.88	0.90	1.93	1.80
K 12000	0.89	0.90	1.98	1.77
K 12500	0.88	0.90	1.99	1.75
K 13000	0.87	0.88	2.00	1.85
K 13500	0.87	0.88	2.05	1.83
K 14500	0.87	0.89	2.02	1.79
G 60	0.90	0.91	2.04	1.69
G 70	0.92	0.92	1.94	1.65
G 80	0.93	0.94	1.87	1.65
G 90	0.91	0.94	1.84	1.62
G 110	0.90	0.97	1.76	1.53
G 120	0.91	0.96	1.78	1.55
G 130	0.92	0.96	1.77	1.53
U 125	0.92	0.91	1.99	1.71
U 150	0.90	0.90	1.99	1.90
U 175	0.91	0.95	1.79	1.56
U 200	0.91	0.95	1.79	1.57
U 225	0.90	0.95	1.75	1.55
U 275	0.88	0.92	1.79	1.71
T 100	0.89	0.90	2.07	1.68
T 125	0.90	0.91	2.01	1.68
T 150	0.90	0.94	1.98	1.75
T 175	0.88	0.92	1.92	1.57
A	0.92	0.93	1.91	1.65

Table 3.2: Planarity of the 3D points in TD1. The first column shows the clustering scheme. Columns 2 and 3 show the ratio of inlying 3D points i.e. ratio of 3D points which are within 5 centimeters from the planes P1 and P2 respectively. The last two columns show the root-mean-squared distance (RMSD) of the inlying 3D points in the two planes P1 and P2 respectively.

the time taken to complete clustering. Results indicate that standard SIFT keypoint matching (numbered 1 to 3) and transitive closure based schemes (numbered 34 to 39) are the fastest followed by k-means based techniques (numbered 4 to 20). Mean-shift (numbered 21 to 33) is the slowest clustering scheme in which some (eg: 24 corresponding to G 80) take nearly 10 times the duration for ATC.

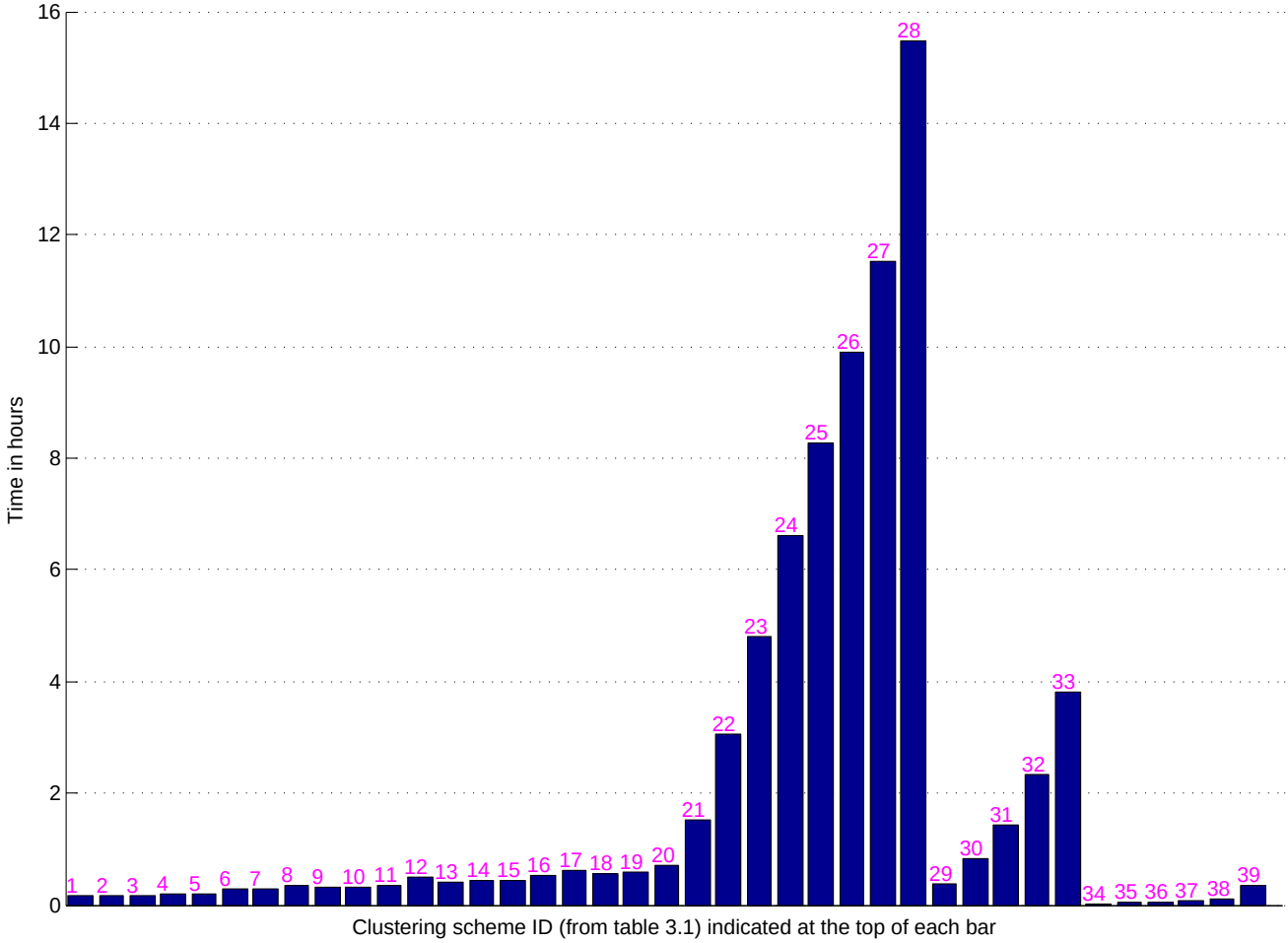


Figure 3.22: Time taken in hours for various clustering schemes presented in table 3.1

3.8 Conclusion

Our experimental results show that ATC based matching is better suited for SfM in many ways. The pruning stage removes many of the repetitive patterns (section 3.7.1) due to which the 2D

matches between images contains few outliers (figure 3.17). Due to the adaptive nature of the matching range, ATC is able to provide large number of acceptable matches for image pairs with wide baseline (table 3.1). This enables us to perform SfM under challenging conditions (figure 3.19) and provides a 3D map with large descriptor participation and long 2D-tracks (figure 3.20). The quality of 3D map did not decrease with the increased feature participation (table 3.2). It is relatively fast, especially, when compared to mean-shift (figure 3.22) clustering. Unlike k-means we need not know the number of clusters apriori to apply ATC. Due to the adaptation using multiple matching thresholds, ATC is less sensitive to the clustering parameter unlike mean-shift and TC. This reduces the task of choosing the right parameter values suitable for a particular situation.

Chapter 4

3D point recognition and pose estimation

In the previous chapter we described different ways of visual word formation through clustering the training vectors. We then associated 3D points with some of the clusters through SfM. In this chapter we present the test stage of the two stage framework i.e. we experiment with different ways to recognize visual words in order to identify the associated 3D points in a given test image. After performing SfM we have a set of N training SIFT descriptors $D = \{x_1, x_2, ..., x_N\}$ and a set of n 3D points $\mathcal{Q} = \{Q_1, Q_2, ..., Q_n\}$ in which each 3D point Q_i is associated with an exclusive set of SIFT descriptors $D_{Q_i} \subset D$. For each vector x_t in the set of test SIFT descriptors D^t extracted from 2D locations of a test image, we use the elements of D to match x_t with the 3D points in $\mathcal{Q}$. The 2D-to-3D matches thus obtained are used to compute pose using RANSAC based PnP algorithms described in chapter 1.

We can define a class label function ω which associates each element in the training set D with the ID of the corresponding 3D point. That is, for a training vector $x \in D$, $\omega(x) = i$ if x is associated with 3D point $Q_i \in \mathcal{Q}$. If x is not associated with any 3D point then $\omega(x) = -1$. Now, visual word recognition can be considered as a supervised pattern classification (SPC) problem [30] in which we have to decide whether a new test SIFT vector x_t belongs to any of the class in the training set. Using SPC framework we experiment with dimensionality reduction and learning discriminant functions for 3D point recognition. Matching with all the training vectors in D is computationally expensive. Hence, in addition to D we experiment with two different training sets derived from D, (i) the set of training vectors associated with 3D points D^{3D} is obtained by union $\cup_{Q \in \mathcal{Q}} D_Q$ (ii) the set of centers of visual words associated with 3D points D^{cc}. The elements of D^{3D} and D^{cc} are also labelled in the same way as D. The elements of D^{3D} and D^{cc} are dependent on the association of training SIFT descriptors with the 3D points. Depending on the clustering scheme used during training stage, this association may differ from each other. Hence, even for the same training set, we may obtain different D^{3D} and D^{cc} for different clustering schemes. Descriptors in D is same for a training set irrespective of the clustering scheme used during training stage.

4.1 Experimental framework

We can assess the outcome of the test stage at two levels, (i)accuracy of 3D point recognition and (ii)accuracy of pose estimation. Accuracy of pose estimation is dependent on the classification

accuracy during 3D point recognition. But, due to the RANSAC step which can tolerate few outliers, we may not find much difference in the estimated pose even if some of the test SIFT descriptors in D^t are misclassified. In addition, the accuracy of the estimated coordinates of the matched 3D points may influence the accuracy of estimated pose. In both cases (i.e. recognition and pose estimation) we need ground truth information in order to evaluate the performance. It is relatively easier to obtain the ground truth camera positions for evaluating the accuracy of pose estimation than to obtain actual value of ω on the test descriptor vectors in D^t. In our experiments we mainly evaluate the accuracy of pose estimation. When we need to evaluate accuracy of 3D point recognition, we obtain the class label for the descriptors in test images by running the training stage in which test vectors are included along with the train vectors while performing clustering and SfM.

Our MPG data set (described in section 3.5.1, camera tracks are shown in this chapter once again in figure 4.2), in which we know the path traced by the camera in the set of test images, is very much suitable for evaluating the quality of pose estimation. In MPG, the images in different training sets have different degree of scene overlap and pose variation with the test images. It is difficult to produce a suitable train-test pair for pose estimation from IDOL[71] and RoboImage[7] data sets (described in section 3.5.2). There are very few portions in the IDOL environment which contain texture information for extracting sufficient point features. As we can see in the SfM results in figure 3.19, within such textured portions only ATC based matching succeeded in producing reasonable reconstruction through SfM. In RoboImage dataset every image covers almost the whole scene (please refer the figure 3.14) and it results in large number of inlying 2D-to-3D matches for all clustering schemes. It is difficult to produce test cases to differentiate various matching schemes based on the accuracy of pose estimation.

4.1.1 Pose estimation on MPG

From each test image in MPG, we extract SIFT descriptors by doubling the size of the image. From figure 4.2 it is clear that in most of the cases the test camera positions are relatively far from the objects in the scene when compared to the train images. By doubling the size we add an additional octave to the DoG pyramid while extracting SIFT descriptors (section 2.1.1) through which we hope to compensate for the scale difference between test and train images. After obtaining the purported 2D-to-3D matches we compute pose through RANSAC based PnP using the threshold of 1 pixel on reprojection error for determining the consensus set. This threshold value is chosen based on the reprojection errors of 3D points on the training images, which were generally less than 1 pixel. We run 500 RANSAC iterations for TD2, TD3, TD5 and TD6. For TD1 and TD4 the overlap with test data is very less. We obtain many incorrect matches from non-overlapping regions due to the repetitive patterns. Hence we use 2500 RANSAC iterations. In our experience, increasing the number of RANSAC iterations did not yield further improvement in the accuracy of the pose. For training sets TD1, TD2 and TD4 the predominant common visible portion of the environment with the test data is planar. Hence we use DLSPnP (section 1.6.2) to compute pose. DLSPnP is slow because the currently available implementation is in MATLAB. For the rest of the training sets i.e. TD3, TD5 and TD6 we use EPnP (section 1.6.1). In [51], in each RANSAC iteration, after randomly choosing the first sample from the set of candidate matches, the subsequent samples are chosen considering only the candidate matches in which the 3D point is co-visible with the 3D point in the first sample in at least one of the training images. In [31], additional strategies like prioritizing the 3D points with longer 2D-track and those which were detected in the temporally close test images

in the past are used while selecting the sample set. We do not employ any such strategy while randomly sampling the candidate matches. While using TD1 and TD4 for training we employ an additional condition for choosing the optimal set of inliers at the end of each iteration. We provide preference to the set containing candidate matches whose 2D locations are widely spread over the test image as will be explained in section 4.2.3.1.

4.1.1.1 Evaluating the accuracy of pose estimation on MPG

We use the ground truth information of the camera positions in the MPG test set to evaluate the pose accuracy as follows. Let $C_{GT} = \{C_1, C_2 ... C_{n_t}\}$ and $C_{Est} = \{C_1^{'}, C_2^{'} ... C_{n_t}^{'}\}$ are ground truth and estimated 3D camera positions respectively where n_t is the number of images in the test data (which is 32 for MPG). Using the closed form solution in [49], we find the optimal similarity transformation (translation, rotation and scale) of points in C_{Est} which minimizes the sum of distances between the respective elements of C_{GT} and C_{Est}. After applying this similarity transformation on C_{Est}, we measure the error as

$$\frac{\sum_i^{n_t} ||C_i - C_i^{'}||}{n_t r_t} \tag{4.1}$$

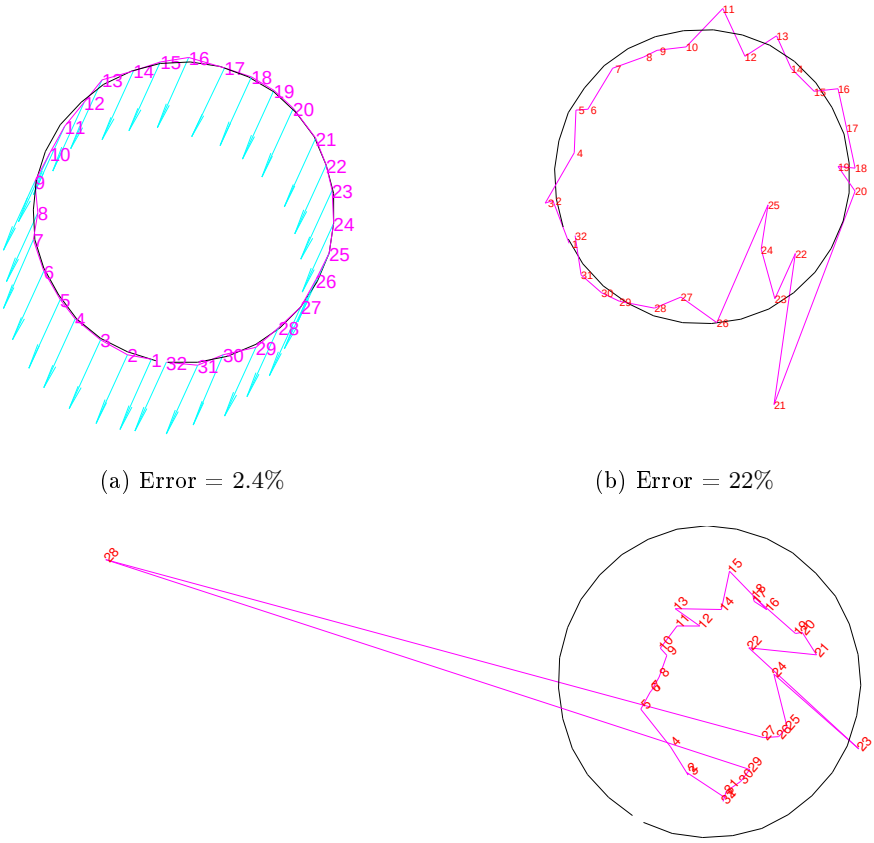
where r_t is the radius of the circular trajectory of the camera positions in the test sequence. Figure 4.1 shows examples of estimated camera poses and the error values with respect to the ground truth.

4.1.2 Successive elimination of reconstruction/recognition strategies

Each 3D point recognition strategy we employ in our experiments should be applied on each clustering scheme used to build 3D map during training stage. The number of different combinations of recognition strategies at test stage and clustering schemes at training stage is very huge. It is difficult to perform recognition and pose computation for all those combinations. Hence, we run our experiments in multiple stages, reducing the combination of parameters at each stage based on the performance in the previous stages. From figure 4.2 we can see that performing pose computation using 3D maps from datasets TD3, TD5 and TD6 is less challenging when compared to that with TD1, TD2 and TD4. Besides, with TD3, TD5 and TD6 we can compute pose quickly using RANSAC based EPnP where as we have to use slow DLSPnP with TD1, TD2 and TD4 due to the predominant planar shape of the 3D map. Hence, we first evaluate pose estimated using 3D point recognition in SIFT descriptor space by applying various matching rules on 3D maps built from datasets TD3, TD5 and TD6. We discard those combinations of clustering schemes and 3D point recognition rules whose pose accuracy is not close enough to the best performing combination. We execute the slow RANSAC based DLSPnP for pose estimation using TD1, TD2 and TD4 for only the retained set of combination of parameters. Using the outcome of these experiments we choose one of the clustering schemes for experimenting with statistical learning methods. We evaluate the performance of 3D point recognition on RoboImage dataset. Based on the recognition performance we choose one of the statistical learning methods to learn 3D point recognition on MPG dataset and evaluate its pose accuracy.

4.1.3 Organization of the chapter

In the first set of experiments presented in section 4.2, we apply recognition techniques on test descriptors in the SIFT descriptor space. In section 4.3 we employ statistical learning techniques



(a) Error = 2.4%

(b) Error = 22%

(c) Error = 68%

Figure 4.1: Illustration of pose error computed through equation 4.1: The black track is obtained by connecting the ground truth camera positions. Track in magenta shows the estimated camera positions for test images numbered 1 to 32. The estimated camera positions are fit to the ground truth using [49]. The error values are indicated below each image. Subfigure (a) and (b) show the configurations corresponding to pose error of 2.4% and 22% respectively. Subfigure (c) shows the case when accuracy of pose estimation is poor and gross errors occur for some test images (image number 28).

for classification. As mentioned above, we successively narrow down different clustering schemes and matching rules in each stage by using the results of the previous stages. Hence the overview of the flow of our experiments are described at the respective stages in this chapter. Finally in section 4.4 we present conclusions we derive from these experiments.

4.2 Recognition in SIFT descriptor space

In the SIFT descriptor space we mainly use two recognition techniques to classify x_t:

1. **NNT** (Nearest Neighbor and Threshold) : Let x_{nn1} be the closest training vector to x_t in the training set. If $||x_t - x_{nn1}||_2$ is less than a threshold and $\omega(x_{nn1}) \neq -1$, then we assign class label $\omega(x_{nn1})$ to x_t. We use threshold values $\{175, 200, 225, 250, 275, 300\}$.

2. **NNR:** (Ratio of distances from two nearest neighbors) : Let x_{nn1} be the closest training vector to x_t in the training set. Let x_{nn2} be the closest training vector such that $\omega(x_{nn2}) \neq \omega(x_{nn1})$. If $\omega(x_{nn1}) \neq -1$ and the ratio $\frac{||x_t - x_{nn1}||_2}{||x_t - x_{nn2}||_2}$ is less than a threshold then we assign class label $\omega(x_{nn1})$ to x_t. We use threshold values $\{.6, .7, .8\}$.

For MS (mean-shift) and ATC based 3D maps we perform an additional recognition technique. For MS, we perform mean-shift iteration (with the same parameters used during training) for each x_t and classify it to the label of the closest training mode if it lies at a distance of ϵ_2 (section 3.6.1) from the mode of x_t. For ATC we match x_t with x_{nn1} in a similar way as in NNT. But, instead of using a fixed threshold we use the adative threshold used while performing ATC during training to obtain the cluster to which x_{nn1} belongs.

We use the above recognition strategies with each of the three training sets namely D, D^{3D} and D^{cc}. Hence, each pose estimation result on the test images of MPG dataset are associated with one set of training images (TD1 to TD6), visual word formation scheme (identified by the label described in section 3.7), training set for recognition (D, D^{3D} or D^{cc}) and one of the recognition methods described in this section. For example, a scheme [TD1 G 100 D^{cc} NNR .8] refers to the scheme in which visual words computed using G 100 on training images in TD1 are used to obtain matches for test images using NNR method with threshold .8 on D^{cc}.

In the next section (4.2.1) we present the result of pose estimation by using TD3, TD5 and TD6 as training images. We evaluate the accuracy of pose estimation for various combinations of parameters in visual word formation and recognition. Using these results we choose a significantly smaller subset of these parameters based on the observation described in section 4.2.2. Then we evaluate pose computation on training images TD1, TD2 and TD4 (for which we have to use slow DSLPnP method for computing pose) using only the reduced set of parameters.

4.2.1 Results on TD3, TD5 and TD6

The result of pose computation using TD3, TD5 and TD6 are shown in tables 4.2, 4.3 and 4.4 respectively. The table entries show the error (in percentage) computed using equation 4.1. Since the radius r_t of the circular trajectory of camera positions in the set of test images of MPG is 42.5 centimeters, 2 to 3 percent of error corresponds to nearly 1 centimeter of shift in estimated test camera positions on average. In tables 4.2, 4.3 and 4.4 columns 3 to 5 show the results corresponding to recognition of test SIFT descriptor vectors through NNT with D, D^{3D} and D^{cc} respectively. The columns 6 to 8 show the corresponding values for NNR. The last column shows

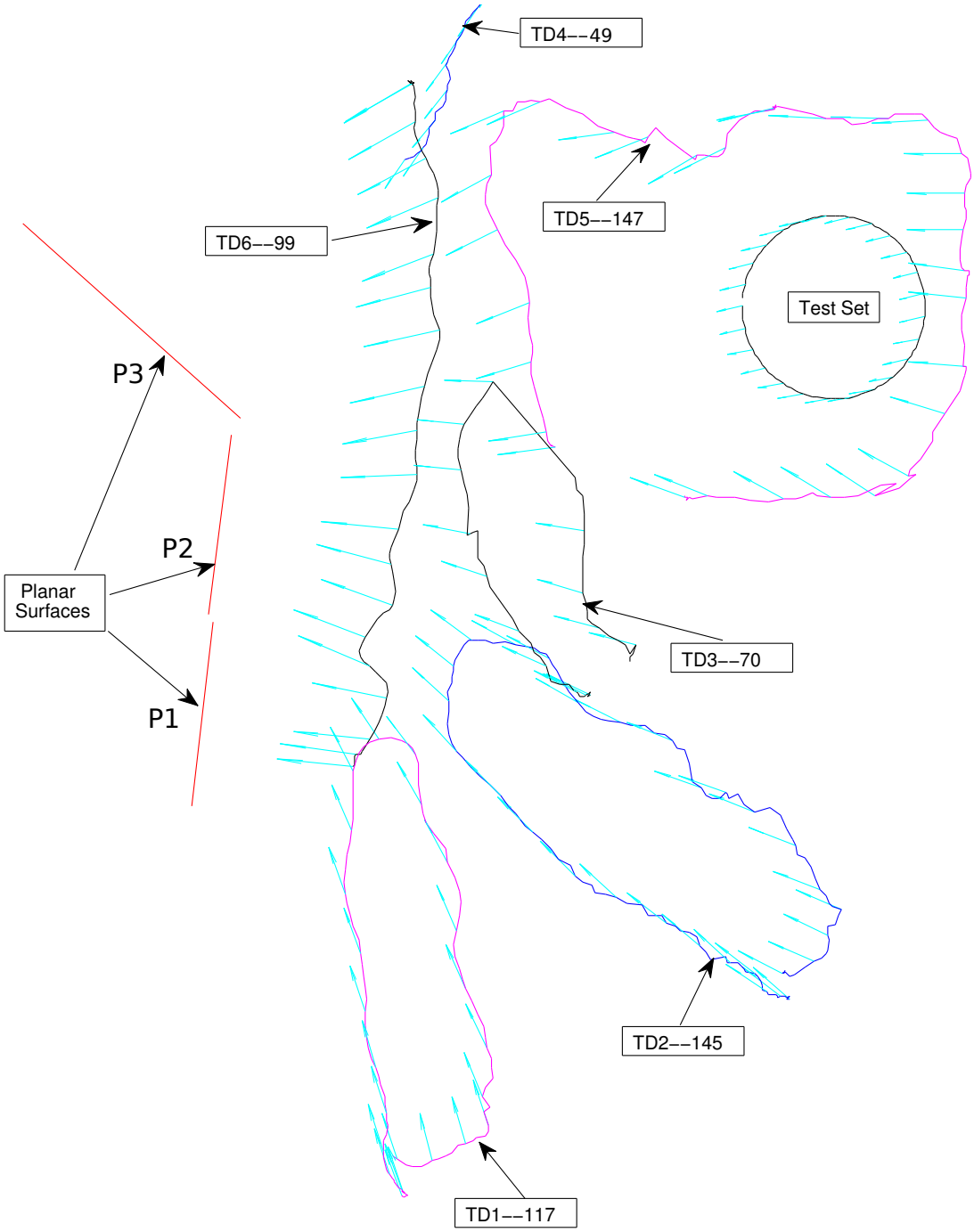


Figure 4.2: Camera tracks in the sequences captured for experiments.

the pose estimation error for the additional recognition technique on MS and ATC (described in the beginning of section 4.2). For rows corresponding to visual word formation schemes other than MS and ATC the last column is left blank. We show only the values corresponding to those visual word formation techniques which manage to provide less than 10% error in at least one of the recognition strategies. In order to fit all the columns of the table in a single page, for NNT and NNR, we show only the values corresponding to the threshold parameter which gives the least error within a recognition method. For example in table 4.2 the cell corresponding to 5th column of row number 17 (with value 6.1/225) corresponds to test vector recognition using the D_{cc} of SfM obtained through G 100 (gaussian mean-shift based visual word with sigma = 100). The recognition strategy corresponding to the cell is NNT (which uses 6 different thresholds $\{175, 200, 225, 250, 275, 300\}$) in which 6.1 is the minimum pose estimation error obtained by the threshold value 225. In the three tables (4.2, 4.3 and 4.4), value in bold letters indicate the best (i.e. least error) in each row. The gray shaded cells indicate the best value in the column. Cell in the red shade indicate the best value in the table. Based on the results in these tables we make following observations.

No advantage in mean-shift or adaptive thresholding : The values in the last column of each table indicate that there is no significant advantage in running mean-shift iteration or using adaptive threshold strategies for recognition. The entries in the last column rarely contain the best of the respective rows and even when they do the difference with the second best is less than 1%.

No scheme is absolutely better : The results indicate that none of the methods are absolutely better over the rest in rows or columns (i.e. in different visual word formation or recognition schemes). For example in table 4.3, the accuracy of NNR on D^{cc} using K 7500 clustering scheme (value 2.7/0.8 in row number 17, column 8) is better than that of T 175 (value 3.9/0.7 in row number 58, column 8). But in table 4.4, T 175 performs (value 3.5/0.8 in row numbered 58, column 8) better than K 7500 (value 8.1/0.8 in row 17, column 8). Still, we can see that ATC based clustering scheme provides either the best or very close to the best performance. We will discuss about this further in section 4.2.2.

Using D^{cc} we can perform matching much faster without significant loss in accuracy: If we compare the values corresponding to the columns using D^{cc} (columns 5 and 8) with rest, we can see that there is no significant loss in accuracy by retaining only the center of clusters corresponding to the visual words. Sometimes (eg: Row number 56 corresponding to T 125 of table 4.2) they provide slightly better accuracy than matching with D and D^{3D}. But performing matching with D^{cc} is much faster when compared to that with D and D^{3D}. Table 4.1 shows the average CPU time taken per test image for matching the test descriptors with the training descriptors in TD5 using fast nearest neighbor search [79]. For all the rows except the last one, the matching is performed with whole training set of descriptors D. Last row shows the time needed for computing two nearest ATC cluster centers in D^{cc}. The first two rows show the time needed for computing first nearest neighbor and 101 nearest neighbors respectively. Subsequently each row indicates the time needed for matching the descriptors in a test image through Gaussian (indicated with label G) and Uniform (indicated with label U) mean-shift operation. For performing NNT we need only the first nearest neighbor for each test descriptor. For D it takes 17.9 seconds as indicated by the first row 'NN 1' of the table. We have observed that computing the first nearest neighbor from D^{3D} also takes similar amount of time (14 seconds for D^{3D} from ATC). For performing NNR based matching we need two nearest neighbors belonging

to different clusters. Hence we cannot use fast nearest neighbor search directly by computing first two nearest neighbors as they may belong to the same cluster. We compute 101 nearest neighbors using fast nearest neighbor search. We found this to be faster than performing linear search to find the second nearest neighbor from a different cluster. For example, it took 95 seconds for computing two nearest neighbors from two different clusters of D^{3D} obtained from ATC on TD5 through linear search. If all of the 101 nearest neighbors obtained through fast nearest neighbor search belong to the same cluster then we label the test vector with the class of the first nearest neighbor. Otherwise we apply the NNR matching threshold using the closest neighbor among the 101 neighbors which belongs to a different cluster than the first nearest neighbor. In our experiments we did not encounter a case in which this alternative strategy gives a different classification. Based on the cost of finding the first nearest neighbor, we can say that performing NNT and NNR on D and D^{3D} takes at least 14 seconds per image. From the table 4.1, we can see that mean-shift is even more costlier in all the cases. The last row indicates that the time taken for computing two nearest neighbors from D^{cc} is just 1.3 seconds per image. Hence, we can say that matching with D^{cc} is at least 10 times faster than the rest of the matching rules.

4.2.2 Eliminating combination of clustering and recognition schemes from available result

We have nearly 50 to 60 different clustering schemes for each training data during training stage. For each clustering scheme we experimented with 27 different recognition rules (6 NNT thresholds and 3 NNR thresholds on D, D^{3D} and D^{cc}). Hence for each training set we perform pose estimation nearly 1500 times. The speed of EPnP (3 to 4 images per second) enables us to finish these experiments in reasonable time. But the currently available implementation of DLSPnP takes nearly a minute per test image to perform pose estimation with 2500 trials per RANSAC. Hence we use the available results on TD3, TD5 and TD6 to reduce the number of methods.

Due to the huge advantage in recognition speed without losing significant accuracy, we use only D^{cc} in our experiments on TD1,TD2 and TD4. We also discard the clustering schemes which did not perform well in on TD3, TD5 and TD6 and for each retained clustering scheme we use an optimal recognition strategy based on the following assessment. We find the best accuracy for each training data (TD3, TD5 and TD6) using D^{cc}. For example in table 4.2, this value is is 4.6 (row number 56, column 8 corresponding to T 125). Then for each recognition scheme using D^{cc} we find the difference of its pose error from the best value. We calculate the maximum of this value i.e. difference with the best across TD3, TD5 and TD6. For example, for the scheme [TD3 ATC D^{cc} NNR .8] (last row column 8 of table 4.2), the difference with the best is $5.2 - 4.6 = .6\%$. In TD5 (value is not displayed in the table 4.3 because threshold .6 outperforms .8) and TD6 this difference is even less. Hence the maximum difference for the recognition scheme [ATC D^{cc} NNR .8] from the best accuracy across the data sets is .6%. Table 4.5 shows the result of this calculation for different schemes. We have showed only those visual word formation schemes which have at least one case with less than 5% error for table 4.5. The values in the bold digits show the best in each row, gray shaded cells correspond to the best in the column and red shade indicates the best of all. We can see that ATC provides the least maximum difference with the best pose accuracy across the three different datasets (TD3, TD5 and TD6).

Matching Rule	CPU time in seconds
NN 1	17.9
NN 101	36.6
G 60	132.2
G 70	217.9
G 80	356.5
G 90	473
G 100	597
G 110	783
G 120	1636
G 130	1820
U 125	18.3
U 150	43.3
U 175	79
U 200	139
U 225	232
U 250	503
A D^{cc} NN 2	1.3

Table 4.1: Duration in seconds of CPU time per test image for matching test descriptors with the 3D map from TD5. Fast nearest neighbor search [79] is used in all the computations. For all the rows except the last one, the matching is performed with whole training set of descriptors D. Last row shows the time needed for computing two nearest ATC cluster centers in D^{cc}. The first two rows corresponding to matching rule 'NN1' and 'NN101' show the time needed for computing first nearest neighbor and 101 nearest neighbors respectively. Subsequently each row indicates the time needed for matching the descriptors in a test image through mean-shift operation. Labels G and U in the first column of a row indicate Gaussian and Uniform mean-shift kernel respectively and the number next to the label indicates the parameter.

	Scheme	NNT			NNR			MS/A
		D	D^{3D}	D^{cc}	D	D^{3D}	D^{cc}	
2	B 0.7	6.1/200	7.8/250	6.8/225	5.6/0.8	**5.5**/0.8	6.0/0.8	-
4	K 1000	8.0/250	8.4/175	**7.8**/200	12.1/0.8	11.1/0.7	11.5/0.7	-
5	K 1500	10.1/225	11.6/200	**9.6**/200	11.9/0.8	10.8/0.7	11.3/0.6	-
6	K 2000	**6.1**/225	6.5/175	6.9/225	7.2/0.8	7.5/0.8	7.6/0.7	-
7	K 2500	**5.3**/225	5.6/175	6.2/175	6.0/0.8	6.3/0.7	5.8/0.7	-
8	K 3000	**6.0**/175	6.6/175	6.4/175	7.2/0.8	6.0/0.7	7.4/0.8	-
9	K 3500	6.3/275	7.9/200	7.6/175	6.1/0.8	6.3/0.8	**5.8**/0.8	-
11	K 4500	5.9/175	6.4/200	**5.9**/175	6.3/0.8	6.4/0.8	6.9/0.8	-
12	K 5000	5.5/175	5.3/175	**5.1**/175	7.4/0.8	6.4/0.7	7.0/0.7	-
15	K 6500	**5.5**/175	6.3/175	6.5/200	6.8/0.7	7.5/0.8	6.5/0.8	-
16	K 7000	7.7/200	9.1/175	9.5/200	**7.6**/0.8	8.5/0.8	8.0/0.8	-
17	K 7500	**6.6**/225	7.2/175	7.9/175	7.7/0.8	7.5/0.7	6.9/0.8	-
18	K 8000	**5.9**/200	6.9/200	7.5/175	6.1/0.8	6.5/0.7	7.7/0.7	-
19	K 8500	**7.5**/175	8.3/175	10.8/175	8.8/0.8	10.3/0.8	8.2/0.8	-
20	K 9000	9.3/175	11.2/175	9.7/200	9.3/0.8	9.5/0.8	**8.6**/0.8	-
21	K 9500	8.1/200	8.8/175	**7.8**/175	9.1/0.8	8.8/0.8	8.6/0.8	-
34	G 40	10.3/250	9.7/200	9.0/175	13.6/0.7	11.3/0.6	**8.9**/0.8	-
35	G 50	6.1/175	6.4/175	**5.5**/175	7.5/0.8	6.3/0.8	6.4/0.8	-
36	G 60	**5.7**/225	5.8/175	6.1/200	6.7/0.8	5.8/0.8	6.8/0.8	6.5
37	G 70	**4.2**/225	5.3/175	6.3/175	4.6/0.8	4.6/0.8	4.8/0.8	5.9
38	G 80	5.0/175	4.7/175	5.2/200	5.0/0.8	**4.6**/0.8	4.6/0.8	5.2
39	G 90	7.6/250	8.2/225	8.0/225	**6.1**/0.8	6.4/0.8	7.2/0.8	9.1
40	G 100	5.0/225	6.5/225	6.1/225	5.9/0.7	6.2/0.6	**5.0**/0.8	6.1
41	G 110	6.4/275	**5.2**/200	6.3/200	5.4/0.7	5.9/0.7	6.8/0.8	7.1
42	G 120	**5.0**/225	5.9/225	5.6/250	5.1/0.8	5.2/0.7	5.1/0.7	8.6
43	G 130	5.8/250	5.6/225	6.5/225	**4.8**/0.7	5.7/0.7	6.3/0.7	8.1
45	U 100	5.9/175	6.9/200	6.6/200	8.9/0.8	**5.7**/0.8	7.2/0.8	9.6
46	U 125	**5.3**/200	6.1/200	7.9/175	6.4/0.8	6.1/0.7	5.9/0.8	7.1
47	U 150	**5.1**/250	5.6/175	6.2/200	6.3/0.8	5.8/0.8	5.9/0.8	6.0
48	U 175	**5.0**/225	5.2/200	5.1/200	5.7/0.8	5.4/0.8	5.3/0.7	6.7
49	U 200	**4.4**/200	4.8/225	5.2/225	5.6/0.8	5.0/0.8	5.1/0.7	5.8
50	U 225	**5.0**/250	6.3/200	6.4/175	6.2/0.8	7.4/0.8	7.0/0.7	5.7
51	U 250	**5.8**/225	7.6/250	6.8/250	6.7/0.7	5.9/0.7	6.1/0.7	7.2
52	U 275	**6.3**/275	7.4/250	6.8/225	6.3/0.8	6.8/0.8	6.9/0.7	7.5
55	T 100	5.9/175	6.3/200	6.8/175	6.0/0.8	**5.1**/0.7	5.4/0.8	-
56	T 125	4.9/200	5.5/200	7.0/200	4.8/0.8	5.5/0.6	**4.6**/0.7	-
57	T 150	4.5/225	6.1/200	6.1/200	**4.5**/0.8	5.2/0.8	5.0/0.8	-
58	T 175	4.5/250	4.9/200	4.8/225	4.2/0.8	3.2/0.7	5.4/0.7	-
59	T 200	5.0/275	5.0/225	5.6/225	**4.6**/0.8	5.2/0.7	5.4/0.8	-
60	T 225	**5.3**/250	6.5/225	6.6/250	6.3/0.8	6.7/0.8	7.7/0.7	-
61	T 250	5.5/225	6.5/250	5.6/250	6.5/0.8	**5.1**/0.6	7.1/0.7	-
65	A	4.8/225	6.6/175	7.0/175	**4.3**/0.8	5.1/0.8	5.2/0.8	5.0

Table 4.2: Pose Estimation by using TD3 for training. In the second column of each row 'L p' represents clustering scheme during training stage labelled as 'L' with p as its parameter. Label 'L' can be G, U, T, A, B, K representing Gaussian mean-shift, Uniform mean-shift, range based TC, ATC, standard SIFT keypoint matching and k-means based clusters respectively. Each column heading indicates the matching rule used during the test stage. NNT indicates thresholded nearest neighbor, NNR indicates ratio of distances from two nearest neighbors, MS indicates mean-shift operation with the same parameter used during training and A indicates adaptive thresholding. Each entry in the cell is in the form x/y where x is the pose error in percentage computed using equation 4.1 and y is the parameter of for the matching rule indicated by column heading. The values in bold faced letter and gray shaded cells indicate the least error in the corresponding row and column respectively. The red shaded cell contains the best result in the table.

	Scheme	NNT			NNR			MS/A
		D	D^{3D}	D^{cc}	D	D^{3D}	D^{cc}	
1	B 0.6	**2.3**/200	2.9/200	2.8/225	2.4/0.8	2.4/0.8	2.6/0.7	-
2	B 0.7	**3.1**/200	3.5/200	3.3/200	3.3/0.8	3.2/0.6	3.1/0.7	-
3	B 0.8	4.6/250	5.9/175	4.7/175	5.8/0.8	4.2/0.7	**3.8**/0.6	-
5	K 1500	5.6/175	5.5/200	5.7/200	**5.3**/0.8	6.0/0.6	5.3/0.8	-
6	K 2000	6.8/225	6.8/175	7.3/175	6.3/0.6	7.5/0.6	**5.9**/0.6	-
7	K 2500	4.1/200	5.5/200	5.0/200	**3.5**/0.7	4.2/0.7	3.7/0.6	-
8	K 3000	3.5/250	4.4/175	4.1/200	**3.4**/0.7	4.0/0.6	3.7/0.7	-
9	K 3500	**3.2**/250	4.2/175	3.6/200	4.1/0.8	4.0/0.7	3.2/0.8	-
10	K 4000	**2.5**/225	3.1/175	2.7/175	2.9/0.8	2.9/0.8	2.7/0.8	-
11	K 4500	7.9/225	**6.7**/175	7.0/175	8.7/0.8	9.1/0.7	8.1/0.8	-
12	K 5000	3.6/200	4.3/225	4.1/200	**3.1**/0.8	3.6/0.6	3.3/0.7	-
13	K 5500	**3.7**/175	5.6/200	4.5/175	5.0/0.7	4.5/0.7	4.6/0.7	-
14	K 6000	4.8/175	4.9/175	5.5/200	**3.7**/0.7	3.9/0.6	4.4/0.8	-
15	K 6500	2.8/200	3.0/200	3.2/200	3.0/0.7	2.7/0.8	**2.4**/0.7	-
16	K 7000	3.8/225	4.0/175	3.9/200	3.8/0.8	**3.7**/0.7	3.8/0.7	-
17	K 7500	3.2/200	3.6/225	3.2/225	4.1/0.8	3.4/0.8	**2.7**/0.8	-
18	K 8000	4.2/225	4.3/175	**4.0**/225	4.3/0.7	4.1/0.7	4.3/0.8	-
19	K 8500	**3.3**/225	3.6/200	4.9/225	3.4/0.8	3.5/0.8	3.6/0.7	-
20	K 9000	4.2/225	5.8/225	4.9/250	**4.1**/0.8	4.8/0.8	5.3/0.7	-
23	K 10500	**3.0**/250	4.0/200	3.6/175	3.2/0.8	4.1/0.8	3.2/0.8	-
25	K 11500	2.3/225	2.7/200	2.4/225	2.5/0.8	**2.1**/0.8	2.8/0.8	-
26	K 12000	**3.2**/275	3.7/250	4.3/200	3.9/0.8	3.8/0.8	4.0/0.8	-
27	K 12500	4.2/200	6.0/175	5.4/225	**3.9**/0.8	4.0/0.7	4.1/0.7	-
28	K 13000	3.4/225	3.6/200	4.1/225	3.5/0.7	**3.2**/0.8	3.4/0.8	-
34	G 40	7.0/275	4.7/175	**4.5**/175	18.4/0.8	5.8/0.7	5.6/0.8	-
36	G 60	3.0/275	3.2/175	3.0/250	3.3/0.8	**2.7**/0.7	3.2/0.7	3.2
37	G 70	2.4/175	2.7/175	2.6/175	2.7/0.8	2.5/0.7	3.0/0.7	**2.3**
38	G 80	2.2/225	2.5/175	2.4/175	2.3/0.7	**2.1**/0.7	2.4/0.8	2.7
39	G 90	2.5/250	2.6/200	2.7/225	2.4/0.7	2.5/0.8	**2.4**/0.8	2.7
40	G 100	2.2/250	2.5/200	2.1/175	**2.0**/0.8	2.3/0.7	2.2/0.8	2.5
41	G 110	2.3/275	2.4/200	2.3/225	2.5/0.8	2.4/0.7	**2.1**/0.8	2.3
42	G 120	**2.1**/200	2.3/175	2.1/225	2.5/0.8	2.2/0.7	2.1/0.7	2.5
43	G 130	3.5/300	3.1/175	3.4/275	**2.9**/0.6	3.4/0.7	3.1/0.6	3.5
47	U 150	2.3/250	2.5/175	2.3/175	2.3/0.8	**2.1**/0.7	2.3/0.7	3.0
48	U 175	**1.9**/225	2.5/225	2.3/225	1.9/0.8	2.2/0.8	2.4/0.7	2.5
49	U 200	**2.1**/225	2.3/200	2.4/200	2.8/0.7	2.6/0.8	2.4/0.7	2.4
50	U 225	**1.9**/225	2.3/225	2.0/175	2.6/0.7	2.2/0.7	2.3/0.7	2.3
51	U 250	2.3/175	**2.3**/200	2.6/250	2.6/0.6	2.3/0.8	2.8/0.7	2.6
54	T 75	5.7/200	5.1/175	4.8/175	7.5/0.8	**4.7**/0.8	5.5/0.7	-
55	T 100	**1.9**/200	2.3/200	2.4/225	2.4/0.8	2.2/0.8	2.2/0.7	-
56	T 125	2.0/200	2.0/175	2.2/225	2.0/0.7	**1.9**/0.8	2.1/0.7	-
57	T 150	1.9/275	2.2/175	2.1/175	2.1/0.8	1.8/0.8	2.0/0.6	-
58	T 175	4.3/250	4.4/175	4.3/225	4.3/0.7	4.4/0.8	**3.9**/0.7	-
59	T 200	**2.4**/225	2.7/200	3.0/225	2.8/0.6	2.7/0.7	2.9/0.6	-
60	T 225	2.8/250	2.9/225	3.1/250	**2.7**/0.8	2.9/0.8	3.3/0.7	-
65	A	2.4/225	2.5/200	2.5/175	2.4/0.7	2.4/0.8	**2.3**/0.6	2.5

Table 4.3: Pose Estimation by using TD5 for training

	Scheme	NNT			NNR			MS/A
		D	D^{3D}	D^{cc}	D	D^{3D}	D^{cc}	
2	B 0.7	4.6/250	5.8/175	4.9/175	3.7/0.8	**3.3**/0.8	3.4/0.8	-
7	K 2500	5.9/200	**4.9**/175	8.8/175	5.4/0.8	6.3/0.7	6.2/0.7	-
8	K 3000	11.0/200	11.4/175	13.7/200	10.2/0.8	**9.7**/0.7	10.7/0.7	-
10	K 4000	5.5/200	7.6/175	8.1/200	5.8/0.8	**4.8**/0.7	4.9/0.8	-
11	K 4500	4.6/175	4.9/175	6.6/175	**4.1**/0.8	4.7/0.8	5.3/0.8	-
12	K 5000	5.3/250	6.3/175	6.0/200	5.8/0.7	5.4/0.8	**5.3**/0.8	-
13	K 5500	**4.9**/225	5.5/175	5.6/200	5.5/0.8	5.2/0.8	5.4/0.8	-
14	K 6000	5.0/200	6.0/175	6.5/175	4.4/0.8	**4.4**/0.7	6.2/0.8	-
15	K 6500	**4.3**/225	5.9/175	6.8/175	4.5/0.8	5.4/0.7	6.0/0.8	-
17	K 7500	8.8/275	9.5/200	10.0/225	8.4/0.8	9.2/0.8	**8.1**/0.8	-
18	K 8000	4.5/200	5.2/200	4.8/175	**4.1**/0.8	4.3/0.8	5.8/0.7	-
19	K 8500	6.0/250	5.9/200	6.0/175	**4.7**/0.8	5.1/0.8	6.2/0.8	-
20	K 9000	9.1/200	16.8/175	18.8/200	8.6/0.8	8.3/0.7	**8.3**/0.8	-
21	K 9500	4.7/200	5.8/200	5.5/175	**4.0**/0.8	4.5/0.8	4.8/0.8	-
22	K 10000	4.4/200	5.0/175	5.4/175	4.7/0.8	**3.7**/0.8	4.7/0.8	-
23	K 10500	**4.2**/200	5.7/175	5.5/175	5.7/0.8	4.9/0.8	5.5/0.8	-
24	K 11000	**4.1**/200	5.4/175	6.7/200	5.5/0.8	6.2/0.7	5.1/0.8	-
26	K 12000	**3.6**/175	4.6/175	5.0/175	4.3/0.8	4.5/0.8	5.1/0.8	-
27	K 12500	4.1/225	5.3/200	4.5/175	**4.0**/0.8	4.5/0.8	5.0/0.8	-
28	K 13000	**7.6**/200	8.7/250	8.2/250	7.9/0.8	9.0/0.8	9.3/0.8	-
35	G 50	**6.2**/200	10.7/175	10.3/175	8.4/0.8	9.1/0.7	8.7/0.8	-
36	G 60	**3.7**/200	4.6/175	5.1/175	4.1/0.8	4.3/0.7	4.5/0.8	3.9
37	G 70	3.4/200	4.3/200	4.1/175	3.8/0.8	**2.8**/0.8	3.8/0.8	3.8
39	G 90	4.3/175	4.0/175	4.7/175	4.5/0.8	**3.9**/0.8	5.0/0.8	6.1
40	G 100	4.4/225	4.8/200	5.6/200	4.0/0.7	**3.8**/0.8	5.2/0.8	6.1
41	G 110	4.2/175	4.2/175	4.3/200	4.6/0.8	**3.9**/0.8	4.7/0.7	6.8
42	G 120	**3.9**/225	4.4/225	4.9/250	4.5/0.8	4.6/0.7	4.9/0.8	14.4
43	G 130	**3.1**/250	3.3/250	3.3/275	3.6/0.8	3.7/0.8	3.7/0.8	4.3
45	U 100	**7.0**/175	10.0/175	9.9/200	12.0/0.8	8.4/0.8	8.9/0.8	-
47	U 150	**3.5**/225	5.1/175	5.2/175	4.1/0.8	4.1/0.8	4.3/0.8	4.1
48	U 175	7.6/175	**6.8**/175	9.9/175	7.4/0.8	8.8/0.7	9.0/0.7	7.4
49	U 200	3.9/175	4.9/200	4.9/175	4.3/0.8	**3.4**/0.7	4.2/0.8	4.8
50	U 225	**4.0**/175	4.4/175	4.6/200	5.3/0.8	4.3/0.7	5.0/0.8	5.1
51	U 250	4.6/275	5.3/250	4.6/225	4.6/0.8	5.2/0.8	6.2/0.8	**4.4**
52	U 275	5.2/250	**5.0**/225	6.9/275	5.5/0.8	5.2/0.7	6.7/0.8	6.5
54	T 75	**7.4**/225	8.9/200	9.9/225	9.7/0.8	9.8/0.8	10.3/0.8	-
55	T 100	5.3/175	5.6/200	5.9/175	**5.1**/0.8	5.9/0.7	5.4/0.8	-
56	T 125	**3.2**/175	4.5/175	4.3/175	3.9/0.7	3.9/0.8	4.5/0.8	-
57	T 150	4.1/200	4.2/175	5.0/175	4.3/0.7	**3.2**/0.8	4.2/0.8	-
58	T 175	3.7/250	**3.3**/175	5.8/175	3.3/0.8	3.8/0.7	3.5/0.8	-
59	T 200	4.2/225	3.9/200	4.4/225	4.0/0.8	**3.6**/0.8	3.7/0.8	-
60	T 225	**5.1**/225	5.4/225	6.9/250	5.3/0.8	7.1/0.8	7.7/0.8	-
62	T 275	**5.7**/275	7.3/275	7.4/250	8.3/0.8	6.8/0.7	8.8/0.8	-
65	A	2.8/175	3.8/175	4.1/200	4.6/0.8	3.7/0.6	2.9/0.8	3.5

Table 4.4: Pose Estimation by using TD6 for training

For each visual word formation scheme we choose the recognition method which gives the least maximum difference from the best pose estimation using D^{cc}. For example, from table 4.5 we can see that for ATC visual word a recognition method using NNR with threshold .8 provides the least maximum difference from the best pose estimation. The scheme B 0.7 (corresponding to the first row of table 4.5 did not yield a reasonable 3D map on TD2 (figure 4.3). Hence we do not use it in the experiments in next section 4.2.3.

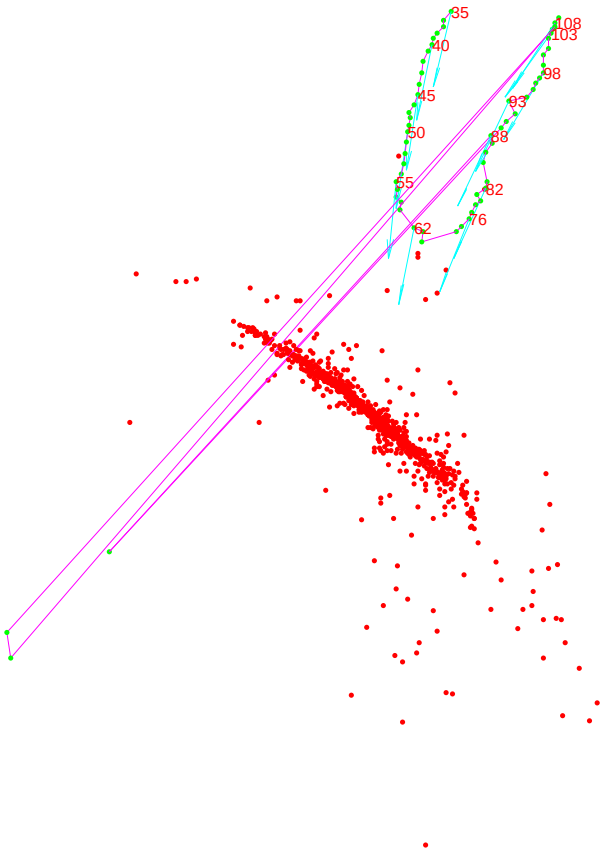


Figure 4.3: 3D map computed from TD2 using scheme B 0.7. The green dots connected by lines in magenta show the camera positions. Red dots show the 3D points. Only 62 camera positions, out of which 3 are incorrectly computed.

4.2.3 Pose estimation using TD1, TD2 and TD4

When using TD1 and TD4 for training, we employ an additional condition while choosing the optimal inlier set during RANSAC based PnP. For other training sets this additional condition is always satisfied and hence we do not use it. In the following subsection we describe how we incorporate this additional condition in RANSAC based DLSPnP and then present the results in section 4.2.3.2.

4.2.3.1 RANSAC with inlier spread priority

In RANSAC algorithm (section 1.4) the choice of a consensus set in an iteration to be considered as the set of inliers is based on the size (i.e. number of elements) of the consensus set. When the commonly visible regions between the test and train data shrinks due to pose variation we

Scheme	NNT D^{cc}			NNR D^{cc}		
	175	200	225	.6	.7	.8
B .7	6.2	4.9	3.0	2.1	2.2	**1.4**
K 8000	4.9	4.1	3.7	6.4	**3.1**	3.3
K 8500	6.2	7.0	9.8	9.3	5.3	**3.6**
G 60	**2.2**	2.8	3.9	4.7	2.7	2.2
G 70	1.7	3.3	3.0	2.2	2.1	**1.3**
G 100	2.8	2.6	2.7	3.1	2.4	**2.3**
G 120	5.5	2.4	2.7	3.0	3.0	**2.0**
G 130	6.5	2.9	**1.9**	3.0	3.2	2.4
U 150	2.3	3.5	4.0	4.4	2.5	**1.4**
U 200	2.0	2.0	2.4	3.1	2.3	**1.3**
U 225	2.8	**2.4**	2.5	3.6	3.0	2.5
U 250	4.7	3.7	**2.8**	5.8	4.0	3.3
T 100	3.0	5.9	7.5	4.2	4.7	**2.5**
T 125	2.7	2.4	4.1	3.4	1.7	**1.6**
T 150	2.1	3.0	4.8	4.1	1.8	**1.3**
A	2.5	2.6	2.6	1.2	0.8	0.6

Table 4.5: Maximum difference from the best result for matching with D^{cc}.

face the following problem (figure 4.4). The top and bottom portion of the figure show the result of two different iterations of the RANSAC. At the left top we show the 2D locations of the purported matches obtained by the test image 22 with TD1. The consensus set in one of the RANSAC iterations are marked by a green star inside a square. The rest are marked by magenta dots. The consensus set contains 12 members (numbered 5, 8, 15, 18, 19, 21, 22, 23, 29, 30, 40, 41) which are spread over a small region of the image. At the right top we show the estimated camera positions. We can see that image number 22 is out of place from the circular trajectory of test image camera positions. At bottom left we show the consensus set for the same image in another RANSAC iteration. It contains only 11 members (numbered 2, 5, 8, 17, 19, 21, 22, 29, 30, 39, 44), but they are spread over a relatively large area of the image. At bottom right the corresponding pose is shown. Through visual observation it is clear that the pose of the image 22 at bottom right is significantly better than that at top right. There are alternative strategies to choose the inliers for pose computation. MSAC [110] penalises each inlier in a consensus set based on its reprojection error unlike RANSAC in which all the inliers are considered equal. In [83], reprojection error of a 3D point is computed based on the uncertainty of its coordinate values.

Since the problem we face appears to be based on the narrow spread of inliers we use the following strategy. In order to make the RANSAC process choose the consensus set corresponding to the bottom of the figure 4.4 we should incorporate some measure of the spread of its members over the image. The length of the smaller side of the minimum bounding[3] rectangle on the 2D locations of the 2D points in a consensus set can be used to measure the spread. We use a similar but simpler strategy. We compute the minimum bounding rectangle aligned with the PCA axes of the 2D locations of the points in the consensus set. In figure 4.4, the axes are

shown in red lines. Clearly, the smaller axes in the left top image is much shorter when compared to the bottom left. We incorporate this measure of spread (i.e. length of the smaller side of the rectangle) in RANSAC as follows.

Let A represent the consensus set computed during the past RANSAC iterations and B be the consensus set obtained from the random sample in the current iteration. Let $nCur$ and $nNew$ be the cardinalities of A and B respectively. Let $condnCur$ and $condnNew$ be boolean variables indicating whether sets A and B satisfy the additional spread condition. Each RANSAC iteration proceeds as follows:

- Compute B, the consensus set in the random sample of the current iteration.
- Assign the cardinality of B to $nNew$. Set boolean variable $condnNew$ to indicate whether B satisfies the spread condition.
- If ($condnNew == true$ && $condnCur == false$) OR ($condnNew == true$ && $nNew > nCur$) OR ($condnCur == false$ && $nNew > nCur$) then:
  $A = B$
  $nCur = nNew$
- If ($condnNew == true$ && $condnCur == false$) OR ($condnNew == true$ && $nNew > nCur$) then:
  $condnCur = true$

Due to the above modification, the RANSAC algorithm prefers a consensus set B which satisfies the spread condition over a set A which does not satisfy the condition, irrespective of the cardinality of A and B. If both A and B satisfy or both do not satisfy the condition, then the one with higher cardinality is preferred.

4.2.3.2 Results

Using TD4 for training we were not able to obtain a reasonable pose from any combination of clustering and recognitions schemes (typical example is shown in figure 4.5). The accuracy of pose estimation using TD1 and TD2 for training is shown in the third and fourth columns of the table 4.6 respectively. On TD2 we obtain the least error of 5.6% using G 70 for clustering which is closely followed by G 100, ATC, G 60, U 100, U 150 and T 125. But on TD1 all these schemes (which are better than or close to the performance of ATC on TD2) perform poorly. We obtain the least error of 14.3% through ATC based 3D map. Even though there is no combination of scheme for clustering and recognition which is absolutely better over the other combinations across different data sets, we can say that ATC based clustering can be used to obtain reasonably good pose estimation consistently across different conditions of pose variation between train and test image. It provides either the best or significantly close to the best performance.

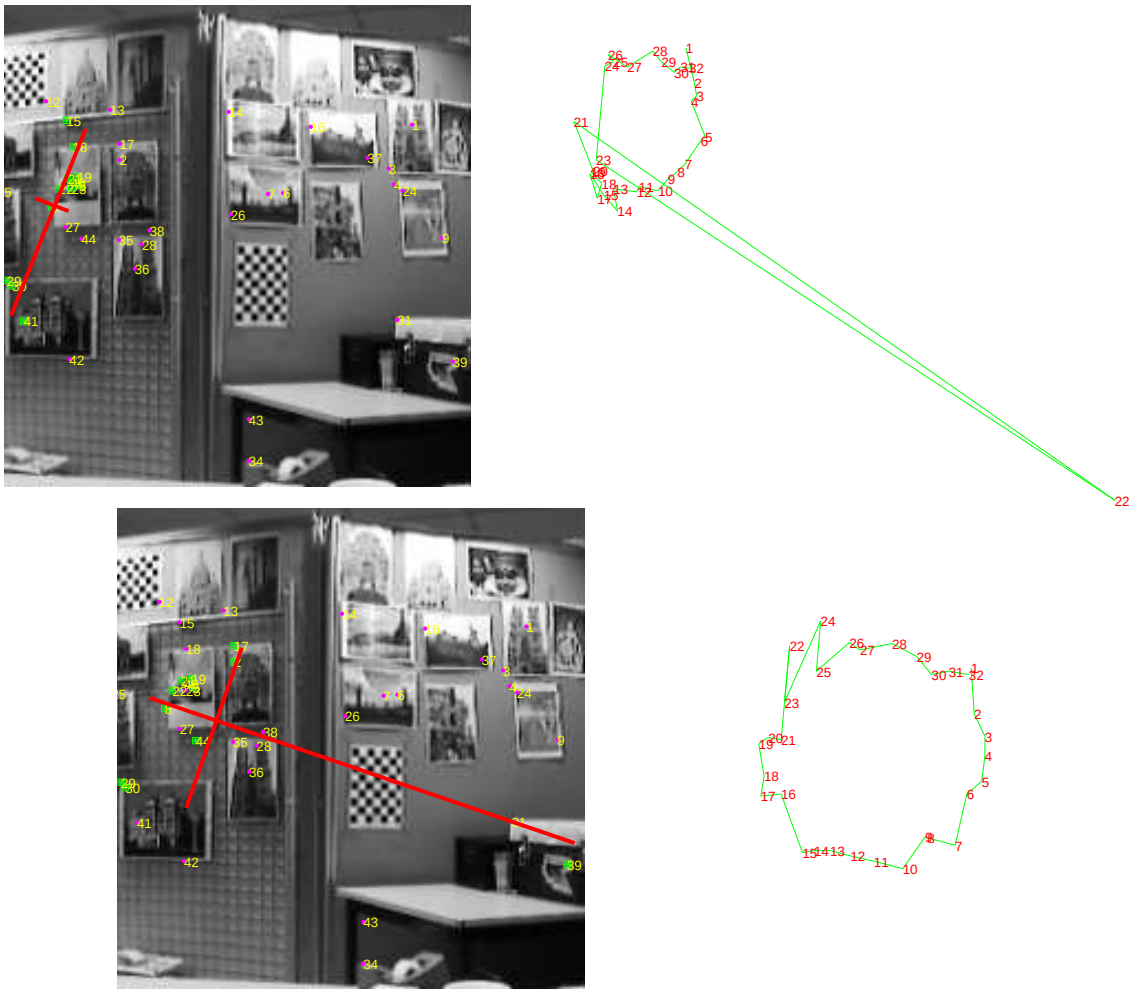


Figure 4.4: Need for condition on the spread of consensus set during RANSAC. The top left image shows test image 22. The 2D location of the consensus set is marked by a green star inside a square and the rest are shown in magenta. The red lines show the spread of PCA axes of the inliers. Since the spread is very small, the estimated pose of image 22 is out of place as shown in the top right. In the bottom row the spread of consensus set covers a significant portion of the image. Hence the estimated pose of image 22 is relatively better.

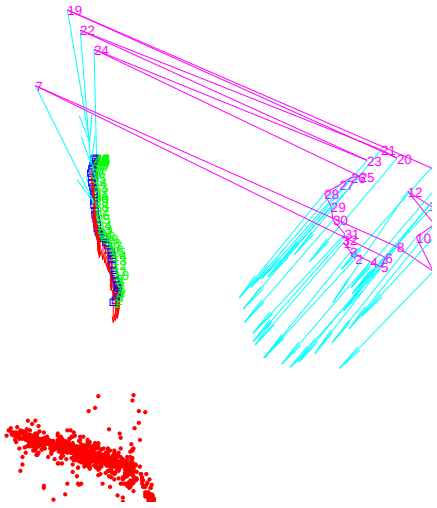


Figure 4.5: Pose estimated on 2D-to-3D matches obtained through scheme [TD4 A D NNR 0.8]. The red dots show the 3D points. Camera positions of images in TD4 are shown at the left top. The estimated camera positions of the test images are shown in magenta. It is difficult to compare the performance of two such pose estimations with huge error at several camera positions.

Scheme	TD1	TD2
K 8000 D^{cc} NNR 0.7	93.8	8.4
K 8500 D^{cc} NNR 0.8	20.5	88.5
G 60 D^{cc} NNT 175	115.1	7
G 70 D^{cc} NNR 0.8	69.1	5.6
G 100 D^{cc} NNR 0.8	99.2	6.5
G 120 D^{cc} NNR 0.8	19.6	14.1
G 130 D^{cc} NNT 225	22.7	15.9
U 150 D^{cc} NNR 0.8	55.5	7.8
U 200 D^{cc} NNR 0.8	58.3	7.2
U 225 D^{cc} NNT 200	63.2	15.4
U 250 D^{cc} NNT 225	77.5	9.1
T 100 D^{cc} NNR 0.8	65.3	102
T 125 D^{cc} NNR 0.8	20.4	8
T 150 D^{cc} NNR 0.8	67	14
A D^{cc} NNR 0.8	14.3	6.9

Table 4.6: Accuracy of pose estimation when using TD1 and TD2 for training

4.3 Recognition using statistical learning techniques

Statistical learning techniques use a set of training vectors to learn functions that can be used for dimensionality reduction and classification. As mentioned in the beginning of this chapter, we can label the training SIFT descriptors using the ID of the associated 3D point. Number of training SIFT descriptors per class we typically obtain is small when compared to the dimension of the SIFT descriptor. For example, in TD1 we have 59713 SIFT descriptors out of which 37093 descriptors (nearly 62%) get associated with 2538 number of 3D points when SfM is applied on ATC based clustering. This amounts to 14.6 SIFT descriptors per 3D point on average. This value is much smaller to the dimension of SIFT descriptor which is 128. The largest class contains 108 SIFT descriptors while the smallest contains only 4.

In this section we describe our experiments with popular statistical learning techniques namely, PCA (Principal Component Analysis), LDA (Linear Discriminant Analysis)[30] and SVM (Support Vector Machine[91]). PCA learns a linear transformation from a given set of training vectors which minimizes the mean-square error of its lower dimensional representation. PCA does not use the class label information of the training vectors for computing the transformation. LDA uses class label information to learn a linear transformation which brings together the vectors belonging to the same class and segregates those belonging to different classes. SVM identifies the vectors at the class boundaries in the training vector space and computes a hyperplane based classifier.

We use the clusters obtained through our ATC algorithm for training classifiers in all the experiments in this section. First we evaluate the classification accuracy of the learning techniques on RoboImage data. Then we evaluate the pose accuracy of the best performing learning method on MPG. In section 4.3.1 we present the way we setup RoboImage data for evaluation of classification accuracy. In section 4.3.2 we describe the way we use PCA, LDA and SVM for recognition. Section 4.3.3 presents the classification results on RoboImage data. We obtain best accuracy using SVM based recognition. In section 4.3.4 we use SVMs for performing pose estimation on MPG data and present the results.

4.3.1 Labelling test descriptors in RoboImage data

In order to measure accuracy of classification we need class labels on test SIFT descriptors. In practice it is very difficult to say whether the image location of a test SIFT descriptor x_t belongs to any 3D point Q_i or not. Moreover, the behavior of the classifier will depend upon the examples with which the classifier is trained. For 3D point recognition, training examples are labelled through a visual word formation technique. Hence we take the following approach. Figure 3.13 shows the known camera positions of 119 images. First we extract SIFT descriptors from the whole set of image and compute visual words. Then we obtain 3D points through SfM on matches obtained through ATC clustering. We treat this 3D point association with the SIFT descriptors as ground truth. That is, the SIFT descriptors associated with a 3D point Q_i are labelled as i both in training and test set. Others are labelled as -1. We select a subset of images (55 images indicated in green in the figure 3.13) and use the set of SIFT descriptors D extracted from them for training and the rest (i.e. SIFT descriptors extracted from the 64 images indicated in red in figure 3.13) for testing. The set of SIFT descriptors associated with 3D points D^{3D} and set of centers of clusters associated with 3D points D^{cc} are derived from D as described in the beginning of this chapter.

4.3.2 Brief description of PCA, LDA and SVM

4.3.2.1 PCA

PCA transformation represents the training vectors in a given dataset using eigenvectors of the covariance matrix of the dataset as basis. It can be used to reduce dimensionality by discarding the eigenvectors corresponding to lower eigenvalues. Suppose $\lambda_1 \geq \lambda_2 \geq, ..., \geq \lambda_n$ are the eigenvalues corresponding to the eigenvectors of the covariance matrix of D. We retain the first l eigenvectors such that the ratio of sum of the eigenvalues corresponding to the retained eigenvectors and the total sum of all the eigenvalues is above a fixed value r. That is, l is the smallest integer such that

$$\frac{\sum_{i=1}^{l} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq r \tag{4.2}$$

After computing the PCA transformation matrix (which consists of first l eigenvectors as its rows) from D, we apply the transformation on D, D^{3D} and D^{cc}. Using transformed D, D^{3D} and D^{cc} for training we perform 3D point recognition using NNT and NNR.

4.3.2.2 LDA and local variant of LDA(LLDA)

LDA tries to find a linear transformation which increases between class scatter and decreases within class scatter of the given set of labelled examples. Let S_w and S_b be the within-class and between-class scatter matrices corresponding to a given set of training vectors. S_w is computed by adding the covariance matrix of training vectors belonging to individual classes. S_b is computed as $\sum(\mu_i - \mu)(\mu_i - \mu)^T$ where μ_i is the mean of the class labelled as i and μ is the mean of whole training set. LDA transformation W^T is computed so as to maximize the objective function $\frac{|W^T S_b W|}{|W^T S_w W|}$ where $|B|$ represents determinant of matrix B. The objective function is the ratio of the product of eigenvalues of S_b and S_w in the transformed domain. It turns out that [30] the columns $z_1, z_2, ...z_d$, of a locally optimal W can be obtained as follows:

1. Compute Λ and Φ which are the matrices containing non-zero eigenvalues and corresponding eigenvectors of S_w respectively.

2. Then first apply the transformation $\Lambda^{-1/2}\Phi^T$. S_w will become an identity matrix under that transform. S_b will be transformed to

$$S_b' = \Lambda^{-1/2}\Phi^T S_b \Phi \Lambda^{-1/2} \tag{4.3}$$

3. Eigenvectors of S_b' are columns of the locally optimal W. Let Ψ be the corresponding projection matrix. Hence the transformation matrix will turn out to be

$$W = \Phi\Lambda^{-1/2}\Psi \tag{4.4}$$

For computing LDA we use D^{3D} i.e. only the training SIFT descriptors associated with a 3D point. For the descriptors not belonging to any 3D point, it is difficult to treat them as belonging to a single class and hence the mean and within-class scatter for those samples do not make sense. After computing the LDA transformation W^T, we apply it on D, D^{3D} and D^{cc}. Using transformed D, D^{3D} and D^{cc} for training we perform 3D point recognition using NNT and NNR.

Local variant of LDA (LLDA)

A global LDA transformation may not be suitable for data in which training vectors belonging to different classes are scattered in different shapes. Hence, we try to modify LDA training process to learn multiple transformations depending upon local data distribution. In LLDA we train LDA function for each class i (except for the class -1) using D_{Q_i} and the descriptors in D which are close to D_{Q_i}. The idea is to explore whether we can locally classify the descriptors using a linear discriminant function. For computing LDA based linear transformation all we need is within class scatter matrix S_w and between class scatter matrix S_b. For class i we compute S_w using only the training samples in D_{Q_i}. To compute S_b, we select from $D \backslash D_{Q_i} = \{x : x \in D \ \ and \ \ x \notin D_{Q_i}\}$ a set of 500 training samples closest to the mean μ_i of D_{Q_i}. Let $D^c_{Q_i}$ be this set. Between class scatter matrix is computed as $S_b = \sum_{x \in D^c_{Q_i}} (x - \mu_i)(x - \mu_i)^T$. Using these matrices we compute the transformation T_i. We also compute a threshold b_i to separate samples of D_{Q_i} and $D^c_{Q_i}$ under this transformation. The value of b_i is the average of the maximum distance of a training sample in D_{Q_i} and the minimum distance of a training sample in $D^c_{Q_i}$ from μ_i under T_i.

A test vector x_t is first matched with the class means in D^{cc}. Let $\mu_1, \mu_2, ..., \mu_k$ be the closest k means before applying any transformation T_i. We compute $d_1, d_2, ..., d_k$ the values their distances from x_t under respective LLDA transformations. We select the least value for $\frac{d_i}{b_i}$. If this value is less than 1 then we assign the class label i to x_t. Other wise we label it as -1.

4.3.2.3 Support Vector Machines (SVM)

Given a set of labelled training vectors belonging to two different classes (namely positive and negative samples), SVM [91] computes the hyperplane which separates the samples belonging to the two classes by minimizing the classification error and maximizing the margin between two classes. The whole computation can be expressed in terms of inner product between the vectors and hence we can use *kernel trick*[4] to perform non-linear classification in a transformed vector space. We use gaussian kernel consisting of a single parameter $\sigma > 0$ through which we can control the adaptability of the classifier to a given training set. The inner product between two vectors using gaussian kernel is computed as follows:

$$\phi(x_i, x_j) = exp(-\frac{||x_i - x_j||^2}{\sigma}) \tag{4.5}$$

We train a gaussian kernel SVM for each class i (except for the class -1) by using D_{Q_i} as positive samples and the rest of the training samples as negative samples. During test stage, for a test vector x_t, we compute the closest k centers from D^{cc}. If only one of the k SVMs corresponding to the respective centers classifies it as +ve then x_t is assigned to its class. Otherwise we label x_t as -1. In our experiments the number of test descriptors which get positively classified by more than one SVMs is negligible (one test vector per two images on average).

Deciding the value of σ: We experiment with different values of σ to find a suitable value. In addition we use the method described in [64] to automatically compute an optimal value for σ. It is based on the following observation. With the kernel defined in the equation 4.5 the norm of a vector x in the transformed vector space is $\sqrt{exp(-\frac{||x-x||^2}{\sigma})} = 1$ for any x. Moreover, the inner product between any two vectors is always greater than zero, which makes the angle between any two different vectors less than 90°. Hence, gaussian kernel always maps the vectors on to the positive orthant of the surface of the unit sphere in the transformed vector space. It implies that the inner product between any two vectors is the cosine value of the angle

between them. The method described in [64] tries to obtain the value for σ so that the angle between vectors belonging to same class is close to zero (i.e. inner product close to 1) and the angle between vectors belonging to different classes is close to 90° (i.e. inner product close to 0) in the transformed vector space. From equation 4.5, we can see that the value of the inner product increases with σ and approaches unity as σ approaches ∞. That is, as σ increases, the vectors in the transformed domain move close to each other. Conversely, as σ approaches 0, the value of inner product approaches zero making all the vectors perpendicular to each other in the transformed descriptor space. For a particular value of σ, let $f_w(\sigma)$ be the average value of inner product in the transformed vector space for the samples belonging to the same class and $f_b(\sigma)$ be the average value of inner product in the transformed vector space for the samples belonging to different classes. We would like to have a σ for which $f_w(\sigma)$ is close to 1 and $f_b(\sigma)$ is close to zero. The following function is evaluated for different values of σ and the one which minimizes it is chosen for learning the SVM:

$$J(\sigma) = 1 - f_w(\sigma) + f_b(\sigma) \tag{4.6}$$

In our experiments we observe that $J(\sigma)$ is a 'U' shaped function of σ i.e. its value increases as the value of σ moves away from the sampled point at which minimum value of $J(\sigma)$ is obtained. For different classes the minimum value of $J(\sigma)$ is always obtained between 1 to 8. Hence, for each class we evaluate $J(\sigma)$ at different values of $\sigma \in \{1, 1.05, 1.1, ..., 8\}$ and use the value of σ corresponding to the minimum value of $J(\sigma)$ to train SVM. Figure 4.6 shows the shape of the function $J(\sigma)$ for one of the clusters used during training.

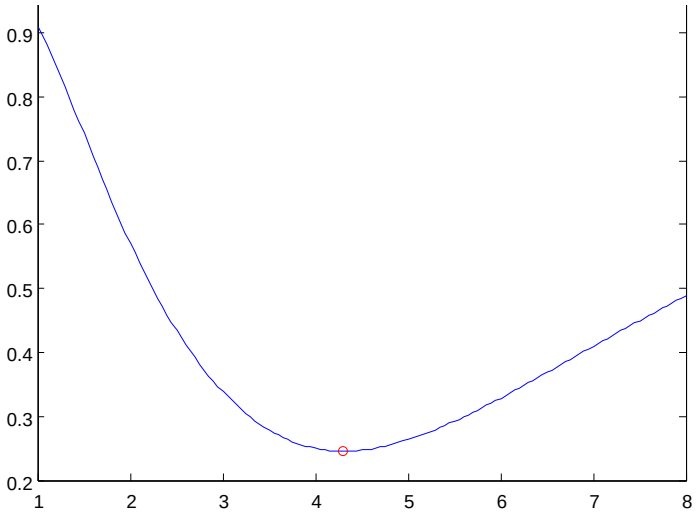


Figure 4.6: Graph of $J(\sigma)$ (defined in equation 4.6) for one of the clusters in our experiments for different values of $\sigma \in \{1, 1.05, 1.1, ..., 8\}$. The function attains minimum value of 0.25 at $\sigma = 4.3$

4.3.3 Classification accuracy on RoboImage data

We have 32800 training vectors (extracted from 55 images) out of which 23349 are labelled to one of 993 classes (or equivalently 3D points). There are 40297 test vectors extracted from 64 images, out of which 20091 labelled to one of 993 classes. First we discuss the result of applying linear transformations based on PCA and LDA in section 4.3.3.1. In section 4.3.3.2 we present the results of using SVM for learning non-linear discriminant function.

4.3.3.1 Linear transformation through PCA and LDA

Table 4.7 and 4.8 show the classification accuracy of using PCA and LDA transformation respectively. Tables show the percentage of error for different cases. The left most column indicates the kind of linear transformation used. Corresponding to each transformation (except LLDA) there are 6 rows each corresponding to a classification scheme indicated in the second column, 3 for NNT on D, D^{3D}, D^{cc} and 3 for NNR on D, D^{3D}, D^{cc} respectively. The entries in each cell of a row indicate the classification error for different parameters used in that particular scheme. In table 4.7, the first part corresponds to 'SIFT 128' indicating that matching is performed in original SIFT descriptor space. We use 8 different thresholds on distances 125:25:300 for NNT. We use 4 different thresholds .6,.7,.8,.9 on NNR. The rest of the 4 columns in a row corresponding to NNR are left blank. These threshold values are indicated in the first row of the table as 'x/y' where x and y correspond to the threshold values used for NNT and NNR respectively. After the part corresponding to original SIFT descriptor space comes PCA based transformation. 'PCA$r-d$' indicates PCA transformation obtained by retaining top eigenvalues summing up to r of the total sum which yields a transformation of dimensionality 'd'. For NNT, when dimensionality is reduced we reduce the threshold values accordingly. We multiply the threshold used in the original SIFT descriptor space by r. In table 4.8, LDA$r-d$ in the first column indicates LDA transformation computed after the application of PCA$r-d$ on the whole training set. Rest of the entries corresponding to these columns are arranged as in table 4.7.

At bottom part of table 4.8 the error rates of LLDA are shown. LLDAr indicates that PCAr has been applied in the beginning on the whole training set. Each LLDAr contains two rows with tags 'LowR' and 'FullSw' respectively. Since LLDA learns a discriminant function for each class using only the training samples corresponding to that class as positive samples, there are only few samples to compute S_w, it may be rank deficient. Usually, in such cases the data is first transformed by discarding the eigenvectors corresponding to the nullspace of S_w. After this transformation we obtain a full rank S_w in a lower dimension. We may loose discriminant information in the nullspace of S_w in this process. On the other hand we can modify the lower eigenvalues of S_w in order to enforce full rank. Let $S_w = \phi^T A \phi$ be the singular value decomposition of S_w. Let $\lambda_1, ..., \lambda_l, \lambda_{l+1}, ..., \lambda_n$ be the diagonal elements of A which are eigenvalues of S_w in the decreasing order in which $\frac{\sum_{i=l+1}^{n} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \leq 5\%$. That is, the sum of eigenvalues $\lambda_{l+1}, ..., \lambda_n$ amounts to less than 5% of the total. We replace these lowest values by λ_l in A to obtain full rank S_w. In table 4.8, 'FullSw' corresponds to the case in which this regulation process is followed. 'LowR' corresponds to the case where no such regulation is performed. Each column of LLDA section corresponds to the number of cluster centers k computed while matching. We have tried values 1, 5, 10, 15, 20, 25, 30, 35 for k. These value of k are indicated in the table in a row which separates the portions of LDA and LLDA entries.

By observing tables 4.7 and 4.8, we can say that within each linear transformation (except LLDA), NNR matching scheme with threhold 0.8 performs best classification. For example, in original SIFT descriptor space, for matching with D^{cc} (row numbers 4 and 7 of table 4.7), NNR with threshold 0.8 gives the least classification error of 20.7%. The least value of error for NNT on D^{cc} is 25.3% corresponding to threshold 225. Matching a test descriptor x_t with whole D gives better accuracy than matching with D^{3D} which in turn gives better accuracy than matching with D^{cc}. The error while using D^{cc} is nearly the double of that while using D for NNR classification scheme in most of the cases where it is more than 2.5 times for NNT. This error gap could not

	Scheme	125/0.6	150/0.7	175/0.8	200/0.9	225/-	250/-	275/-	300/-
SIFT 128	NNT D	15.5	13.0	11.8	11.3	11.5	12.3	13.7	15.2
	NNT D^{3D}	16.8	15.9	16.9	19.5	22.7	26.7	31.8	37.6
	NNT D^{cc}	38.6	33.2	29.0	26.7	26.1	27.6	31.1	36.5
	NNR D	13.9	11.2	10.0	10.8	-	-	-	-
	NNR D^{3D}	13.9	13.3	15.0	22.7	-	-	-	-
	NNR D^{cc}	29.0	23.4	20.7	25.1	-	-	-	-
PCA0.9-51	NNT D	14.5	12.5	11.6	11.6	12.0	13.1	14.7	16.6
	NNT D^{3D}	16.3	16.2	17.8	20.7	24.2	28.6	34.4	40.7
	NNT D^{cc}	37.4	32.3	28.7	26.6	26.5	28.4	32.5	38.3
	NNR D	13.9	11.4	10.3	11.5	-	-	-	-
	NNR D^{3D}	14.1	13.7	15.9	24.5	-	-	-	-
	NNR D^{cc}	28.7	23.9	21.7	27.4	-	-	-	-
PCA0.8-30	NNT D	14.4	12.7	12.0	12.2	12.9	14.3	16.2	18.3
	NNT D^{3D}	16.6	16.9	18.7	22.0	26.1	31.0	37.6	44.1
	NNT D^{cc}	37.3	32.6	29.3	27.3	27.5	29.8	34.7	41.2
	NNR D	14.3	11.8	11.0	12.5	-	-	-	-
	NNR D^{3D}	14.6	14.5	17.2	26.5	-	-	-	-
	NNR D^{cc}	29.4	24.9	23.4	30.0	-	-	-	-
PCA0.7-20	NNT D	14.6	12.9	12.5	13.0	14.4	16.3	18.5	20.2
	NNT D^{3D}	17.2	17.5	19.8	23.6	28.5	34.6	41.7	47.7
	NNT D^{cc}	37.5	33.0	29.8	28.6	29.4	32.3	37.7	45.0
	NNR D	15.0	12.6	11.8	13.6	-	-	-	-
	NNR D^{3D}	15.2	15.3	18.7	28.5	-	-	-	-
	NNR D^{cc}	30.3	26.2	25.7	33.4	-	-	-	-
PCA0.6-14	NNT D	15.3	13.9	13.6	14.6	16.6	19.0	21.0	22.3
	NNT D^{3D}	18.0	18.8	21.3	25.9	31.7	38.9	46.0	51.0
	NNT D^{cc}	38.4	34.3	31.5	30.6	31.9	35.4	41.5	49.0
	NNR D	16.2	14.0	13.2	15.4	-	-	-	-
	NNR D^{3D}	16.3	16.6	20.5	31.0	-	-	-	-
	NNR D^{cc}	32.3	28.6	28.9	36.7	-	-	-	-

Table 4.7: Classification error for NNT and NNR matching schemes under PCA

	Scheme	125/0.6	150/0.7	175/0.8	200/0.9	225/-	250/-	275/-	300/-
LDA SIFT 128	NNT D	17.3	14.0	12.4	11.8	11.6	12.2	13.3	14.9
	NNT D^{3D}	18.2	16.3	16.8	18.8	21.8	25.9	30.5	35.9
	NNT D^{cc}	39.3	33.7	28.9	25.9	25.3	26.9	30.5	35.7
	NNR D	14.5	11.5	9.8	10.3	-	-	-	-
	NNR D^{3D}	14.2	12.9	13.9	20.3	-	-	-	-
	NNR D^{cc}	29.4	22.6	18.6	21.3	-	-	-	-
LDA0.9-51	NNT D	15.5	13.0	11.7	11.4	11.5	12.3	13.4	15.0
	NNT D^{3D}	16.6	15.9	16.5	18.9	21.9	25.8	30.3	36.1
	NNT D^{cc}	38.2	33.0	28.8	25.8	24.7	25.9	29.0	33.6
	NNR D	13.7	11.1	9.8	11.1	-	-	-	-
	NNR D^{3D}	14.0	13.4	15.3	23.7	-	-	-	-
	NNR D^{cc}	28.0	22.5	19.9	24.8	-	-	-	-
LDA0.8-30	NNT D	15.0	12.9	11.9	11.8	12.2	13.1	14.7	16.6
	NNT D^{3D}	16.7	16.4	17.7	20.1	23.7	28.0	33.5	39.6
	NNT D^{cc}	37.8	33.0	29.3	26.9	26.1	27.3	30.6	36.1
	NNR D	14.0	11.6	10.7	12.0	-	-	-	-
	NNR D^{3D}	14.4	14.2	16.5	25.3	-	-	-	-
	NNR D^{cc}	28.6	23.8	22.0	28.1	-	-	-	-
LDA0.7-20	NNT D	14.7	12.9	12.2	12.5	13.5	15.1	17.1	18.9
	NNT D^{3D}	17.1	17.1	19.2	22.5	26.9	32.7	39.2	45.3
	NNT D^{cc}	37.2	32.8	29.4	27.7	27.9	30.4	35.3	41.8
	NNR D	14.8	12.4	11.5	13.1	-	-	-	-
	NNR D^{3D}	15.1	15.0	18.1	28.2	-	-	-	-
	NNR D^{cc}	29.6	25.4	24.6	31.9	-	-	-	-
LDA0.6-14	NNT D	14.8	13.5	13.5	15.0	17.0	19.4	21.4	22.5
	NNT D^{3D}	17.8	19.0	22.1	27.2	33.4	41.0	47.5	52.0
	NNT D^{cc}	37.1	33.1	30.6	30.1	32.2	36.6	43.7	51.2
	NNR D	16.1	13.8	13.1	15.3	-	-	-	-
	NNR D^{3D}	16.3	16.5	20.2	30.8	-	-	-	-
	NNR D^{cc}	31.5	27.7	28.3	36.5	-	-	-	-
		1	5	10	15	20	25	30	35
LLDA 0.9	LowR	29.5	28.9	28.8	28.8	28.8	28.8	28.9	28.9
	FullSw	19.3	18.3	18.3	18.2	18.2	18.2	18.2	18.2
LLDA 0.8	LowR	30.1	29.3	29.3	29.3	29.3	29.3	29.3	29.3
	FullSw	19.8	18.6	18.5	18.5	18.5	18.5	18.5	18.5
LLDA 0.7	LowR	29.5	28.2	28.1	28.1	28.1	28.1	28.1	28.1
	FullSw	20.7	19.0	18.9	18.8	18.8	18.8	18.8	18.8
LLDA 0.6	LowR	29.0	27.3	27.0	26.9	26.9	27.0	27.0	27.0
	FullSw	22.3	20.3	20.1	20.0	20.0	20.0	20.0	20.0

Table 4.8: Classification error for LDA and LLDA

be reduced even by using LDA. Only LLDA with full rank regulation on S_w performs slightly better than matching x_t with D^{cc} in lower dimensions. For example LLDA 0.7 with FullSw and k=5, gives 19% error while minimum error in that dimension for PCA and LDA is 24.6% (which corresponds to LDA 0.7, NNR 0.8 on D^{cc}). But using D with LDA 0.7, we can obtain much better accuracy of 11.5% through NNR 0.8.

4.3.3.2 Learning non-linear classifier through RBF-SVM

Similar to LLDA we train RBF-SVM for each class. First we experiment by fixing a value for σ for all classes. For each class we use the training vectors belonging to that class as +ve samples and 500 training vectors closest to the mean of the +ve samples which do not belong to the class as -ve samples. For each x_t we classify x_t with the SVMs belonging to k closest centers. If only one of the k SVMs classifies it as +ve then x_t is assigned to that class. Otherwise we label it as -1. We experiment by fixing different values for σ and k. Then we experiment with computing σ automatically for each class using the method described in [64]. Table 4.9 shows the classification accuracy. The first column indicates the σ value and the subsequent columns show the classification accuracy for different values of k. The entries are in 'x/y' form in which x is the classification error in percentage and y is the number support vectors with which a test vector x_t had to be compared on average in order to perform classification. Out of the four different values for σ, we obtain best results with $\sigma = 5$. Using $\sigma = 5$ we once again learned SVM by using all the negative training samples (not just 500). The results are shown in the row '5 All Neg'. We can see that there is not much difference in accuracy, but the number of support vectors need to be compared is reduced slightly. We feel that the presence of all the negative samples has helped SVM learning algorithm to choose the support vectors more optimally. Hence while learning SVMs using automatically computed σ values for each class, we used all the available -ve samples. The results are shown in the last row of table 4.9. We can see

σ	k=1	k=5	k=10	k=15
4	14.1/61	12.4/301	12.3/609	12.2/926
5	13.4/36	11.6/180	11.4/364	11.4/553
10	13.6/22	12.6/109	12.9/219	13.1/330
25	15/19	16.9/94	19.8/188	22.1/282
50	15.3/19	17.9/96	21.5/192	24.4/288
5 All Neg	13.2/33	11.5/166	11.3/333	11.3/503
Auto σ	13.6/43	11.9/214	11.7/431	11.7/551

Table 4.9: Classification error while using k-SVMs. The first column indicates the σ value and the subsequent columns show the classification accuracy for different values of k, where k is the number of SVMs chosen based on the k closest centers to the test vector x_t. The entries are in 'x/y' form in which x is the classification error in percentage and y is the number support vectors with which x_t had to be compared on average in order to perform classification.

that using the method in [64] we can obtain error rate as low as 11.9% for $k = 5$ which needs 214 additional vector comparisons. There is no significant gain in accuracy by increasing k further. This result indicates that using SVMs we can speed up the recognition process by performing k-nearest neighbor search on D^{cc} and then using the SVMs to obtain the class label. Using the method in [64] we can automate the task of choosing the right value for σ.

4.3.4 Pose accuracy of SVM based recognition on MPG data

Based on the results on RoboImage data we learn SVMs with automatically computed σ for each 3D point in order to recognize them in a test image. Since in MPG we have much more 3D points (nearly 2500 points compared to 993 of RoboImage data), we apply PCA $r = 0.9$ based dimensionality reduction before learning SVMs in order to reduce the memory needed for storing each SVM. As in our earlier experiments, we use DLSPnP for TD1 and TD2, and EPnP for the rest of the training sets. For TD1 we performed 2500 trials during RANSAC and for the rest we used only 500 trials.

Table 4.10 shows the accuracy of pose computed from SVM based 3D point recognition using TD1,TD2,TD3, TD5 and TD6 for training. The first row shows the pose accuracy obtained using SVM based 3D point recognition. The last row shows the pose accuracy of NNR 0.8 using D^{cc} for comparison (copied from different tables in the section 4.2.1 and 4.2.3.2). With SVMs the pose accuracy seems to be reduced slightly for TD1 and TD6, with no significant change in other cases. We do not see any clear advantage of the better classification performance of SVMs when compared to NNR with D^{cc}.

In order to analyze this further we reduced the number of RANSAC trials while performing pose computation using TD1. We computed pose multiple times using only 500 RANSAC samples (instead of 2500) each time. A better 3D point recognition method will contain less number of outliers and hence it will be less effected by this reduction in RANSAC trials. The results are shown in table 4.11. The first three columns correspond to recognition through NNR with threshold 0.8. The first column uses the whole training set D, second column uses only D^{cc} and the third column uses D^{cc} after reducing its dimension using a PCA transformation learned from D with retained eigenvalue ratio 0.9. Last column corresponds to SVM based recognition learnt from D after reducing its dimension using a PCA transformation learned from D with retained eigenvalue ratio 0.9. The entries corresponding to error above 50% are shaded in gray. Error above 50% indicates poor pose estimation accuracy with gross errors similar to that depicted in figure 4.1(c). Using D^{cc} in reduced dimensional space of PCA $r = 0.9$ (third column of table 4.11) produces gross errors 10 out of 20 i.e. 50% of the times. Using D^{cc} without any dimensionality reduction (second column of table 4.11) produces gross errors 3 out of 20 times. SVM based 3D point recognition (last column of table 4.11) leads to pose estimation which is as stable as using the whole D (first column of table 4.11) for recognition i.e. there was no gross error with less number of RANSAC trials.

Using SVMs significantly reduces the time needed to perform recognition. Even if we consider just the cost of finding only the first nearest neighbor while matching with D of TD1, we need 9 seconds per image. We need only 0.32 seconds for performing k-nearest neighbor search in D^{cc} for k=5. In addition we had to perform 3×10^{05} vector comparisons on average per image to perform SVM based classification. This needs nearly 1.6 seconds of cputime. Adding the two durations together, we need 1.92 seconds per image which is significantly less than 9 seconds.

4.4 Conclusion

In this chapter we experimented with various strategies on the 3D map obtained from different clustering schemes to perform 3D point recognition. Due to the huge number of combination of parameters at training and test stage we successively eliminated test cases based on the results

Scheme	TD1	TD2	TD3	TD5	TD6
SVM (PCA r = 0.9),k=5	15.74	6.8	5.42	2.34	5.16
D^{cc} NNR 0.8	14.3	6.9	5.2	2.3	2.9

Table 4.10: Pose accuracy : SVM (PCA .9) vs NNR on D^{cc} based 3D point recognition. The first column shows the recognition scheme used to identify 3D points in a test image. The subsequent columns show the pose error for performing pose computation using 3D maps obtained by different training sets of MPG.

NNR 0.8			SVM k=5
D	D^{cc}	PCA D^{cc}	PCA D
16.7	16.8	68.1	18.9
18.5	16.9	21.6	19.7
15.3	59.7	61.4	20.3
18.3	16.3	67.8	19.5
16.8	19.6	22.4	20.2
15.9	18.2	21.9	21.5
18.3	19.5	65.2	18.1
17.4	59.6	65.5	19.9
16.8	16.9	21.4	20.7
17.4	17.0	96.0	21.6
15.8	17.7	27.5	19.1
15.6	22.8	22.8	20.7
17.3	21.0	65.9	21.6
16.9	21.6	33.1	20.9
17.7	19.6	64.0	21.1
18.7	63.5	19.0	17.5
19.7	16.2	70.4	19.2
16.8	23.7	25.8	21.2
15.9	15.9	64.0	18.4
14.7	18.8	19.2	20.4

Table 4.11: Result of estimating pose repeatedly 20 times using 3D map computed from TD1. Only 500 RANSAC trials are performed. It is much less than the 2500 trials we used in previous experiments using TD1. The first three columns correspond to recognition through NNR with threshold 0.8. The first column uses the whole training set D, second column uses only D^{cc} and the third column uses D^{cc} after reducing its dimension using a PCA transformation learned from D with retained eigenvalue ratio 0.9. Last column corresponds to SVM based recognition learnt from D after reducing its dimension using a PCA transformation learned from D with retained eigenvalue ratio 0.9. The entries corresponding to error above 50% are shaded in gray.

we obtained at different stages. Unlike previous chapter in which we were able to obtain results showing clear advantage of ATC clustering scheme over other methods for building 3D map, the results in this chapter showed marginal improvement due to ATC in few cases. The most significant of them being the ability to perform pose estimation consistently over different pose variation between train and test images. The clustering methods which performed better than ATC in some cases failed to provide reasonable pose on TD1. Subsequently, we experimented with statistical learning techniques on clusters obtained through ATC. Out of the various descriptor space transformation and discriminant functions SVM based methods showed significantly better performance in classification accuracy on RoboImage data. But we could not obtain results on accuracy of pose computation which clearly follow its advantage in classification. But when the number of trials during RANSAC was reduced, we were able to see the advantage of SVMs in terms of stability when compared to matching with D^{cc} and speed when compared to matching with D. With TD4 we were not able to compute reasonably good pose in any combination of clustering and recognition scheme. We feel that the main reason behind this is the poor quality of 3D map built due to the lack of wide baseline view of the scene in TD4. Combining the 2D-to-2D match constraints between the test and train images along with the 2D-to-3D matches [104] can be an interesting topic for future work for TD4.

Chapter 5

Accelerating Mean-Shift Clustering

5.1 Introduction

In chapter 3, we used Mean-Shift Clustering (MSC) based visual words to build 3D map. MSC is a non-parametric clustering technique[35, 22, 23] which is becoming increasingly popular in computer vision. Initially applied in relatively low-dimensional spaces (dimension up to 11, eg: object tracking[9], image segmentation [121]), it is now being applied on high dimensional feature descriptors like SIFT[51]. The simple iterative averaging process in MSC has strong theoretical support. It can be considered as mode seeking[23] gradient-ascent algorithm over the density function obtained through performing kernel density estimation (KDE) on the training set of vectors. The step-size of the gradient ascent is adaptive and guarantees convergence. Since MSC is based on non-parametric density estimation it can handle clusters of arbitrary shape. One more distinctive feature of MSC technique is that we do not need to know the number of clusters *a priori*. This is particularly advantageous in computer vision tasks for clustering objects with similar visual patterns, since in most of the cases, the number of distinctive objects (or clusters) in the scene is not known in advance. Despite these nice properties, MSC has a major disadvantage. The computational cost of MSC becomes prohibitively huge as the dimension of the feature space increases. The major source of this cost is the range search process [36] which involves computing, in each iteration, the set of training vectors within a range w from the current mean vector.

In chapter 3, we observed that MSC is the slowest among the different clustering methods (please refer to figure 3.22) in our experiments. In this chapter we propose a strategy to accelerate MSC which tries to divide the training set into different groups such that the mean-shift computation for a vector in one group can be performed without comparing it with vectors in other groups. These groups are obtained by performing transitive closure (TC) based clustering presented in section 3.4.1.1. The matching threshold R for TC clustering depends on value of the range w with which range search is performed in each MSC iteration. In fact we prove that if $R = \sqrt{2}w$, then the modes we obtain by performing MSC on individual TC clusters are same as the modes we obtain by performing MSC on whole training set. We perform exact computations unlike methods which use subsampling of D[34] or approximate computations[20]. But, we obtain moderate acceleration compared to these inexact methods, for a specific range of parameter values used to perform mean-shift.

5.1.1 Background

Let $D = \{x_1, x_2 \dots x_n\}$ be the set of n training vectors, where each $x_j \in \mathbb{R}^N$. For each training vector x_j, MSC algorithm iteratively searches for peaks of the density function. Starting with the initial estimate $\mu_j^0 = x_j$, the mean vector is updated in each iteration as follows:

$$\mu_j^{k+1} = \frac{\sum_{x_i \in D} K(x_i - \mu_j^k) x_i}{\sum_{x_i \in D} K(x_i - \mu_j^k)} \tag{5.1}$$

where K is the kernel function from $\mathbb{R}^N$ to $\mathbb{R}$. As we can see from equation (5.1), for each vector $x_j \in D$, in each iteration k, we need to compute the distance of μ_j^k with all the vectors of D. Hence the computational complexity of standard mean-shift procedure is $O(\tau N n^2)$[119] where τ is the average number of iterations per data point. Usually the value of $K(x_i - \mu_j^k)$ depends only on the scalar value $\frac{||x_i - \mu_j^k||}{h}$, where h is the *bandwidth* parameter and $||.||$ is the Euclidean norm. The bandwidth may be *adaptive* i.e. h is derived based on neighborhood of x_i (e.g. h taking value of distance of x_i from its k^{th} nearest-neighbor in D) or *fixed* i.e. a single value for all x_i. In practice almost all kernel functions have compact *support* i.e. there is a finite range w such that $K(x - \mu_j^k) = 0$ for all $||x - \mu_j^k|| > w$.

Very small portion of D usually fall within the range w of μ_j^k in a MSC iteration. On the other hand, many of the elements in D eventually converge to the same mean at the end of MSC. Hence there are two ways to accelerate MSC computation: (i)using fast range search techniques[79] to reduce the number of comparisons needed in an iteration (ii)identifying $D' \subset \mathbb{R}^N$ which is significantly smaller than D but containing elements which can lead to the modes of D through MSC. In [34] a set D' which is much smaller than D is obtained by randomly sampling KDE of D without explicitly evaluating the KDE. MSC is performed on D' and at the end the elements of D are associated with the closest modes of D' thus obtained. In [119], D is divided into KD-tree clusters and each cluster is represented by a single sample. MSC is performed on this reduced set of representative samples. In [28] MSC is performed hierarchically. Small value for the range w is used to compute mean-shift clusters in the initial levels which merges elements of D within small regions. Higher range values are successively applied on the centroids of the previous level till the desired range size is reached. With small range values, tree structures (eg KD-tree) for fast range search provide good acceleration. For higher range values the number of data points get reduced due to merging. In [36, 117] locally sensitive hashing and dual tree based data structures are used respectively to organize the elements of D for fast range search. In [122] the datapoints in D which are likely to converge to the same mean are identified in each MSC iteration and merged. In [20] gaussian mean-shift is treated as *expectation maximization* (EM) in which computing the coefficient of each x_i in the right hand side of the equation 5.1 becomes E step and updating μ_j becomes M step. Sparse EM [81] is employed for reducing computation in E step. EM step is combined with Newton's method in order to achieve faster convergence near modes.

5.1.2 Overview

All the acceleration strategies discussed above use approximation, i.e, the final result may not be same as performing MSC directly on D. In contrast the strategy we present produces exactly same result as the original MSC on D. We try to reduce the required number of comparisons by dividing D into a finite set of partitions $\{P_i\}$, in such a way that for any $x_j \in P_i$, it is

guaranteed that $||x - \mu_j^k|| > w$ for all the training vectors $x \notin P_i$. The computation of the partitions can be performed during the first MSC iteration on each training vector with some additional cost. For the subsequent iterations the range search for a vector in a partition need to be done only within the partition it belongs. Our acceleration strategy is applicable for all kernel types which have compact support and fixed bandwidth. In section 5.2 we illustrate how MSC update in an iteration can be considered as convex combination operation. We present the necessary terminology in section 5.2.1 to systematically analyse our point of view. This point of view leads us to a *reachability* criteria presented in section 5.3 which can be used to divide D and reduce number of comparisons in MSC. Later in section 5.4 we present mathematical results to find the best possible values for this criteria. In section 5.5.1 we present the result of using our acceleration strategy for gaussian mean-shift operation on SIFT descriptors. Finally, in section 5.6 we summarize our observations based on the experimental results.

5.2 Mean-Shift update as bounded convex combination

We can rewrite the equation 5.1 as follows:

$$\mu_j^{k+1} = \sum_{x_i \in D} \frac{K(x_i - \mu_j^k)}{\sum_{x_i \in D} K(x_i - \mu_j^k)} x_i = \sum_{x_i \in D} \theta_i x_i \tag{5.2}$$

where $\theta_i = \frac{K(x_i - \mu_j^k)}{\sum_{x_i \in D} K(x_i - \mu_j^k)}$, which implies that $0 \leq \theta_i \leq 1$ and $\Sigma_1^n \theta_i = 1$. Hence, in each iteration, the new mean vector μ_j^{k+1} lies within the convex hull of the training vectors which are within distance w from its current position μ_j^k.

5.2.1 Terminology

- The Euclidean norm of a vector x is represented by $||x||$.
- If A and B are two sets of vectors, then the dissimilarity measure between the two sets is defined as $||A, B|| = \inf_{x \in A, y \in B} ||x - y||$. If one of the sets is singleton containing a vector x, then $||x, B|| = \inf_{y \in B} ||x - y||$
- If A and B are two sets of vectors, the $A + B = \{x + y : x \in A, y \in B\}$. If one of the sets is singleton containing a vector x, then $x + B = \{x + y : y \in B\}$
- In each iteration μ_j^{k+1} depends only on μ_j^k and the training vectors that are within distance w from it. Hence, when we deal with the process within a single iteration, we refer to the value of the mean vector in the beginning of the iteration (i.e. μ_j^k) as μ_0 and the value at the end of the iteration (i.e. μ_j^{k+1}) as μ_1.
- We say that a vector y is *visible* from vector x if $||x - y|| \leq w$.
- The set of training vectors visible from μ_0 is called V. Elements of V are referred as $v_1, v_2 \ldots v_m$.
- Convex hull of a set of vectors A is referred as $CH(A)$.

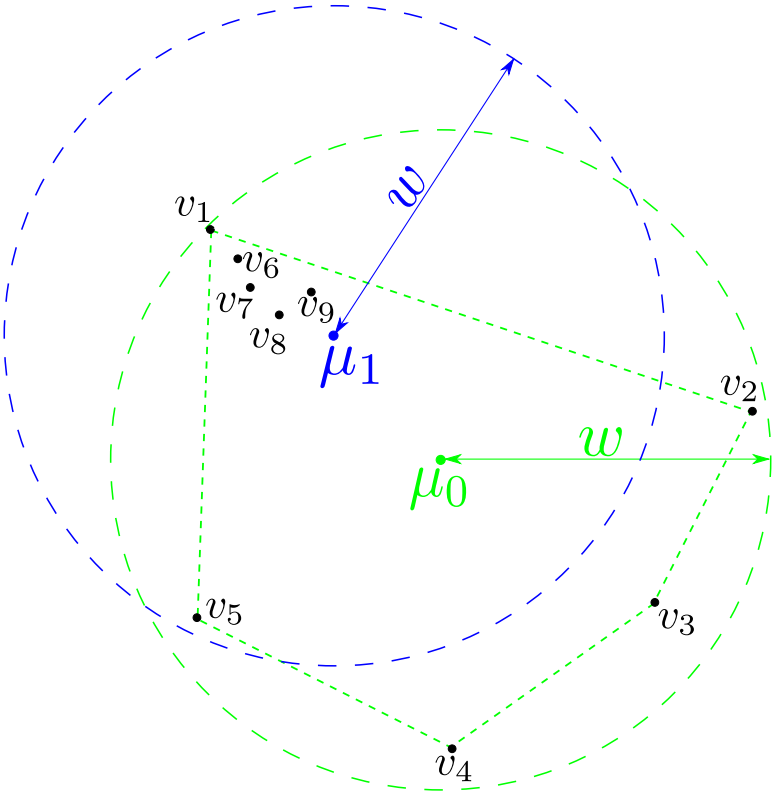


Figure 5.1: μ_1 is computed by convex combination of vectors v_i which lie within a distance w from μ_0

- While it is guaranteed that $\mu_1 \in CH(V)$, the converse is not true. Any vector in $CH(V)$ cannot be a valid μ_1. For example, for uniform kernel the only possible μ_1 that can be computed from a visible set V is the centroid of V. For any finite set of vectors A, we denote $CH_K(A)$ as the set of vectors which can be obtained from A using only the convex combination through the kernel function K as in equation (5.2). Mathematically :

$$CH_K(A) = \bigcup_{\mu_0 \in X_A} \left\{ \sum_{y_i \in A} \theta_i y_i \text{ where } \theta_i = \frac{K(y_i - \mu_0)}{\sum_{y_j \in A} K(y_j - \mu_0)} \right\}$$

  where X_A is the set of vectors in $\mathbb{R}^n$ from which all the elements of A are visible i.e. $X_A = \{x \in \mathbb{R}^n : ||x - y|| \leq w \text{ for all } y \in A\}$.

5.2.2 Illustration in 2D

Bounded convex combination update of mean vector is illustrated in figure 5.1. μ_0 is the current position of the mean vector. $V = \{v | v \in D \text{ and } ||v - \mu_0|| \leq w\} = \{v_1, v_2 \dots v_9\}$ i.e. training vectors within the green circle of radius w around μ_0. New mean vector is computed by convex combination as $\mu_1 = \sum_i \theta_i v_i$. μ_1 will lie inside the $CH(V)$ (In the figure $CH(V)$ is indicated by dashed polygon in green color). In the next iteration we search for training vectors within distance w from μ_1 (i.e. the training vectors lying inside the blue circle).

5.3 Reachability relation on elements of D

As we can see in the figure 5.1, the vectors in V constrain the possible values of μ_1 and consequently constrain the set of training vectors visible from μ_1. Let x be any vector in $\mathbb{R}^N$ which is visible from μ_1. We want to compute a reasonable upper bound value R for the set of values $||x, V||$ can take. The R we want to define here is the maximum distance from a vector visible from μ_1 to the set of training vectors visible from μ_0 i.e. V. In other words, R is the distance threshold such that, if μ_1 is a convex combination of vectors in V obtained through a kernel function and x is any vector satisfying $||\mu_1 - x|| \leq w$, then we are assured that there is at least one $v_i \in V$ such that $||x - v_i|| \leq R$. If such a distance threshold is available, then, in the next iteration, we need to compare μ_1 only with those training vectors which lie within distance R from V. We define R as:

$$R = \sup_{||r|| \leq w} ||V, CH_K(V) + r|| \tag{5.3}$$

Consider any training vector $x \in D$ such that $||V, x|| > R$. Since, by definition in equation (5.3), R is the supremum of the distance values from V for a vector lying within distance w from $CH_K(V)$, we can say that $||V, x|| > R \implies ||x, CH_K(V)|| > w$. Since $\mu_1 \in CH_K(V)$, we have $||x - \mu_1|| > w$. Hence we get the following conclusion:

$$||V, x|| > R \implies ||x - \mu_1|| > w \tag{5.4}$$

This condition is true for any upper bound of R

If we can compute R (or a reasonable upper bound of R) in advance, then we can define a *reachability* relation S on D. We say that y is *reachable* from x if y lies within distance R from x i.e. $S = \{(x, y) : x, y \in D, ||x - y|| \leq R\}$. This relation is reflexive and symmetric. For $x \in D$, let $B_x = \{\text{Elements in } D \text{ which are reachable from } x\} = \{y : (x, y) \in S\} = \{y \in D : ||x - y|| \leq R\}$. If we know R in advance, we can compute B_x for all the training vectors in advance. At the end of each MSC iteration we can compute the *reachability set* $B = \bigcup_{v_i \in V} B_{v_i}$. In other words, B is the set of training vectors within distance R from the vectors in V. For any $y \in D$, by definition of B, we can say that if $y \notin B$ then $||V, y|| > R$. Using equation (5.4), we can say that $||y - \mu_1|| > w$. To summarize, any training vector which does not belong to B will lie at a distance greater than w from μ_1. Hence, we need to compare μ_1 only with the vectors in B in next iteration.

5.3.1 Partitioning through transitive closure on reachability relation

If we apply transitive closure operation on relation S, we obtain a partition $T = \{P_i\}$ on D i.e. $(\cup P_i = D)$ and $(P_i \cap P_j = \phi \text{ for } i \neq j)$. Since T is obtained by transitive closure operation on S, we can say that $(y \in P_i) \implies (B_y \subseteq P_i) \implies (B_y \cap P_j = \emptyset \ \forall i \neq j)$. Hence, in any iteration on vectors in P_i, the reachability set B remains within P_i. Hence MSC can be performed on each partition independently.

The accelerating technique mentioned above can be performed in the same way with any upper bound of R. But, it is desirable to have a distance threshold as small as possible. In the following section we present the theoretical results we have on the bounds of R.

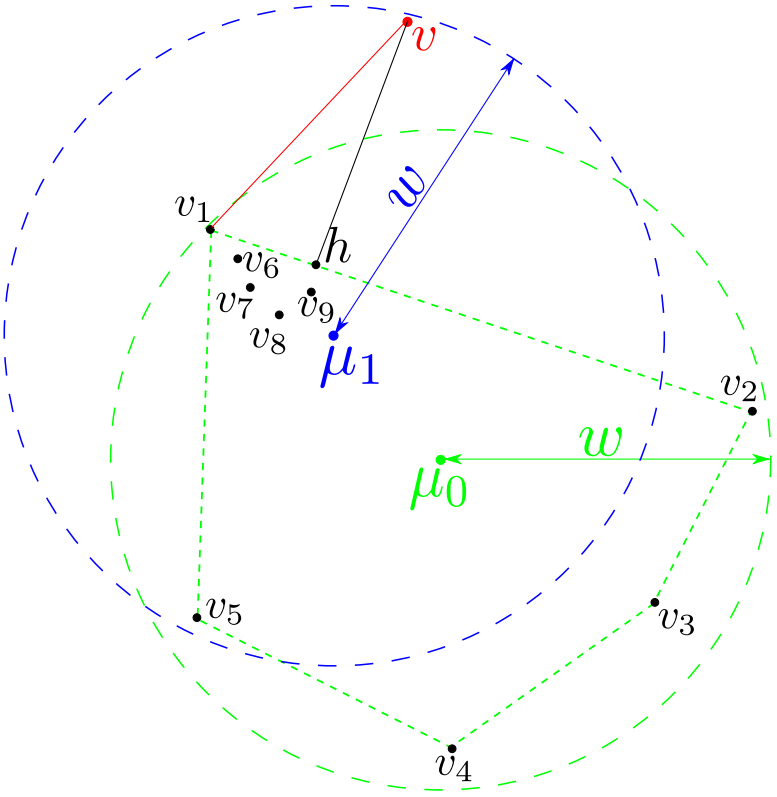


Figure 5.2: For any vector $v \notin CH(V)$, the nearest point h in a convex hull would either be a vertex or the line connecting v and h will be orthogonal to the surface of the convex hull.

5.4 Bounds on reachability threshold R

It is difficult to compute the exact value of R, since it is difficult to compute $CH_K(V)$. Hence we substitute $CH(V)$ to $CH_K(V)$ in equation (5.3) to define:

$$R_c = \sup_{r \in \mathbb{R}^N, ||r|| \leq w} ||V, CH(V) + r|| \tag{5.5}$$

Since $CH_K(V) \subseteq CH(V)$, we can say that $R \leq R_c$. In section 5.4.1, we show that $R_c = \sqrt{2}w$. In section 5.4.2, using a hypothetical D and uniform kernel, we show that $R \geq \frac{\pi}{\sqrt{6}}w$. Hence, in numerical terms we can say that $1.28w \leq R \leq 1.41w$.

5.4.1 Upper bound on R

In this section we show that $R_c = \sqrt{2}w$. The overview of the proof is shown in the figure 5.2. Since $\mu_1 \in CH(V)$, any point v within distance w from μ_1 will have its closest point h in $CH(V)$ such that $||h - v|| \leq w$. We will show that there is at least one vertex $v_i \in V$ such that $||h - v_i|| \leq w$ and $(v_i - h)$ is orthogonal to $(v - h)$. Hence, $||v - v_i||^2 = ||v - h||^2 + ||h - v_i||^2 \leq 2w^2$. In section 5.4.1.1, we present mathematical results which will be used to prove the main theorem in section 5.4.1.2.

5.4.1.1 Lemmas

Lemma 5.4.1 *Let x_0 be a vector in $\mathbb{R}^N$ and t be a positive real number. Let $X = \{x_1, x_2 \dots x_m\}$ be a finite set of vectors in $\mathbb{R}^N$ satisfying $||x_i - x_0|| \leq t$. For any $x \in CH(X)$, $\exists x_j \in X$ such that $||x - x_j|| \leq t$.*

Proof Without loss of generality, we assume that x_0 is the origin. Hence $||x_i|| = ||x_i - x_0|| \leq t$. Let $d_i = ||x - x_i||$. We need to prove that at least one $d_i \leq t$. On the contrary, if $d_i > t$ for all $x_i \in X$ then:

$$\begin{aligned} & d_i^2 = (x - x_i)^T(x - x_i) = x^T x - 2x^T x_i + x_i^T x_i > t^2 \\ \implies & x^T x - 2x^T x_i > 0 \text{ for all } x_i \in X, \text{ since } x_i^T x_i = ||x_i||^2 \leq t^2 \\ \implies & 2x^T x_i - x^T x < 0 \text{ for all } x_i \in X \end{aligned}$$

If we define a hyperplane $f(z) = u^T z - b$ where $u^T = 2x$ and $b = x^T x$, we can see that $f(x_i) < 0$ for $x_i \in X$ and $f(x) = 2x^T x - x^T x = x^T x \geq 0$. This implies that the hyperplane f separates x from X, which is a contradiction since $x \in CH(X)$. Hence we have at least one $d_j \leq t$ that is $\exists x_j \in X$ such that $||x - x_j|| \leq t$.(PROVED)

The following lemma is a classic theorem in theory of optimization [70]. We will be using it to prove lemma 5.4.3.

Lemma 5.4.2 *If G is a closed convex set in Euclidean space $\mathbb{R}^N$ and $x \in \mathbb{R}^N$, then there exists a unique vector $g_0 \in G$ such that $||x - g_0|| = \inf_{g \in G} ||g - x||$. The necessary and sufficient condition for g_0 to be minimum distance vector from x is that $(g_0 - x)^T(g_0 - g) \leq 0$ for all $g \in G$.*

Lemma 5.4.3 *Let $V = \{v_1, v_2 \dots v_m\}$ be a finite set of vectors in $\mathbb{R}^N$. Let v be a vector outside $CH(V)$. Let $h = \sum_{v_i \in V} \theta_i v_i$ be the vector in $CH(V)$ having the minimum distance from v. Then $\theta_i > 0 \implies (h - v)^T(h - v_i) = 0$.*

This lemma implies that the line connecting v and the closest point h is orthogonal to the line connecting h and any element of V having nonzero co-efficient in the convex combination of h.

Proof Let $b_i = (h - v)^T(h - v_i)$. Since $h \in CH(V)$ is the closest point to v and $v_i \in CH(V)$, using lemma 5.4.2, it is clear that $b_i \leq 0$. We need to prove that, if $\theta_i > 0$ then $b_i = 0$.

$$\sum_{i=1}^{m} \theta_i b_i = \sum_{i=1}^{m} (h - v)^T(\theta_i h - \theta_i v_i) = (h - v)^T(\sum_{i=1}^{m} \theta_i h - \sum_{i=1}^{m} \theta_i v_i) = 0 \tag{5.6}$$

If $b_j < 0$ for some j and $\theta_j > 0$ then $\theta_j b_j < 0$. Moving $\theta_j b_j$ in the summation of equation (5.6) to the right side, we get:

$$\sum_{\substack{i=1 \\ i \neq j}}^{m} \theta_i b_i = -\theta_j b_j > 0 \tag{5.7}$$

Since $\theta_i \geq 0$, equation 5.7 implies that one of the b_i's in the summation of the equation should be greater than zero, which is a contradiction.(PROVED)

The following corollary directly follows from lemma 5.4.3:

Corollary 5.4.4 *In Lemma 5.4.3, the vector h can be obtained using convex combination of only those vectors $v_i \in V$ which satisfy $(h - v)^T(h - v_i) = 0$.*

5.4.1.2 Proof of $R \leq \sqrt{2}w$

Now we have all the necessary tools to prove what we have described in the beginning of this section 5.4.1.

Theorem 5.4.5 *Let $\mu_0 \in \mathbb{R}^N$ and $V = \{v_1, v_2 \dots v_m\}$ be a finite set of vectors in $\mathbb{R}^N$ satisfying $||v_i - \mu_0|| \leq w$. Let v a vector outside $CH(V)$ such that $||v, CH(V)|| \leq w$. Let h be the vector in $CH(V)$ having the minimum distance from v. Then there exists $v_i \in V$ such that $||v_i - h|| \leq w$ and $(h - v)^T(h - v_i) = 0$.*

Proof From corollary 5.4.4, h is a convex combination of vectors $U \subseteq V$ such that each $u \in U$ satisfies $(h - v)^T(h - u) = 0$. Since all of those vectors in U are within distance w from μ_0, and h is convex combination of those vectors, from lemma 5.4.1, there exists at least on $u_i \in U$ which such that $||u_i - h|| \leq w$. Hence the distance between $u_i \in U \subseteq V$ and v i.e. $||u_i - v|| \leq \sqrt{w^2 + w^2} = \sqrt{2}w$.

5.4.2 Lower bound on R

In section 5.4.1 we proved that $R \leq \sqrt{2}w$. In this section we prove that R, in general, cannot be less than $\pi/\sqrt{6}$. The following result [5] will be used to derive this:

$$\sum_{j=1}^{\infty} \frac{1}{j^2} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} \dots\dots = \frac{\pi^2}{6} \tag{5.8}$$

5.4.2.1 Hypothetical training set

Consider a set of n vectors in $n-1$-dimensional space, $D = \{x_1, x_2 \dots x_n\} \subset \mathbb{R}^{n-1}$, as follows:

$$\begin{aligned}
x_1 &= (0, 0, 0, 0, 0 \dots 0) \\
x_2 &= (w, 0, 0, 0, 0 \dots 0) \\
x_3 &= (\frac{w}{2}, w, 0, 0, 0 \dots 0) \\
x_4 &= (\frac{w}{2}, \frac{w}{3}, w, 0, 0 \dots 0) \\
x_5 &= (\frac{w}{2}, \frac{w}{3}, \frac{w}{4}, w, 0 \dots 0) \\
x_6 &= (\frac{w}{2}, \frac{w}{3}, \frac{w}{4}, \frac{w}{5}, w, 0 \dots 0) \\
&\vdots \\
x_k &= (\frac{w}{2}, \frac{w}{3}, \frac{w}{4}, \frac{w}{5}, \dots \frac{w}{k-1}, w, 0 \dots 0) \\
&\vdots \\
x_n &= (\frac{w}{2}, \frac{w}{3}, \frac{w}{4}, \frac{w}{5}, \dots \frac{w}{n-1}, w)
\end{aligned}$$

where $w > 0$. These vectors are generated according to the rule:

$$x_{ij} = \begin{cases} \frac{w}{j+1} & 1 \leq j \leq i-2 \\ w & j = i-1 \\ 0 & j \geq i \end{cases} \tag{5.9}$$

Lemma 5.4.6 *The nearest neighbor for any $x_l \in D$, in set $\{x_1, x_2 \dots x_{l-1}\}$ is x_1.*

Proof For any k such that $2 \leq k < l$, we need to prove that, $||x_l - x_k||^2 - ||x_l - x_1||^2 \geq 0$. We denote Δ^{kl} for vector of absolute difference between co-ordinates of x_k and x_l. For any two elements $x_k, x_l \in D$, where $k, l > 1$, we can obtain Δ^{kl} as follows

$$\begin{aligned} x_k &= (\underbrace{\frac{w}{2}, \frac{w}{3} \cdots \frac{w}{k-1}}_{k-2 \text{ terms}}, \quad w \quad , 0, 0 \dots 0) \\ x_l &= (\underbrace{\frac{w}{2}, \frac{w}{3} \cdots \frac{w}{k-1}}_{k-2 \text{ terms}}, \quad \frac{w}{k} \quad , \underbrace{\frac{w}{k+1}, \frac{w}{k+2} \cdots \frac{w}{l-1}}_{l-k-1 \text{ terms}}, w, 0, 0 \dots 0) \\ \Delta^{kl} &= (\underbrace{0, 0 \dots\dots\dots 0}_{k-2 \text{ zeros}}, \quad \frac{(k-1)w}{k} \quad , \underbrace{\frac{w}{k+1}, \frac{w}{k+2} \cdots \frac{w}{l-1}}_{l-k-1 \text{ entries}}, w, 0, 0 \dots 0) \end{aligned} \tag{5.10}$$

From equations in (5.10), for any $2 \leq k < l$:

$$||x_l - x_k||^2 - ||x_l - x_1||^2 = ||\Delta^{kl}||^2 - ||x_l||^2 = \Big(\frac{k-2}{k} - \sum_{j=2}^{k-1} \frac{1}{j^2}\Big) w^2 = f(k) w^2 \tag{5.11}$$

where $f(k) = (\frac{k-2}{k} - \sum_{j=2}^{k-1} \frac{1}{j^2})$. It can be easily verified that

$$f(k) - f(k-1) = \frac{k-2}{k(k-1)^2} \geq 0 \quad \text{for all } k \geq 2 \tag{5.12}$$

which implies that $f(k)$ is increasing function of k. Thus, the minimum value of $f(k)$ is attained when $k = 2$ which is $f(2) = 0$. Hence we can say that $f(k) \geq 0$. Substituting this in equation (5.11) we get

$$||x_l - x_k||^2 - ||x_l - x_1||^2 = f(k) w^2 \geq 0 \tag{5.13}$$

(PROVED)

5.4.2.2 MSC with uniform kernel on D

MSC with uniform kernel shifts the mean vector in each iteration to the average of the set of training vectors visible from the current mean vector.

Lemma 5.4.7 *If we execute MSC iterations with uniform kernel of support size w on $x_1 \in D$ by starting the first iteration with $\mu_0 = x_1$, then, at the beginning of k^{th} iteration, the set of training vectors visible from the mean μ_{k-1} will be $V_k = \{x_1, x_2 \dots x_{k+1}\}$.*

Since the uniform kernel takes the average of the visible training vectors, we get the following corollary directly by taking average of first $k+1$ training vectors:

Corollary 5.4.8 *μ_k, the mean vector at the end of the iteration will be $(\frac{w}{2}, \frac{w}{3}, \frac{w}{4}, \dots \frac{w}{k}, 0, 0 \dots 0)$ i.e. :*

$$\mu_{kj} = \begin{cases} \frac{w}{j+1} & 1 \leq j \leq k \\ 0 & j > k \end{cases} \tag{5.14}$$

Proof We apply mathematical induction on the iteration k to prove lemma 5.4.7. It is easy to verify that the statement is true for $k = 1$, since $\mu_0 = x_1$ and, only x_1 and x_2 are visible from μ_0. We assume that the statement is true for $k = m-1$ i.e. at the beginning of $(m-1)^{th}$ iteration, $V_{m-1} = \{x_1, x_2 \dots x_m\}$. Hence the mean vector at the beginning of m^{th} iteration is the mean of V_{m-1} i.e. $\mu_{m-1} = (\frac{w}{2}, \frac{w}{3}, \frac{w}{4}, \dots \frac{w}{m}, 0, 0 \dots 0)$. We have to show that $V_m = \{x_1, x_2 \dots x_{m+1}\}$ i.e. for any $l = 1 \dots m+1$ the distance $||\mu_{m-1} - x_l||$ is less than or equal to w and for any $l \geq m+2$, the distance $||\mu_{m-1} - x_l||$ is greater than w. For $l \geq m+2$, it is trivial, since x_l will have at least two additional non-zero co-ordinates where μ_{m-1} is zero and one of those co-ordinates in x_l will have value w. For $l = m+1$:

$$\begin{aligned} ||x_{m+1} - \mu_{m-1}||^2 &= ||(\frac{w}{2}, \frac{w}{3}, \frac{w}{4}, \dots \frac{w}{m}, w, 0, 0 \dots 0) - \\ &\qquad (\frac{w}{2}, \frac{w}{3}, \frac{w}{4}, \dots \frac{w}{m}, 0, 0 \dots 0)||^2 \\ &= w^2 \end{aligned}$$

For $l \leq m$:

$$\begin{aligned} ||x_l - \mu_{m-1}||^2 &= ||(\frac{w}{2}, \frac{w}{3}, \frac{w}{4}, \dots \frac{w}{l-1}, \quad w \quad , 0, 0 \dots 0) - \\ &\qquad (\frac{w}{2}, \frac{w}{3}, \frac{w}{4}, \dots \frac{w}{l-1}, \quad \frac{w}{l} \quad , \frac{w}{l+1} \dots \frac{w}{m}, 0, 0 \dots 0)||^2 \\ &= \Big(\frac{(l-1)^2}{l^2} + \sum_{k=l+1}^{m} \frac{1}{k^2}\Big) w^2 \\ &= f(l) w^2 \end{aligned} \tag{5.15}$$

where $f(l) = (\frac{(l-1)^2}{l^2} + \sum_{k=l+1}^{m} \frac{1}{k^2})$. It is easy to verify that

$$f(l+1) - f(l) = \frac{l-1}{l^2(l+1)} \geq 0 \qquad \text{for all } l \geq 1$$

This shows that $f(l)$ is increasing function of l, attaining its maximum $\frac{(m-1)^2}{m^2}$ when $l = m$. Substituting this in equation (5.15)

$$||x_l - \mu_{m-1}||^2 = f(l) w^2 \leq \frac{(m-1)^2}{m^2} w^2 < w^2$$

(PROVED)

5.4.2.3 Proof for $R \geq \frac{\pi}{\sqrt{6}}$

From lemma 5.4.7, it is clear that the set of training vectors visible from the mean keeps on growing by adding one element in each iteration. The only vector visible from μ_k and not a

member of V_k is x_{k+2}. From lemma 5.4.6, the closest neighbor to x_{k+2} in V_k will be x_1. Hence the distance of the newly visible training vector from the currently visible training vectors is :

$$||x_{k+2} - x_1||^2 = ||x_{k+2}||^2 = \Big(\sum_{j=1}^{k+1} \frac{1}{j^2}\Big) w^2 \tag{5.16}$$

which according to equation 5.8, can be made arbitrarily close to $\frac{\pi^2}{6}w^2$ by choosing large k on a training set D with large n . (PROVED)

5.5 Implementation details

Let $T = \{P_1, P_2, \ldots, P_{ntc}\}$ be the partitions (i.e. clusters) of $D = \{x_1, x_2 \ldots x_n\}$, obtained through transitive closure based clustering (TC) using matching threshold $\sqrt{2}w$. Let c_i be the number of vectors in P_i and the partitions are indexed in the decreasing order of their size i.e. $i < j \implies c_i \geq c_j$. P_1 is the largest partition with c_1 members. Based on the proof presented in section 5.4.1, we can perform MSC on D with range w by performing mean-shift on each P_i independently. If size of each P_i is significantly smaller than that of D, then the range search within each P_i will be faster when compared to that with whole D. Hence, we can say that if $c_1 << n$ then TC partitions can be used to accelerate MSC.

We refer to the MSC algorithm which performs range search with whole D in each iteration as SMSC (Standard MSC). The algorithm which uses TC partitions to perform MSC is referred to as TCMSC. In our experiments we compare the time required to perform SMSC and TCMSC. In our implementation of TCMSC, we perform TC on D during the first iteration of MSC by performing range search with range $\sqrt{2}w$ instead of w. During subsequent iterations, for a vector $x_j \in P_i$ we need to search only in P_i for neighbors within range w from the mean vector corresponding to x_j.

Our acceleration strategy is useful only when the size of each partition P_i is significantly smaller compared to n. It requires that the different clusters in D are well separated from each other so that the transitive closure operation on range based matching (i.e. reachability criteria) can provide partitions in which each partition is significantly smaller than D. When the data are well separated, fast range searching techniques can also increase the efficiency of MSC by employing a pre-processing step in which the training vectors are arranged in a tree structure. Different subtrees of this structure hold vectors belonging to separate regions. This arrangement will help to confine the range search to a limited region around the query vector. In our experiments we would like to see whether TC partitions can further accelerate SMSC which employs a fast range searching technique for performing range search in each iteration. We use [79] for fast range search in SMSC and TCMSC. We perform exact range search without using approximation option available in [79]. When using fast range search methods, the cost of first mean-shift iteration in TCMSC in which we use range $\sqrt{2}w$ (which is larger than w) may increase. In order to obtain acceleration through TCMSC, this additional cost during the first iteration should be compensated by the faster range search in subsequent iterations due to the partitioning of D. Hence, in addition to the size of TC partitions, the advantage of TCMSC depends also on the subsequent number of iterations needed for convergence.

5.5.1 Experiments with gaussian MSC on MPG data

Table 5.1 gives the details of running gaussian TCMSC and SMSC on SIFT descriptors extracted from TD1 to TD6 of MPG dataset (section 3.5.1). Left most column indicates training data. Each subsequent column corresponds to a σ parameter value of the gaussian kernel. We use values $\{30, \ldots, 110\}$. The support range value w, termination criteria ϵ_1 corresponding to each σ are chosen as explained in section 3.6.1. For each dataset there are four rows. The first row shows (in bold faced letters) the ratio of CPU time utilized to perform SMSC and TCMSC. Whenever SMSC takes more time than TCMSC this value is greater than 1 and vice-versa. The second row shows $\frac{c_1}{n} \times 100$ i.e. the percentage of training vectors in the largest TC partition. The third row shows the ratio of CPU time utilized by TCMSC and SMSC during the first mean-shift iteration. This is always greater than 1 because TCMSC needs more time to perform range search in the first iteration using threshold $\sqrt{2}w$ which is larger than w used in SMSC iterations. For example, if its value is 2, then it indicates that the first mean-shift iteration of TCMSC needs nearly the time of two SMSC iterations. The fourth row shows the average number of mean-shift iterations performed for a training vector.

As σ increases, the range value w increases. This increases the number of vectors getting merged into a cluster through transitive closure operation on range based matching. Hence, as σ increases, TC produces partitions with large number of elements. Therefore, the value in the second row (the percentage of training vectors in the largest TC partition) corresponding to a dataset increases as we move from left to right. In most cases, as σ increases, the ratio of CPU time (third row of each data) needed to perform a range search with threshold $\sqrt{2}w$ and w seems to be moving closer to 1. For example, in the third row of TD4, this value decreases from 3.11 to 1.37 from the first to the second last column and increases slightly to 1.58 in the last column. Average number of mean-shift iterations (fourth row of each data) usually increases with σ because larger range will involve vectors far from the current mean vector in mean-shift update. The acceleration provided by TCMSC will be high when the size of the partitions (second row of each data) is less and the average mean-shift iterations (fourth row of each data) is high. For low value of σ (eg. $\sigma = 30$) we do not obtain much acceleration through TCMSC because the mean-shift iterations are not sufficient to compensate for the additional cost incurred during the first TCMSC iteration. As σ takes higher values the TC partitions (second row of each data) grow and eventually the whole D is merged into a single partition. We obtain highest acceleration through TCMSC for σ in the rage of 40 to 60 (best is 4.82 on TD4 for $\sigma = 50$. At worst case TCMSC seems to be 5% slower than SMSC (last column of TD3).

5.6 Conclusions

We presented a strategy for accelerating mean-shift operation without using any approximation. The speedup of the strategy depends on three different factors (i)size of TC partitions, (ii) additional cost incurred during the first iteration of TCMSC and (iii) number of mean-shift iterations needed for convergence. Our experiments show that for gaussian kernel we obtain acceleration up to 4 times for σ value near 50. In the worst case the speed reduced by 5%. Our results are comparable to that in [20] (acceleration between 1.5 to 3 most of the time). In contrast the method in [34] provides a speed up by a factor close to 1000. But it involves subsampling D by a factor of 1024 before beginning MSC. For 3D point representation in which a 3D point associated with features in few images (nearly 20 in our case) such a huge subsampling is not reasonable. In [122] a speed up by a factor close to 30 is achieved by eliminating the

	30	40	50	60	70	80	90	100	110
TD1	**1.48**	**2.99**	**2.93**	**1.64**	**1.16**	**0.99**	**0.99**	**0.98**	**0.96**
	1.36	9.60	36.14	69.69	89.30	97.52	99.76	99.99	100.00
	3.00	3.11	2.58	2.07	1.74	1.52	1.43	1.50	2.10
	5.78	10.60	14.48	17.15	18.75	19.48	19.89	20.96	21.15
TD2	**2.00**	**3.76**	**3.34**	**1.80**	**1.19**	**1.01**	**0.98**	**0.98**	**0.95**
	0.90	10.88	38.02	68.26	88.72	97.86	99.82	100.00	100.00
	3.03	3.21	2.68	2.13	1.77	1.55	1.44	1.49	2.07
	7.75	13.96	18.24	20.14	20.89	20.30	19.94	20.08	20.94
TD3	**1.82**	**3.34**	**4.57**	**2.31**	**1.39**	**1.06**	**0.99**	**0.97**	**0.95**
	3.90	7.79	19.34	55.99	79.40	94.08	99.32	99.99	100.00
	3.07	3.11	2.49	1.97	1.67	1.49	1.40	1.43	1.74
	7.32	12.08	15.22	16.98	16.78	15.59	15.18	15.06	15.16
TD4	**1.28**	**2.67**	**4.82**	**3.89**	**1.79**	**1.17**	**0.99**	**0.98**	**0.97**
	0.59	4.14	14.77	37.63	69.18	89.24	98.80	99.97	100.00
	3.11	3.12	2.42	1.90	1.60	1.45	1.37	1.37	1.58
	5.96	10.27	13.70	16.15	17.16	16.90	16.98	18.05	18.14
TD5	**1.32**	**2.45**	**3.19**	**2.09**	**1.27**	**1.01**	**0.98**	**0.98**	**0.96**
	2.22	12.45	29.35	57.38	83.37	96.33	99.63	100.00	100.00
	3.00	3.23	2.64	2.10	1.74	1.54	1.44	1.51	1.99
	5.19	9.24	12.80	15.14	15.83	16.75	17.17	17.89	18.72
TD6	**1.40**	**2.77**	**4.63**	**3.68**	**1.92**	**1.10**	**0.98**	**0.97**	**0.96**
	0.29	2.76	14.08	36.59	66.20	91.74	99.44	100.00	100.00
	3.07	3.24	2.61	2.00	1.65	1.46	1.38	1.39	1.72
	5.58	9.63	12.34	13.60	13.90	13.77	13.71	13.62	14.23

Table 5.1: Comparing computational cost of gaussian TCMSC and SMSC. Left most column indicates the name of the dataset. Each subsequent column corresponds to a σ parameter of the gaussian kernel. For each dataset there are four rows. The first of these four rows shows the ratio of CPU time utilized to perform SMSC and TCMSC. If the ratio is greater than 1, then TCMSC is faster than SMSC and vice-versa. The second of the four rows shows the percentage of training vectors in the largest TC partition. The third row shows the ratio of CPU time utilized by TCMSC and SMSC during the first mean-shift iteration. The last of the four rows shows the average number of mean-shift iterations performed for a training vector.

training vectors which are likely to converge to the same mode. But the results are shown on a dataset containing more than 300 samples per class. During our experiments on pose estimation we obtained better pose results for σ values above 50, for which the TCMSC acceleration is moderate. As future work we would like to see whether we can use our strategy for performing SfM through hierarchical mean-shift in which clustering is initiated with low σ values.

Chapter 6

Features from simulated views through ASIFT

As pose difference between the test image and the images in the training set increases, it becomes difficult to identify a sufficient number of 3D points in the test image for pose estimation. If new views of the environment can be simulated from one of its view then such techniques can be used to generate new views of the environment from training and test images. Some of the views simulated from training images may be closer to that of the test image and vice-versa. In this chapter we present the experiments in which we attempt to use features from simulated views in order to improve 2D-to-3D point matches under large pose variation between test and training images. For all the experiments in this chapter we use ATC based visual words (section 3.4) for performing SfM.

6.1 Background

Single image of a planar object contains all the parts of the object that are visible from different views. Hence, a different view of a planar object can be simulated by applying the appropriate transformation (usually homography or affine) to one of its views. Locally planar assumption on the shape of the target environment provides scope to achieve robustness against change of viewpoint. This can be done in two ways : (a)Surface normal based warping, (b)View simulation.

Surface normal based techniques try to estimate the local surface normal and generate the desired view of the features to be matched to improve matching accuracy. In [118], dense textured 3D model of the environment is used to enhance the view invariance of the SIFT descriptors by normalizing the image patches. The dominant planes are computed in the dense 3D model. Then, for each dominant plane, texture is projected into an orthographic camera with viewing direction parallel to the plane's normal. SIFT descriptors extracted from these orthographic patch projections are used to match two different 3D models of a scene. In [76], dynamically estimated camera pose and 3D position of the tracked features under a SLAM model are used to refine the surface normal from an initial estimate at each feature positions by successively warping the image patches around it through homography. [21] uses similar warping strategy on training images for which the camera positions are computed in an offline training step. In this case the camera pose values are more reliable due to which the estimated surface normals are more accurate.

Second approach is to obtain feature descriptors from additional views simulated from available images. Affine transformations (ASIFT[78], Ferns[89]) are widely used to generate new views from the existing views. The parameters of the affine transformation are generated assuming the frontoparallel view of the locally planar surface and ignoring perspectivity[66]. We adopt view simulation based approach because unlike surface normal warping it does not rely on the accuracy of the estimated surface normal.

6.1.1 Overview

In section 6.2 we give a brief description of the process of view simulation in Ferns and ASIFT and justify our choice of using ASIFT over Ferns through an experiment. Section 6.3 presents the different approaches used in our experiments to incorporate features from simulated views into 3D map. In section 6.4 we describe the experimental framework for evaluating the performance of these different approaches. Section 6.5 presents the result of this evaluation and the conclusions are presented in 6.6.

6.2 SIFT vs Ferns

6.2.1 View simulation in Random Ferns[89]

The random ferns method generates random affine transformations by

$$A = R_\theta R_{-\phi} diag(\lambda_1, \lambda_2) R_\phi \tag{6.1}$$

where the $diag(\lambda_1, \lambda_2)$ is a 2×2 diagonal matrix having λ_1, λ_2 as its diagonal values, R_α represents 2D rotation by an angle α. The values of θ, ϕ, λ are randomly chosen from the range $[0 : 2\pi], [0 : \pi], [.5 : 1.5]$ respectively. User specified number of synthetic images are generated iteratively by applying a random affine transformation on the original training image in each iteration. Interest points that are repeatedly detected in multiple simulations are retained.

6.2.2 View simulation in ASIFT[78]

Instead of randomly generating a huge number of affine transformations for view simulation, ASIFT method systematically samples the *view-hemisphere* as depicted in the figure 6.1 (It is exact reproduction of figure 14 of [78]). SIFT descriptors [69] are extracted from each simulated image. The simulations are modeled as a rotation ϕ caused by longitude and a tilt $t = |\frac{1}{cos\theta}|$ caused by the latitude θ. ASIFT does not simulate in-plane rotation and scale change since the SIFT feature extraction process itself is invariant to it. Hence the affine mapping is generated by

$$\begin{pmatrix} t & 0 \\ 0 & 1 \end{pmatrix} R_\phi \tag{6.2}$$

Where R_ϕ is the rotation matrix corresponding to angle ϕ. Images are generated for $t \in \{1, \sqrt{2}, 2, ..., \sqrt{2}^n\}$ and $\phi \in \{0, b/t, ..., kb/t\}$ with $b = 72^o$ and $k = \lfloor t/b.180^o \rfloor$.

Representing a keypoint in Random Ferns involves storing the distribution of values of a large number of binary functions computed from image patch around the keypoint. It occupies a huge amount of memory (nearly 480KB for a keypoint containing 30 ferns each with 12 binary functions) which makes it difficult to represent a 3D map with large number of 3D points. Moreover,

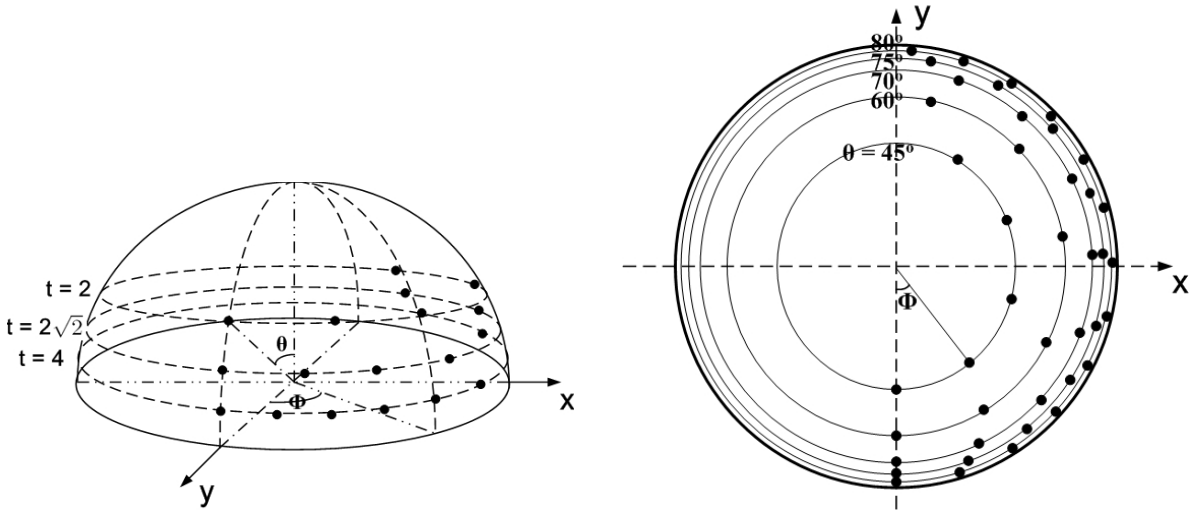


Figure 6.1: (Exact reproduction of figure 14 of [78]) Sampling of the parameters $t = |\frac{1}{cos\theta}|$ and ϕ. The samples are the black dots. Left: perspective illustration of the view hemisphere (only $t = 2, 2\sqrt{2}, 4$ are shown). Right: zenith view of the observation hemisphere. The values of θ are indicated on the figure.

experimentally SIFT descriptors extracted from original image (without any view simulation) have performed better than random fern based descriptors [65] in terms of ability and accuracy of matching. Our own experiments confirm these results. We used the code provided by the authors on MPG dataset to detect planar region in test images using a single training image. We select one image from TD2 and mark the planar region. After training random ferns on this single image we use the images in TD1 as test images. We compute homography from the matches computed on test images using the learned keypoints in the train image. Using the homography we demarcate the planar region in test images. We repeat the same experiment with SIFT features (without using view simulation) by matching test images with the train image using standard SIFT keypoint matching with threshold 0.6. The results are evaluated manually looking at the demarcated planar region in the test image. Figure 6.2 shows the training image, and one example for each case of correct detection and wrong detection. The detection count for SIFT and Fern based methods is listed in table 6.1. The result clearly indicates that SIFT descriptors are more robust compared to Ferns. Hence in our work we use ASIFT framework to obtain features from simulated views.

6.3 Incorporating features from simulated views into 3D map

We explore different ways of incorporating the descriptors from simulated views into 3D map of the environment. A straight forward way is to extract ASIFT features (which consist of SIFT features extracted from simulated images along with those from the original image) from each training image and build the 3D map as described in chapter 3. This straight forward method (referred to as Method1) is described in section 6.3.1. Through ASIFT descriptors one would expect to enrich the visual representation of 3D points by having additional SIFT descriptors from simulated views. In our experiments we observe that the SIFT features extracted from simulated views more often produce new 3D points than providing additional descriptors to the 3D points computed from the original images. Hence, we design additional strategies to add

(a) (b) (c)

Figure 6.2: (a) Training image chosen from TD1 and the marked planar region. (b) Correct detection (c) Wrong detection

Method	Detected Correctly	Wrong Detection
Fern $N = 100$	3	114
Fern $N = 200$	6	111
Fern $N = 400$	6	111
Fern $N = 800$	3	114
Fern $N = 1600$	5	112
SIFT	70	47

Table 6.1: Detection counts for the planar region on 117 images of TD1 using one of the images from TD2 for training (Please refer to figure 6.2) . The first column shows the method used (Fern[89] or SIFT[69]) for extracting features. For Fern method we have to specify N the number of features to be extracted from an image. The two subsequent columns show the number of Correct and Wrong detections respectively.

SIFT descriptors for simulated views to the 3D map computed from original images. We first build the 3D map through SfM on 2D-to-2D matches obtained from SIFT features extracted from original training images. Then, for each 3D point we try to add SIFT descriptors from views simulated from training images. We perform this post-SfM addition of descriptors in three different ways referred to as Method2, Method3 and Method4. They are presented in section 6.3.2. Throughout our presentation, the '2D location of a SIFT descriptor' refers to its 2D coordinates in the original image, irrespective of whether the descriptor is extracted from the original or a simulated image.

6.3.1 Method1 : Using ASIFT features during SfM

We build the 3D map as described in chapter 3 using the ATC clusters (section 3.4) on ASIFT descriptors extracted from training images. The pruning step presented in section 3.3 discards the clusters having multiple instances in a single training image. The set of ASIFT features may contain multiple descriptors of a 3D point in a single training image due to view simulation. Hence, we make a minor modification in the pruning step. Suppose a cluster contains multiple SIFT descriptors from a single training image I. Let $\{q_1, ..., q_k\}$ be the 2D locations of those descriptors in I. We compute the maximum Euclidean distance d_{max} between the 2D locations $\{q_1, ..., q_k\}$. If d_{max} is greater than one pixel, then the cluster is discarded. Otherwise it is used to generate 2D-to-2D matches. The 3D map thus obtained will contain the descriptors from simulated views.

6.3.2 Post-SfM addition of simulated features to 3D map

For each 3D point in the 3D map computed from original training images through ATC based clustering, we associate SIFT descriptors from simulated views in the following three different ways.

6.3.2.1 Method2 : Adding descriptors based on pixel distance

If the 2D locations of two different SIFT descriptors are sufficiently close to each other then we can say that they belong to the same 3D point. In Method2, SIFT descriptors from simulated views of training images are added to the 3D map based on the proximity of their 2D locations to that of the descriptors from original images which are associated with 3D map. Let q be the 2D location of a SIFT descriptor associated with a 3D point Q in one of the training images I. We extract SIFT features from simulated views generated from I. All the SIFT features from simulated views whose 2D locations when back projected to the original image, lie within a distance of 1 pixel from q are associated with Q.

6.3.2.2 Method3 : Computing descriptors from a given location

The 2D location of a SIFT descriptor associated with a 3D point can be mapped to a simulated view using the affine transformation used to generate the view. In Method3, we try to compute additional SIFT descriptors for each 3D point from these mapped 2D locations in simulated views. In addition to the 2D location we have to specify the scale at which the descriptor need to be computed. We use the same scale as the original descriptor. That is, if a 3D point Q is associated with SIFT descriptor extracted from a 2D location q and scale s of a training image I, then from each simulated view of I we compute SIFT descriptor from the location corresponding to q at scale s and associate it with Q.

6.3.2.3 Method4 : Choosing the best scale for a given location

Method4 is similar to Method3 except that we try to determine the scale at which we want to compute the descriptor from a given location q in the simulated image. The values in the different layers of the DoG (Difference of Gaussian) pyramid (section 2.1.1) of the simulated view is searched along the location corresponding to q. If the value of DoG at the lowest layer of the pyramid is -ve, then we look for the layer in the pyramid which has the lowest value for DoG at q. If the value of DoG at the lowest layer of the pyramid is +ve, then we look for the layer in the pyramid which has the highest value for DoG at q. If the extremum value occurs in the bottom most or the top most layer, then we do not compute any descriptor at q. Otherwise, SIFT descriptor is computed at the scale of the layer at which extremum occurs.

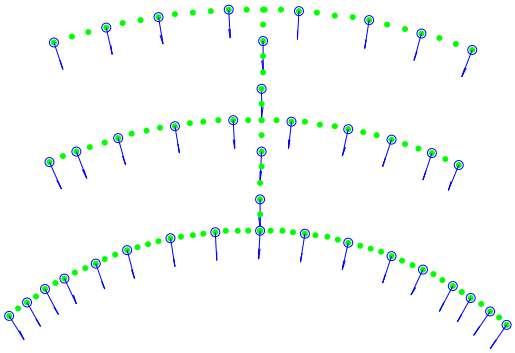
6.4 Experimental framework

We want to compare the quality of 2D-to-3D matches in a test image obtained through a 3D map with and without having features from simulated views. We mainly use the reprojection error corresponding to 2D-to-3D matches to measure the quality. There are many factors other than the usage of simulated views which influence this measure. For example, ATC needs smooth variation of pose in the training images in order to be able to establish 2D-to-2D matches, the accuracy of the coordinates of 3D points depends on the length of the baseline available in the training images. In order to reduce such influences we design our experiments in the following way.

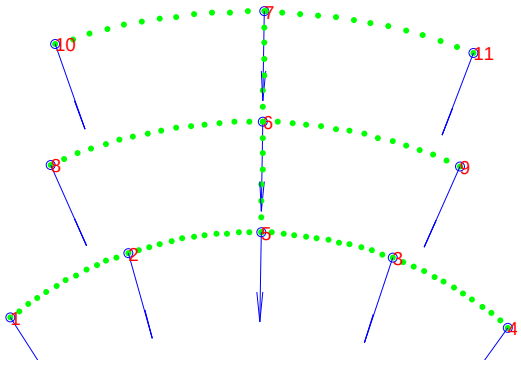
We select a set of test images and different sets of training images from RoboImage[7] (described in section 3.5.2) dataset. Figure 6.3.b shows the positions of images included in the test set RITest. Figures (c) to (f) of 6.3 show the camera positions of the images in four different training sets RITrain1 to RITrain4. We can see that the change of camera pose in the training images (especially in RITrain3) is not always smooth. Hence we choose a super train set (Figure 6.3.a) in which the pose variation is smooth. It contains all the images in the training sets RITrain1 to RITrain4. We build 3D map through SfM on super train set. The coordinates of each 3D point in the 3D map is recomputed by performing triangulation using the ground truth camera positions of the super train set. 3D points having mean reprojection error higher than 1 pixel are discarded. Using this process we hope to obtain a good quality super 3D map in the ground truth 3D coordinate system. For each training set we obtain its 3D map by choosing the 3D points from the super 3D map which are associated with its descriptors. Since these 3D maps are in ground truth 3D coordinate system and we have ground truth camera pose information of test images, we can compute the reprojection error for each 2D-to-3D match in a test image.

6.5 Results

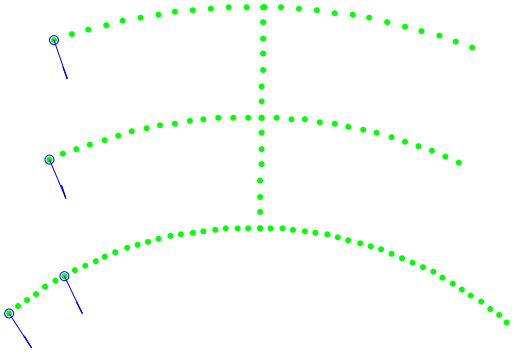
For generating simulated views we use only one additional tilt i.e. $t \in \{1, \sqrt{2}\}$ in equation 6.2. Using more tilts will increase the computational cost of extracting and matching SIFT descriptors. For $t = \sqrt{2}$ four simulated images are generated corresponding to different longitudes. Without using simulation we obtain around 600 SIFT keypoints per image. With simulation we obtain nearly 2000 to 2500 keypoints per image. After computing the 3D map and adding features from simulated views in different ways as described in the section 6.3, we match the feature descriptors from test images with D^{3D} i.e. the training feature descriptors associated with 3D



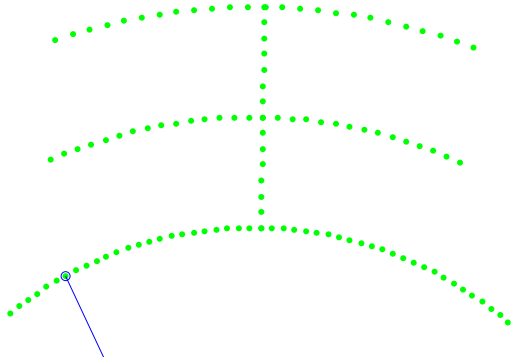
(a) Super train set, 39 images



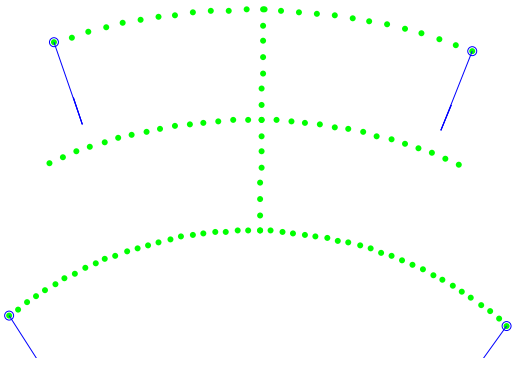
(b) RITest, 11 images



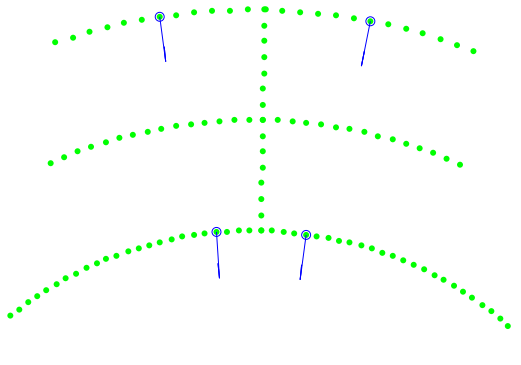
(c) RITrain1, 4 images



(d) RITrain2, 2 images



(e) RITrain3, 4 images



(f) RITrain4, 4 images

Figure 6.3: Images selected from RoboImage dataset for experiments in this chapter. We select 6 different subsets shown in the 6 plots above. Each plot shows the camera position of 119 images in green dots. The camera positions of the selected images of a subset are indicated by arrows in the direction of the orientation of the camera. Plot (a) shows the subset camera positions used to build the 3D map during training. Plot (b) shows the subset used as test set. The test images are numbered from 1 to 11. The rest, i.e, plots (c) to (f) are different training sets. RITrain1 and RITrain2 provide views only from one extreme end. RITrain2 contains only one image from the front row which leads to a huge scale variation with test images in the last row. RITrain3 provides views from both sides but middle views are missing. RITrain4 provides only the views from the middle portion. Super train set in plot (a) contains all the images in the 4 training sets. It is used to obtain a reliable 3D map which is projected on the different training sets based on the subset image positions.

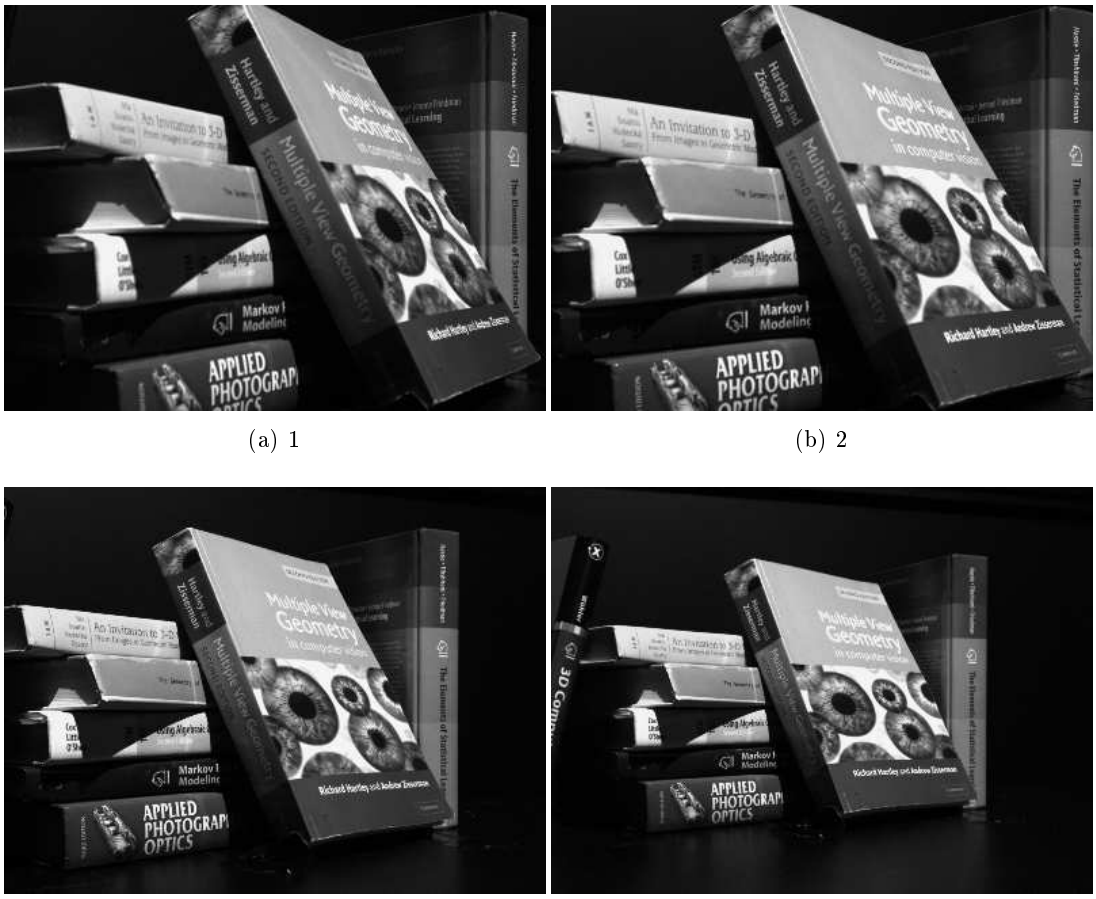
points. We apply a threshold of 0.8 on the ratio of distances with the two closest training feature descriptors associated with two different 3D points (this matching scheme is same as NNR with D^{3D} presented in section 4.2). We compute pose using EPnP with 500 RANSAC iterations. A 2D-to-3D match with reprojection error less than 1 pixel with respect to the ground truth pose is considered as inlier.

The results for each test image (numbered 1 to 11 in figure 6.3(b)) is shown separately in tables 6.2 to 6.12 respectively. In each table, the first column indicates the training set used to obtain the 3D map. The subsequent columns are divided into two parts each containing 5 columns. For the first part we do not use simulated views on the test images. For the second part we use $t \in \{1, \sqrt{2}\}$ to extract features through ASIFT. In each part M0 corresponds to the case in which only SIFT features from the original training images are used during training. M1 to M4 correspond to 4 different ways of using simulated features in 3D map as described in section 6.3.

In each table, there are 4 rows corresponding to each training set. The first three rows indicate the quality of the 2D-to-3D matches obtained for the test image. The first row shows the number of inlying 2D-to-3D matches in the test image with respect to the ground truth pose of the camera. Second row shows the % of these inliers out of the total 2D-to-3D matches obtained for the image. Third row shows the spread (in % computed as described in section 4.2.3.1) of 2D locations of inlying 2D-to-3D matches in the image. The fourth row shows the accuracy of the pose computation in terms of the error of estimated camera position from the ground truth. Let C_g and C_e be the ground truth and estimated camera position for the test image corresponding to the table. Let Q_μ be the mean of the 3D coordinates of the 3D points available in the ground truth information of the RoboImage data. The error value for the fourth row corresponding to each training set is computed as $\frac{||C_g - C_e||_2}{||C_g - Q_\mu||_2} \times 100$.

Method1 to Method4 increase the number of descriptors associated with the 3D map by a ratio of 3, 1.7, 5 and 1.6 respectively on average over different training datasets. From the tables 6.2 to 6.12 we can see that method M0 provides reasonably good pose accuracy without using view simulation on test images. Except for test image 11 (table 6.12) with RITrain2 (for which the pose error is 40.8%) the pose error is less than 6%. The spread of inlying 2D-to-3D matches is also more or less similar across different methods (figure 6.5 and 6.6 illustrate this). Hence, in most cases we have to rely on the number of inlying 2D-to-3D matches for comparing the performance of different methods. We compare the performance of M1 to M4 with M0.

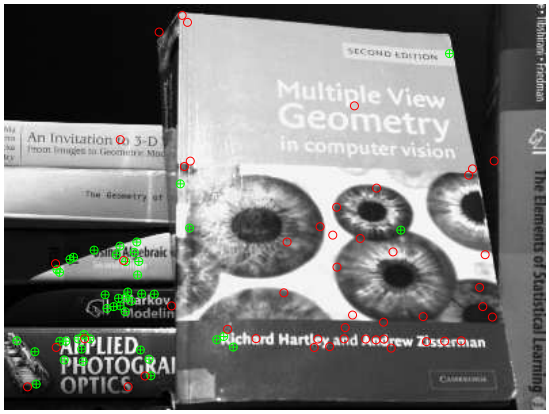
In most cases Method1 reduces the number of inlying 2D-to-3D matches for a test image when view simulation is not performed on the test image. This point is evident if we compare the first row of two columns M0 and M1 (in the left half portion of each table) corresponding to each training set. For example it reduces from 277 to 262 for test image 1 with RITrain1 (table 6.2). The percentage of inliers also decrease in these cases. There are few exceptions (eg. for test image 6, 7, 9, 10 and 11 with RITrain2 in tables 6.7, 6.8, 6.10, 6.11 and 6.12 respectively) where there is a marginal increment in the number of inlying 2D-to-3D matches. This suggests that the additional descriptors in the 3D map obtained through M1 are creating ambiguity for matching with test SIFT descriptors from original test images. Only when the total number of available features in the 3D map are very small (eg: for RITrain2 which has features from only one training image) M1 increases the number of matches. When view simulation is not performed on test images M2, M3 and M4 provide more or less the same number of inlying 2D-to-3D matches and



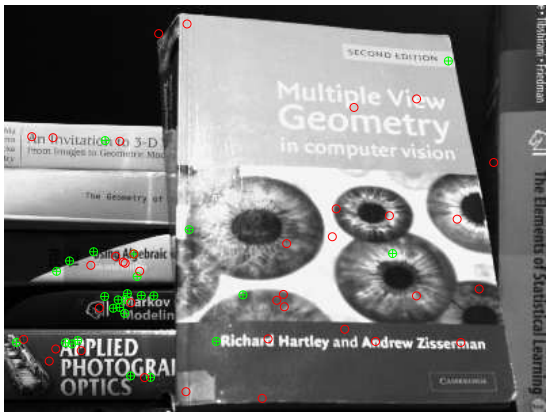
(a) 1 (b) 2

(c) 3 (d) 4

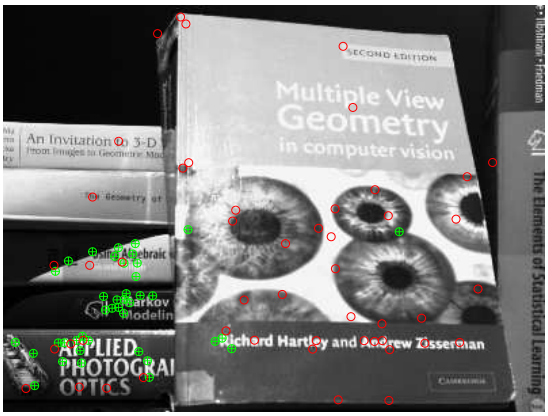
Figure 6.4: The four training images in RITrain1. Image number 2 is the only training image used in RITrain2.



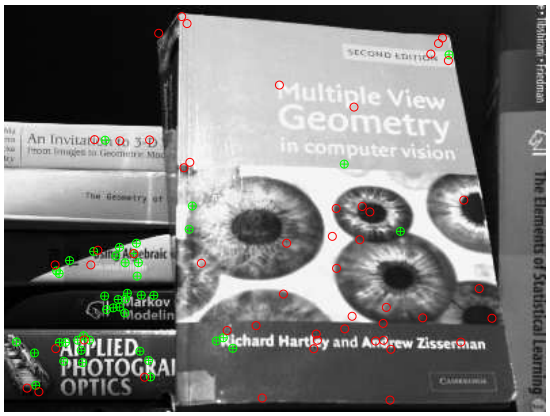
(a) M0 In=45 Out=50



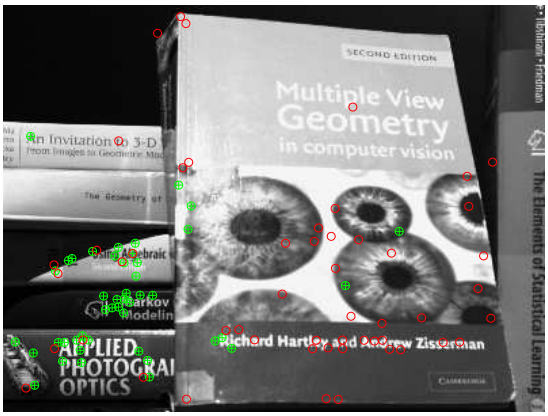
(b) M1 In=26 Out=37



(c) M2 In=41 Out=51

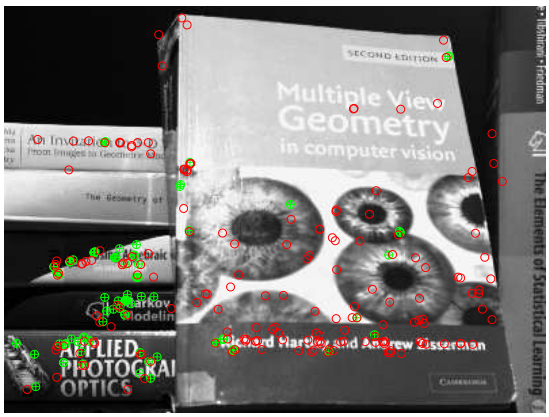


(d) M3 In=46 Out=51

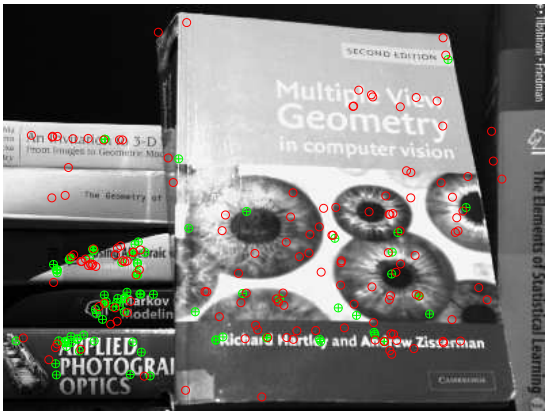


(e) M4 In=46 Out=54

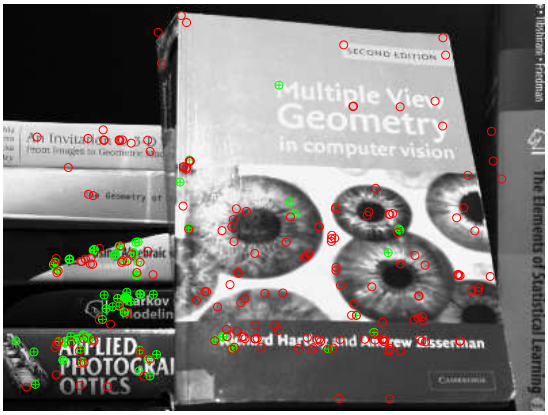
Figure 6.5: 2D locations of the 2D-to-3D matches for test image 3 with RITrain1 (the four images in this training set are shown in figure 6.4)when view simulation is not performed on the test image. The green '⊕' and red 'o' marks indicate inliers and outliers respectively with respect to ground truth camera pose. Each subfigure corresponds to method M0 to M4 (The 5 columns of the left portion corresponding to RITrain1 in table 6.4). The number of inliers and outliers are indicated along with each figure. We can see that Except for the one green '⊕' mark near the left top portion, M1 to M4 provide inliers from the same region as M0.



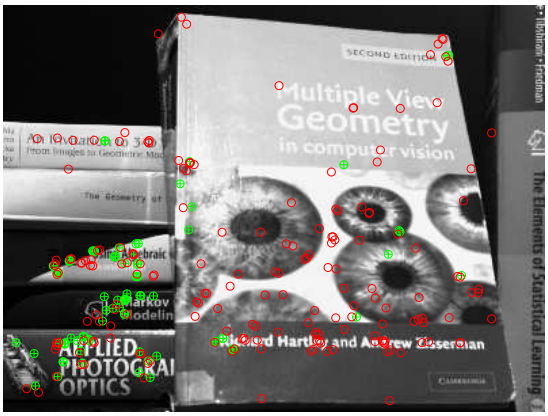
(a) M0 In=90 Out=186



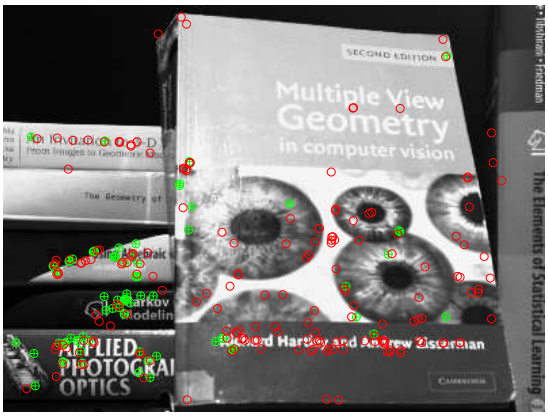
(b) M1 In=97 Out=149



(c) M2 In=90 Out=201



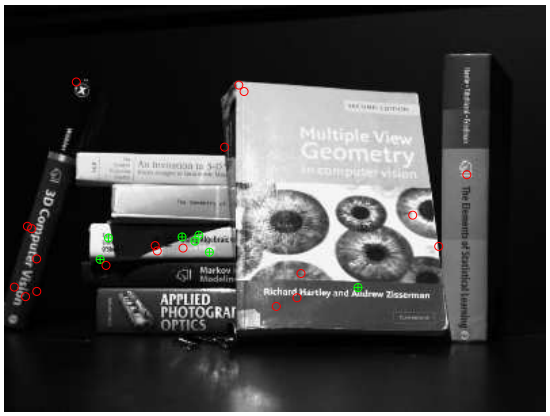
(d) M3 In=91 Out=193



(e) M4 In=84 Out=193

Figure 6.6: 2D locations of the 2D-to-3D matches for test image 3 with RITrain1 when view simulation is performed on the test image. The green '⊕' and red 'o' marks indicate inliers and outliers respectively with respect to ground truth camera pose. Each subfigure corresponds to method M0 to M4 (The 5 columns of the right portion corresponding to RITrain1 in table 6.4). The number of inliers and outliers are indicated along with each figure. We can see that Except for the one green '⊕' mark near the left top portion, M1 to M4 provide inliers from the same region as M0. The regions with green '⊕' mark seems to be same even if we compare with M0 of figure 6.5 in which view simulation is not performed on the test image.

(a) M0 In=2 Out=35



(b) M1 In=7 Out=20

(c) M2 In=7 Out=29

(d) M3 In=2 Out=32

(e) M4 In=2 Out=35

Figure 6.7: 2D locations of the 2D-to-3D matches for test image 11 with RITrain2 when view simulation is not performed on the test image. The only training image in RITrain2 is shown in subfigure (b) of figure 6.4. The green '$\oplus$' and red 'o' marks indicate inliers and outliers respectively with respect to ground truth camera pose.

(a) M0 In=13 Out=115

(b) M1 In=33 Out=94

(c) M2 In=18 Out=101

(d) M3 In=18 Out=107

(e) M4 In=13 Out=119

Figure 6.8: 2D locations of the 2D-to-3D matches for test image 11 with RITrain2 when view simulation is performed on the test image. The only training image in RITrain2 is shown in sub-figure (b) of figure 6.4. The green '⊕' and red 'o' marks indicate inliers and outliers respectively with respect to ground truth camera pose.

pose accuracy.

When view simulation is used on test images, we obtain significant number of additional 2D-to-3D matches in all cases from M0 to M4. This is evident when we compare the left portion with the right portion of each table. M1, in most cases, further increases 2D-to-3D matches and the ratio of inliers from M0. Even with this significant increase in the number of matches the spread of matches over the test image seems to be unchanged (as can be seen in figure 6.6).

When using SIFT features only from the original images (i.e. method M0 without performing view simulation on test image) we obtain highest pose error of 40.8% on test image 11 with RITrain2 (table 6.12). Figure 6.7 and 6.8 show the distribution of the 2D location of the 2D-to-3D matches with and without view simulation on the test image respectively. Method M1 and M2 significantly improve the number of matches in both cases.

The viewing angle of the images in training sets RITrain1 and RITrain2 (figure 6.3) are very close to each other. RITrain2 contains only one image, whereas RITrain1 contains multiple images which cover the view at different scales. This produces huge difference in the number of matches obtained for a test image using RITrain1 and RITrain2. For examle, for test image 9 (table 6.10), the number of inlying 2D-to-3D matches for with RITrain1 is more than twice of that with RITrain2 in most cases. For M0 (i.e. without view simulation on training images) this ratio is nearly 5. Both RITrain1 and RITrain2 provide almost same number of matches for test image 1 (table 6.10) which is captured at a scale close to the image in RITrain1. This indicates the limitation of the ability of SIFT features to handle scale variation.

6.6 Conclusion

In our experiments we could not obtain significant advantage in 2D-to-3D matches and pose computation despite employing various strategies. While performing image to image matching, ASIFT provides significantly more number of matches when compared to using only SIFT descriptors from original image (online demo available [6]). We could not translate these benefits in 3D map representation. The main difference between these two scenarios is that the 3D map contains descriptors from various images and their simulated views. When performing image to image matching, a test descriptor is matched with descriptors extracted from a single view (original or simulated). Hence the ambiguity in the descriptor space is certainly much less while matching two images than matching with 3D map. We may need more counter measures to get rid of the ambiguity created by descriptors from synthetic views[31] before we could use it for 3D point representation.

We need to design better test cases in order to increase the scope of the experiments with view simulation. Each image in the dataset we used covers a large portion of the environment at a reasonably good pixel resolution. Hence it is difficult to produce cases in which pose computation fails with a training set containing more than two images when view simulation is not used. Moreover we used only one scheme of matching which may not be suitable to handle the ambiguities due to huge number of features generated from simulated views. In future we would like to perform experiments with data captured from larger environment with more tilts while generating view simulation and different strategies to match 3D points.

	No view simulation on Test images					Views simulated on Test images				
	M0	M1	M2	M3	M4	M0	M1	M2	M3	M4
RITrain1	277	262	277	292	283	455	888	513	495	467
	81.7	75.1	81.2	78.9	80.2	54.4	75.6	55.1	52.7	52.9
	105.5	99.1	105.1	102.5	103.3	105.2	100.1	104.2	104.0	103.8
	0.1	0.0	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.1
RITrain2	228	215	230	244	229	336	708	393	388	351
	81.4	75.7	81.0	81.3	79.8	54.0	72.3	54.7	53.5	53.0
	98.1	94.3	100.2	100.1	97.4	99.4	102.2	103.2	102.8	97.6
	0.0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.0	0.1
RITrain3	265	249	267	276	271	408	851	473	450	414
	80.1	74.1	78.8	80.7	79.0	53.4	75.1	54.8	52.3	51.8
	106.5	96.3	106.1	104.8	104.7	105.1	100.0	105.1	104.3	103.6
	0.1	0.0	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1
RITrain4	58	48	54	64	60	105	144	112	119	109
	59.2	49.5	50.9	52.5	54.5	38.9	46.3	36.7	36.1	37.7
	83.6	91.7	83.8	87.1	90.1	84.4	96.4	87.2	84.8	93.2
	0.4	0.3	0.4	0.4	0.3	0.3	0.2	0.5	0.9	0.3

Table 6.2: TestIm 1

	No view simulation on Test images					Views simulated on Test images				
	M0	M1	M2	M3	M4	M0	M1	M2	M3	M4
RITrain1	279	223	273	280	279	501	803	533	514	502
	78.4	68.8	74.0	76.5	76.9	51.0	70.1	49.8	49.9	49.1
	100.0	100.9	99.4	99.4	99.7	100.6	103.8	100.1	99.9	100.0
	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.3	0.2
RITrain2	288	229	284	285	287	446	799	502	496	466
	86.0	75.1	85.8	82.4	84.4	54.9	72.3	55.2	52.4	53.0
	101.5	102.0	101.8	100.4	101.7	103.6	104.8	104.8	102.2	104.1
	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.3	0.1	0.3
RITrain3	137	105	140	132	142	289	383	314	294	305
	64.0	54.7	60.6	60.0	64.8	44.8	55.2	43.3	42.0	43.7
	102.1	103.2	101.7	103.1	100.8	101.8	100.8	101.8	100.7	101.4
	0.2	0.1	0.2	0.2	0.2	0.3	0.1	0.2	0.3	0.2
RITrain4	275	237	268	283	264	452	750	493	476	459
	80.2	69.5	77.7	78.2	78.3	54.6	70.0	54.2	52.5	53.6
	98.5	97.9	96.9	98.3	98.9	101.5	99.7	98.4	100.0	100.8
	0.0	0.1	0.1	0.1	0.0	0.1	0.1	0.2	0.2	0.2

Table 6.3: TestIm 2

	No view simulation on Test images					Views simulated on Test images				
	M0	M1	M2	M3	M4	M0	M1	M2	M3	M4
RITrain1	45	26	41	46	46	90	97	90	91	84
	47.4	41.3	44.6	47.4	46.0	32.6	39.4	30.9	32.0	30.3
	41.3	58.2	33.8	60.1	65.2	68.6	71.7	66.1	62.3	65.8
	0.5	0.4	0.7	0.8	0.7	3.3	0.6	0.9	0.9	2.7
RITrain2	44	27	42	44	43	71	90	80	79	75
	48.4	39.7	47.7	51.2	50.6	29.0	38.3	30.8	29.9	31.8
	68.5	46.5	68.5	68.2	68.4	79.0	56.9	79.8	79.3	76.9
	0.7	0.8	0.4	0.5	1.6	5.3	0.7	2.5	2.8	0.7
RITrain3	154	114	156	160	155	255	354	289	276	258
	67.2	58.5	67.8	64.8	67.4	44.7	57.6	46.3	43.1	43.3
	86.7	85.6	86.1	86.0	74.9	85.4	84.7	85.2	85.2	80.0
	0.2	0.1	0.1	0.1	0.3	0.1	0.2	0.1	0.2	0.6
RITrain4	253	200	251	245	253	430	675	464	445	434
	75.5	68.5	75.1	76.1	76.2	53.7	70.2	54.1	51.9	53.0
	91.0	77.7	90.8	91.5	90.6	89.7	78.3	89.1	88.5	89.6
	0.1	0.1	0.0	0.1	0.1	0.3	0.1	0.2	0.1	0.0

Table 6.4: TestIm 3

	No view simulation on Test images					Views simulated on Test images				
	M0	M1	M2	M3	M4	M0	M1	M2	M3	M4
RITrain1	27	16	26	24	28	49	53	54	45	52
	45.8	37.2	39.4	42.9	46.7	27.5	36.8	28.3	26.8	29.5
	58.3	53.6	58.6	54.5	58.1	56.8	56.1	56.9	53.8	57.1
	0.7	0.6	0.2	0.9	0.2	1.4	0.4	0.3	1.2	0.8
RITrain2	26	13	23	24	27	35	44	45	37	38
	47.3	28.3	41.8	46.2	51.9	23.3	29.3	25.7	21.9	24.8
	55.0	51.3	52.7	52.5	55.2	53.9	52.1	53.7	53.6	53.9
	0.2	108.8	0.1	0.4	0.2	0.5	0.4	0.2	0.4	3.1
RITrain3	259	235	264	266	263	363	729	459	422	379
	80.9	75.8	78.6	81.6	80.7	55.7	76.6	57.1	53.8	53.4
	104.4	103.6	104.7	104.7	104.6	104.4	101.9	103.2	103.7	104.2
	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.1
RITrain4	92	60	84	88	87	164	197	170	165	167
	68.7	52.6	63.2	61.1	61.3	45.1	52.0	43.6	40.4	43.0
	61.5	60.8	61.4	87.8	61.3	70.2	69.4	70.2	80.7	70.4
	0.1	0.1	0.1	0.1	0.2	0.2	0.3	0.4	0.3	0.2

Table 6.5: TestIm 4

	No view simulation on Test images					Views simulated on Test images				
	M0	M1	M2	M3	M4	M0	M1	M2	M3	M4
RITrain1	102	67	90	93	93	199	235	204	203	187
	52.0	39.6	47.6	50.3	48.4	34.9	42.3	33.3	34.6	31.6
	97.8	94.5	97.3	96.0	98.2	98.9	94.4	99.9	101.7	99.8
	0.3	1.1	1.3	0.6	0.5	1.2	0.9	2.1	0.3	1.0
RITrain2	92	64	89	90	91	174	238	191	185	182
	60.9	48.5	57.8	58.4	57.6	38.8	48.1	37.8	36.8	37.7
	91.3	71.0	92.2	95.1	93.1	84.0	81.3	83.5	81.6	90.6
	0.2	1.8	0.5	0.2	0.3	0.7	0.4	0.5	0.4	0.2
RITrain3	85	63	93	99	85	193	224	214	220	197
	46.7	35.4	46.3	51.6	46.7	35.9	39.9	35.4	36.5	35.4
	86.4	94.7	89.2	88.2	86.8	87.2	92.5	86.3	86.1	87.1
	0.1	0.5	0.3	0.1	0.2	0.1	0.6	0.3	0.2	0.4
RITrain4	392	353	392	394	393	625	1231	704	681	645
	80.7	75.6	79.5	80.1	80.4	55.1	77.5	56.4	54.2	54.0
	99.8	96.7	99.8	99.8	99.8	99.1	94.0	98.2	98.6	98.8
	0.2	0.1	0.1	0.1	0.2	0.1	0.1	0.2	0.2	0.4

Table 6.6: TestIm 5

	No view simulation on Test images					Views simulated on Test images				
	M0	M1	M2	M3	M4	M0	M1	M2	M3	M4
RITrain1	119	98	110	121	115	214	337	219	221	215
	70.8	62.8	65.1	66.5	66.5	42.6	59.5	40.3	39.2	39.7
	90.9	90.5	91.5	91.8	89.2	89.3	92.7	89.7	90.3	88.2
	0.5	0.5	0.6	0.5	0.2	0.3	0.4	0.8	0.9	0.1
RITrain2	50	57	65	75	49	79	154	114	126	91
	53.8	57.0	59.1	59.5	52.1	29.6	42.9	35.3	34.7	29.5
	55.3	57.5	54.1	53.4	53.3	53.7	58.0	52.6	53.4	53.0
	0.7	0.1	0.9	0.2	0.7	0.4	0.6	0.3	1.3	0.4
RITrain3	147	106	150	145	139	254	335	271	274	255
	71.0	59.6	68.5	65.0	67.8	45.9	56.7	44.2	42.7	44.3
	82.9	78.7	80.3	80.1	81.2	80.3	79.7	80.4	80.3	80.3
	0.1	0.3	0.2	0.3	0.1	0.8	0.7	0.1	0.2	0.5
RITrain4	242	166	239	229	229	406	558	440	414	399
	76.6	66.4	73.8	75.3	74.4	49.6	63.5	48.8	47.6	46.8
	80.1	81.5	80.1	80.1	80.0	80.3	80.9	80.4	80.3	80.2
	0.2	0.3	0.2	0.1	0.2	0.3	0.2	0.6	0.2	0.4

Table 6.7: TestIm 6

	No view simulation on Test images					Views simulated on Test images				
	M0	M1	M2	M3	M4	M0	M1	M2	M3	M4
RITrain1	107	91	103	104	111	156	333	183	162	170
	65.6	59.5	64.0	65.4	66.9	37.9	60.2	38.7	35.5	37.9
	63.9	66.7	64.0	64.0	64.0	66.8	85.5	73.8	66.3	66.2
	0.9	0.2	0.4	0.2	0.2	0.4	1.0	1.1	0.4	1.1
RITrain2	16	24	27	23	20	27	65	44	42	33
	34.8	42.1	43.5	35.4	38.5	15.7	32.0	22.3	21.1	17.3
	36.2	34.9	49.4	41.7	38.0	39.9	51.9	50.0	39.7	42.1
	2.1	2.5	2.2	1.6	1.3	1.7	0.6	1.3	2.0	3.1
RITrain3	127	124	122	133	129	186	411	209	198	196
	67.6	68.1	64.2	67.2	67.2	41.6	65.4	40.6	38.1	40.5
	65.9	67.1	67.1	68.0	66.6	66.1	80.3	67.7	66.1	66.0
	0.7	0.5	0.5	0.3	0.4	0.4	1.0	1.5	0.4	2.3
RITrain4	187	197	188	184	185	261	692	314	279	271
	70.3	73.8	70.1	71.3	70.1	40.5	70.1	43.0	41.3	38.7
	67.7	65.6	67.7	67.6	68.2	67.4	73.0	66.2	67.1	68.0
	0.3	0.1	0.2	0.6	0.4	0.9	0.3	0.7	0.3	0.4

Table 6.8: TestIm 7

	No view simulation on Test images					Views simulated on Test images				
	M0	M1	M2	M3	M4	M0	M1	M2	M3	M4
RITrain1	278	258	284	289	284	400	803	473	434	427
	81.0	76.1	79.3	81.6	80.5	49.9	71.5	50.9	47.3	49.4
	88.6	87.1	88.4	88.0	88.1	88.1	92.9	87.8	87.2	87.6
	0.2	0.1	0.2	0.1	0.0	0.1	0.1	0.2	0.4	0.3
RITrain2	138	138	164	161	140	179	309	247	229	199
	63.6	62.4	69.5	67.1	66.0	38.4	48.0	44.2	41.0	39.6
	74.2	71.9	76.6	74.2	73.1	73.5	73.8	75.7	73.7	73.3
	0.3	0.4	0.4	0.3	0.1	0.4	0.2	0.3	0.5	0.6
RITrain3	168	147	174	172	169	267	445	299	282	281
	74.3	67.1	75.3	73.2	73.2	43.1	58.5	44.0	39.7	42.4
	90.2	88.2	90.1	88.1	89.8	90.5	91.0	91.9	88.6	90.0
	0.2	0.2	0.3	0.3	0.1	0.2	0.2	0.3	0.3	0.6
RITrain4	151	93	161	145	151	233	339	275	243	239
	77.8	52.8	73.2	75.9	75.1	44.7	53.3	45.2	42.0	43.6
	90.3	88.1	93.3	91.7	88.7	91.4	93.0	94.0	91.7	89.3
	0.1	0.3	0.2	0.1	0.1	0.4	0.3	0.8	0.3	0.5

Table 6.9: TestIm 8

	No view simulation on Test images					Views simulated on Test images				
	M0	M1	M2	M3	M4	M0	M1	M2	M3	M4
RITrain1	52	41	50	57	49	75	136	83	88	72
	62.7	56.9	58.8	63.3	57.0	34.6	54.8	34.2	39.1	31.4
	76.4	45.6	70.8	71.0	71.6	77.0	72.7	69.8	74.4	71.1
	0.3	0.9	0.2	1.1	0.4	5.1	1.5	3.2	0.9	6.4
RITrain2	8	25	19	18	10	17	45	28	31	21
	20.0	54.3	35.2	36.0	21.7	12.1	31.5	18.5	19.6	14.6
	55.5	20.0	46.9	47.5	54.7	52.1	26.3	48.6	47.5	54.8
	5.2	0.5	7.9	2.4	11.0	4.4	1.9	23.5	1.2	2.3
RITrain3	139	100	141	149	138	224	297	233	248	224
	72.4	62.9	72.3	71.6	72.3	43.7	54.5	43.2	42.5	43.3
	89.8	81.7	90.1	90.5	90.0	89.7	83.8	89.8	89.9	89.8
	0.2	0.9	0.2	0.2	0.0	0.3	0.4	0.3	0.4	0.2
RITrain4	147	100	141	149	138	217	289	233	239	211
	75.4	59.5	71.6	71.0	72.6	46.3	53.4	45.2	43.4	45.2
	89.9	61.1	90.0	90.0	90.0	89.7	85.1	89.7	89.9	89.8
	0.2	0.2	0.2	0.1	0.3	0.2	0.5	0.2	0.4	0.8

Table 6.10: TestIm 9

	No view simulation on Test images					Views simulated on Test images				
	M0	M1	M2	M3	M4	M0	M1	M2	M3	M4
RITrain1	169	189	174	175	170	215	586	269	246	230
	72.2	72.7	69.9	72.0	70.0	40.6	64.3	42.0	39.1	38.5
	67.7	67.3	67.3	67.6	67.3	67.2	75.6	71.7	68.4	67.1
	0.3	0.3	0.1	0.3	0.1	0.5	0.3	0.5	0.3	0.5
RITrain2	45	55	62	58	54	65	112	89	82	80
	52.9	48.7	55.9	55.2	55.1	24.6	32.6	29.6	27.6	26.4
	40.3	38.8	47.9	40.2	40.1	50.6	45.9	51.2	48.1	49.2
	1.6	0.5	0.9	0.8	0.7	1.4	1.1	1.5	0.6	0.9
RITrain3	162	169	165	167	165	213	545	253	239	224
	74.0	70.4	72.1	74.2	70.8	43.1	64.2	42.3	40.0	40.7
	67.6	67.3	66.9	67.2	67.4	67.0	76.8	70.9	67.6	67.1
	0.3	0.4	0.3	0.2	0.1	0.4	0.2	0.5	1.4	0.3
RITrain4	148	151	152	146	150	207	484	243	211	215
	70.8	70.2	70.4	68.2	69.8	42.3	61.7	42.2	39.7	40.0
	66.7	66.6	66.6	67.0	66.7	69.4	79.7	73.2	68.3	68.9
	0.4	0.3	0.5	0.4	0.2	1.2	0.2	0.1	1.0	0.3

Table 6.11: TestIm 10

	No view simulation on Test images					Views simulated on Test images				
	M0	M1	M2	M3	M4	M0	M1	M2	M3	M4
RITrain1	31	29	32	26	30	51	102	58	49	52
	43.1	46.0	44.4	39.4	43.5	26.4	45.3	27.6	25.3	25.1
	62.8	51.9	61.8	68.7	62.3	65.5	84.4	66.7	66.5	68.6
	2.4	6.0	1.6	2.8	2.6	0.6	3.5	1.4	2.9	2.4
RITrain2	2	7	7	2	2	13	33	18	18	13
	5.4	25.9	19.4	5.9	5.4	10.2	26.0	15.1	14.4	9.8
	0.0	9.6	16.5	0.0	0.0	37.8	46.3	41.2	34.4	33.6
	40.8	31.5	12.8	3.7	35.8	52.9	2.1	6.1	2.9	17.2
RITrain3	148	126	148	151	150	196	424	246	226	211
	65.8	67.0	66.7	66.5	64.7	40.0	63.1	44.5	40.4	39.7
	72.3	72.1	72.3	72.3	72.2	71.9	73.0	71.7	72.6	71.8
	0.3	0.5	0.6	0.7	0.1	1.0	0.1	1.2	1.8	0.4
RITrain4	115	106	113	117	113	149	380	177	163	158
	61.8	56.4	60.8	61.9	61.4	35.6	58.6	38.6	35.3	35.7
	72.3	68.3	72.3	72.4	72.4	72.1	72.0	71.9	72.2	72.1
	0.5	0.0	0.6	0.9	0.4	0.5	1.0	1.4	0.6	1.5

Table 6.12: TestIm 11

Chapter 7

Conclusion

Our work addressed the problem of producing 2D-to-3D matches for PnP (Perspective n-Point) based camera pose estimation. Based on the recent developments in the area we adopted a two stage framework in which SIFT technique is used for interest point detection and description. During the training stage we explored different ways of matching SIFT descriptors and during test stage we tried different ways of 3D point representation. Using ASIFT technique we explored obtaining additional feature descriptors from simulated views of the environment. We also presented a strategy for accelerating mean-shift operation which we conceived during our experiments with different clustering techniques during training stage.

For matching SIFT descriptors during training stage we propose a novel method (ATC - Adaptive Transitive Closure) for clustering in which range based matching is performed adaptively. Results of our evaluation presented in chapter 3 indicates many advantages of our method for establishing 2D-to-2D matches to perform SfM. It avoids the need of knowing the number of clusters apriori and the task of choosing the right parameter to represent intra-class distance.

During test stage we experimented with various strategies on the 3D map obtained from different clustering schemes to perform 3D point recognition. Unlike, the training stage in which we were able to obtain results showing clear advantage of ATC clustering scheme over other methods for building 3D map, the results in the test stage showed marginal improvement due to ATC in few cases. The most significant of them being the ability to perform pose estimation consistently over different pose variation between train and test images. We also demonstrated the advantage of using SVM classifier for 3D point representation.

Our strategy for accelerating mean-shift operation is based on mathematical support which proves that we obtain exactly same result as the conventional mean-shift with our modification. But, compared to the existing strategies which use approximation and subsampling, we obtain moderate acceleration in specific range of parameters.

Using features from simulated views did not show any clear advantage for pose computation unless the number of feature descriptors in the 3D map is extremely scarce. We need to run experiments with simulated views on a dataset captured in a larger environment and design new strategies to handle the ambiguity in feature matching due to the huge number of features from simulated views.

7.1 Future work

Using high dimensional features like SIFT makes the matching process computationally inefficient. We need a high degree of parallel processing (eg: parallel matching for each test vector) for our method to be usable in applications which need real-time speed. In order to reduce computational cost it would be interesting to experiment with fast classification techniques[108, 89].

The features extracted from training images which could not participate in the 2D-to-2D matching process are not included in the 3D map. This is particularly true for the features in the images whose camera positions are at the extremities of the views available in the training set. Some of these excluded features may match with the test image. We would like to explore whether combining the 2D-to-2D match constraints between the test and train images along with the 2D-to-3D matches during the test stage can lead to better pose estimation. These 2D-to-2D matches between test and train images can be useful when the available 2D-to-3D matches are less or the accuracy of the 3D coordinates of the matched 3D points is poor due to short baseline between the training camera positions from which they are computed. The combination of 2D-to-2D and 2D-to-3D matches can also help to reduce the jitter in pose estimated through independently recognized 3D points in successive images. For using simulated views this provides a scope for exploring new strategies for choosing the appropriate simulation based on the approximate location of the test image which can be guessed from previous image. We would also like to compare the cost of computing features from simulated views with that of updating the 3D map with new features from test images.

For mean-shift acceleration we want to explore a different experiment in the same framework presented in chapter 5. During the first iteration, instead of performing transitive closure using range $\sqrt{2}w$, we can just keep a list for each training vector that stores all the training vectors that are within distance $\sqrt{2}w$ from it. After each iteration, we need to perform range search with only those vectors which are in the list associated with the currently visible set of training vectors.

Bibliography

[1] http://nlp.stanford.edu/IR-book/completelink.html. 52

[2] http://en.wikipedia.org/wiki/Coefficient_of_variation. 74

[3] http://en.wikipedia.org/wiki/Minimum_bounding_rectangle. 92

[4] http://http://en.wikipedia.org/wiki/Kernel_trick. 98

[5] http://en.wikipedia.org/wiki/Basel_problem. 114

[6] http://demo.ipol.im/demo/my_affine_sift/. 134

[7] Henrik Aanæs, Anders Lindbjerg Dahl, and Kim Steenstrup Pedersen. Interesting interest points. *International Journal of Computer Vision*, 97(1):18–35, 2012. 59, 80, 126

[8] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, 24(5):1027–1037, 2008. 42, 44, 45

[9] S. Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, 2007. 107

[10] Ronald T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997. 7, 17

[11] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics Automation Magazine*, 13(3):108–117, 2006. 7, 17, 37

[12] Stephen T. Barnard and William B. Thompson. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):333–340, 1980. 37

[13] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. 36, 41

[14] M.-O. Berger. How to track efficiently piecewise curved contours with a view to reconstructing 3D objects. In *ICPR*, 1994. 38

[15] K. K. Srikrishna Bhat, Marie-Odile Berger, Gilles Simon, and Frédéric Sur. Transitive closure based visual words for point matching in video sequence. In *ICPR*, 2010. 49

[16] K. K. Srikrishna Bhat, Marie-Odile Berger, and Frédéric Sur. Visual words for 3D reconstruction and pose computation. In *3DIMPVT*, 2011. 49

[17] Tom Botterill, Steven Mills, and Richard Green. Bag-of-words-driven, single-camera simultaneous localization and mapping. *Journal of Field Robotics*, 28(2):204–226, 2011. 43

[18] A. J. Bray. Tracking objects using image disparities. *Image Vision Computing*, 8(1):4–9, 1990. 37

[19] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzinski, C. Strecha, and P. Fua. BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, 2012. 37

[20] M.A. Carreira-Perpinan. Acceleration strategies for gaussian mean-shift image segmentation. In *CVPR 2006*. 13, 23, 107, 108, 118

[21] Baptiste Charmette, Eric Royer, and Frédéric Chausse. Matching planar features for robot localization. In *International Symposium on Advances in Visual Computing: Part I*, 2009. 121

[22] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995. 42, 45, 107

[23] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. 13, 22, 42, 45, 63, 107

[24] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004. 46

[25] Anders Lindbjerg Dahl, Henrik Aanæs, and Kim Steenstrup Pedersen. Finding the best feature detector-descriptor combination. In *3DIMPVT*, 2011. 36

[26] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV 2003*. 8, 18, 39

[27] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. MonoSLAM : Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007. 7, 17, 37

[28] Daniel Dementhon. Spatio-temporal segmentation of video by hierarchical mean shift analysis. 2002. 108

[29] Zilong Dong, Guofeng Zhang, Jiaya Jia, and Hujun Bao. Keyframe-based real-time camera tracking. In *ICCV*, 2009. 42

[30] R. Duda, P. Hart, and D. Stork. *Pattern classification*. John Wiley and Son, 2001. 12, 21, 40, 45, 79, 96, 97

[31] Michele Fenzi, Ralf Dragon, Laura Leal-Taixé, Bodo Rosenhahn, and Jörn Ostermann. 3D Object Recognition and Pose Estimation for Multiple Objects Using Multi-Prioritized RANSAC and Model Updating. *Pattern Recognition*, 7476:123–133, 2012. 43, 80, 134

[32] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 8, 10, 17, 20, 28, 29

[33] W. Forstner. A feature based correspondence algorithm for image matching. *ISPRS*, 1986. 36

[34] D. Freedman and P. Kisilev. Fast mean-shift by compact density representation. In *CVPR*, 2009. 13, 23, 107, 108, 118

[35] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975. 107

[36] B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: a texture classification example. *ICCV 2003.* 13, 22, 63, 107, 108

[37] Chris Goad. Readings in computer vision: issues, problems, principles, and paradigms. chapter Special purpose automatic programming for 3D model-based vision, pages 371–381. Morgan Kaufmann Publishers Inc., 1987. 37

[38] Iryna Gordon and David G. Lowe. What and where: 3D object recognition with accurate pose. In *Toward Category-Level Object Recognition*, 2006. 9, 11, 19, 20, 42, 45, 46

[39] M. Grabner, H. Grabner, and H. Bischof. Learning features for tracking. In *CVPR*, 2007. 9, 18, 39

[40] W E L Grimson and T Lozano-Perez. Model-based recognition and localization from sparse range or tactile data. *The International Journal of Robotics Research*, 3(3):3–35, 1984. 37

[41] C Harris and M Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988. 41, 43

[42] Chris Harris. Tracking with rigid objects. In *MIT Press*, 1992. 8, 18, 38

[43] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, Second edition, 2004. 25, 30, 32

[44] R.I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997. 30, 31

[45] Joel A. Hesch and Stergios I. Roumeliotis. A direct least-squares (DLS) solution for PnP. In *ICCV 2011.* 7, 17, 25, 34

[46] Stefan Hinterstoisser, Vincent Lepetit, Selim Benhimane, Pascal Fua, and Nassir Navab. Learning real-time perspective patch rectification. *International Journal of Computer Vision*, 91(1):107–130, 2011. 9, 18, 39

[47] W. A. Hoff, K. Nguyen, and T. Lyon. Computer vision-based registration techniques for augmented reality. In *Proceedings of Intelligent Robots and Control Systems XV, Intelligent Control Systems and Advanced Manufacturing*, 1996. 8, 18

[48] J. Hong, X. Tan, B. Pinette, R. Weiss, and E.M. Riseman. Image-based homing. *IEEE Control Systems*, 12(1):38–45, 1992. 40

[49] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *The Journal of the Optical Society of America A*, 4(4):629–642, 1987. 33, 81, 82

[50] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 1981. 38

[51] Edward Hsiao, Alvaro Collet, and Martial Hebert. Making specific features less discriminative to improve point-based 3D object recognition. In *CVPR*, 2010. 9, 19, 42, 46, 80, 107

[52] A. Irschara, C. Zach, and H. Bischof. Towards wiki-based dense city modeling. In *ICCV*, 2007. 45, 46

[53] Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. 38

[54] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. *CVPR 2009*. 48

[55] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *ISMAR*, 2007. 9, 18, 39

[56] Georg Klein and David Murray. Full-3D edge tracking with a particle filter. In *BMVC 2006*. 38

[57] Dieter Koller, Gudrun Klinker, Eric Rose, David Breen, Ross Whitaker, and Mihran Tuceryan. Real-time vision-based camera tracking for augmented reality applications. In *ACM VRST*, 1997. 8, 18

[58] Henner Kollnig and Hans-Hellmut Nagel. 3D pose estimation by directly matching polyhedral models to gray value gradients. *International Journal of Computer Vision*, 23(3):283–302, 1997. 38

[59] J. Leng and H. Wang. Tracking as recognition: a stable 3D tracking framework. In *International Conference on Control, Automation, Robotics and Vision*, 2004. 40

[60] V. Lepetit and P. Fua. Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005. 15, 24, 35

[61] V. Lepetit and P. Fua. Keypoint Recognition using Randomized Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006. 41

[62] V. Lepetit, J. Pilet, and P. Fua. Point Matching as a Classification Problem for Fast and Robust Object Pose Estimation. In *CVPR*, 2004. 41

[63] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An Accurate $O(n)$ Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2):155–166, 2009. 7, 17, 25, 28, 33

[64] Cheng-Hsuan Li, Chin-Teng Lin, Bor-Chen Kuo, and Hui-Shan Chu. An automatic method for selecting the parameter of the RBF kernel function to support vector machines. In *IGARSS*, 2010. 13, 22, 98, 99, 103

[65] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab. A dataset and evaluation methodology for template-based tracking algorithms. In *ISMAR*, 2009. 37, 123

[66] Wei Liu, Yongtian Wang, Jing Chen, Junwei Guo, and Yang Lu. A completely affine invariant image-matching method based on perspective projection. *Machine Vision and Applications*, 23(2):231–242, 2012. 122

[67] M.I. A. Lourakis and A.A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009. 31, 32

[68] D G Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987. 37

[69] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 9, 19, 36, 41, 44, 122, 124

[70] D. G. Luenberger. Optimization by vector space methods, 1969. 113

[71] Jie Luo, Andrzej Pronobis, Barbara Caputo, and Patric Jensfelt. The KTH-IDOL2 Database. Technical Report CVAP304, KTH Royal Institute of Technology, CVAP/CAS, Stockholm, Sweden, October 2006. 59, 80

[72] Eric Marchand, Patrick Bouthemy, and Francois Chaumette. A 2D-3D model-based approach to real-time visual tracking. *Image and Vision Computing*, 19(13):941–955, 2001. 38

[73] K. Mikolajczyk and J. Matas. Improving descriptors for fast tree matching by optimal linear projection. In *ICCV 2007*. 46

[74] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005. 36

[75] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *ECCV*, 2002. 41

[76] N. D. Molton, A. J. Davison, and I. D. Reid. Locally planar patch features for real-time structure from motion. In *BMVC*, 2004. 121

[77] J. Mooser, S. You, and U. Neumann. Real-time object tracking for augmented reality combining graph cuts and optical flow. In *ISMAR 2007*. 38

[78] Jean-Michel Morel and Guoshen Yu. ASIFT: A new framework for fully affine invariant image comparison. *SIAM: SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009. 5, 14, 23, 122, 123

[79] David M. Mount and Sunil Arya. ANN: A library for approximate nearest neighbor searching. http://www.cs.umd.edu/~mount/ANN. 14, 23, 63, 64, 85, 87, 108, 117

[80] Shree K. Nayar, Sameer A. Nene, and Hiroshi Murase. Real-time 100 object recognition system. In *ICRA*, 1996. 40

[81] Radford M. Neal and Geoffrey E. Hinton. Learning in graphical models. chapter A view of the EM algorithm that justifies incremental, sparse, and other variants, pages 355–368. MIT Press, 1999. 108

[82] U. Neumann and S. You. Natural feature tracking for augmented reality. *IEEE Transactions on Multimedia*, 1(1):53–64, 1999. 38

[83] Stéphane Nicolau, Xavier Pennec, Luc Soler, and Nicholas Ayache. Evaluation of a new 3D/2D registration criterion for liver radio-frequencies guided by augmented reality. In *Proceedings of the International Conference on Surgery simulation and soft tissue modeling*, 2003. 92

[84] Huazhong Ning, Wei Xu, Yihong Gong, and T. Huang. Discriminative learning of visual words for 3D human pose estimation. In *CVPR 2008*. 46

[85] D. Nister. An efficient solution to the five-point relative pose problem. In *CVPR*, 2003. 30

[86] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *CVPR*, 2004. 8, 18, 39

[87] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. *CVPR 2006*. 11, 20, 42, 43, 44

[88] Esko Nuutila. Efficient transitive closure computation in large digraphs. *Acta Polytechnica Scandinavia: Mathematics and computing in engineering series.*, 74:1–124, 1995. 52

[89] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–461, 2010. 5, 14, 23, 36, 37, 41, 122, 124, 142

[90] M. Ozuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature Harvesting for Tracking-by-Detection. In *ECCV*, 2006. 9, 18, 39, 44

[91] Sastry P.S. Computing and information sciences: Recent trend. chapter An introduction to support vector machines. Narosa Publishing, New Delhi, 2002. 96, 98

[92] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *ECCV*, 2006. 36

[93] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66(3):231–259, March 2006. 7, 9, 17, 19, 41, 42

[94] Eric Royer, Maxime Lhuillier, Dhome Michel, and Jean-Marc Lavest. Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, 74(3):237–260, 2007. 39, 43

[95] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV 2011*. 37

[96] F. Schaffalitzky and A. Zisserman. Automated location matching in movies. *Computer Vision and Image Understanding*, 92:236–264, 2003. 48

[97] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, pages 1–7, 2007. 44

[98] Jianbo Shi and C. Tomasi. Good features to track. In *CVPR 1994*. 39

[99] R. Sim and G. Dudek. Mobile robot localization from learned landmarks. In *International Conference on Intelligent Robots and Systems*, 1998. 40

[100] R. Sim and G. Dudek. Learning visual landmarks for pose estimation. In *ICRA*, 1999. 40

[101] G. Simon and M.-O. Berger. A two-stage robust statistical method for temporal registration from features of various type. In *ICCV*, 1998. 38

[102] G. Simon and M.-O. Berger. Pose estimation for planar structures. *IEEE Computer Graphics and Applications, IEEE*, 22(6):46–53, 2002. 39

[103] G. Simon, A.W. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *ISAR*, 2000. 37

[104] Gilles Simon, Vincent Lepetit, and Marie-Odile Berger. Computer vision methods for registration: mixing 3D knowledge and 2d correspondences for accurate image composition. In *Proceedings of the international workshop on Augmented reality : placing artificial objects in real scenes*, pages 111–127, 1999. 43, 106

[105] Josef Sivic and Andrew Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV 2003*. 11, 20, 44

[106] Noah Snavely. http://phototour.cs.washington.edu/bundler/. 25, 28, 31

[107] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3D. In *ACM SIGGRAPH*, 2006. 9, 19, 25

[108] C. Strecha, A.M. Bronstein, M.M. Bronstein, and P. Fua. LDAHash: Improved Matching with Smaller Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):66–78, 2012. 9, 19, 36, 37, 142

[109] M. Tamaazousti, V. Gay-Bellile, S.N. Collette, S. Bourgeois, and M. Dhome. Nonlinear refinement of structure from motion reconstruction by taking advantage of a partial knowledge of the environment. In *CVPR*, 2011. 44

[110] P.H.S. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000. 92

[111] Miroslav Trajkovic and Mark Hedley. Fast corner detection. *Image and Vision Computing*, 16(2):75–87, 1998. 39

[112] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3D tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1385–1391, 2004. 7, 17, 44

[113] J. Valls Miro, Weizhen Zhou, and G. Dissanayake. Towards vision based navigation in large indoor environments. In *International Conference on Intelligent Robots and Systems*, 2006. 8, 18, 39

[114] J.C. van Gemert, C.J. Veenman, A.W.M. Smeulders, and J.-M. Geusebroek. Visual word ambiguity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1271–1283, 2010. 46

[115] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/. 63

[116] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 40

[117] Ping Wang, Dongryeol Lee, Alexander G. Gray, and James M. Rehg. Fast mean shift with accurate and stable convergence. *Journal of Machine Learning Research - Proceedings Track*, 2:604–611, 2007. 108

[118] Changchang Wu, B. Clipp, Xiaowei Li, J.-M. Frahm, and M. Pollefeys. 3D model matching with Viewpoint-Invariant Patches (VIP). In *CVPR*, 2008. 121

[119] Chunxia Xiao and Meng Liu. Efficient Mean-shift Clustering Using Gaussian KD-Tree. *Computer Graphics Forum*, 29(7):2065–2073. 108

[120] Jianxiong Xiao, Jingni Chen, Dit-Yan Yeung, and Long Quan. Structuring visual words in 3D for arbitrary-view object localization. In *ECCV*, 2008. 42

[121] C. Yang, R. Duraiswami, N.A. Gumerov, and L. Davis. Improved fast gauss transform and efficient kernel density estimation. In *ICCV 2003*. 107

[122] Xiaotong Yuan, Bao-Gang Hu, and Ran He. Agglomerative mean-shift clustering via query set compression. In *Proceedings of the SIAM International Conference on Data Mining*, 2009. 108, 118

Résumé

L'estimation de la pose (position et orientation) d'une caméra par rapport à un système fixe de coordonnées 3D à partir d'une image de l'environnement a de nombreuses applications. Dans notre travail, nous établissons des correspondances de points 2D-3D entre l'image et l'environnement afin d'appliquer l'algorithme Perspective-n-Point (PnP) pour calculer la pose. Chaque correspondance 2D-3D associe les coordonnées 2D d'un point de l'image aux coordonnées 3D du point correspondant dans l'environnement. Afin d'établir une correspondance 2D-3D entre points à travers un point 3D capturé dans une image, il faut connaître (i) les coordonnées 3D du point (ii) des caractéristiques visuelles de ce point qui peuvent être utilisés pour identifier sa position 2D dans une image donnée. Nous proposons un cadre dans lequel nous calculons la carte 3D, c'est à dire les coordonnées 3D et les caractéristiques visuelles de quelques-uns des points dans l'environnement, par l'intermédiaire d'une étape hors ligne en utilisant un ensemble d'apprentissage d'images de l'environnement. étant donné une nouvelle image de test, nous établissons les correspondances de points 2D-3D à l'aide de la carte 3D pour détecter un certain nombre de points 3D visibles dans l'image. Au cours de l'étape de construction nous utilisons un algorithme SfM (Structure from Motion) sur des images d'apprentissage pour calculer les coordonnées 3D de quelques-uns des points de l'environnement. Pour effectuer SfM nous avons besoin d'ensembles de points 2D en correspondance dans les images d'apprentissage. Chacun est constitué d'un ensemble de coordonnées dans les images 2D d'un seul point 3D. Nous établissons ces ensembles en regroupant les descripteurs SIFT calculés à partir des images d'apprentissage. Les positions 2D des descripteurs SIFT ainsi regroupés sont utilisés comme ensemble de points en correspondance. L'algorithme SfM calcule les coordonnées 3D de ces points en correspondance en minimisant l'erreur de reprojection. Pendant l'étape de test, les descripteurs SIFT associés à un point 3D sont utilisés pour reconnaître le point 3D dans une image donnée. Ce processus est semblable au cadre des mots visuels utilisés dans différents domaines de la vision par ordinateur. Pendant l'apprentissage, les mots visuels sont construits par clustering et pendant les tests les points 3D sont identifiés grâce à la reconnaissance de mots visuels. Nous menons des expériences avec différentes méthodes de clustering (k-means et moyenne-shift) et proposons un nouveau schéma pour la formation de mots visuels. Nous évaluons différents aspects de la qualité de l'ensemble de descripteurs associé et des points 3D calculés à partir de ces méthodes. Pendant la phase de test, nous menons des expériences avec différentes méthodes de mise en correspondance, y compris quelques méthodes bien connues de classification supervisée pour effectuer la reconnaissance des mots visuels. Nous évaluons les différentes stratégies de reconnaissance, sur la base de la précision du calcul de pose sous divers degrés de variation de la pose entre les images d'apprentissage et de test. Afin d'assurer la robustesse à des variations de pose entre images d'apprentissage et de test, nous testons différentes façons d'intégrer des descripteurs SIFT extraits de vues synthétiques générées à partir des images d'apprentissage. En faisant des expériences de clustering avec l'algorithme mean-shift, nous avons conçu une stratégie pour accélérer son calcul en divisant l'ensemble des vecteurs de formation en groupes tels que les vecteurs dans un groupe ne seront jamais influencés par le calcul des clusters dans un autre groupe. Nous présentons la stratégie d'accélération et la preuve mathématique avec une évaluation expérimentale.

Mots-clés: Estimation de la pose, Mots visuel, Représentation de points 3D, Accélération de mean-shift, Veu simulation

Abstract

Estimating the pose (position and orientation) of a camera with respect to a fixed 3D coordinate system from an image of the surrounding environment captured in that pose has many applications. In our work we try to establish 2D-to-3D point correspondences between the image and the environment in order to apply Perspective-n-Point (PnP) algorithm to compute the pose. Each 2D-to-3D point correspondence associates the 2D image coordinates of an image point to the 3D coordinates of the corresponding point in the environment. In order to establish a 2D-to-3D point correspondence through a 3D point captured in an image, we should know (i)3D coordinate of the point and (ii)quantified visual characteristics of the point which can be used to identify its 2D location in a given image. We propose a framework in which we compute the 3D map i.e. 3D coordinates and visual characteristics of some of the points in the environment, through an offline training stage using a set of training images of the environment. Given a new test image we establish the 2D-to-3D point correspondences by using the 3D map to detect some of the 3D points visible in the image. During the training stage we perform SfM (Structure from Motion) on training images to compute 3D coordinates of some of the points in the environment through . In order to perform SfM we need 2D-tracks of 3D points in the training images. Each 2D-track consists of a set of 2D image coordinates of a single 3D point in different training images. We establish 2D-track by clustering the SIFT descriptors computed from training images. The 2D positions of the SIFT descriptors in a cluster is used to establish a 2D-track. SfM computes the 3D coordinates of the points corresponding to these 2D-tracks by minimizing the reprojection error. During the test stage, the SIFT descriptors associated the 2D-track of a 3D point is used to recognize the 3D point in a given image. The overall process is similar to visual word framework used in different fields of computer vision. During training, visual word formation is performed through clustering and during testing 3D points are identified through visual word recognition. We experiment with different clustering schemes (k-means and mean-shift) and propose a novel scheme for visual word formation. We evaluate different aspects of the quality of 2D-tracks and the 3D points computed from these schemes during training. During test stage, we experiment with different matching rules including some of the popular supervised pattern classification methods to perform visual word recognition. We evaluate the various recognition strategies based on the accuracy of pose computation under varying degree of pose difference between training and test images. In order to achieve robustness against pose variation between train and test images, we explore different ways of incorporating SIFT descriptors extracted from synthetic views generated from the training images. While experimenting with mean-shift clustering we conceived a strategy to accelerate its computation by dividing the set of training vectors into groups such that vectors in one group will never influence the computation of clusters in another group. We present the acceleration strategy and mathematical proof with experimental evaluation.

Keywords: Pose estimation, Visual word, 3D point representation, Mean-shift acceleration, View simulation