Optical flow-based navigation algorithms for virtual characters

Axel López

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

Axel LÓPEZ

Optical flow-based navigation algorithms for virtual humans

Thèse présentée et soutenue à l'IRISA, le 16 décembre 2019
Unité de recherche : IRISA / Inria Rennes
Thèse N° :

Rapporteurs avant soutenance :

Ronan BOULIC	Maître d'enseignement et de recherche, EPFL, Lausanne
Nuria PELECHANO	Associate Professor, Universitat Politecnica de Catalunya, Barcelone

Composition du Jury :

Président :	Patrick BOUTHEMY	Directeur de recherche Inria, Inria, Rennes
Examinateurs :	Ronan BOULIC	Maître d'enseignement et de recherche, EPFL, Lausanne
	Nuria PELECHANO	Associate Professor, Univ. Politècnica de Catalunya, Barcelone
	Marilena VENDITTELLI	Associate Professor, Université La Sapienza, Rome
Dir. de thèse :	Julien PETTRE	Directeur de recherche Inria, Inria / IRISA, Rennes
Co-dir. de thèse :	François CHAUMETTE	Directeur de recherche Inria, Inria / IRISA, Rennes

ACKNOWLEDGEMENT

First, I would like to thank my supervisors Julien Pettré and François Chaumette who helped me with invaluable guidance and help through these three years. Special thanks to Eric Marchand as well for your advice for my work.

I would like to thank the members of the Rainbow team for a pleasant experience in Rennes to support my work.

Many thanks to all friends and family who supported me and encouraged during this special period of my life.

And a special mention to my parents who pushed me, supported me and made possible my academic career during these years of studies.

TABLE OF CONTENTS

LIST OF FIGURES

RÉSUMÉ EN FRANÇAIS

Cette thèse a abordé le problème du guidage d'un personnage. Ce travail consiste à trouver une méthode pour générer duLE mouvement d'un personnage dans un environnement soumis à certaines contraintes. Les agents peuvent avoir besoin d'atteindre un certain objectif ou de conserver une certaine formation [Rey99] avec d'autres agents. Sur le chemin, il peut y avoir des obstacles qui doivent être détectés et évités. Le guidage d'un personnage peut inclure des comportements faisant en sorte qu'un agent navigue dans un environnement comme dans la figure 1. Par conséquent, la conduite d´un personnage doit générer une trajectoire, généralement une courbe 2D, en percevant le monde qui l'entoure pour effectuer une tâche de navigation donnée. La perception de l'environnement dépend d'une mèthode particulière.

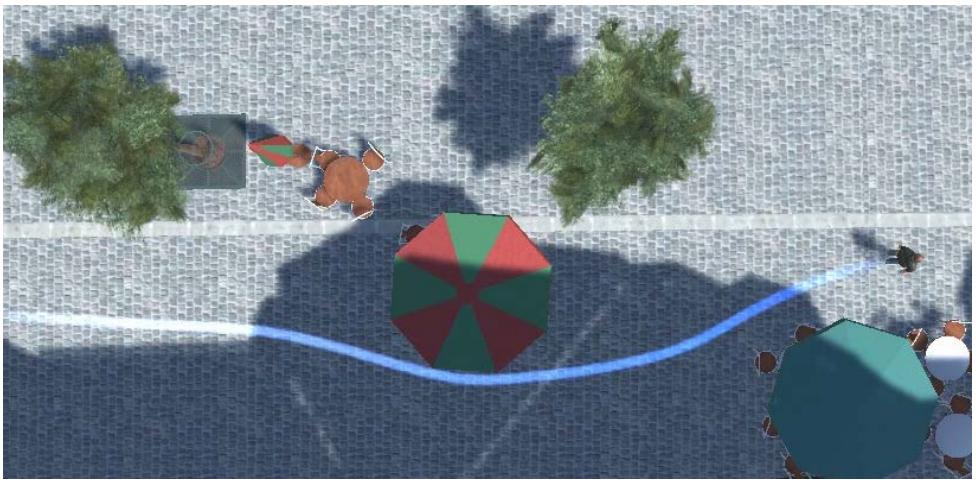


Figure 1 – Exemple de LA trajectoire dU personnage. L'Agent se déplaçant dans une scène virtuelle en évitant les collisions.

Le problème du guidage d'un personnage a été abordé par diverses disciplines telles que la simulation de foule, la robotique et les sciences cognitives. La résolution du pilotage a un large éventail d'applications dont quelques-unes sont illustrées à la figure 2. La communauté de simulation de foule a proposé de nombreuses solutions pour animer des êtres humains, des animaux et d'autres personnages dans des environnements virtuels. Leur objectif est de simuler un grand nombre d'individus le plus

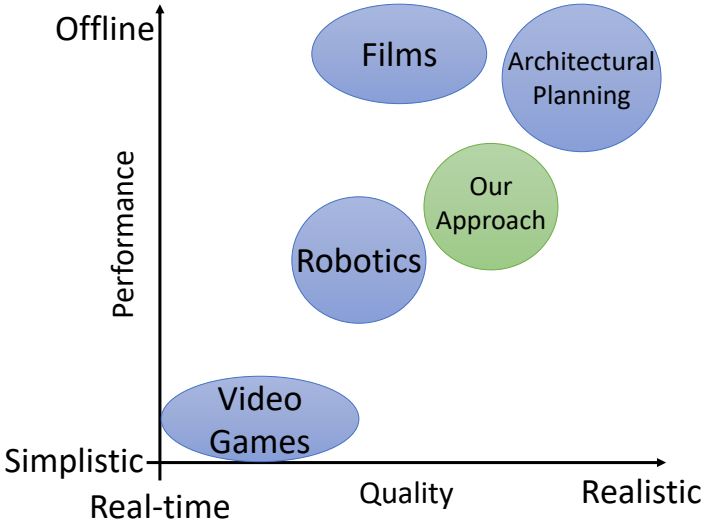


Figure 2 – Exigences de performance et réalisme de la conduite du personnage dans diverses applications.

rapidement possible. Il existe différentes applications de la simulation de foule:

Cinéma. Dans l'industrie du cinéma, les algorithmes de pilotage sont utilisés pour remplir des scènes du film avec des personnages virtuels. Cela permet aux réalisateurs de réduire les coûts en réduisant le nombre de suppléments nécessaires ou en animant des créatures de fiction. L'industrie cinématographique n'exige pas de performance en temps réel et insiste sur le réalisme de ses personnages. Reynolds a initialement proposé l'algorithme de pilotage de Boids [Rey87] pour générer le mouvement des troupeaux. Néanmoins, les artistes veulent garder le contrôle total du personnage virtuel pour répondre aux besoins du scénario en question.

Jeux vidéo. Les jeux vidéo nécessitent généralement la simulation d'un grand nombre de personnages virtuels en temps réel avec les ressources de calcul limitées d'un ordinateur personnel. Cela impose une grande contrainte sur la complexité et le réalisme du comportement des humains. De nombreux jeux implémentent une version de RVO [vdBLM08] [vdBGLM11] qui n'essaie pas de reproduire un comportement réaliste des humains mais fournit une solution très rapide pour éviter les collisions.

Planification architecturale. Les architectes cherchent à améliorer la disposition des bâtiments afin de réduire les encombrements et de faciliter la circulation

des personnes. Ils ont donc besoin d'un modèle de pilotage reproduisant le plus fidèlement possible le comportement et la perception humaine. Ceci est particulièrement important pour la conception de plans d'évacuation.

La communauté de la robotique a exploré des solutions à ce problème. Leur objectif est de permettre aux robots de naviguer dans des environnements réels. Un robot n'a qu'une perception limitée de l'environnement, ce qui augmente la puissance de traitement requise, mais il a également besoin d'un temps de réaction rapide pour naviguer en toute sécurité. Certaines applications de la communauté robotique incluent les entrepôts automatisés, le transport de robots et la gestion des marchandises stockées dans ces derniers. Dans cette situation, les robots sont généralement conscients de l'emplacement de tous les autres robots et un algorithme de pilotage rapide peut piloter plusieurs robots dans une petite zone. Une autre application est la création de robots entièrement autonomes naviguant le long d'êtres humains dans les villes et autres environnements. Cela signifie qu'ils ne peuvent compter que sur des capteurs locaux, tels que des caméras, pour percevoir leur environnement, ce qui entraîne une augmentation spectaculaire du traitement requis pour traiter leur perception limitée.

La communauté des sciences cognitives a également proposé plusieurs méthodes de pilotage. Ils étudient le comportement humain réel en s'appuyant sur des données expérimentales. Ensuite, ils construisent des modèles pour reproduire le plus fidèlement possible les données enregistrées. Leur objectif est d'étudier comment l'homme perçoit l'environnement et sa réaction. Notre travail s'inscrit dans le champ des sciences cognitives car nous proposons un nouveau modèle de contrôle du personnage virtuel qui utilise une perception visuelle simulée similaire à la façon dont les humains perçoivent leur environnement.

Les humains naviguent dans un monde dynamique où les sources d'information sont limitées. Quoi qu'il en soit, ils sont très capables de traverser tout type d'environnement connu ou non. Explorer les règles qui contrôlent la navigation humaine est un problème ouvert. En simulation, de nombreuses approches basent le comportement sur la connaissance de la géométrie de la scène, souvent simplifiée. Cela simplifie beaucoup le problème et un grand nombre de propositions ont été présentées pour produire des trajectoires réalisables sans collision. Cependant, les vrais humains ne peuvent utiliser que leurs sens limités pour traverser leur environnement, principalement la vision et le son. Ces capteurs fournissent une connaissance de l'environnement d'une manière très différente comme dans les simulations mentionnées. Cette distinction affecte la

façon dont les humains traversent la scène.

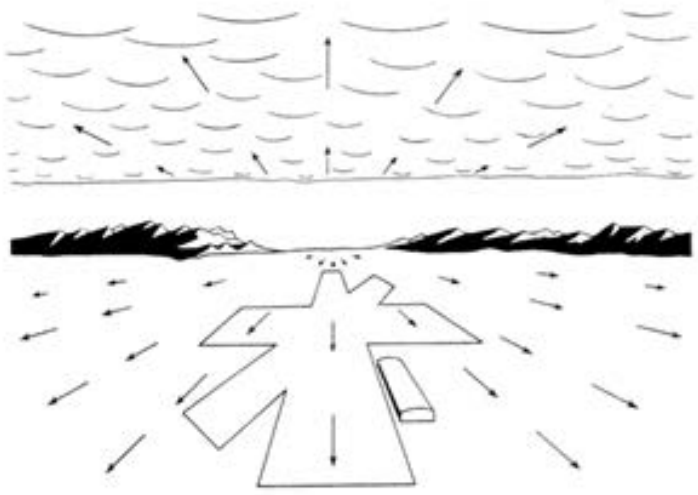


Figure 3 – Représentation du flux optique [Gib86].

De nombreux travaux ont été réalisés pour apprendre comment un être humain réel interagit avec l'environnement. À l'aide de dispositifs de capture de mouvement, la trajectoire de l'individu peut être enregistrée et étudiée. En particulier, le rôle de la vision en navigation a fait l'objet de beaucoup d'attention car c'est l'une des principales sources d'information sur l'environnement. Warren et al. [WKZ+01] à montrer comment le flux optique joue un rôle clé dans la navigation humaine. Le flux optique est un champ vectoriel qui code le mouvement perçu au fil du temps dans le champ de vision. Il code toutes sortes de mouvements visuels allant du mouvement des objets à la variation de l'intensité de la couleur due au mouvement des sources lumineuses.

Notre hypothèse est que, pour reproduire fidèlement le comportement humain, un agent virtuel a besoin de la même information sensorielle pour traverser l'environnement. Afin de se rapprocher de cet objectif, des approches synthétiques basées sur la vision ont été proposées. Ondrej et al. [OPOD10] et Dutra et al. [DMCN+17] a proposé différents algorithmes pour permettre à l'agent de naviguer dans une scène virtuelle en utilisant la vision synthétique. Ces algorithmes n'enregistrent pas d'informations visuelles, mais ils enregistrent les propriétés géométriques de la scène dans le champ de vision. Par conséquent, dans cette thèse, nous proposons et étudions un nouvel algorithme de navigation de caractères basé sur le flux optique. Nos agents enregistreront le flux optique perçu dans leur champ de vision et détecteront les indices qui

Figure 4 – Exemple de navigation par flux optique. (gauche) Vue de dessus de la scène et de la trajectoire de l'agent. (milieu) Scène perçue du point de vue de l'agent. (droite) Flux optique perçut par l'agent.

leur permettront de suivre une trajectoire sans collision. La figure 4 donne un exemple de la perception de nos agents.

Navigation avec Flux Optique

Dans cette thèse, nous fournissons deux boucles de contrôle distinctes utilisant des informations de perception différentes. Cette section fournit un aperçu général des boucles de contrôle qui seront décrites dans cette thèse. La figure 5 est un résumé des étapes détaillées dans les sections suivantes.

Perception

La phase de perception de notre boucle de contrôle correspond à l'enregistrement d'informations de la scène. Nos agents virtuels sont équipés de plusieurs capteurs, qui sont des caméras virtuelles placées au sommet de leur tête et jouant le rôle de l'œil. Les agents peuvent percevoir les images en couleur, les flux optiques synthétiques, la profondeur et les images sémantiques. L'image en couleur est un rendu complet de la scène du point de vue de l'agent. L'image de flux optique synthétique CODE le mouvement relatif réel des objets dans la scène par rapport à l'agent et est générée à l'aide de shaders. La profondeur est simplement la carte de profondeur du cadre de l'agent. Enfin, l'image sémantique CODE l'identifiant de l'objet derrière chaque pixel perçu par l'agent.

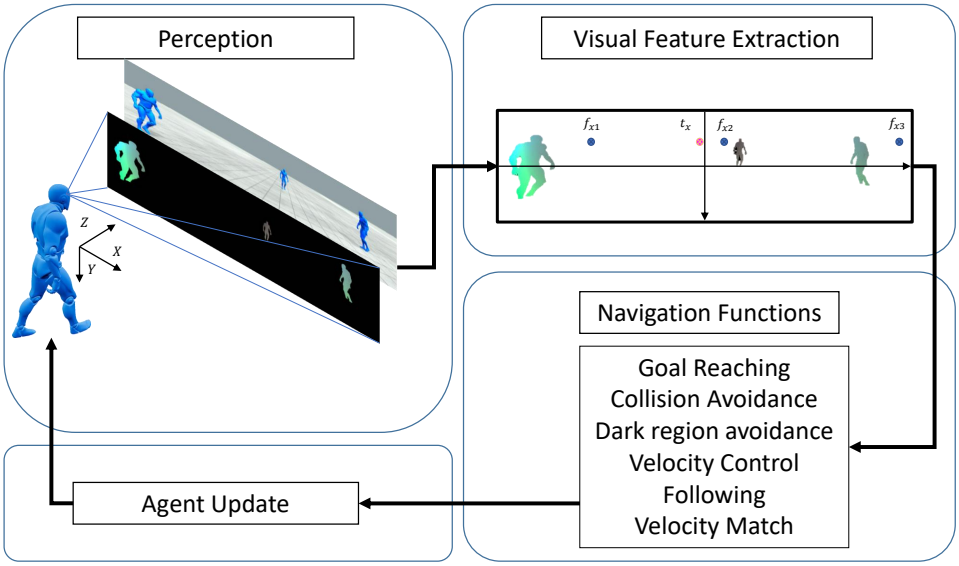


Figure 5 – Illustration des 4 étapes de la boucle de contrôle basée sur la vision pour la navigation de personnage.

Un seul sous-ensemble de ces capteurs est utilisé à la fois, lorsque des images en couleur à flux optique synthétique et des informations sémantiques ne sont généralement pas nécessaires. D'autre part, le flux optique peut être calculé numériquement à partir d'une séquence d'images couleur. Dans cette situation, le flux résultant est bruyant et moins précis, nécessitant l'utilisation d'une segmentation sémantique.

Extraction de caractéristiques visuelles

Dans cette phase, les images perçues doivent être traitées pour extraire des informations utilisables sur l'environnement. La première étape consiste à segmenter les objets de l'arrière-plan. Ceci est fait en utilisant l'image sémantique. S'il n'est pas disponible, il peut être calculé en détectant les discontinuités dans le flux optique.

Ensuite, pour chaque segment, un ensemble d'entités est calculé. À l'aide de la carte de flux optique, nous pouvons calculer le centre d'expansion (FOE), le délai de collision, etc. Ces caractéristiques sont liées au mouvement relatif de l'objet avec l'agent et peuvent être utilisées pour corriger la trajectoire.

Fonctions de navigation

Cette phase utilise les fonctionnalités visuelles extraites pour définir la tâche de navigation. Un ensemble de fonctions de contrôle qui sont des fonctions d'erreur exprimant une petite tâche telle que l'atteinte d'un objectif ou l'évitement de collision. Ensuite, ces fonctions de contrôle sont fusionnées dans la fonction de navigation.

Adaptation des variables de contrôle

Enfin, la fonction de navigation est optimisée à chaque image pour mettre à jour les variables de contrôle de l'agent. Plusieurs méthodes sont possibles pour effectuer cette minimisation. Dans ce travail, nous utilisons une simple descente de gradient qui s'est avérée suffisante.

Conclusion

Cette thèse propose un nouvel algorithme de pilotage pour les personnages virtuels. Notre objectif est de réduire davantage l'écart entre la simulation humaine virtuelle et le mouvement humain réel en utilisant un système de perception visuelle virtuelle qui émule la façon dont les humains perçoivent leur environnement. En particulier, nous introduisons l'utilisation du flux optique, composant clé de la locomotion humaine, en tant que source du mouvement relatif de l'environnement dans les scènes dynamiques.

Notre première contribution consiste à établir l'ensemble des caractéristiques visuelles pertinentes pour la navigation. Nous proposons un nouveau pipeline de perception qui enregistre d'abord les informations dans le champ de vision de l'agent sous forme d'images. Ces images, qui contiennent le flux optique, la profondeur, la couleur, etc., doivent être analysées et traitées afin de pouvoir être utilisées pour la navigation. Le flux optique code le mouvement de chaque point du champ de vision, ce qui signifie que nous pouvons en déduire le mouvement relatif des objets et leur temps de collision. Nous fournissons ici deux pipelines, le premier utilisant un flux optique synthétique calculé à partir de la connaissance du monde virtuel qui fournit un flux de haute qualité. Le second pipeline utilise un flux optique numérique, calculé à partir d'une séquence d'images. Cela se traduit par un flux optique de qualité inférieure mais se rapproche mieux de la perception humaine.

La deuxième contribution établit les relations entre le mouvement humain et les caractéristiques visuelles extraites du système de perception. Pour chaque objet détecté dans les images perçues, nous calculons sa position, son centre d'expansion et son délai de collision. Ces caractéristiques et les vecteurs de flux eux-mêmes sont pris en compte pour la navigation. Nous fournissons l'expression de la motion de ces caractéristiques en ce qui concerne les actions de l'agent.

La dernière contribution est la conception d'un schéma de contrôle utilisant les caractéristiques visuelles et leurs relations avec les variables de contrôle de l'agent. Nous exprimons des tâches simples telles que l'atteinte des objectifs ou l'évitement de collision en termes de caractéristiques visuelles. Ensuite, ces tâches sont combinées dans des fonctions de navigation qui expriment un comportement plus complexe, comme atteindre un objectif tout en évitant les obstacles. De plus, nous concevons une tâche de navigation capable de prendre en compte les conditions d'éclairage sans aucune connaissance préalable de la lumière de la scène et en s'appuyant uniquement sur leurs capteurs visuels.

INTRODUCTION

This thesis addresses the problem of character steering. This problem consists in solving how to generate the motion of a character through an environment subject to certain constraints. Agents may need to reach a certain goal or to keep a certain formation [Rey99] with other agents. On the way there may be obstacles which need to be detected and avoided. Character steering may include any behavior that causes an agent to navigate an environment such as in Figure 6. Therefore, character steering needs to generate a trajectory, typically a 2D curve, by perceiving the world around the character to complete a certain navigation task. The perception of the environment depends on the particular application.

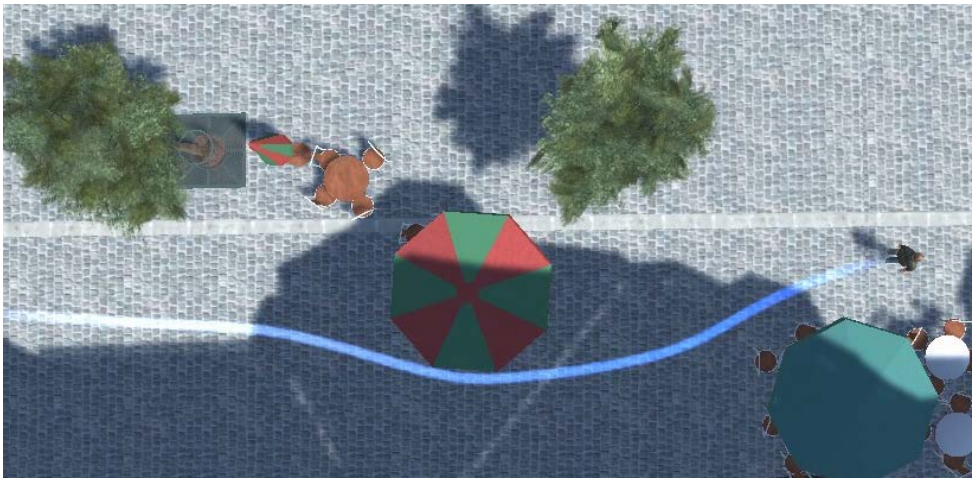


Figure 6 – Character trajectory example. Agent moving in a virtual scene avoiding collisions.

The steering problem has been addressed by various disciplines such as crowd simulation, robotics and cognitive science. Solving the steering has a wide array of applications with a few of them shown in Figure 7. The crowd simulation community has proposed many solutions to animate humans, animals and other characters in virtual environments. Their goal is the simulation of a large number of individuals as fast as possible. There are various applications of crowd simulation:

Cinema. Steering algorithms are used in the cinema industry to fill movie scenes

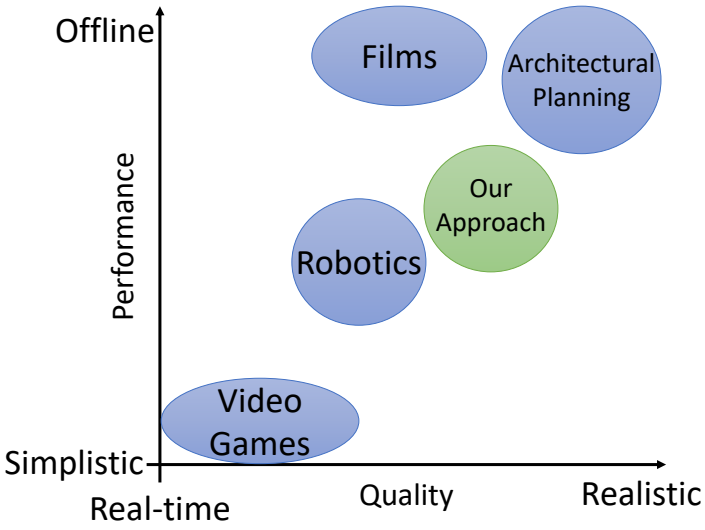


Figure 7 – Requirements of performance and realism of character steering in various applications.

with virtual characters. This allows filmmakers to reduce cost by reducing the number of extras needed for animating large crowds or fictional creatures. The film industry does not require real-time performance and it emphasises the realism of their characters. Reynolds initially proposed the Boids steering algorithm [Rey87] to generate the motion of flocks and herds. Still, artists want to keep full control of the virtual character to satisfy the needs of the particular scenario.

Video Games. Video games typically require large number of virtual characters to be simulated in real-time with the limited computation resources of a personal computer. This imposes a big constraint on how complex and realistic the behavior of the humans may be. Many games implement a version of RVO [vdBLM08] [vdBGLM11] which does not try to reproduce realistic behavior of humans but provides a very fast solution to collision avoidance.

Architectural Planning. Architects seek to improve the layout of buildings to reduce congestion and facilitate the flow of people. Therefore, they require a steering model that reproduces as closely as possible the human behavior and perception. This is particularly important for designing evacuation plans.

The robotics community has explored solutions to this problem. Their goal is to enable robots to navigate real environments. A robot has only a limited perception of the environment and requires a fast reaction time to navigate safely. Some applications

of the robotics community include automated warehouses were robots transport and manage the goods stored in them. In this situation, robots are usually aware of every other robot's location and fast steering algorithm can drive multiple robots in a small area. Another application is the creation of fully autonomous robots navigating along real humans in cities and other environment. This means that they can only rely on local sensors such as cameras to perceive their environment typically the using expensive image processing algorithms.

The cognitive science community has also proposed several steering methods. They study real human behavior relying on experimental data. Then, they build models to reproduce the recorded data as close as possible. Their goal is to study how human perceive the environment and their reaction. Our work falls in the scope of cognitive science as we propose a new virtual character control model that uses a simulated visual perception similar to how real humans perceive their environment.

Humans navigate in a dynamic world with limited sources of information about it. Regardless, they are very capable of traversing any kind of environment known or new. Exploring the rules that control human navigation is an open problem. In simulation, many approaches base behavior on the knowledge of the geometry of the scene, often simplified. This makes the problem much simpler and a large number of proposals have been presented to produce feasible, collision-free trajectories. However, real humans can only use their limited senses to traverse their environment, mainly vision, sound and haptic. These sensors provide knowledge of the environment in a very different way as in the mentioned simulations. This distinction affects the way humans traverse the scene.

There have been many works aimed to learn how a real human interacts with the environment. Using motion capture devices, the trajectory of the individual can be recorded and studied. In particular, the role of vision in navigation has received a lot of attention as it is one of the main sources of information regarding the environment. Warren et al. [WKZ+01] showed how optical flow plays a key role in human navigation. Optical flow is a vector field that encodes the perceived motion over time in the field of view (see Figure 8). It encodes all kinds of visual motions from the motion of objects to the variation of color intensity due to the motion of light sources.

Our hypothesis is that a way to truthfully reproduce human behavior, is to use the same sensory information that humans use to traverse the environment. In order to move closer to this goal synthetic vision-based approaches were proposed. Ondrej et

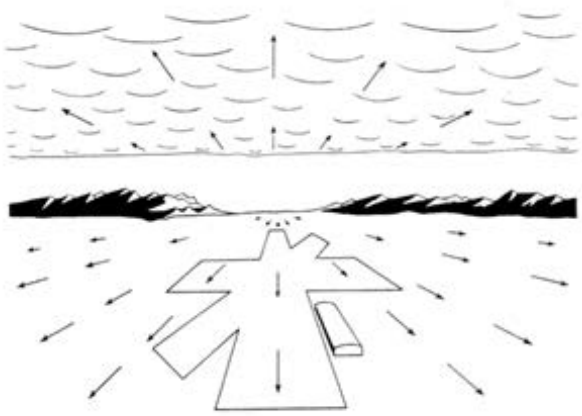


Figure 8 – Representation of optical flow [Gib86].

Figure 9 – Optical flow navigation example. (left) Top view of the scene and trajectory of the agent. (middle) Scene perceived from the agent's point of view. (right) Optical flow perceived by the agent.

al. [OPOD10] and Dutra et al. [DMCN+17] proposed different algorithms to allow agent navigate a virtual scene using synthetic vision. These algorithms do not record visual information but instead they record geometrical properties of the scene in the field of view. Therefore, in this thesis we propose and study new algorithms for character navigation based on optical flow. Our agents record optical flow perceived in their field of view and detect the clues that allow them to follow a collision-free trajectory. Figure 9 provides an example of the perception of our agents.

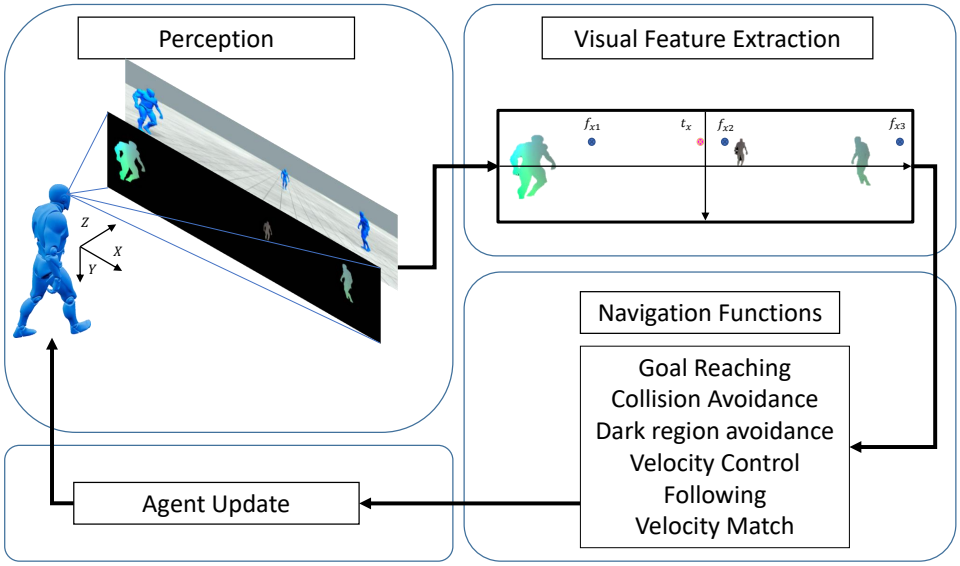


Figure 10 – Illustration of the 4 stages of the vision-based control loop for character navigation.

Optical Flow Navigation

In this thesis we provide two distinct control loops using different perceptual information. This section provides a general overview of the control loops that are described through this thesis. Figure 10 is a summary of the steps that are detailed in the following sections.

Perception

The perception phase of our control loop corresponds to the recording of information of the scene. Our virtual agents are provided with only one visual sensor that provides different types of data which is virtual camera placed at the top of their head, playing the role of eyes. Agents can perceive color, synthetic or estimated optical flow, depth and semantic images. The color image is a full render of the scene from the point of view of the agent. The synthetic optical flow image encodes the true relative motion of the objects in the scene with respect to the agent and is generated using shaders. The depth is simply the depth map from the agent's frame which humans acquire using stereoscopic vision. Finally, the semantic image encodes the id of the object behind every pixel perceived by the agent.

Only a subset of these data can be used: when using synthetic optical flow, color images and semantic information are usually not necessary. On the other hand, when optical flow is numerically computed from a sequence of color images, the resulting flow is noisy and less accurate, requiring the use of semantic segmentation.

Visual Feature Extraction

In this phase the perceived images need to be processed to extract usable information about the environment. The first step is to segment objects from the background. This is done using the semantic image. If it is not available, then it can be computed by detecting discontinuities in optical flow.

Then, for each object, a set of features are computed. Using the optical flow map we can compute the focus of expansion (FOE), time-to-collision (TTC), etc. These features are related to the relative motion of the object with the agent and are used to determine the agent motion.

Navigation Functions

This phase uses the visual features extracted to define the navigation task. A set of control functions which are objective functions expressing low-level task such as goal reaching or collision avoidance are selected, then, these control functions are blended into the navigation function.

Agent Update

Finally, the navigation function is optimized at every frame to update the control variables of the agent. Several methods are possible to perform this minimization. In this work we use a simple gradient descent which proved to be sufficient.

Contributions

This thesis proposes new steering algorithms for virtual characters. Our goal is to further close the gap between virtual human simulation and real human motion using

a virtual visual perception system that emulates the way humans perceive their environment. In particular we introduce the use of optical flow, a key component in human locomotion, as a source of the relative motion of the environment in dynamic scenes.

Our first contribution is establishing the set of visual features relevant for navigation. We propose a new perception pipeline that first, records information in the field of view of the agent in form of images. These images, which contain optical flow, depth, color, etc, need to be analyzed an processed in order to be usable for navigation. Optical flow encodes the motion of every point in the field of view, which means that we can deduce the relative motion of objects and their time-to-collision. We provide two pipelines here, the first one uses synthetic optical flow computed from the knowledge of the virtual world which provides a high quality flow. The second pipeline uses numeric optical flow, computed from a sequence of images. This result in a lower quality optical flow but approximates better the human perception.

The second contribution establishes the relations between the human motion and the visual features extracted from the perception system. For every object detected in the perceived images we compute its position, FOE and TTC. These features as well as flow vectors themselves are taken into account for navigation. We provide the expression of the motion of these features with respect to the agent's possible actions.

The last contribution is the design of a control scheme using the visual features and their relations with the agent's control variables. We express simple tasks such as goal reaching or collision avoidance in terms of the visual features. Then these tasks are combined into navigation functions which express a more complex behavior such as reaching a goal while avoiding obstacles. In addition, we design a navigation task which is able to consider lighting conditions without any previous knowledge of the light in the scene and solely relying on the visual perception.

Thesis Outline

This thesis is structured as follows. Chapter 1 provides an overview of previous works related to this subject. It describes what solutions were proposed to address the steering problem and how this thesis improves upon that. Chapter 2 describes in detail how agents perceive the environment and how the visual features are extracted. Agent's are equipped with a visual sensor that allows the extraction of visual features. We describe the relevance of these features and how they relate with the motion of

the agent. Chapter 3 defines a navigation framework that uses visual information. It defines different navigation tasks using the visual features and how to adapt the agent's motion accordingly. We present two different control loops that use different perceptual information and result in different behaviors. Chapter 4 shows several examples of our control loops in different scenarios. We compare our control loops to previous models and analyze the effect of parameters. Chapter 5 provides a discussion upon the results of our framework, its limitations and future work. Finally, this document ends with a conclusion to summarize the contributions.

CHAPTER 1

STATE-OF-THE-ART

This thesis seeks to steer virtual characters in a dynamic environment using visual information, therefore, simulating the human vision-locomotion loop. This problem is directly related to the fields of Computer Graphics and Robotics as they both attempt to achieve agents or robots capable of autonomously navigating an environment. The main distinction between a robot and a virtual agent is the way they perceive the environment as robots need to be equipped with sensors and filter the information while a virtual agent may directly access the information from the simulation. In this PhD we attempt to reduce this distinction by providing agents with virtual sensors similar to what a robot may have access. This work also relates to Crowd Simulation as this discipline works in a very dynamic environment where agents need to take into account the motion of other agents and moving obstacles. Furthermore, our work seeks to be a step forward towards human-like navigation as we propose a control loop based on vision and optical flow, information that are well known to be used by humans and animals to navigate an environment. Therefore, this work is related to experimental psychology, the community that studies how real humans behave.

1.1 Character Motion: planning and controlling the locomotion of characters

The simulation of virtual characters is a problem with many variants. Character steering consists on computing the trajectory of agents from an initial state to a final destination. When generating trajectories it is also important to be able to generate a feasible set of animations and behaviors. The resulting trajectories are required to satisfy constraints such as avoid obstacles, make the character jump to overcome a gap in the ground or approach an object in a certain way to grab it (similar to how real humans react). Many approaches have been proposed to plan the motion of agents in

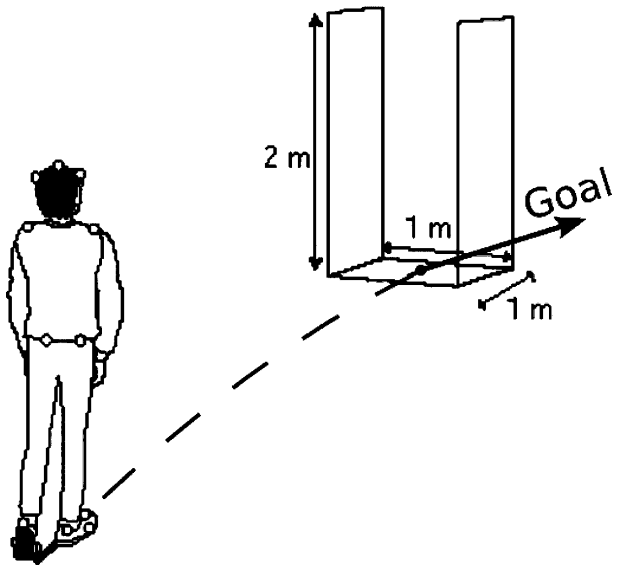


Figure 1.1 – Illustration from [ALHB08]. Example of goal reaching experiments of real humans with motion capture devices.

a variety of environments.

To be able to realistically simulate human navigation we need to study how they actually traverse their environment, what kind of trajectories they produce and how they reach their goal. Several authors [BC89] [ALHB08] [HPA+07] have studied how they move using motion capture techniques to record the human position over time as illustrated in Figure 1.1. The idea is to study how humans change their path to reach a goal in a particular orientation starting from different initial positions. Using these data they generate new trajectories with similar properties to the real data recorded.

Another problem is how a virtual human should traverse the scene given a complex environment and limited amount of indication as actions or goals. As depicted in Figure 1.2, the whole animation needs to be generated to take into account the terrain, obstacles and the requested actions. Many approaches have been proposed to solve this problem [SH07] [CLS03] [KL00] typically using finite state machines and a pool of simple animations. These animations are then blended to produce the final result of agent. Some approaches focus on footstep planning [CLC+05], the simulation of many characters in dynamic environments [LK05] or cooperative planning [EAPL06].

In searching for higher realism to move characters, human models have also been proposed to represent their locomotion. Boulic et al. [BTT90] proposed a human skeleton model capable of walking using a cinematic model in real-time which was then used

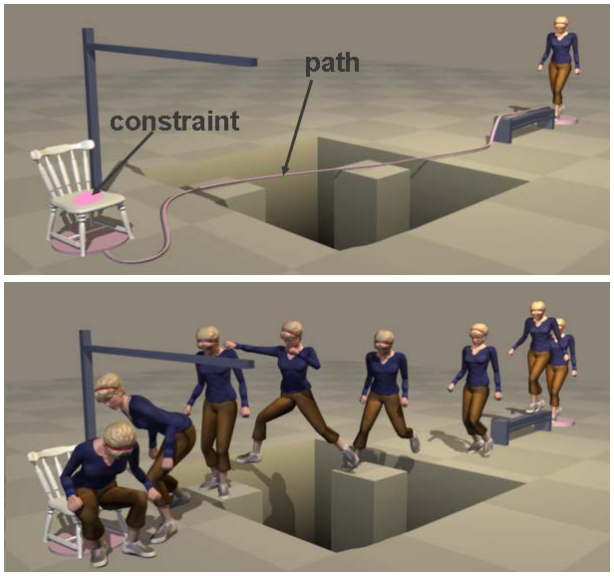


Figure 1.2 – Illustration from [SH07]. Example of motion planning. In the upper image the user defines a path or only a set of way-points with a set of actions that need to be completed. The lower image shows the generated animation.

by Glardon [GBT04] to develop a human walking engine. This allows the generation of better walking animations that adapt to the motion of the character. Later Baerlocher [BB04] proposed a skeleton model and reproduced various poses using inverse kinematics while satisfying the constraints in the joints of the human body.

Discussion

Typically, agents have access to a precise representation of the 3D environment to plan the entire motion which allows to easily generate good-looking trajectories but it misses the perceptive capabilities of real humans. In this work, we describe a reactive behavior, our agents have only limited knowledge of the environment from their visual sensors that reproduce human the vision system. Existing reactive behaviors are described in Section 1.2

1.2 Crowd Simulation: steering agents in highly dynamic environment

Crowd simulation attempts to reproduce the behavior of real crowds in a virtual environment. This problem has been typically addressed in two different ways. First, the so-called macroscopic models, where crowds are modeled as continuous fluid-like matter [NGCL09] [TCP06] and they obey fluid dynamic equations. These methods provide an accurate high level description of the crowd (density, flow, etc) in high density scenarios. However, real crowds exhibit behaviors as complex as the humans that are part of it. In order to accurately model the details of a crowd it is necessary to model human behavior and locomotion system individually, these are the microscopic models. Several approaches have been proposed to imitate humans [DDH13] such as physically-based models, velocity-based, behavioral models, rule-based models and more.

Physically-based models

Physically-based models propose considering virtual agents as point particles. Then, the interaction of an agent with the rest of the world is modeled with forces. Tasks such as goal reaching are modeled with an attractive force and collision avoidance as a repulsive force. Social forces [HM95] is a well-known example of this kind of model. Several improvements to this model have been proposed. Bouvier et al. [EB97] proposed a generic particle system to demonstrate its applicability to model crowds. These models are capable of generating large crowds in real-time and can be easily calibrated to produce different types of crowd behavior. With the capability of simulating large crowds some studies focused on model panic scenarios [HFV00].

Further approaches such as predictive models [KHvBO09] improve realism in physically-based approaches. Future possible collisions are predicted, then the repulsive force is computed for this situation and it is applied in the present configuration as shown in Figure 1.3.

Later Karamouzas et al. [KSG14] designed a repulsive interaction between agents to reproduce a power-law pattern present in the evolution of time-to-collision over time when real humans perform collision avoidance.

Kuffner et al. [KL99] used the idea of label images and introduced the use of mem-

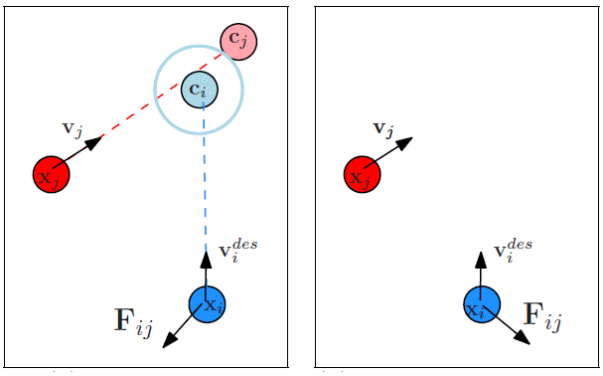


Figure 1.3 – Illustration from [KHvBO09]. The right panel shows the typical repulsion in a social forces model. The left panel shows the approach presented by Karamouzas et al. where the repulsive force is computed at the position of the future collision.

ory and learning. Objects are registered when they are first seen and forgotten after some time if they are not seen again. Therefore, the agent builds an internal representation of the world while navigating it.

However, human navigation is an extremely complex system that exhibits behaviors that a simple physics model cannot reproduce. These models rely on a very simple representation of the environment and agents and simple repulsive and attractive forces produce simple behaviors.

Rule-based models

A way to obtain a greater variety of behaviors for virtual agents is to combine different tasks. With this in mind Reynolds [Rey87] [Rey99] pioneered rule-based models. The interactions of agents consisted of a set of rules to perform a complex navigation task. For example Figure 1.4 illustrates the multiple rules that define flocking, which is a crowd moving in a flexible formation. Figure 1.5 shows multiples behaviors (collision avoidance, target reaching, fleeing, etc.) that need to be combined to enable agents to complete these tasks at the same time. Many of the rules proposed for collision avoidance and other tasks may be contradictory with each other and need to be blended in particular ways [SKH+11].

Shao W. [ST05] proposed a complete model taking into account multiple needs and desires to simulate humans in a scene for generating crowd behavior. Each agent has a set of priorities that drive his behavior while performing collision detection and avoidance with other agents and static objects. Static obstacles are detected by performing

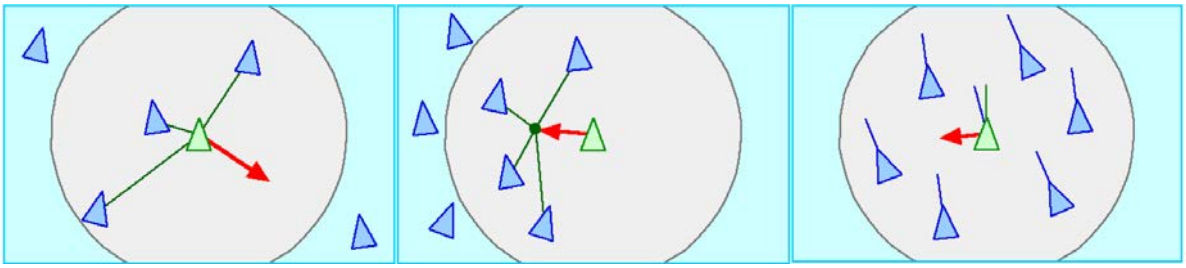


Figure 1.4 – Rules defining flocking proposed by Reynolds [Rey99].

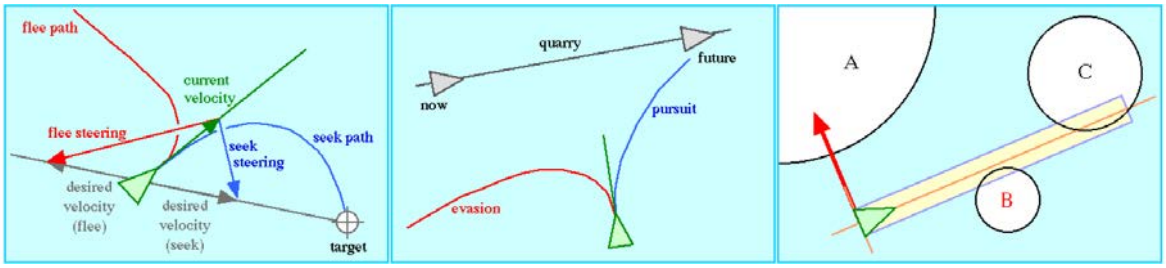


Figure 1.5 – Rules defining different behaviors (seeking, flee, collision avoidance, etc.) proposed by Reynolds [Rey99]

ray tracing with the geometry of the world. Then the agent selects a path with the farthest collision distance. Regarding dynamic obstacles, a set of basic interactions with other dynamic obstacles are modeled and agents select a combination of these rules to solve every situation.

Moussaïd et al. [MHT11] included aspects from cognitive science to study possible dangers within a crowd. They simulated human perception by computing the time to collision with objects in the scene across the horizontal field of view. Then they used an heuristic model to select a new direction with greater time to collision that does not deviate too much from the desired destination of the agent.

Kapadia et al. [KSHF09] [KSH+12] proposed affordance fields. These fields evaluate how easy it is for the agent to navigate the local vicinity. They vary their value according to obstacles and proximity to the goal. This information is used to plan the trajectory of the agent.

In most models high densities in crowds usually result in congestions and blocking situations. Pelechano et al. [PAB07] proposed a model to handle low and high density crowds. They modeled the psychology of the agents of the crowd enabling communication to solve dead-locks.

Velocity-based models

Another well known algorithm used in crowd simulation is RVO [vdBLM08] [vdBGLM11]. It is an efficient collision avoidance approach that finds optimal collision free velocities for agents in the crowd in the near future. It is based on the concept of velocity obstacles VO_B^A of an agent B to an agent A. This is the set of velocities of A that cause a future collision. Figure 1.6 illustrates this concept. There are many variations of the RVO formula and velocity-based models [PPD07].

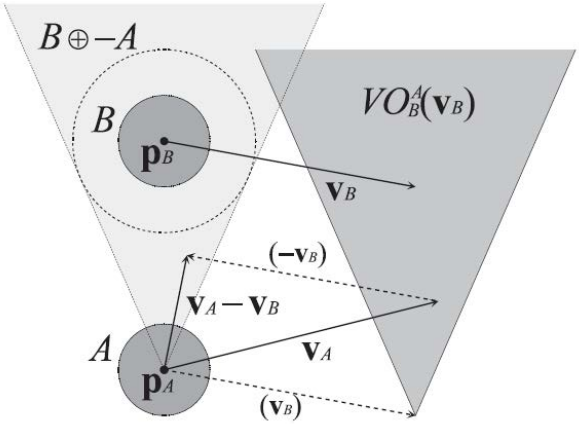


Figure 1.6 – Velocity obstacles. $VO_B^A(v_B)$ is the set of velocities of agent A that causes a future collision with agent B. [vdBLM08]

Other approaches such as the one proposed by Patil et al. [PvC+11] used guidance fields to facilitate crowd motion and prevent or reduce congestions while using traditional approaches. These fields are annotations in the world that guide agents through a particular path depending on their destination. These fields can be manually edited by the user or extracted from real crowd data.

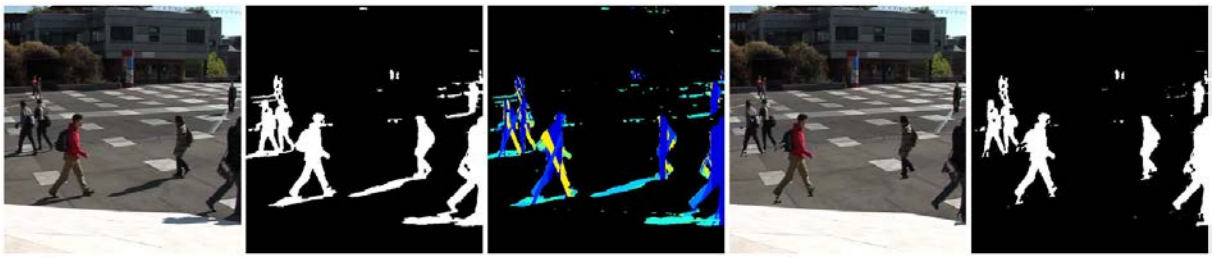


Figure 1.7 – Extraction of human trajectories from camera recordings performing background and shadow removal. [AGS+11]

Real crowds have been studied many times to discover new avoidance approaches [POO+09] [KO12]. Other approaches [JCP+10] generate crowd behavior from existing data. Lerner et al. [LCL07] proposed a data-driven model for agent interaction. Ahn et al. [AGS+11] proposed a method to extract of crowds from camera recordings (see Figure 1.7) and used these trajectories to generate paths for virtual humans. With this set up, collision detection is easy as the trajectories of the agents are known in advance. They handle collisions by shifting the trajectory of the agents. Then, they proposed an improved model [AWTB12] and performed user studies to evaluate the perceived realism of the crowd in VR with respect to other models.

Discussion

Many other crowd systems have been proposed [ANMS13]. These models however, focus on achieving high performance with a large amount of agents. For this reason they rely on simple representation of the environment and the agents themselves. In order to improve the realism of the simulation we require to study what information relying on vision real humans use to navigate an environment.

1.3 Synthetic Vision Character Control

Every algorithm that enables a virtual agent navigate a virtual environment requires a mean of perceiving their environment. Real humans use their senses (mainly sight but also sound) to detect obstacles and avoid them. In character animation this problem has usually been addressed by processing entities within a certain radius of the agent. Therefore, agents are able to perceive more information than real humans could possibly perceive at the same time. This results in a lack of realism as real human navigation and interactions depend on what they can perceive and what they cannot. Real humans perceive the color of the environment. They are then processed in order to adapt the motion of the individual to reach a certain goal. In practice this is a very computational expensive operation and it has not been completely solved yet. For this reason virtual characters typically rely on various kinds of approximations. The steering algorithms proposed in this thesis fall into the vision-based control category.

Renault et al. [RTT90] designed a synthetic vision model to enable navigation. In this method a virtual camera draws the scene from the point of view of the agent.

However, instead of capturing the color of every pixel, they captured the semantic information of the scene, every pixel holds a label that identifies the underlying object. In addition, the depth map is also processed. Using this information agents can perform local navigation in a given environment. Noser et al. [NRTT95] later combined this idea with path planning.

In addition to humans, animal locomotion can also be simulated by using synthetic vision. Tu et al. [TT94] searched to reproduce the entire locomotion system of fishes from the motion of the body to the visual stimulus used to navigate. The synthetic vision in this work consists on queries to the simulation for every element in the field of view of the fish. The fish could then access to the distances, shape, size and identities of the perceived objects. Later Terzopoulos et al. [TR95] introduced the use of color images perceived by the fishes. They reproduce the fish color perception by using multiple cameras with different resolutions to reproduce the distortion of fish eye lenses. Then the color images are analyzed by detecting color histogram patterns. Every pattern corresponds to a specific type of object in the environment allowing the fish to identify the perceived entities. They also introduced the use of optical flow to stabilize the motion of the fish.

Courty et al. [CMA03] used a synthetic vision approach to simulate gazing behavior. Their approach use images to detect interesting features (salient) that may draw the attention of the virtual character in order to reproduce how humans analyze their environment.

Other works attempted to simulate large crowds in real time while still reproducing human perception limitations. Silva et al. [SLC10] proposed a GPU method to roughly simulate occlusion in crowds with a large amount of agents. Based on the flocking behavior presented by Reynolds [Rey87], for every agent, they process the first N agents within the field of view of the agent. Then the rest of agents are assumed to be occluded by the crowd and ignored.

This thesis builds on top of some recent synthetic vision models for character steering. Ondrej et al. [OPOD10] presented a collision avoidance method using synthetic vision. They render the scene from the point of view of the agent but instead of capturing color information they capture the variation of the bearing angle of each pixel and time-to-interaction. Figure 1.8 illustrates the relevance of the bearing angle regarding future collisions. They then update the velocity of the agent in order to obtain a collision-free trajectory. Later Wu et al. [WJDL13] applied the model presented by

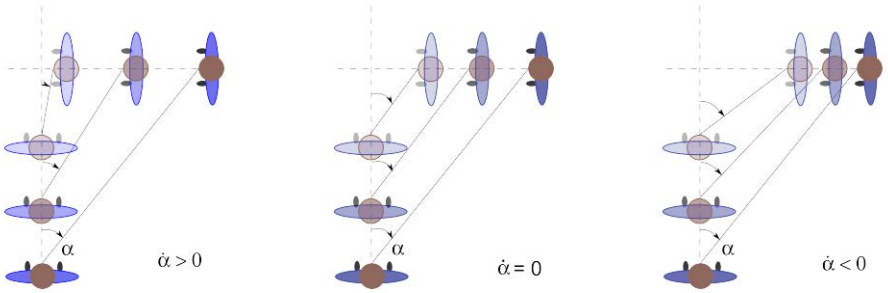


Figure 1.8 – Image by Ondrej et al. [OPOD10]. This figure illustrates the importance of the bearing angle α to prevent collisions. The cases where the bearing angle increases or decreases over time (left and right situation) indicates no collision will occur. When the bearing angle remains constant (center) this means that a collision will occur.

Ondrej to groups and studied to apply synthetic vision to collective motion.

Finally one of the most recent approaches in synthetic vision steering was presented by Dutra et al. [DMCN+17]. They presented a synthetic vision model that perceives distance and time to collision for every pixel in the field of view of the agents. With this information they build a set of cost functions C that evaluate the risk of collision. Then the speed and direction of motion of the agent is updated to minimize the cost functions using a gradient descent approach. Figure 1.9 shows an overview of the control loop used to steer agents.

Discussion

Several solutions have been proposed to steer virtual agents using various kinds of synthetic vision methods. However, very few of them actually use visual information resulting in a big distance from the information used by real humans to navigate. We seek to close this gap by introducing the use of optical flow and depth maps for navigation. Optical flow is the perceived motion of the scene from an individual point of view and it is known to play an important role in human navigation.

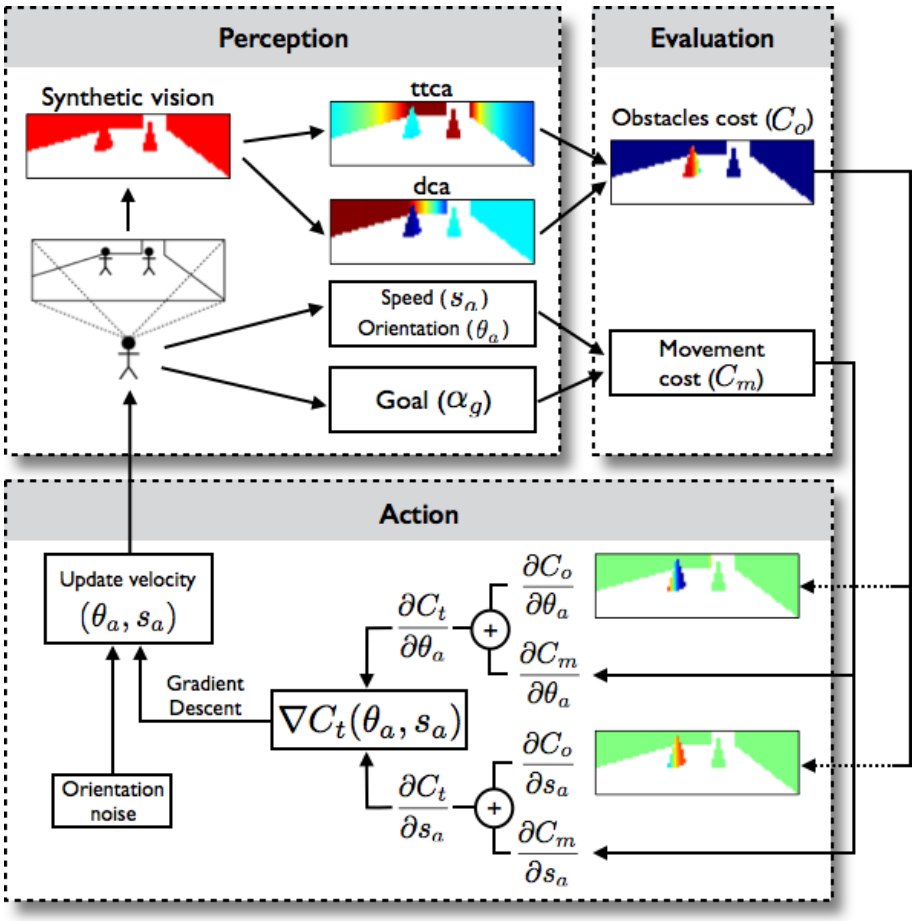


Figure 1.9 – Image by Dutra et al. [DMCN+17]. This figure illustrates the control loop presented by Dutra. First the synthetic images are generated in the perception stage containing time-to-closest-approach ($ttca$) and distance-of-closest-approach (dca). Then this information is combined in the second stage to evaluate the risk of collision. Finally, the cost functions are optimized by updating the speed s_a and orientation θ_a of the agent.

1.4 Experimental psychology: the human visuo-locomotor loop

In the experimental psychology field, there have been numerous studies about the internal relation between vision and human locomotion. In particular and related to this thesis is what kind of visual information humans use to adjust their trajectory when navigating an environment. What clues alert the human visual system about possible collisions and how to decide a new trajectory.

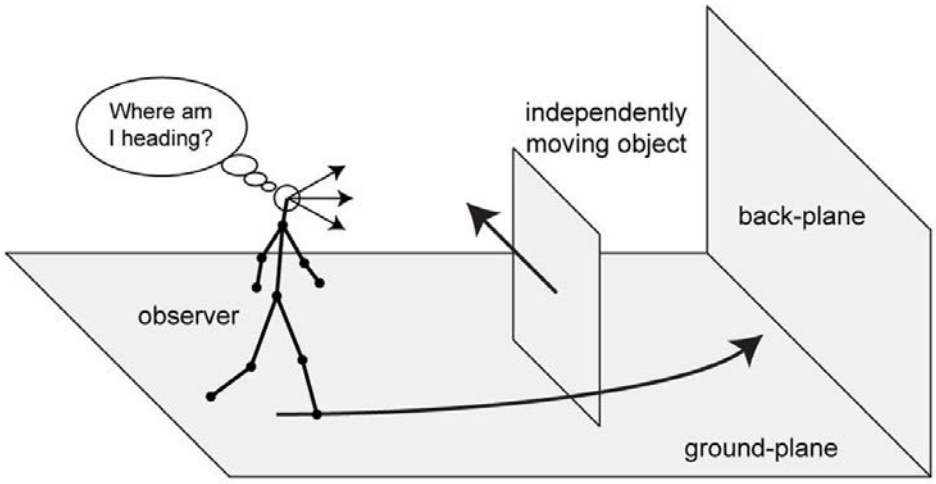


Figure 1.10 – Experiment with moving obstacles [RN13].

Studies suggest that humans perform a combination of motion planning and reactive behavior [GLRLR+07] to navigate various environments. This was suspected for some time as there are experiments suggesting prior motion planning [PTI04] while other point towards reactive behavior. In particular, Gibson [Gib86] suggested that optical flow was used by humans to estimate self motion and navigate environments.

Optical flow is defined as the apparent motion of objects projected in the retina. Experiments by Warren et al. [WKZ+01] suggested that it plays a key role in human navigation. They performed several experiments to record humans moving in an environment as depicted in Figure 1.10. The optical flow generated by an object in this situation can be analytically estimated and analyzed. Experiments were made first with static [WMBM91] and latter with dynamic obstacles [WJS95]. They found that the expansion center of objects, a particular point in the optical flow map, was a relevant feature that hinted the human visual system about possible collisions.

This problem is still being studied and experiments are revised to explain contradictory results. Raudies and Neumann [RN13] reviewed previous experiments [WJS95] [RH96] [FK02] as shown in Figure 1.10. The resulting trajectories yield contradictory conclusions in this examples. Raudies and Neumann discussed that the experimental conditions of the different experiments allow the use of different cues for object detection by the human visual system which may result in different trajectories.

Discussion

There is still a debate regarding the role of optical flow in human navigation but many experiments suggest that it is an important map perceived by the human visual system. Our work simulates the human perception system to generate optical flow in virtual environments and defines control laws to link the perceived information with locomotion. The accuracy of the resulting trajectories compared with the motion of real humans is out of the scope of this thesis. However, this provides a new tool to evaluate how real humans behave with different visual information.

1.5 Robotics: implementing cyber visuo-locomotor loops

The robotics community has explored many methods to acquire visual information and process them to influence the motion of a robot. In particular, there has been a study of the use of optical flow which is a dense map that encodes a lot of information about the motion of the environment, from self motion to the relative motion of objects.

The first step is to control geometrical points or other visual features obtained through the robot sensors to obtain a specific configuration. Chaumette et al. [CH06] present how to approach the control problem using visual features. To accomplish a task it is required that a certain image feature $\mathbf{s}$ achieves a certain state $\mathbf{s}^*$. Then the error function is simply defined as $\mathbf{e} = \mathbf{s} - \mathbf{s}^*$. To minimize $\mathbf{e}$ the dependency of $\mathbf{s}$ with respect the control variables $\mathbf{v}_c$ is represented by the interaction matrix L, so that $\mathbf{s} = L\mathbf{v}_c$. To ensure an exponential minimization of the error function $\dot{\mathbf{e}} = -\lambda\mathbf{e}$ the control variables have the expression $\mathbf{v}_c = -\lambda L^{+}\mathbf{e}$.

Cretual et al. [CC01] designed two control schemes for camera positioning. They measured the motion in the image (optical flow) due to the camera's own movement and use this information to perform certain tasks. The camera is able to align its position with a plane or follow a 3D surface. These tasks can be accomplished without previous knowledge of the scene as the perceived motion is sufficient to guide the camera.

Controlling visual features can be used to achieve corridor following. Faragasso et al. [FOPV13] designed a control model based on visual information. The robot sensors detect the edges of the corridor that converge in the vanishing point. Keeping these edges at the same distance in the field of view allows the robot to navigate through the middle of the corridor. Analysis of the perceived edges allows them to handle turns or

T junctions as in this situations the edges of one or both sides of the image eventually disappear indicating the corresponding situation.

Many authors proposed visual and control system for aerial robots. Grabe et al. [GBR12] used optical flow to estimate the self motion of the robot. Their idea is to take into account point in the dominant plane of the image (such as the ground) then use this optical flow to determine the motion of the robot. Lee and Song [SJ04] used optical flow to obtain a more robust estimation of self motion and even to compensate the trajectory of the robot after an external perturbation in its trajectory. Hérissé et al. [HHMR12] developed a controller to allow aerial robots to land safely on mobile surfaces. A camera is aimed towards the ground and the optical flow is computed from the images of this camera. Then the optical flow is used to stabilize the robot with respect to the mobile landing platform.

Detecting obstacles is a key feature in robot navigation and optical flow has been used to solve this problem. Braillon et al. [BPCL06] designed a robot that uses optical flow to detect obstacles. As the speed of the robot could be estimated accurately, the optical flow generated by the ground was known. Therefore, pixels with optical flow different from the one expected indicate the presence of an obstacle. A similar approach was considered by Ferro et al. [FPCV16] where discrepancies between expected and computed optical flow point towards the presence of obstacles.

The previous approaches method detects obstacles by detecting discrepancies in the expected motion perceived by the robot. An alternative to this is to detect the dominant motion in the scene. Odobez and Bouthemy [OB94] applied this idea to design an obstacle detection algorithm. The dominant motion produced by the background is detected and motions of objects that deviate from this are considered obstacles.

Fernandez et al. [FCCMCMT10] designed a surveillance robot to detect humans and their motion using infrared cameras. While the robot is static a simple image subtraction operation is performed so that differences between consecutive frames are detected. When the robot is moving, optical flow is computed using the Lucas and Kanade algorithm [LUC81] to detect humans.

Griffiths et al. [GSC+06] designed a miniature aerial device (MAV). They use a laser based system to detect obstacles in front of the vehicle. The MAV adjust its trajectory until the obstacle is no longer detected ensuring a collision free trajectory. In addition, the MAV has cameras looking at the sides of the vehicle. The images from these cameras are then processed to compute the optical flow. This is used to prevent lateral

collisions and stay close to the center of a corridor.

Hrabar et al. [HSC+05] proposed a set up for navigation in land and air. The robot consists on two frontal stereo cameras that capture the scene in front of the robot and two lateral cameras capturing the scene at the sides. The stereo cameras are used to reconstruct the 3D shape of the world and detect any obstacles in the way of the robot. The lateral cameras are used to compute the optical flow at the sides of the robot. Then the robot turns to balance the flow from both cameras (e.g. moving through the center of a corridor).

Zingg et al. [ZSWS10] used optical flow for collision avoidance. They proposed a mobile aerial robot with fish eye cameras pointing towards the lateral sides of the robot with respect to the direction of motion. In a static scene, optical flow is proportional to the distance of the perceived objects and the speed of the robot. As the speed of the robot is known, it is possible to obtain the depth of perceived points. Then the robot tries to keep a certain distance with all objects perceived.

A relevant point in detecting obstacles is the expansion center of an object or Focus of Expansion. Nelson et al. [NA89] studied how different patterns in optical flow correspond to the 3D world, in particular, the divergence pattern that indicates the presence of an obstacle in the field of view. Souhila and Karim [SK07] explored the use of expansion centers in optical flow to detect obstacles and the time to collision to them. Then they proposed a robot navigating using a optical flow balance strategy to maneuver it through the scene. Sundareswaran et al. [SBC96] proposed a control solution to align a camera mounted on a robot with its direction of motion. They use the optical flow to compute the Focus of Expansion of the scene. Then, the robot moves so that the FOE is aligned with the center of the image.

Discussion

The robotic community has proposed many approaches to solve the robot navigation problem, using visual servoing, optical flow and other types of sensory information. However, there has not been attempts to use optical flow for highly dynamic environments. In our work we propose a navigation model using optical flow to navigate dynamic environments with multiple obstacles and little knowledge of the scene.

CHAPTER 2

PERCEPTION AND VISUAL FEATURES

Humans navigate their environment using their perception and knowledge of the environment. Experiments [GLRLR+07] suggest that a certain combination of planning and reactive behavior affect the way humans move in a given environment. Distances and personal space [GLRM05] is taken into account to obtain a prior plan of their trajectory. Then, humans rely on their visual perception system to adapt their trajectory to the details or unknowns of the environment [TT94] [WKZ+01]. In particular, humans are known to rely on optical flow, the apparent motion of objects in the field of view. Optical flow contains information regarding the relative motion of every object in the scene with respect to the observer. Using this, humans are capable to deduce their own motion with respect to static objects (e.g. the ground) as a reference and the relative motion of dynamic obstacles like other humans. The pattern of optical flow also contains information about the remaining time until collisions. This allows humans to adapt their trajectory to avoid obstacles or reach a goal. Therefore, optical flow contains a large amount of information that can be used as the basis of a perception scheme.

The human visual perception system is not limited to optical flow but it can identify objects or perceive distances. We seek to simulate the visual perception of a human and find those features that may allow a virtual agent to navigate a virtual environment based on this visual information. Therefore, we design a perception pipeline that is able to record different types of visual information as seen in the first column of Figure 2.1, color images emulating the human eye, optical flow recording the apparent motion of objects in the field of view, depth as the distance of every point in the field of view and semantic information as the classification of every perceived object. We propose an agent model equipped with a virtual camera at the height of the eyes. This camera renders the scene from the point of view of the agent and records the various images required for navigation. First, we have color images which consist of a normal render of the scene from the agent's point of view. Then, there is synthetic optical flow which is a special render of the scene that encodes the velocity of every point in the image.

This is done using a specialized shader. Depth information is also acquired together with synthetic optical flow as the variation of the flow depends directly on the distance between the agent and the corresponding point. Finally, the cameras are capable of rendering semantic information, again using a special shader to assign a label to every object in the scene. This is necessary in cases were the quality of optical flow is reduced such as when a numeric solver is used to compute the optical flow from a sequence of images.

Optical Flow

Optical flow is defined as the apparent motion of objects perceived by an observer. In the case of humans it represents the variation over time of colors perceived by the retina. It is a 2D vector field for each point in the field of view of a camera and encodes how the camera and objects move. Different objects moving differently with respect to the observer produce distinct image motion which allows object segmentation by analyzing continuity in the flow map as done by [OB98]. In order to use optical flow for navigation we first need to remove any contribution from rotational motion, in particular, due to the observer's change of orientation. Real humans are able to naturally compensate this effect through the vestibulo-ocular reflex [Ang04]. The resulting optical flow map contains the linear motion and exhibits for each object a radial pattern around a particular point, the focus of expansion (FOE). The location of the FOE with respect to the object depends on the relative velocity of the object with respect to the observer. This property can be used to detect risks of collision. The novelty of this work is the direct manipulation of optical flow to solve navigation problems. Therefore, optical flow is a rich feature map that needs to be processed to extract the relevant features for navigation. In this thesis we explore two methods for exploiting optical flow, synthetic and numeric. Table 2.1 summarizes the traits of each method.

The first method of computing optical flow is called synthetic flow. This approach uses the simulation data, that is, the known relative velocity between agent and objects, to generate an artificial version of optical flow that encodes the projected motion of objects in the field of view of the agent. The main advantage of this method is its accuracy, as optical flow can be computed with arbitrary quality and precision. This allows to perform object segmentation using the optical flow alone. The downside of this approach is that it is not the real flow that humans perceive but an approximation and it is limited to a virtual simulation.

The second method of computing optical flow is to use a numerical solver. Numerical optical flow is computed by finding correspondent pixels in a pair of consecutive color frames which is closer to the way humans perceive optical flow. Usually, this solution generates flow with noise and lower accuracy but it is not limited to virtual simulation and is also usable in real environments. The main downside to this approach is the higher computation time required with respect to synthetic optical flow.

Synthetic Optical Flow	**Numeric Optical Flow**
Faster Computation Time	Higher Computation Time
Highest Quality and Accuracy	Higher Realism
Limited to virtual environments	Usable in real environments
Easy to filter rotational flow	Noisy and hard to filter rotational flow

Table 2.1 – Comparison of synthetic and numeric optical flow.

Perception Pipeline

This chapter details a new agent model detailed in Section 2.1 that uses a new perception pipeline that allows our agents to perceive and process visual information. Figure 2.1 shows an overview of the three steps involved, perception, image segmentation and visual feature extraction. The perception stage gathers the visual information for later processing. Our agents have access to 4 types of visual data: color images, numeric or synthetic images, depth map and semantic information. The details on how each of these images are obtained is detailed in Section 2.1.3. The second step is image segmentation. In order to react to obstacles or other objects in the scene the agent needs to detect them in its field of view. In the case of numeric optical flow this is done thanks to the semantic map, that identifies objects in the scene. When using synthetic optical flow we search for discontinuities in the flow to find the boundaries of the different objects. The details of this stage are described in Section 2.2.1. Then, we move to the last column in Figure 2.1 the visual features stage. After image segmentation the agent has detected a number of objects O_i. For each of these objects a set of visual

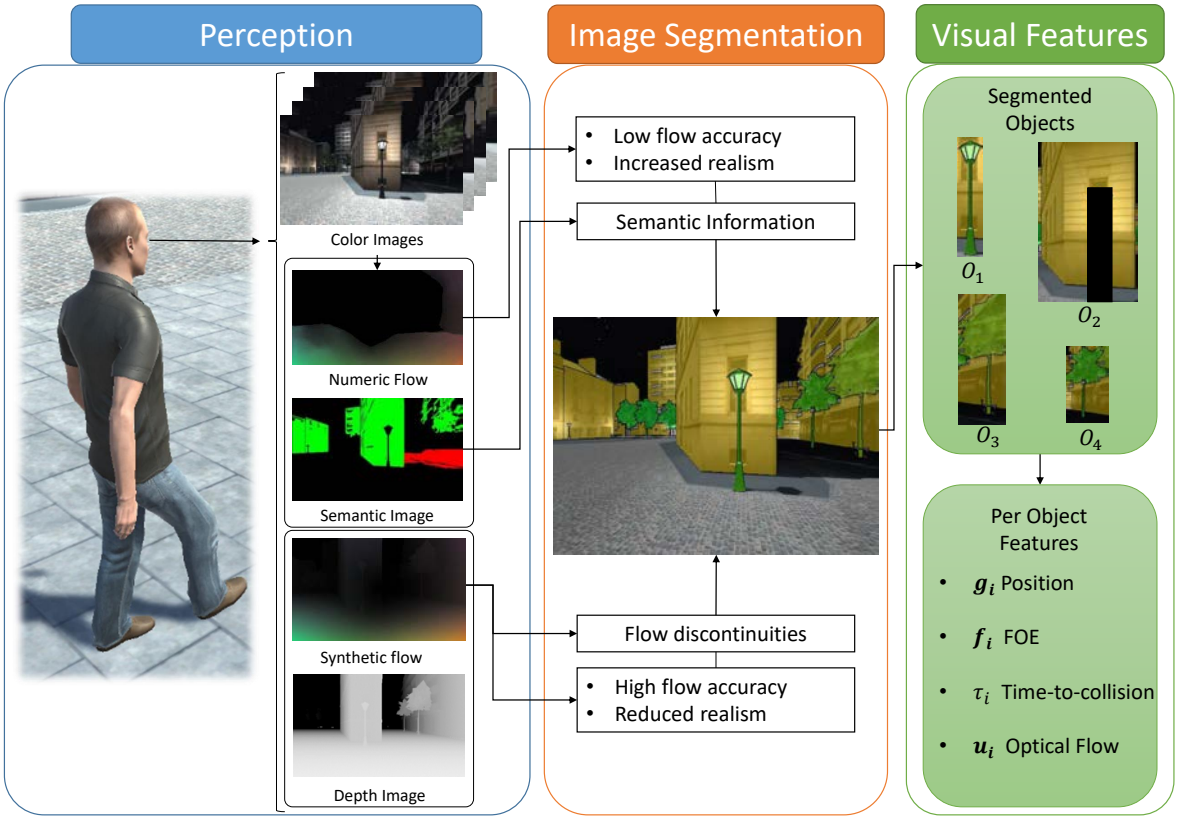


Figure 2.1 – Scheme of the agent's visual perception system. Agents can record various kinds of visual information which are processed to extract individual objects and compute their visual features.

features are computed using the optical flow map such as the Focus of Expansion and time-to-collision. All visual features and their relation with the agent's control variables are developed in Section 2.2.2.

2.1 Human Model

This section describes the characteristics of the agents considered in this thesis. We present how they are controlled, how they capture visual information of the scene and the visual features that can be extracted from their perception. Agents are equipped with a virtual camera simulating eyes. This camera is used to capture color images, optical flow, depth map and semantic information. The resulting images are available to the agent at every iteration of the control loop. First, we describe the con-

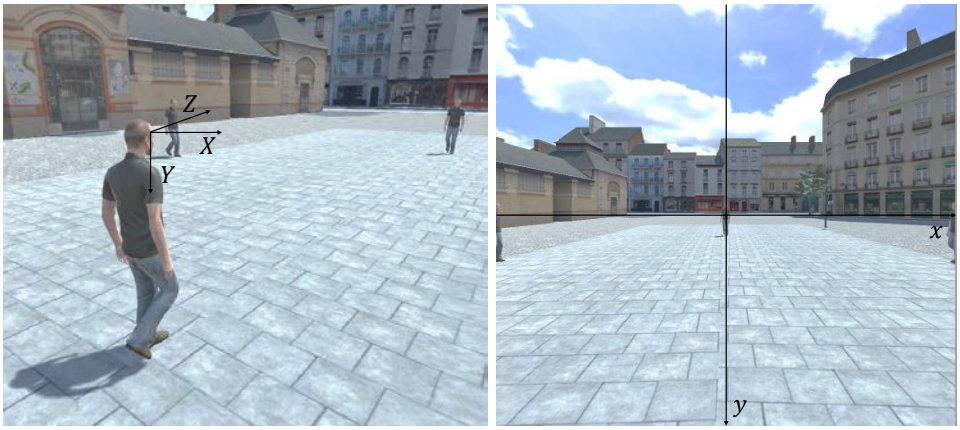


Figure 2.2 – Coordinate frames. (left) 3D coordinate frame, (right) 2D image coordinate frame.

trol variables of the agent in Section 2.1.2 and then we list and explain the different sensory information they have access through virtual perception in Section 2.1.3.

2.1.1 Coordinate Frames

All the variables and notation shown in this thesis are expressed in the local frame of the agent. Figure 2.2 shows the 3D local frame (left image) of the agent. This frame is located at the position of the agent's eyes and is used to express any 3D variables such as velocities and positions. The Z component is always pointing towards the direction of motion of the agent and the Y vector points towards the ground. The right image shows the image space local frame and is used to express any 2D variables such as pixel position, optical flow and all visual features. It is a frame in normalized image coordinates with the origin at the middle of the image with the vertical vector pointing towards the ground.

2.1.2 Control Variables

The control variables of an agent are the inputs to change its motion. They are updated at every iteration of the control loop to enable an agent to accomplish a certain task. We consider an agent with two degrees of freedom: forward acceleration $\boldsymbol{a_c} = (0, 0, a_c)$ and turning rate ω_{cy} around the vertical axis expressed in the 3D coordinate frame of the agent. This control scheme corresponds to the unicycle model.

Figure 2.3 – Agent's control variables. The output of the navigation algorithm updates the acceleration and the turning rate.

Acceleration allows the agent to move faster or slower and turning rate is the angular velocity of the agent around its vertical axis, which allows the agent to change its direction of motion. Lateral motion is not considered in this model. Figure 2.3 illustrates the degrees of freedom of the agent. The control variables are clamped to a maximum value to stabilize the simulation. The maximum value of $\omega_{cy} = 3rad/s$ and the maximum acceleration is $a_c = 1m/s^2$.

2.1.3 Visual Sensors

Agents are equipped with a virtual sensor capable of recording different types of visual information as shown in Figure 2.1. This sensor is a camera situated roughly at the same location as human eyes are and points in the direction of motion. Depending on the specific control loop agents are executing they use different visual input. As already said, there are 4 types of visual information our agents can perceive: synthetic or numeric optical flow, depth map, semantic images and color images. This section details how these images are obtained and describes their relevance for navigation purposes.

Color Image

Color images are full renders of the scene from the agent point of view. These images are a monocular version of human eyes and they have two purposes. First, numerical optical flow is computed using color images from consecutive frames. Second, it allows the agent to detect dark regions. Measuring the intensity of pixels we can use these images to detect regions in the images with poor visibility or lack of information. The corresponding algorithm is described in Section 2.2.1.

Synthetic Optical Flow Image

Optical flow is defined as the apparent motion of objects in the visual field. In a virtual simulation the information of the scene is available to an agent. Therefore, optical flow can be computed by approximating it as the projected motion of every point in the image plane. A dynamic observer with angular velocity ω_{cy} perceives a point j with coordinates $\mathbf{X}_j = (X_j, Y_j, Z_j)$ of an object i, with relative translational velocity $\mathbf{v}_i = (v_{ix}, v_{iy}, v_{iz})$. We neglect optical flow components due to the object's own rotation under the assumption that these rotations are small compared to the other contributions. The observer perceives optical flow according to the following equation [SBC96],

$$\begin{cases} u_j = \left(v_{ix} - x_j v_{iz}\right)/Z_j - \omega_{cy}\left(x_j^2 + 1\right), \\ v_j = \left(v_{iy} - y_j v_{iz}\right)/Z_j - \omega_{cy} x_j y_j, \end{cases} \tag{2.1}$$

with $(x_j, y_j) = (X_j/Z_j, Y_j/Z_j)$ being the projection of point $\mathbf{X}_j$ in the image plane.

Equation (2.1) takes into account the angular velocity of the observer. We need to remove this contribution from the perceived flow. The reason is twofold: first, humans are able to compensate rotational component through the Vestibulo-ocular reflex [Ang04], and second, it allows us to define the FOE and other features derived from it. Then, Equation (2.1) with contributions from the relative linear motion alone is reduced to the following expression,

$$\begin{cases} u_j = \left(v_{ix} - x_j v_{iz}\right)/Z_j, \\ v_j = \left(v_{iy} - y_j v_{iz}\right)/Z_j. \end{cases} \tag{2.2}$$

Synthetic optical flow images are then computed by encoding Equation (2.2) at every pixel. This version of optical flow is of very high quality which allows easy object

segmentation by detecting discontinuities (see Section 2.2.1).

Numeric Optical Flow Image

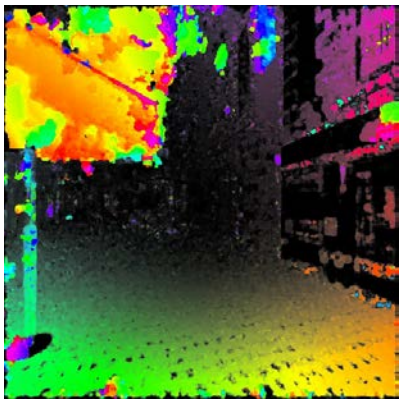
Synthetic optical flow allows the agent to obtain a high quality map that can be used for navigation. However, it is not a realistic representation of the perception of real humans. They can only perceive optical flow from the variation of color in the retina. In order to replicate this, optical flow can be numerically computed from two color images at two consecutive frames. A strategy for optical flow solvers is to search for every pixel of the first image the corresponding pixel in the second image resulting in a displacement map. The solver needs to face some challenges such as how to handle occlusions, new objects appearing in the scene or fast moving objects.

Many approaches have been proposed to solve compute optical flow, from time-consuming stochastic slow models [HB93] to polynomial approximations [Far03] to solve the motion estimation problem. We have tested a number of optical flow solvers easily available as open source projects and present their results in Figure 2.4.

The first method tested is Farneback's [Far03] approach. They approximated the displacement of the neighborhood of each pixel with a polynomial expansion. The resulting flow presents a large amount of noise although is not computationally expensive. Kroeger et al. [KTDVG16] proposed an inverse search method. They find correspondences in small patches in the image and then refine every pixel to obtain the dense map. This approach is the fastest we tested although its results are lower quality than other models. Zach et al. [ZPB07] proposed a variational method with a GPU implementation to reduce computation time. Their approach produces good results however edges tend to be noisy and smaller details are missed. Brox et al. [BBPW04] proposed a warping technique to compute optical flow. Their approach is still computationally expensive and the result is a bit noisy. Deepflow [WRHS13] introduced a descriptor to search dense correspondence using convectional and pooling layers. Their approach produces good results however the computation time of their approach is high. Finally, we decided to use FlowNet2.0 in our experiments. This model is a deep learning approach to compute optical flow and provided the best results. We found this solution to be a good compromise between accuracy, preservation of details and performance.

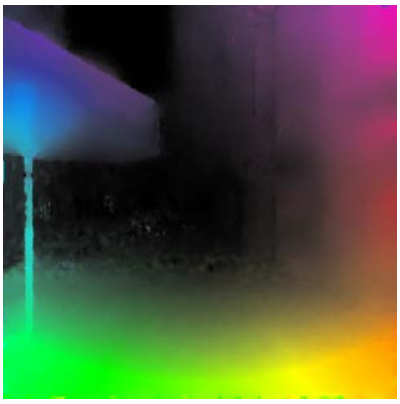
In order to effectively remove the rotational flow, the rotation of the agent is temporarily stopped so optical flow is computed from two frames with only linear motion.

(a) Consecutive color frames.

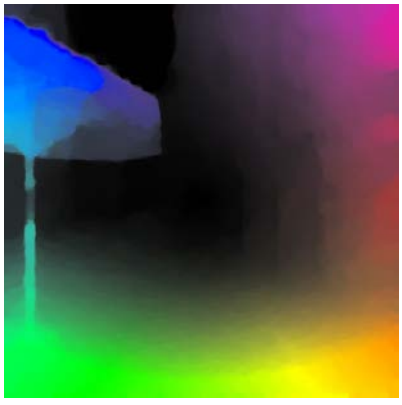


(b) Farneback [Far03].

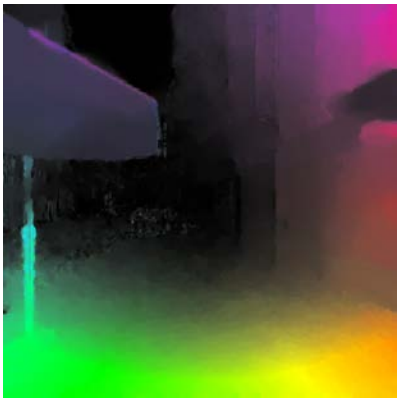
(c) Kroeger [KTDVG16].



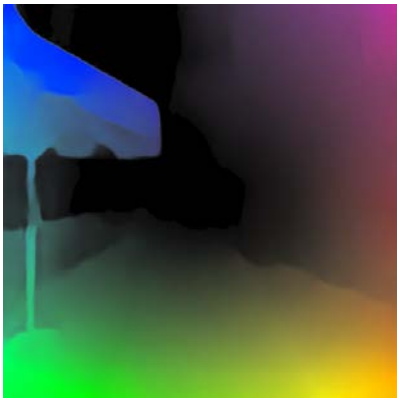
(d) Zach [ZPB07].



(e) DeepFlow [KTDVG16].



(f) Brox [BBPW04].



(g) FlowNet2.0 [IMS+17].

Figure 2.4 – Comparison of multiple optical flow methods.

Even though, this method of computing optical flow does not correspond exactly to solving Equation (2.2), the apparent motion is not always equal the projected motion. However, it provides a good enough approximation for the optical flow from the numerical solver.

Figure 2.5 shows a comparison between numerical and synthetic optical flow. Both solutions provide a similar result despite the fact that numerical optical flow is prone to noise and errors as opposed to perfect synthetic flow. Despite this, it can still be used for character navigation while being closer to real human perception.

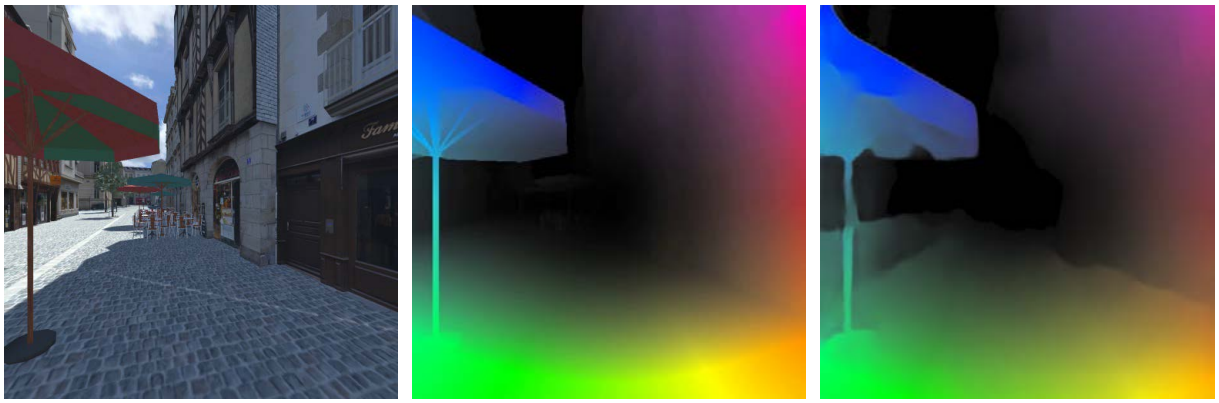


Figure 2.5 – Comparison between synthetic and numeric optical flow. (left) Color image, (middle) synthetic optical flow, (right) numerical optical flow using FlowNet2.0 [IMS+17].

Figure 2.6 shows a second comparison of numeric and synthetic optical flow. In this case, parts of the scene are dark and some areas are barely visible. This generates a noticeable difference of flow in the dark regions of the image as the optical flow solver has little color information. For this reason it is necessary to identify these regions of lack of information as the solver cannot provide reliable information about possible collisions. The synthetic optical flow does not work with light and therefore it provides the same solution regardless of the shadows.

Depth Image

The depth image contains for each pixel in the field of view the depth Z_j of the corresponding 3D point. Many of the visual features described in the following sections have a dependency with the distance between the objects and the agent. Therefore, having the depth map available to the agent allow precise control of the visual features during navigation. Figure 2.7 shows an example of the perceived depth images. However it

Figure 2.6 – Comparison between synthetic and numeric optical flow. (left) Color image, (middle) synthetic optical flow, (right) numerical optical flow using FlowNet2.0 [IMS+17].

Figure 2.7 – Illustration of depth perception. (left) Color image. (right) Depth map in a grey scale representation.

is worth mentioning that it is not an essential point and certain approximations can be made when this information is missing.

Semantic Image

The semantic image allows the agent to identify every pixel in the field of view and its boundaries. It is a render of the scene generating an image that encodes an id at every pixel that identifies the object underneath it. The purpose of this map is to

Figure 2.8 – Illustration of semantic images.

allow easy image segmentation, to extract each object present in the field of view to later compute its visual features. This map is necessary when optical flow quality is not sufficient to perform segmentation. Figure 2.8 shows an example of semantic image.

2.2 Analysis of Perceived Images and Visual Features

This section details how the perceived visual information is processed to enable navigation. First, we present two different segmentation methods that are used depending on the type of computed optical flow (synthetic or numeric) as depicted in the middle column of Figure 2.1. The segmentation process detects a set of objects present in the field of view that need to be analyzed. Then, we describe a set of per-object visual features that are encoded in the optical flow relevant for navigation and we show how they are related to the agent's own motion.

2.2.1 Object Segmentation

The first step to process the captured images is to consider every object present in the scene independently. Figure 2.1 illustrates the segmentation process that in the end retrieves a collection of connected pixels. With every object as a separate entity it is possible to compute the visual features described in the next section. We propose two methods to perform object segmentation in the scene, detecting optical flow discontinuities in the synthetic flow and to use semantic images to identify objects in the field of view.

Optical Flow Discontinuities

The first method consists in using the optical flow image to detect discontinuities. This requires a very high quality optical flow images or a synthetic optical flow computation. This allows the agent to detect different objects in the scene which are processed as rigid bodies. We use the Graph Segmentation algorithm [FH04] from the OpenCV library to detect discontinuities in the flow map.

The ground generates optical flow as the rest of objects, however it should not be considered an obstacle. We assume a flat ground and knowing the agent's speed v_c and height h, the flow has the following expression,

$$\begin{cases} u_j = x_j y_j v_c / h, \\ v_j = y_j^2 v_c / h. \end{cases} \tag{2.3}$$

Optical flow pixels that satisfy this expression are removed.

Semantic Information

The second method requires the semantic image in addition to optical flow. As already said, the scene is rendered from the point of view of the agent, but instead of color, every pixel encodes the id of the underlying object. This allows our agent to distinguish different objects in the scene without the need of an expensive processing of the optical flow. This method is necessary when optical flow quality is low such as in the case of the solution from a numerical solver.

Detecting Dark Objects

Dark objects are detected by scanning the color image. When a pixel is detected as dark, that is, its intensity is less than a particular threshold, its id is replaced by a special id common for every dark pixel. This allows our agents to extract two types of objects, visible and dark.

Dark objects are connected groups of pixels that have been labelled as dark. These objects often lack contrast making the optical flow solver unreliable for these pixels. As a result we cannot compute visual features related to optical flow for these points in the image (FOE, time-to-collision) and only know the location and size of these regions in the image. In addition, we discard dark objects which are in the upper half of the image as we assume there is no collision risk with dark areas above the ground.

Figure 2.9 illustrates a semantic image in a given situation. The objects at the left, tree and buildings are marked as visible objects in green. Dark pixels are marked in red such as the shadow region on the right of the scene. Pixels in black are contours or discarded pixels belonging to visible ground or sky. We also discard any object with a number of pixels less than 0.1% of the resolution of the image.

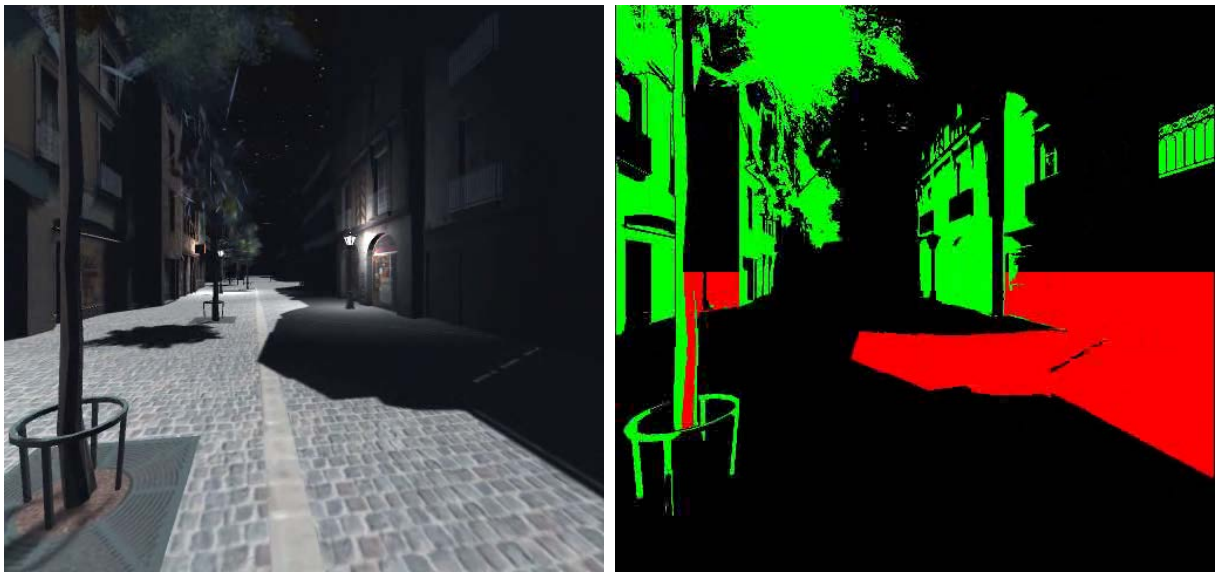


Figure 2.9 – Semantic information. (right) Color image of the scene. (left) Label image, green regions indicate visible obstacles, black regions are visible regions without obstacles or flow information, red regions are pixels labelled as uncertain.

2.2.2 Visual Features

In this section, we expose the relevant features encoded in an optical flow image and show their dependency with the control variables. All features presented in this section are object specific. As explained in Section 2.2.1, objects are extracted from the background. Figure 2.10 illustrates the various features contained in the optical flow image. Note that, as the motion of our agents is restricted to the horizontal plane, we focus on the x component of the features.

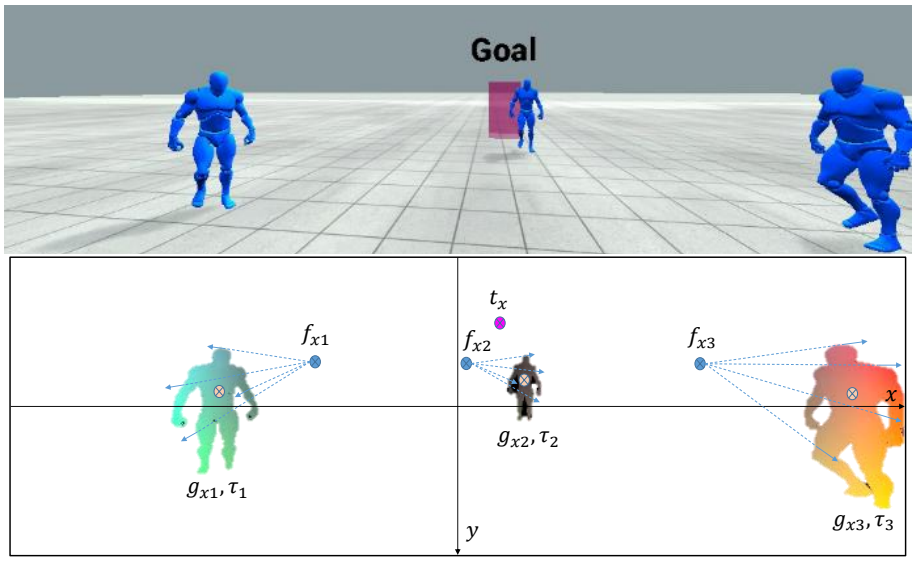


Figure 2.10 – Segmentation of the optical flow image. (top) Color image perceived by the agent. (down) Visual features extracted from optical flow. The goal $\mathbf{t}$ of the agent corresponds to the pink point. For every object the visual features extracted are the FOE $\mathbf{f}_i$ (blue points), time-to-collision τ_i, object center $\mathbf{g}_i$ (orange points) and optical flow vectors (blue arrows), which exhibit a radial configuration around the FOE allowing its computation as the intersection of many lines. The pink object is a representation of the goal and does not generate any optical flow.

Object Position

The first visual feature of an object i is its position g_{xi} in the image. This is the central point to every pixel belonging to that object. From Equation (2.1) Its equation of motion is expressed by,

$$\dot{g}_{xi} = \left(v_{ix} - g_{xi} v_{iz}\right) / Z_i - \left(g_{xi}^2 + 1\right) \omega_{cy}, \tag{2.4}$$

with Z_i being the average depth of the object obtained through the depth map.

This allows our agents to achieve certain goals such as target reaching which consists on having the target at the center of the image.

Focus of Expansion

The FOE is a per-object feature. It is defined as a point with null optical flow despite having a non-zero 3D velocity. It is the center of growth in the image of an object and corresponds to a point moving towards the observer in a collision trajectory. Therefore, controlling the FOE allows us control the relative direction of motion of the obstacle and the observer and prevent collisions. From (2.2) and enforcing $\mathbf{u}_j = \mathbf{0}$ and $\|\mathbf{v}_i\| \neq 0$ the FOE is expressed as,

$$\begin{cases} f_{xi} = v_{ix}/v_{iz}, \\ f_{yi} = v_{iy}/v_{iz}, \end{cases} \tag{2.5}$$

which is the projection of the relative velocity of the object and the observer in the image.

Optical flow vectors exhibit a radial configuration around this point, which allows the FOE to be computed as the intersection of many lines. These lines are defined as supporting point $\mathbf{x}_j$ and direction vector $\mathbf{u}_j$. Figure 2.10 illustrates the flow vectors configuration.

In order to use the FOE for navigation we need to know how the FOE behaves when the agent control variables are changed. From Eq. (2.5) we obtain a temporal derivative of the FOE,

$$\dot{f}_{xi} = \frac{\dot{v}_{ix}v_{iz} - v_{ix}\dot{v}_{iz}}{v_{iz}^2}. \tag{2.6}$$

Using Equation (2.5) this equation can be rewritten,

$$\dot{f}_{xi} = \frac{\dot{v}_{ix} - f_{xi}\dot{v}_{iz}}{v_{iz}}. \tag{2.7}$$

The velocity $\mathbf{v}_i$ is the relative velocity between the object and the observer. As the agent changes its direction of motion, the orientation of the camera changes as well as it always points towards the direction of motion which has to be taken into account to compute $\dot{\mathbf{v}}_i$. Therefore, the composition of the change of direction of motion with angular velocity ω_{cy} coupled with change of orientation of the camera results in the

following expression,

$$\begin{cases} \dot{v}_{ix} = -v_c\omega_{cy} - v_{iz}\omega_{cy}, \\ \dot{v}_{iz} = -a_c + v_{ix}\omega_{cy}, \end{cases} \tag{2.8}$$

where we assume the velocity of the object is constant and only the camera can accelerate with a_c.

Finally, we combine Equations (2.5) and (2.8) to obtain the variation of the FOE with respect to the control variables,

$$\dot{f}_{xi} = -\left(\frac{v_c}{v_{iz}} + f_{xi}^2 + 1\right)\omega_{cy} + \frac{f_{xi}}{v_{iz}}a_c. \tag{2.9}$$

Time-to-Collision

The time-to-collision is defined as the remaining time until an object will cross the plane perpendicular to the direction of motion of the agent. It is expressed as,

$$\tau_i = -\frac{Z_i}{v_{iz}} \tag{2.10}$$

From the FOE and the optical flow it is possible to compute the time-to-collision without any 3D geometrical information nor 3D velocity. For each pixel of the object we define its distance to the FOE as $\Delta_j \equiv \|\mathbf{x}_j - \mathbf{f}_i\|$ and we have [TGS91],

$$\tau_i = \frac{1}{N_i}\sum_{j\in i}\frac{\Delta_j}{\|\mathbf{u}_j\|}, \tag{2.11}$$

with N_i being the number of pixels of object i. It should be noted that although Δ_j and $\mathbf{u}_j$ are pixel dependent values, τ_i is not. When noise is present, in order to obtain a robust result, τ_i is computed as the average for all pixels of the object as expressed in (2.11).

From Equation (2.10) the derivative of τ_i is,

$$\dot{\tau}_i = -\frac{\dot{Z}_i}{v_{iz}} + \frac{Z_i}{v_{iz}^2}\dot{v}_{iz}, \tag{2.12}$$

using $\dot{Z}_i = v_{iz}$ and Equation (2.10) we obtain,

$$\dot{\tau}_i = -1 - \tau_i\frac{\dot{v}_{iz}}{v_{iz}}. \tag{2.13}$$

Finally, combining this equation with Equation (2.8) we have the following expression,

$$\dot{\tau}_i = -1 + \frac{\tau_i}{v_{iz}} a_c - \tau_i f_{xi} \omega_{cy}. \tag{2.14}$$

Optical Flow Dynamics

Finally, optical flow is used as a visual feature as well. The temporal derivative of (2.2) is,

$$\dot{u}_j = -\frac{\dot{\tau}_i}{\tau_i^2}(x_j - f_{xi}) + \frac{1}{\tau_i}(u_j - \dot{f}_{xi}). \tag{2.15}$$

As already said, we need to remove the contributions of rotational flow. In this case, it means reducing the expression of the FOE to $\dot{f}_{xi} = -\frac{v_c}{v_{iz}}\omega_{cy} + \frac{f_{xi}}{v_{iz}} a_c$. Then, we provide the final expression of the dynamics of the optical flow,

$$\dot{u}_j = \frac{2}{\tau_i} u_j + \left(u_j f_{xi} + \frac{v_c}{Z_i} \right) \omega_{cy} + \left(\frac{\tau_i u_j - f_{xi}}{Z_i} \right) a_c. \tag{2.16}$$

2.3 Contribution

This chapter has presented a new virtual human model with a new perception system to reproduce human perception. We provide two contributions here, the perception capabilities of the virtual human and the processing of this information to enable navigation.

We have described an agent model capable of perceiving visual information to emulate human eyes. Our agents are able to perceive color, optical flow, depth and semantic images. These images contain a lot of information about the environment that is processed to detect objects and extract the relevant visual features for navigation. In particular, we introduce optical flow for virtual agents which is known to play an important role in human navigation. This places our agents a step closer to real humans which use this information to move in the world.

Finally, we have described the relevant visual features encoded in optical flow and shown how they are related to the agent's motion. We showed their dynamic equations with respect to the agent actions and why they are relevant for navigation.

CHAPTER 3

AGENT CONTROL USING VISUAL FEATURES

The previous chapter exposed a perception pipeline capable of obtaining and processing visual information as images. Objects in the field of view are segmented and a set of visual features are extracted for each one of them: position in the field of view, optical flow pixels, focus of expansion and time-to-collision. In this chapter, this information is used to design a control scheme that allows agents to navigate dynamic environments. We propose an optimization-based model to enable agents react to their perception. Figure 3.1 shows the general scheme of how the agent is controlled.

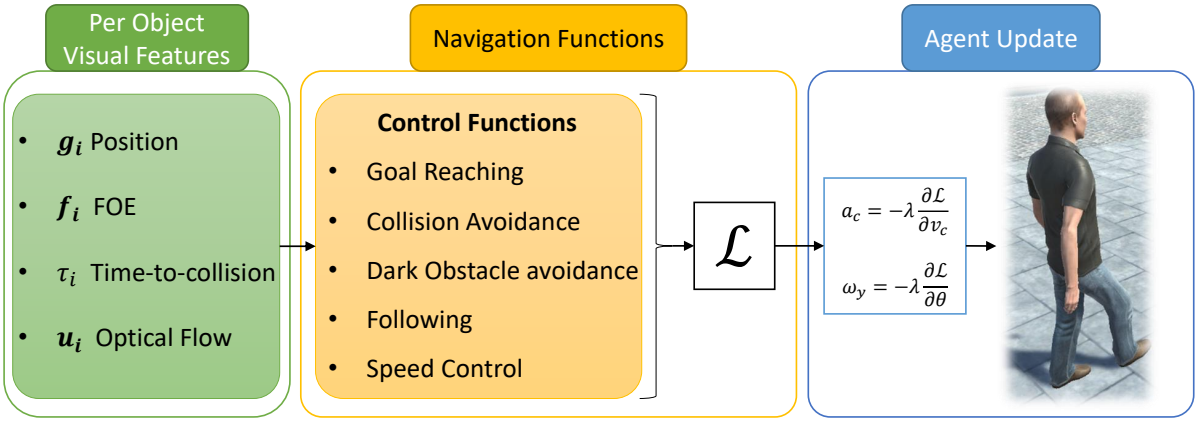


Figure 3.1 – Scheme of the agent's control system. Agents use the visual features to define a navigation function $\mathcal{L}$ that encodes a set of behaviors. Then, the gradient of the function is computed to update the control variables of the agent.

In this thesis we refer to Navigation Function for those functions that need to be minimized in order to update the agent's control variables. These functions use the visual features to combine different behaviors such as goal reaching with collision avoidance or following. These behaviors are usually contradictory as to avoid an obstacle, we

probably need to deviate our path from the fastest route to the goal. These behaviors are weighted in the navigation function with a set of parameters, prioritizing certain tasks such as avoidance. Therefore, the navigation function encodes the behavior of the agent, it defines the priority for every kind of behavior defining a complete navigation task.

The building blocks of the navigation functions are called control functions. These are functions that express a simple task such as define a preferred speed for the agent or align the direction of motion of the agent with a certain object. The control functions are mostly an expression of the visual features captured by the perception pipeline. They define the ideal conditions to complete a certain task, for instance, when the agent needs to reach a goal, the corresponding control function is minimum when the goal is aligned in the center of the field of view as this means the agent moves straight towards it. Control functions are not limited to control visual information but also the agent's own variables such as its speed. Therefore, we define five different control functions which perform the following tasks:

- **Goal reaching**. Aligns the direction of motion of the agent to move towards the goal.
- **Collision Avoidance**. Avoids obstacles in the field of view.
- **Dark Obstacle Avoidance**. Avoids dark regions in the field of view.
- **Velocity Alignment**. Aligns the velocity of the agent with the velocity of another object.
- **Speed Control**. Directly controls the speed of the agent by defining a preferred default speed or matching the one of a specific object.

This set of control functions is sufficient to build a variety of navigation functions with different behaviors.

The main visual information map that our agents need for navigation is optical flow as it encodes the relative motion of objects in the scene. As we explained in the previous chapter, our perception model allows two ways of obtaining optical flow, synthetic and numeric. Due to the differences of the two maps, different pipelines and navigation functions can be used for different visual inputs. Therefore, we define two different control loops. The first one uses synthetic optical flow which allows to obtain more accurate information about the objects and obtain smother trajectories. The second control loop, uses numerical flow and introduces a novel method to navigate an environment taking into account lighting conditions.

This chapter is structured as follows. The control functions are defined using the visual features which are described in Section 3.1. The control functions are then combined into the navigation functions, further described in Section 3.2. Finally, the control variables of the agent are updated using a gradient descent minimization on the navigation function. In Section 3.3 we describe the two control loops proposed for virtual character navigation.

3.1 Control Functions

Control functions $\mathcal{L}_k$ are the building blocks of the navigation scheme presented in this thesis. They define a small specific task and are expressed using the visual features from the perception pipeline. Control functions use a small subset of the visual features, relevant to the particular task. Their value becomes minimum when the task is achieved therefore, they express the ideal configuration of the visual features present in the field of view. Control function do not only rely on the visual features but also on the agent's own parameters such as its speed.

In this section we present multiple control functions for various tasks that are then used to construct the navigation functions to enable agents to traverse virtual environments and complete different navigation tasks.

Our control variables are the agent's acceleration a_c and angular velocity ω_{cy} around the vertical axis. At every update step, the gradient of the navigation function is evaluated and the control variables are updated by,

$$a_c = -\lambda \frac{\partial \mathcal{L}_k}{\partial v_c},\ \omega_{cy} = -\lambda \frac{\partial \mathcal{L}_k}{\partial \theta}, \tag{3.1}$$

with λ being the gradient descent step size parameter. We have empirically found that a value $\lambda = \frac{1}{dt}$, with dt being the time step in the simulation, performs well.

3.1.1 Target Reaching

Target reaching is the basic task for any navigation algorithm. In our agent model, the optic axis, the direction that the agent's sensor face is coupled with its direction of motion. Therefore, target reaching is expressed as the deviation of the target t_x from the center of the field of view. In other words, we want the goal t_x to be placed at the

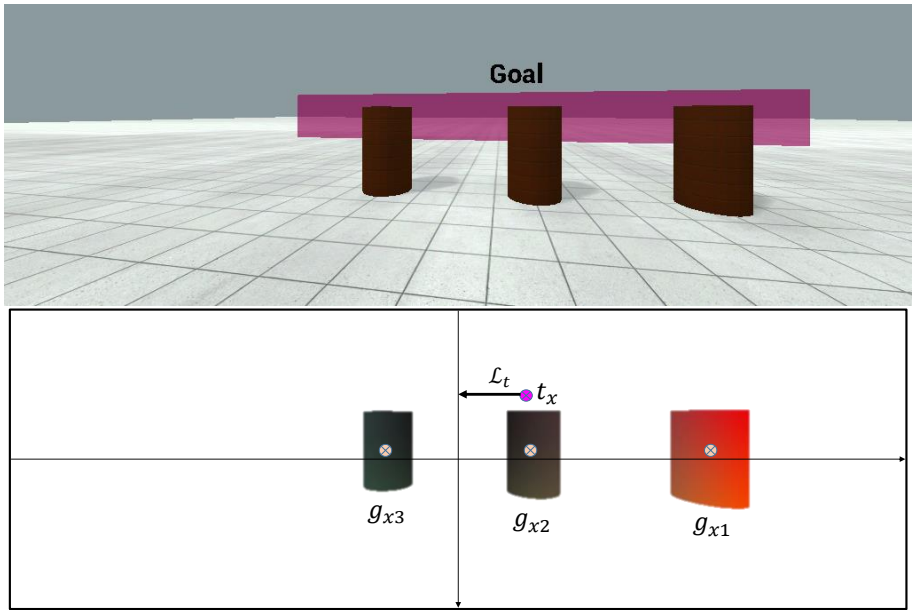


Figure 3.2 – Effect of target reaching control function. (up) Color image from the agent's point of view. (down) Visual features perceived: position of objects g_{xi} and goal closest point t_x. $\mathcal{L}_t$ shows the effect of the target reaching control function.

center of the image. This is a control of geometrical points problem which is well-known in robotics [CH06]. We use the following control function,

$$\mathcal{L}_t = \frac{1}{2} t_x^2, \tag{3.2}$$

The minimum value of this function is for $t_x = 0$ and prevents the agent from deviating too much from the goal. The goal is a known point in space and in the image. The goal corresponds to a fixed 3D point in space therefore, it follows (2.4) which can be rewritten taking into account $v_{ix} = 0$ and Equation (2.10),

$$\dot{t}_x = t_x/\tau_t - (t_x^2 + 1)\omega_{cy}. \tag{3.3}$$

In the cases where the goal is a region and not a point, t_x is considered as the closest point of the region to the agent. Figure 3.2 illustrates the effect of $\mathcal{L}_t$, it shows an agent moving in an environment with a certain goal t_x which is not completely aligned to the center of the image. The control function updates the agent's direction of motion

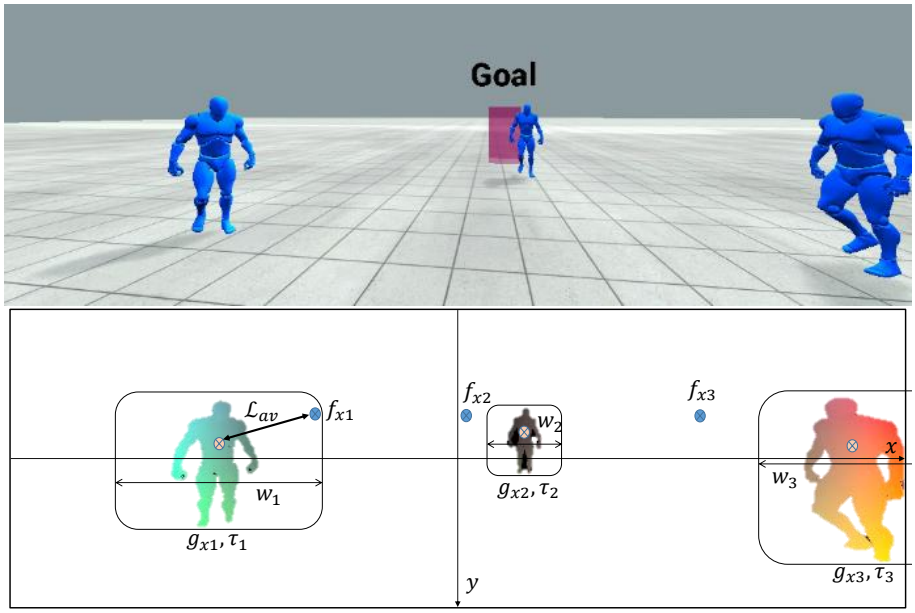


Figure 3.3 – Collision avoidance example. (up) Color image from the agent's point of view. (down) Visual features perceived, obstacle position g_{xi}, obstacle width w_i, obstacle FOE f_{xi} and obstacle time-to-collision τ_i. $\mathcal{L}_{av}$ shows the effect of the collision avoidance control function.

to move t_x towards the center. The control variables have the following expression,

$$\begin{aligned} a_c &= 0, \\ \omega_{cy} &= \lambda t_x (t_x^2 + 1). \end{aligned} \tag{3.4}$$

3.1.2 Collision Avoidance

In order to allow an agent to safely move in any environment it requires a means to avoid obstacles in the field of view. As explained in Section 2.2.2 the FOE f_{xi} is the key element in preventing collisions. It is the expansion center of the object and when it overlaps with the obstacles it means a future collision will occur. Therefore, in order to avoid obstacles, an agent needs to adapt its trajectory to separate the FOE from the corresponding object. In addition, the own agent's size needs to be taken into account to separate these two points. Figure 3.3 shows a bounding box around every obstacle that includes the size of the observer agent with width w_i. To this end, we propose a control function that penalizes the FOE f_{xi} and the object's center g_{xi} being closer than

$w_i/2$,

$$\mathcal{L}_{av} = \sum_{i}^{n} I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i}\right). \tag{3.5}$$

where $\Delta x_i = g_{xi} - f_{xi}$, $\sigma_i = \frac{w_i/2}{ln(10)}$. The function $I(\tau_i) = a\tau_i + b$ is a linear function introduced to give a higher priority to objects with lower time-to-collision τ_i. In our implementation we set $a = -2$ and $b = 10$ and the contribution of objects with $\tau_i > 5s$ is clamped to 0. Figure 3.3 illustrates the effect of $\mathcal{L}_{av}$ separating all objects labeled as visible that need to be avoided.

From Equations (2.1) and (2.9), Δx_i has the following dependency with the control variables,

$$\dot{\Delta x_i} = (g_{xi} - f_{xi})/\tau_i + \left(f_{xi}^2 - g_{xi}^2 - \frac{\tau_i}{Z_i} v_c\right) \omega_{cy} + \frac{\tau_i}{Z_i} f_{xi} a_c, \tag{3.6}$$

and the control variables have the following expression,

$$\begin{aligned} a_c &= \lambda \sum_i I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i}\right) \frac{sign(\Delta x_i)}{\sigma_i} \frac{f_{xi}\tau_i}{Z_i}, \\ \omega_{cy} &= -\lambda \sum_i I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i}\right) \frac{sign(\Delta x_i)}{\sigma_i} \left(\frac{\tau_i}{Z_i} v_c + g_{xi}^2 - f_{xi}^2\right). \end{aligned} \tag{3.7}$$

3.1.3 Dark Object Avoidance

As mentioned in Section 2.1.3, optical flow solver produces inaccurate results in dark regions due to the lack of color and contrast. This renders optical flow computed for dark regions unreliable and need to be differentiated from the rest of visible obstacles considered in normal collision avoidance. We consider that dark regions may contain obstacles and should be avoided. Figure 3.4 shows an example of a scene with dark regions. In the lower image objects are segmented and classified as visible or dark. Visible objects in green are considered in normal collision avoidance control function. Dark regions in red need to be taken into account in a different manner as we only know the position and size of these objects. We follow a similar approach to normal collision avoidance however, in this case we only separate these objects from the center of the field of view as depicted in Figure 3.4. We propose the following control function to avoid these obstacles,

$$\mathcal{L}_{dark} = \sum_{i}^{dark} \exp\left(-\frac{|g_{xi}|}{\sigma_i}\right), \tag{3.8}$$

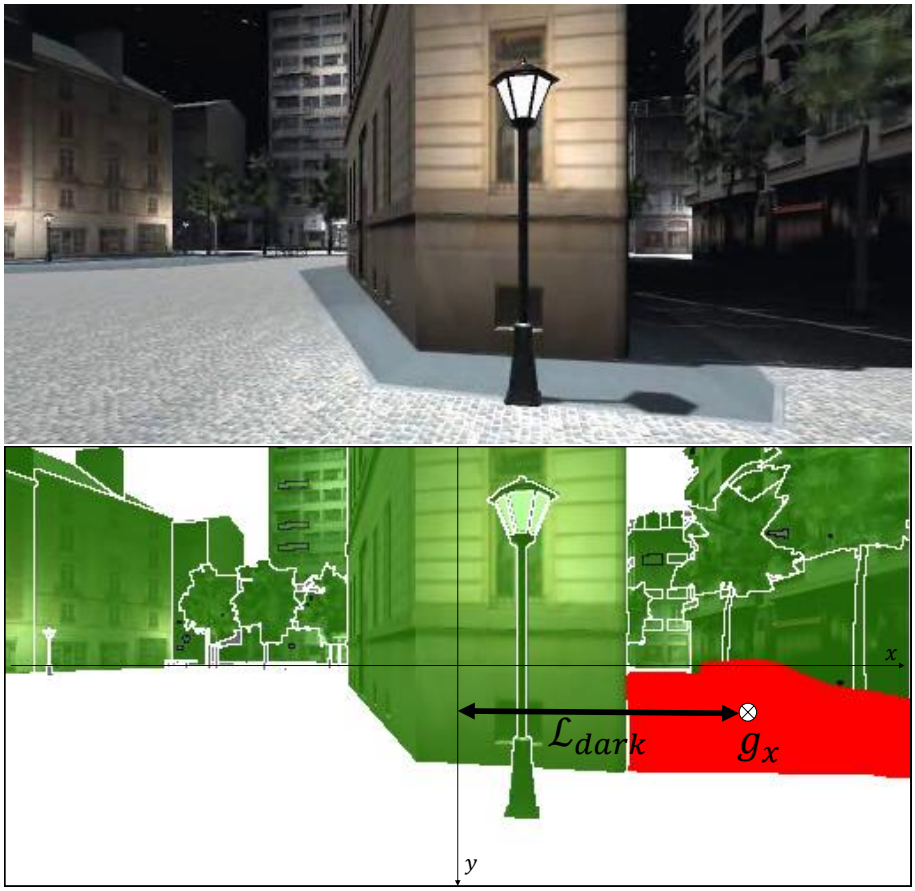


Figure 3.4 – Dark Obstacle Avoidance example. (up) Color image from the agent's point of view. (down) Segmentation of visible objects (green) and dark objects (red) with dark obstacle position g_{xi}. $\mathcal{L}_{dark}$ shows the effect of the dark collision avoidance control function.

with σ_i being computed in the same way as in Equation (3.5). The control variables have the following expression,

$$\begin{aligned} a_c &= 0, \\ \omega_{cy} &= -\lambda\delta \sum_{i}^{dark} \exp\left(-\frac{|g_{xi}|}{\sigma_i}\right) \frac{sign(g_{xi})}{\sigma_i} \left(g_{xi}^2 + 1\right). \end{aligned} \tag{3.9}$$

3.1.4 Velocity Alignment

Velocity alignment is essential in following scenarios. An agent following a leader needs to adjust its velocity to match the one of the leader. This is the situation illustrated in Figure 3.5. When the agent matches its velocity with the leader, the relative

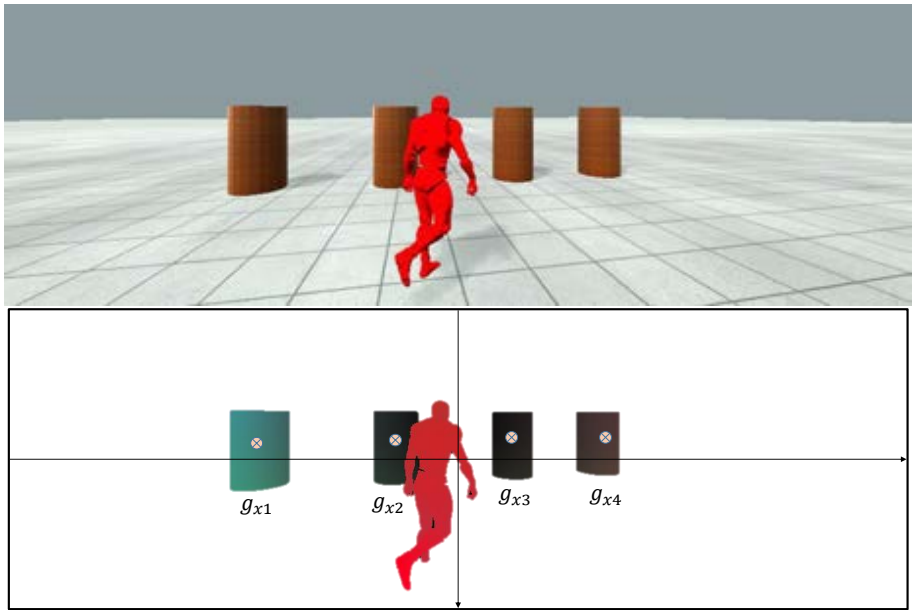


Figure 3.5 – Velocity alignment example. (up) Color image from the agent's point of view. (down) Visual features perceived, the agent tries to align its velocity with the velocity of the red object.

velocity between them is zero as they move in the same direction at the same speed. This results in no optical flow generated by the leader as it becomes a static object in the field of view. For this reason we model the velocity alignment problem as the minimization of perceived optical flow. The corresponding control function has the following expression,

$$\mathcal{L}_{min} = \frac{1}{N^2}\left(\sum_{j\in i} u_j\right)^2, \tag{3.10}$$

being N the number of pixels of object i. As the goal of velocity alignment and following is to make the relative velocity v_{iz} between the object and the agent tend to zero we cannot rely on the FOE $f_{xi} = v_{ix}/v_{iz}$. Therefore, set $f_{xi} = 0$ to remove its contributions and Equation 2.16 is rewritten to,

$$\dot{u}_j = \frac{2}{\tau_i}u_j + \frac{v_c}{Z_i}\omega_{cy} + \frac{\tau_i u_j}{Z_i}a_c, \tag{3.11}$$

resulting in the following expression for the control variables,

$$\begin{aligned} a_c &= \lambda \frac{2\tau_{red}}{N^2 Z_{red}} \left(\sum_{j \in red} u_j \right)^2, \\ \omega_{cy} &= \lambda \frac{2v_c}{N Z_{red}} \sum_{j \in red} u_j. \end{aligned} \tag{3.12}$$

3.1.5 Speed Control

Some control functions may change the speed of the agent without any limit on their own. To control this behavior a preferred speed v_c^* is defined, that plays the role of the default speed when no other behavior is active and limits the range of possible values it may take. This is modeled by,

$$\mathcal{L}_v = \frac{1}{2}(v_c - v_c^*)^2, \tag{3.13}$$

and from the definition of v_c, $\dot{v_c} = a_c$. The control variables have the following expression,

$$\begin{aligned} a_c &= -\lambda(v_c - v_c^*), \\ \omega_{cy} &= 0, \end{aligned} \tag{3.14}$$

Alternatively, we may want to match the relative speed $v_{iz} = \frac{Z_i}{\tau_i}$ between the camera and an object. This is for example useful to follow an object. To this end, we use,

$$\mathcal{L}_\tau = \frac{1}{2} v_{iz}^2. \tag{3.15}$$

As this equation makes v_{iz} tend to zero, the FOE becomes unreliable, therefore, $\dot{v}_{iz} = -a_c$. And the control variables have the following explression,

$$\begin{aligned} a_c &= \frac{Z_{red}}{\tau_{red}}, \\ \omega_{cy} &= 0. \end{aligned} \tag{3.16}$$

3.2 Navigation Functions

Navigation functions express a higher-level navigation task which allows the agent to satisfy multiple control functions at the same time. The key idea is to blend these functions to generate a navigation function $\mathcal{L}$, which captures a complex behaviour, such as reaching a target while avoiding collisions with multiple objects, or following a mobile target while avoiding collision with obstacles. The main objective of the navigation function is to assign priorities to the control functions. It assigns a weight to each task that the agent needs to complete which defines how the agent behaves. The general expression of a navigation function is,

$$\mathcal{L} = \sum_k \alpha_k \mathcal{L}_k, \tag{3.17}$$

where $\mathcal{L}_k$ is a control function presented in Section 3.1 and α_k is a parameter that controls the weight of each control function.

3.3 Control Loops

In this section we propose two strategies using the framework presented in this chapter and the perception scheme presented in the previous chapter. The general idea is the following, agents acquire visual information (images) from their sensors, then these data are processed to perform segmentation, extract the visual features and build the control functions. Then, the navigation function combines the control functions and the agent's motion is updated by minimizing the navigation function as in Equation (3.1). The main difference between the two control loops is the usage of either synthetic or numeric optical flow. Due to the nature of each flow computation method our agent exhibits different behaviors and therefore use different navigation functions.

3.3.1 Synthetic Optical Flow

In this section we describe the control loop associated to using synthetic optical flow. Figure 3.6 shows a scheme of this control loop. We detail the perception and navigation functions, describing what are the input information of these agents and their behaviors.

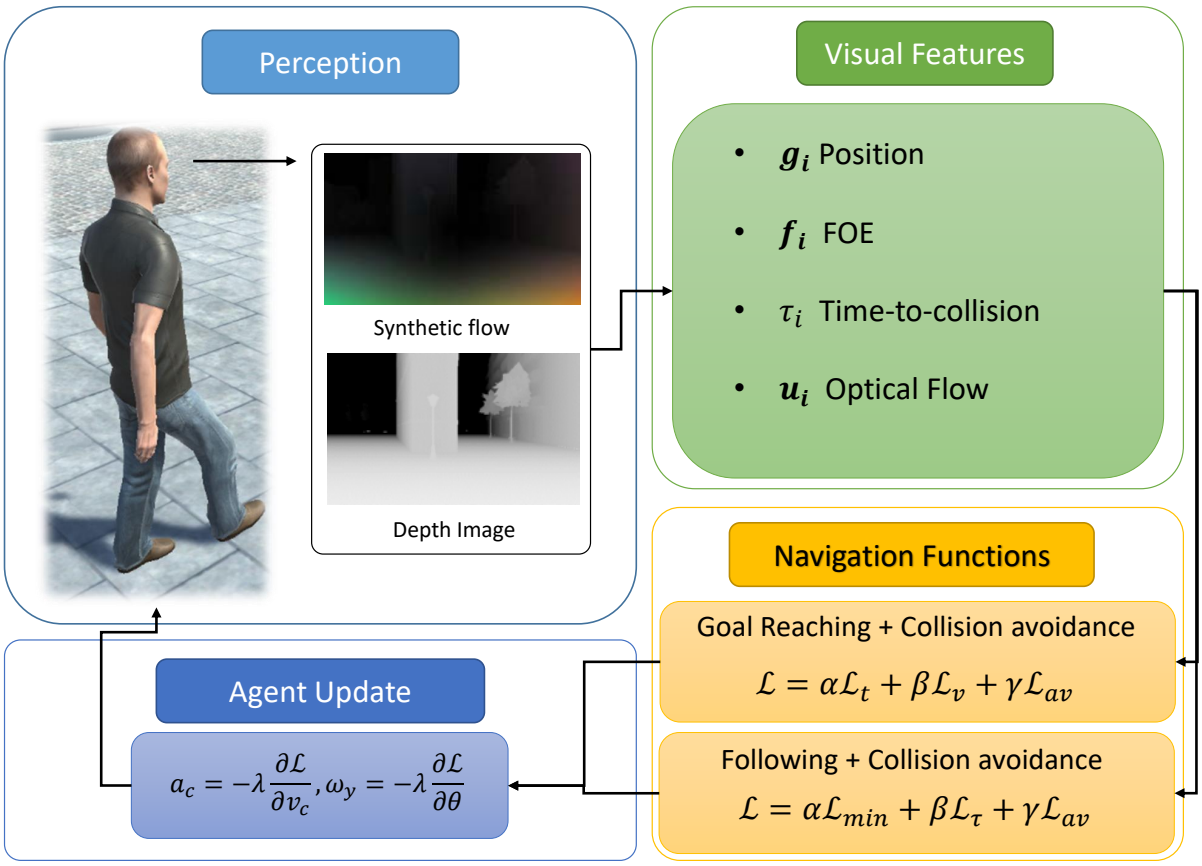


Figure 3.6 – Agent control loop using synthetic optical flow.

Perception

In this situation, the agent is equipped with sensors to record synthetic optical flow and the depth map. The depth map is retrieved from the render engine and optical flow encodes Equation (2.2), typically generated using the render engine. This optical flow is of very high quality, which means that we can use simple segmentation algorithms to segment objects from the optical flow map. In particular we use the OpenCV library to detect discontinuities in the flow map through the Graph Segmentation algorithm [FH04]. Additionally, we can remove non-obstacle objects in the scene such as the ground or sky as the flow generated by these objects is known. Then, visual features are computed for every object.

Agent Behavior

We present two similar navigation functions that function well from the perception scheme presented in the previous section.

The first navigation function performs goal reaching and collision avoidance and is expressed in the next equation,

$$\mathcal{L} = \alpha\mathcal{L}_t + \beta\mathcal{L}_v + \gamma\mathcal{L}_{av}, \tag{3.18}$$

with α, β, γ being parameters. $\mathcal{L}_t$ brings the goal to the center of the field of view, $\mathcal{L}_v$ defines an ideal velocity for the agent to maintain and $\mathcal{L}_{av}$ processes every visible obstacle to separate its FOE from its center to avoid collisions.

The second navigation function involves following an object or an agent and uses the next equation,

$$\mathcal{L} = \alpha\mathcal{L}_{min} + \beta\mathcal{L}_\tau + \gamma\mathcal{L}_{av}, \tag{3.19}$$

again, with α, β, γ being parameters. $\mathcal{L}_{min}$ and $\mathcal{L}_\tau$ process a specific object in the field of view, which is the object the agent needs to follow. $\mathcal{L}_{min}$ has a big effect on the direction of motion of the agent and $\mathcal{L}_\tau$ focuses on its speed. Finally, $\mathcal{L}_{av}$ processes every other obstacle to perform collision avoidance.

3.3.2 Numeric Optical Flow

The next control loop involves numerical optical flow. Figure 3.7 illustrates the control loop associated to using numerical flow. Again, we detail the perception of agents using this control loop and how they behave. The goal here is to move a step forward to human perception and to consider lighting conditions.

Perception

In this control loop, our agents do not have access to synthetic optical flow. In addition, this control loop does not use depth images to reduce the amount of data processed and to enable future monocular robotic applications. Instead, they are equipped with sensors capable of recording color images and the semantics of every pixel. The color images are then used to compute optical flow. From two consecutive frames a flow image is computed using the FlowNet 2.0 solver [IMS+17]. The solution of numer-

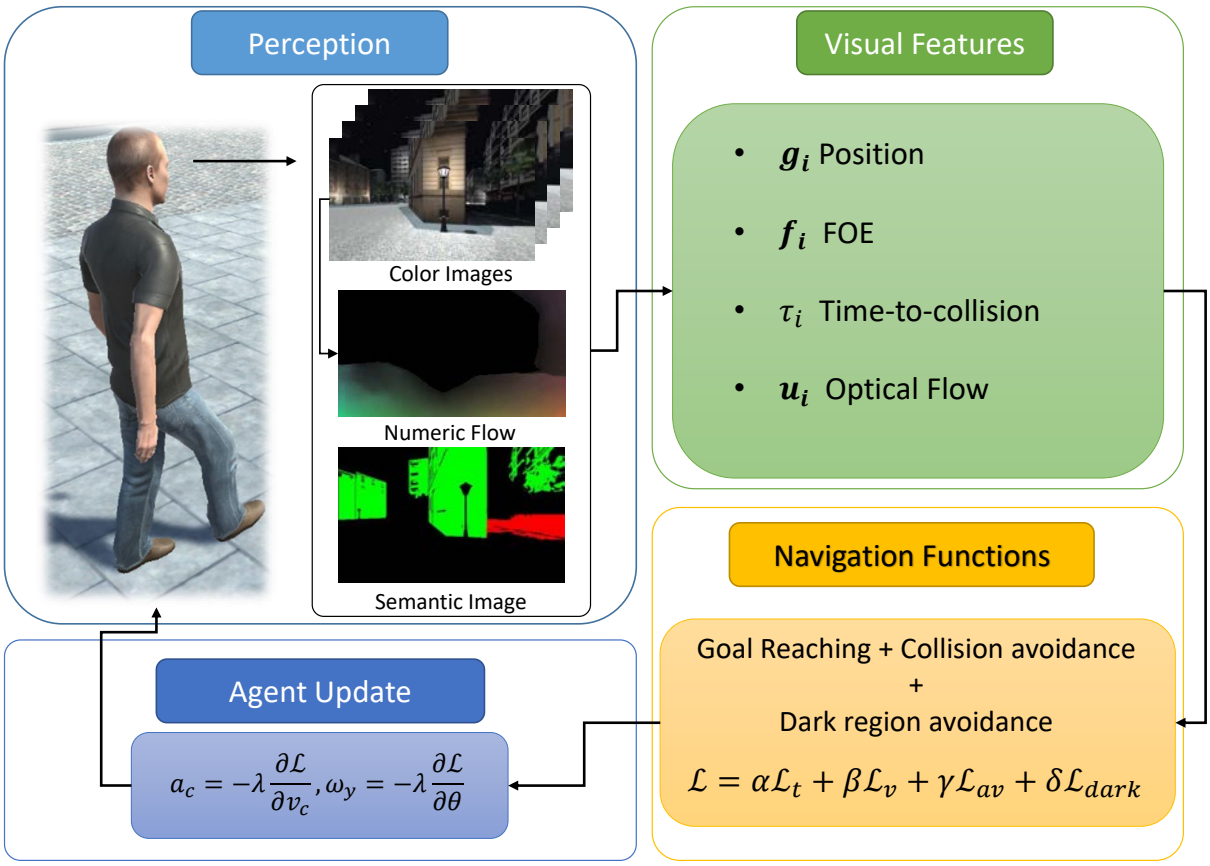


Figure 3.7 – Agent control loop using numeric optical flow.

ical solvers are usually noisy and may contain errors, therefore, the semantic image is used to perform object segmentation. In addition, the color images are used to detect dark regions or objects.

Agent Behavior

We present a navigation function capable of navigating an environment while avoiding obstacles and dark regions on its way,

$$\mathcal{L} = \alpha\mathcal{L}_t + \beta\mathcal{L}_v + \gamma\mathcal{L}_{av} + \delta\mathcal{L}_{dark}, \tag{3.20}$$

with $\alpha, \beta, \gamma, \delta$ being parameters of the model. $\mathcal{L}_t$ and $\mathcal{L}_v$ play the same role as in Equation (3.18), $\mathcal{L}_{av}$ processes those objects labelled as visible to perform avoidance and

$\mathcal{L}_{dark}$ performs avoidance with the dark objects. We ensure $\gamma \geq \delta$ so that visible objects have higher priority over dark objects.

3.4 Contribution

This chapter closes the loop to enable character navigation in virtual environments using visual information. The contribution of this chapter can be split in two parts, definition of navigation functions and definition of control loops.

In the previous chapter we described the visual features that can be extracted from optical flow and how they relate to the agent's own motion. In this chapter, these relations have been exploited to design control functions. These functions encode the ideal conditions to satisfy a certain task and then they are combined into navigation functions. Optimizing these functions allows the agent to traverse the environment avoiding collisions, following an object, etc. The novelty lies in the new control framework that allows navigation using visual features in virtual environments. We used this framework to propose algorithms for goal reaching, following and collision avoidance.

The second part of the contribution is the definition of two strategies using different navigation functions. The first control loop focuses on the use of synthetic optical flow. Agents using this model are capable of navigating an environment reaching a goal or following another obstacle. The second control loop, uses numerical optical flow computed from a sequence of images. In this second model, our agents have a perception system closer to that of humans which depends on the light of the scene. This allows for a novel agent model capable of reacting to lighting conditions without any prior knowledge of the luminosity of the scene, relying only in its visual sensors.

CHAPTER 4

RESULTS AND ANALYSIS

Chapter 2 exposed a new visual perception model for virtual humans and how to process this information to obtain the relevant features for navigation. Chapter 3 closed the loop to use the visual features to define navigation functions that encode the behavior of the virtual human. In this chapter we demonstrate our approach on various scenarios, from the simplest to illustrate our technique to more complex ones to show their efficiency.

In the previous chapter we exposed two different control loops using different visual inputs, respectively based on synthetic and numeric optical flow. First, in Section 4.1, we demonstrate the use of synthetic optical flow corresponding to the control loop presented in Section 3.3 in static and dynamic environments. We show how agents can navigate various environments using only the visual information provided by the simulation (synthetic optical flow and depth maps). First, agents use the navigation function in Equation (3.18) in this control loop, reaching a goal while avoiding obstacles in their way. Then, the second navigation function is used to demonstrate how agents can follow a leader while avoiding obstacles.

The second control loop presented in Section 3.3.2 uses numeric optical flow and is demonstrated in Section 4.2. In this situation agents are capable of perceiving color images that are then processed to obtain optical flow and semantic images. The use of color information allows to take into account the lighting conditions of the scene. We demonstrate how agents react to the different lighting conditions while reaching a goal.

Then, we compare our approach with other approaches for character steering in Section 4.3. We show our model versus RVO [vdBGLM11] and Dutra's model [DMCN+17]. We chose RVO as it is one of the most known algorithms for crowd simulation and Dutra as it is also a synthetic vision approach to character steering. These models are built for crowd simulation and even though our model is not aimed to simulate crowds we can still compare the results of each model. We also compare the two control loops proposed in this thesis to show the effects of using synthetic or numeric optical flow.

We study the contribution of each control function in the agent behavior in Section 4.4. Navigation functions weight every control function with a parameter α, β, γ and δ. The value of these parameters affect how the different control function interact and therefore how the agent behaves.

Finally, we analyze the performance of the control loops used in this chapter. Our model is computationally expensive as it needs to perform intense image processing operations.

4.1 Synthetic Optical Flow

This section shows the results of using the control loop described in Section 3.3 that uses synthetic optical flow. We demonstrate this navigation technique by testing our agents in various scene with static, dynamic objects and other agents moved with the same algorithm. This control loop proposed two navigation functions, target reaching with collision avoidance and following with collision avoidance. We demonstrate the results of each function in the following sections.

4.1.1 Collision-free target reaching navigation

We first demonstrate our technique in the classic situation of a character reaching a goal while avoiding the obstacles as depicted in Figure 4.1. To model this behavior we use the control loop defined in Section 3.3.1 using the navigation function shown in Equation (3.18). There are various parameters that need to be defined for the following example, first the comfort velocity defined by the speed control function is set to $v_c^* = 1.5m.s^{-1}$. We have empirically found that the values $\alpha = 1, \beta = 0.1, \gamma = 1$ for the parameters in the navigation function perform well in most situations, the examples in this section use these values. We analyze the particular effect of these parameters in Section 4.4.

The control variables of the agent are updated every frame applying a gradient descent optimization method to the navigation function in Equation (3.18), this results

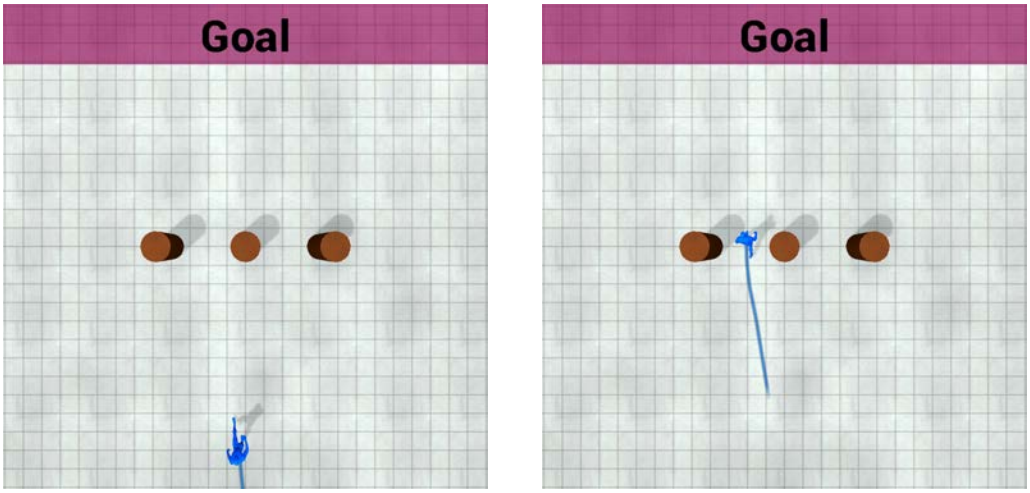


Figure 4.1 – Target reaching example. Agent traversing an scene with cylindrical obstacles. The goal is any location in the red region displayed on top of the figure.

in the following expression,

$$
\begin{aligned}
a_c = \lambda \Bigg[& -\beta(v_c - v_c^*) + \\
& \gamma \sum_i I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i}\right) \frac{sign(\Delta x_i)}{\sigma_i} \frac{f_{xi}\tau_i}{Z_i} \Bigg], \\
\omega_{cy} = \lambda \Bigg[& \alpha t_x (t_x^2 + 1) - \\
& \gamma \sum_i I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i}\right) \frac{sign(\Delta x_i)}{\sigma_i} \left(\frac{\tau_i}{Z_i} v_c + g_{xi}^2 - f_{xi}^2\right) \Bigg],
\end{aligned}
\tag{4.1}
$$

Static Environments

In a static environment, the apparent motion of objects is only due to the agent's own motion. As the visual sensor of the agent (virtual cameras) point towards the direction of motion of the agent at all times, the FOE remains at the center of the image regardless of the direction of motion of the agent. To avoid collisions, agents need to change their direction of motion to separate the shape of the objects from the FOE.

Figure 4.2 shows agents moving in a room with objects of various sizes and shapes. We show the visual features perceived by one of the agents, the closer objects have a larger contribution in the collision avoidance control function. Therefore, our agents are capable of finding their way through this scene without any planning. Each agent takes

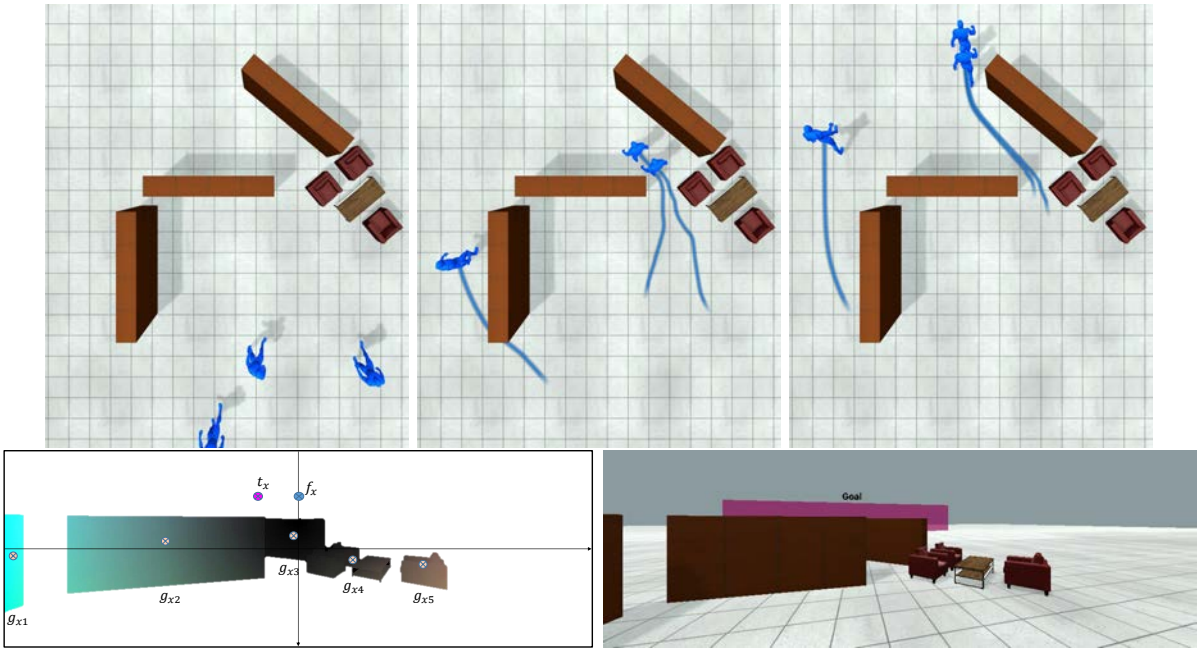


Figure 4.2 – Agent trajectory with static obstacles. (up) Top view. Three agents need to cross the room avoiding obstacles. Various objects of different shapes and sizes are present in this scene. Agents are placed in different starting position and orientation and they choose different paths to walk out the room. (bottom left) Visual features perceived by one of the agents. (bottom right) Color image perceived corresponding to the visual features.

a different route depending on its initial position, one agent finds an easier path around the walls and others manage to move close to the other objects through the scene. Figure 4.3 shows how the control functions for target reaching and collision avoidance and the position of the goal in the field of view of the same agent varies over time. The cost of avoidance increases and decreases very fast as objects enter and leave the field of view. Once all obstacles are cleared, goal reaching becomes dominant and the goal position converges quickly to the center ($t_x = 0$).

Figure 4.4 shows an example of an agent in a city-like environment. The agent is capable of detecting the thin fence as an obstacle allowing him to avoid it. This proves that any geometrical shape can be processed and considered as an obstacle and perform collision avoidance.

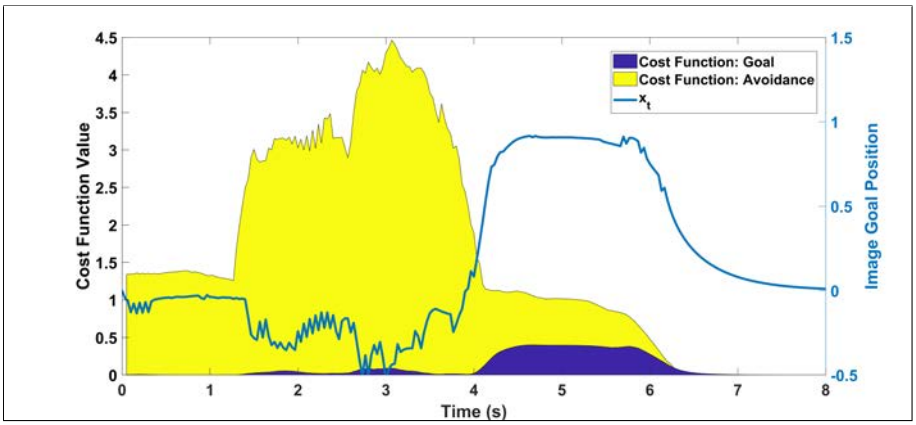


Figure 4.3 – Variation of the cost function (left axis) and goal t_x (right axis) over time. It corresponds to the front agent of the scenario shown in Figure 4.2.

Figure 4.4 – Agent trajectory in a city-like environment with urban obstacles.

Dynamic Environments

We now test our algorithm in dynamic environments which includes scenes with multiple agents blocking each other's path such as in Figure 4.5 and/or other objects following a predefined path that need to be avoided. This means that the FOE is no longer limited to the center of the image but can appear in any place of the screen. Regardless, the basic principle of collision avoidance remains the same as in static obstacles, to separate the FOE from the object's shape.

Figure 4.6 shows a circle scenario. Six agents are arranged in a circular formation and are tasked to reach the opposite side of the circle. This is a very symmetrical situation and because of that a particular pattern appears in the trajectory of the agents as they choose to avoid the other agents in the same manner. This effect is due to

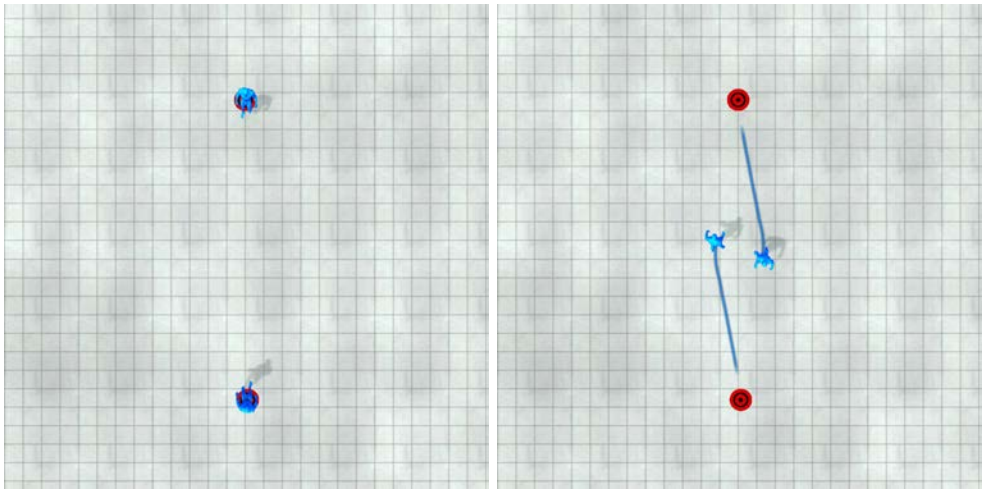


Figure 4.5 – Target reaching example with two agents moving in opposite directions.

the high accuracy of the synthetic optical flow where a perfectly symmetrical situation causes all agent to react in the same way. Figure 4.7 shows a similar situation with 4 agents. In this situation we added a small randomized displacement to the initial position of the agents. This breaks the symmetry and the pattern disappears.

Even though the computation time of our algorithm is too high to be usable for crowds we tested our approach in high density situations such as in the example given in Figure 4.8. Despite the high density of the situation agents are able to find a way through the scene.

Our method is not only able to coordinate its motion with other agents to perform avoidance but can also avoid objects moving arbitrarily through the scene. Figure 4.9 shows an example of an agent moving in a scene with soccer balls bouncing around. The agent does not want to touch any soccer ball and wants to find a way through them. Our control model is able to detect every ball and find a way between them.

Figure 4.10 shows a scene of multiple agents in a street with cars. Cars move at a constant speed and do not react to pedestrians in the road. Some agents are tasked to cross the road and have to find a secure path that prevents an accident. The navigation function is able to slow down our agents until there is a path they can follow safe from the cars. This shows that our approach can handle many types of dynamic obstacles (other agents and constant moving objects) at the same time.

The examples above show agents moving in open spaces or plenty of room to move in the scene. The following two examples test our approach in a more restricted scenario where agents need to also avoid the boundaries of the scene. Figure 4.11 shows

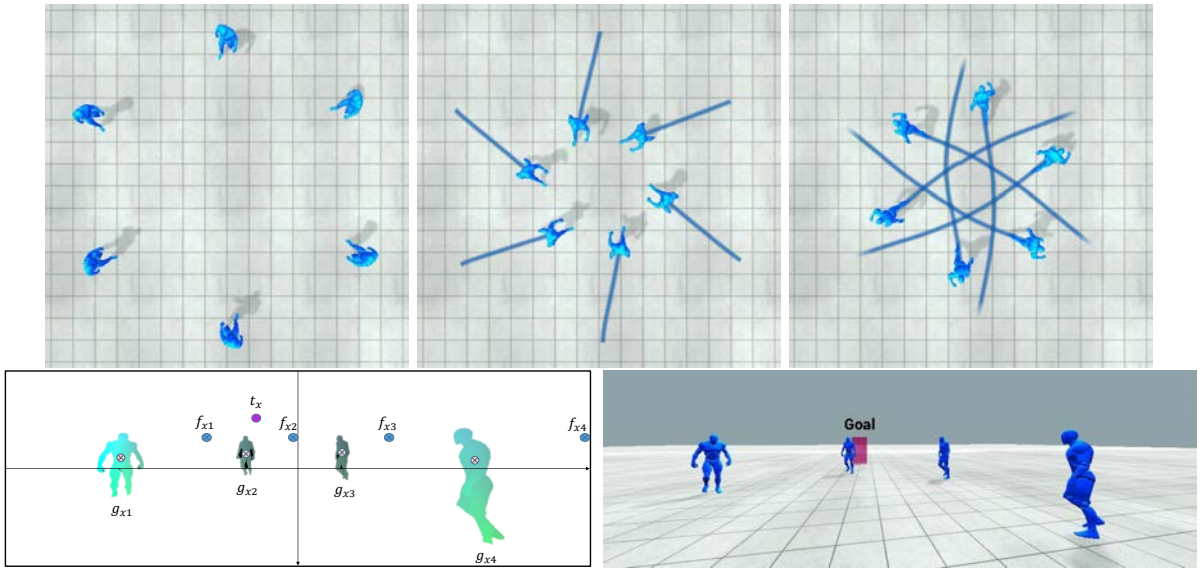


Figure 4.6 – Circle scenario. (up) Six agents are initially arranged in circular formation and need to reach opposite side of the circle. (bottom left) Visual features perceived by one agent. (bottom right) Color image corresponding to the same agent.

a first example with 4 agents that need to go to opposites sides of the parking scene. Agents detect each other and adjust their trajectory to perform mutual avoidance. They also need to take into account the cars and also try to keep a certain distance from them. Figure 4.12 shows a second example of this were two agent can barely pass at the same time. Our control loop is capable of finding a way without collision and maximizing the distance to all the objects at the same time.

4.1.2 Following with Collision Avoidance

In this section we use our synthetic flow-based control loop to produce a following behavior for our agents. The set up in this experiments goes as follow, there is a leader (illustrated in red color) with a predefined path and other agents that uses our control loop (illustrated in blue color) like in the example of Figure 4.13. We understand following as matching the velocity of the leader both in direction and magnitude. The following behavior is the result of applying the navigation function in Equation (3.19). The values of the parameters in the following experiments are $\alpha = 1, \beta = 1, \gamma = 1$. In this situation there is no preferred velocity as the agent matches the velocity of the leader.

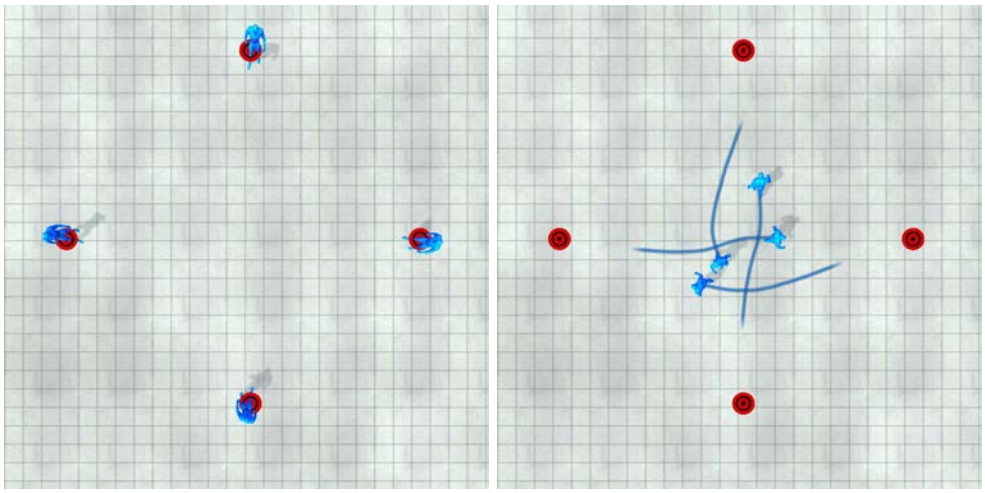


Figure 4.7 – Example of target reaching with 4 crossing agents with added random variation to the initial position.

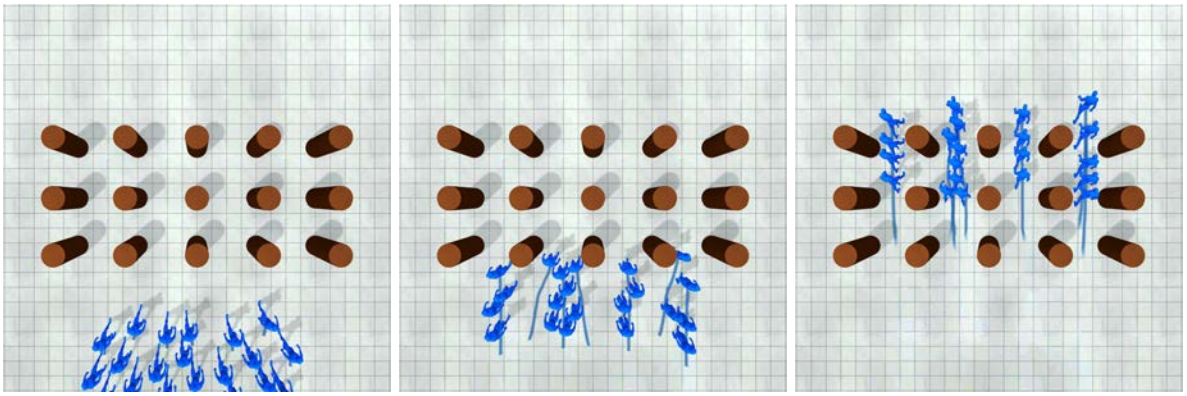


Figure 4.8 – Example with many agents. Agents need to cross the scene in the same direction through the cylindrical obstacles.

Figure 4.9 – Agent avoiding soccer balls which bounce through the scene.

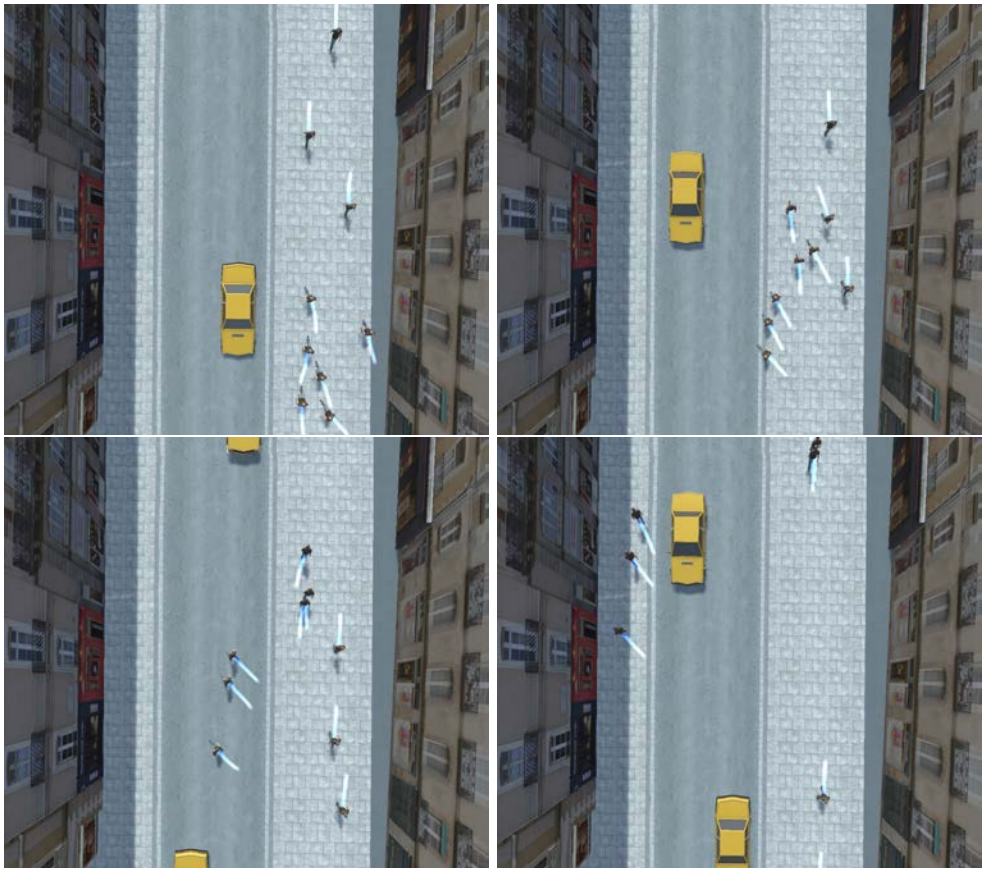


Figure 4.10 – Agent trajectory with complex obstacles in a city-like environment. Some agents are tasked to cross the road while others just walk along the sidewalk.

Similarly as we did for in the previous section, we derive the expression of the control variables from the navigation function in Equation (3.19),

$$
\begin{aligned}
a_c = \lambda \Bigg[& \alpha \frac{2\tau_{red}}{N^2 Z_{red}} \left(\sum_{j \in red} u_j \right)^2 + \beta \frac{Z_{red}}{\tau_{red}} + \\
& \gamma \sum_i I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i} \right) \frac{sign(\Delta x_i)}{\sigma_i} \frac{-f_{xi}\tau_i}{Z_i} \Bigg], \\
\omega_{cy} = \lambda \Bigg[& \alpha \left(\frac{2v_c}{N Z_{red}} \sum_{j \in red} u_j \right) + \\
& \gamma \sum_i I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i} \right) \frac{sign(\Delta x_i)}{\sigma_i} \left(\frac{\tau_i}{Z_i} v_c + g_{xi}^2 - f_{xi}^2 \right) \Bigg],
\end{aligned}
\tag{4.2}
$$

to update the simulation.

Figure 4.11 – Agents in a parking scenario. Agents need to avoid each other while not hitting any car in the way.

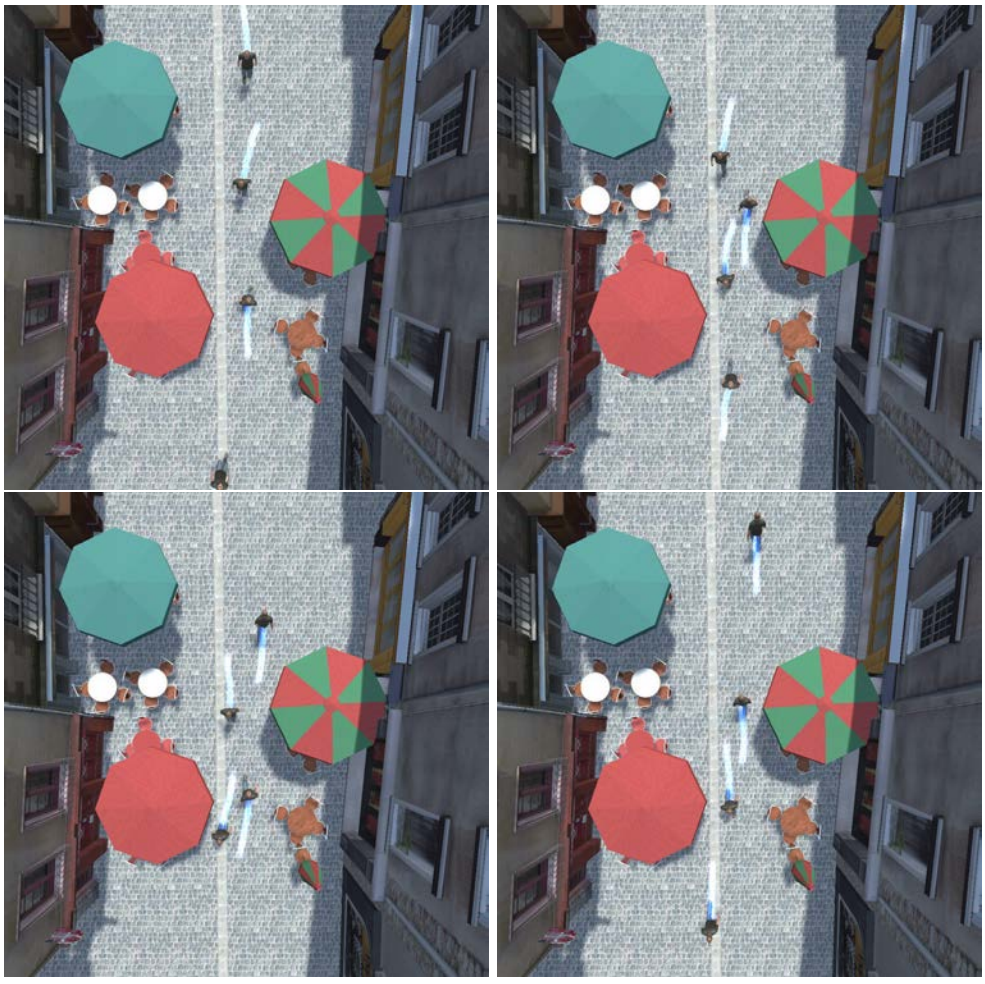


Figure 4.12 – Agents in a street with table and chairs. Agents need to avoid each other while not hitting any obstacle in their way.

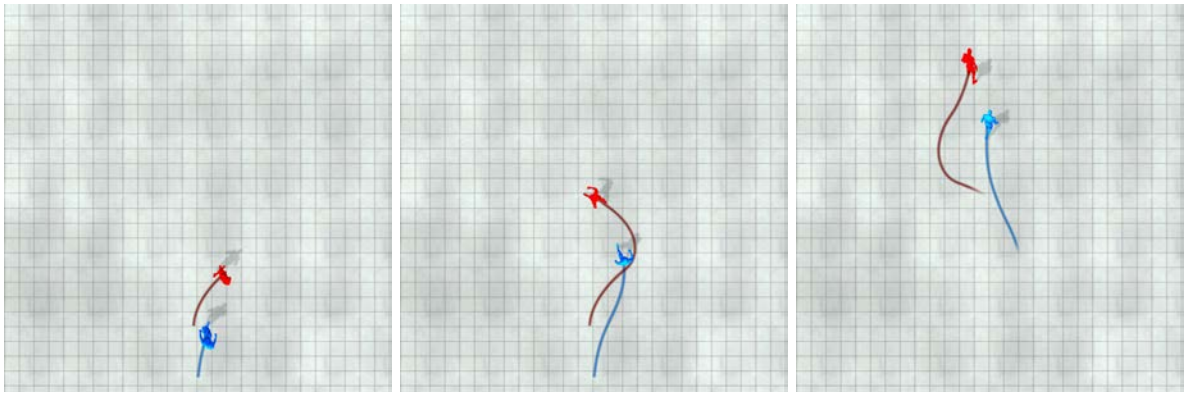


Figure 4.13 – Following example. The red agent is a leader and follows a predefined path. The blue agent needs to adjust its velocity to match the one of the leader.

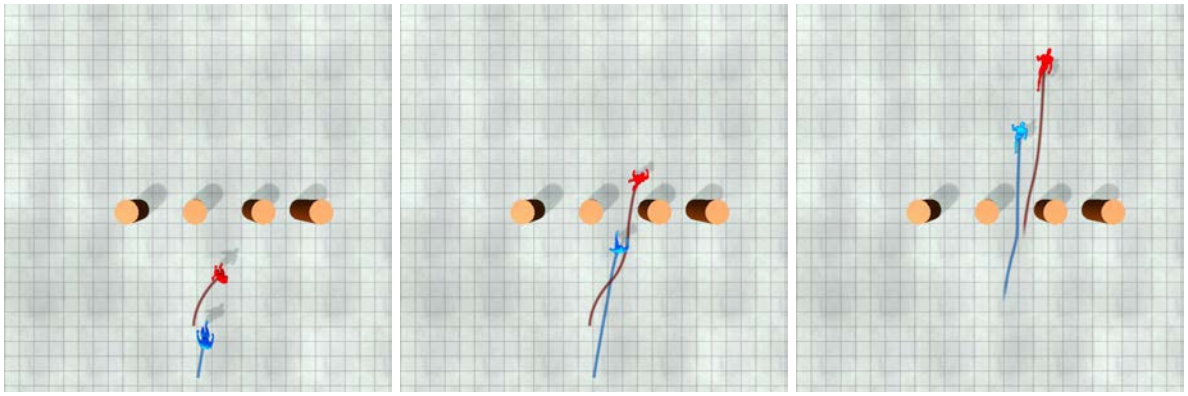


Figure 4.14 – Following a leader with cylindrical obstacles. The red agent follows a predefined path around the obstacles. The blue agent finds a path to avoid the obstacles while trying to match the velocity of the leader.

As seen in Figure 4.13 agents using our model are capable of easily adapting to the motion of other agents. There is a certain delay to the adaptation of the velocity of the agent when the leader changes its direction quickly. This depends on the parameter α and we study its effect in Section 4.4.

Furthermore, our control loop supports obstacle avoidance at the same time as following another agent. Figure 4.14 shows a following example with obstacles in the way. The red agent follows a curved path without collisions. The blue agent has to match the velocity of the red agent, however, it detects that there is a risk of collision with the cylindrical obstacles. Therefore, the agent turns to move between the cylinders to prevent a collision. Once the obstacles are cleared, the agent matches the direction of motion of the leader.

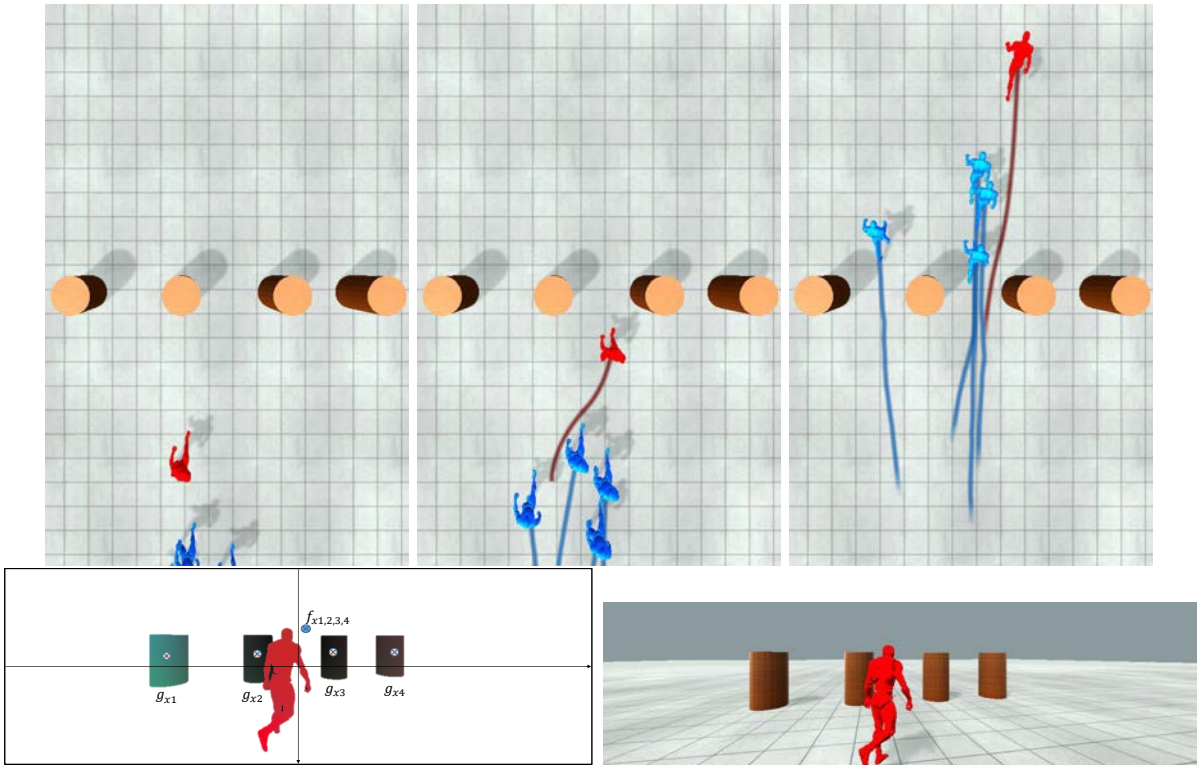


Figure 4.15 – Following scenario. (up) Four agents (blue) follow a leader (red) while avoiding obstacles. (bottom left) Visual features perceived by the agents. (bottom right) Color image corresponding to the middle image.

Figure 4.15 shows another example of following with obstacles in the same scene as in the previous example. This time, there are multiple agents present following the same leader. Some agents perform the same maneuver as in the previous example, one agent takes a different path as its initial position facilitates avoidance in this way. Collision avoidance usually takes precedence to any other behavior as it is one of the critical tasks that any navigation system needs to satisfy.

4.2 Numeric Optical Flow

In this section we show the results of the control loop presented in Section 3.3.2 using numerical optical flow. As opposed to synthetic optical flow, numeric optical flow requires the use of color images to perform the computation. The effect of this is that the optical flow quality depends on the brightness of the scene. This reduces the reliability of numerical optical flow and introduces the necessity of taking into account

light conditions. This control loop uses the navigation function in Equation (3.20) which performs target reaching with collision avoidance similar to the previous control loops. In addition, this model takes into account the light of the scene perceived in the color images to avoid dark regions. We present results of agents in scenes with high contrast scenes where they need to find a way avoiding dark areas. In addition, we prove our method in changing lighting conditions.

In this situation the control variables of the agent take the following expression,

$$\begin{aligned} a_c &= \lambda \Bigg[-\beta(v_c - v_c^*) + \\ &\quad \gamma \sum_i^{visible} I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i}\right) \frac{sign(\Delta x_i)}{\sigma_i} \frac{f_{xi}\tau_i}{Z_i} \Bigg], \\ \omega_{cy} &= \lambda \Bigg[\alpha t_x (t_x^2 + 1) - \\ &\quad \gamma \sum_i^{visible} I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i}\right) \frac{sign(\Delta x_i)}{\sigma_i} \left(\frac{\tau_i}{Z_i} v_c + g_{xi}^2 - f_{xi}^2\right) \\ &\quad -\delta \sum_i^{dark} \exp\left(-\frac{|g_{xi}|}{\sigma_i}\right) \frac{sign(g_{xi})}{\sigma_i} \left(g_{xi}^2 + 1\right) \Bigg], \end{aligned} \tag{4.3}$$

with $\alpha = \beta = \gamma = \delta = 1$ and $v_c^* = 1m/s$ for all experiments in this section.

In this control loop we do not have any knowledge about the depth Z_i of the objects as we lack the depth map. In order to compensate this missing information we make the following approximation $\frac{\tau_i}{Z_i} v_c = \frac{1}{2}$ for every object. This approximation is the result of assuming that the objects are moving in the opposite velocity as the agent and at the same speed. We demonstrate that this assumption allows agents to avoid obstacles with velocities within the same order of magnitude, including static obstacles.

4.2.1 Static Lighting

In this section we demonstrate our algorithm in action with static lighting conditions. We use dark environments with hard shadows and bright moonlight as in Figure 4.16 to show the effect of limited visibility. Then, we compare the result of not taking into account light for navigation using a synthetic optical flow approach to visualize how agents take different routes.

Figure 4.16 shows an agent moving in a street with a very hard shadow at his

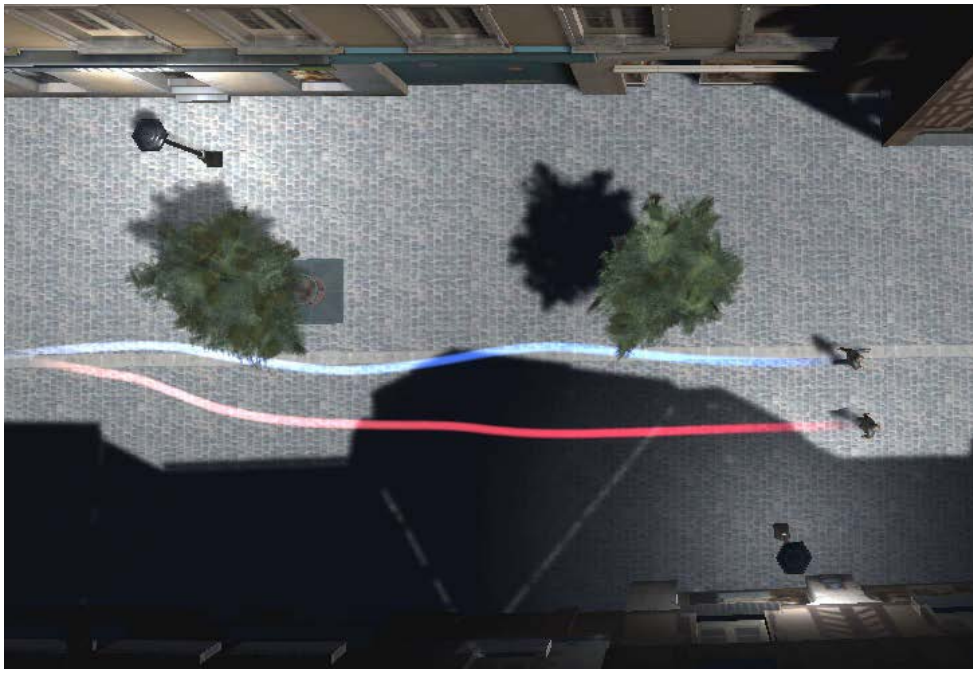


Figure 4.16 – Agent trajectory in dark street. The shadow creates a very dark region in the field of view of the agent which it tries to avoid. The blue trajectory is an agent using the numerical flow navigation model, the red trajectory is an agent using the synthetic optical flow model.

right. These shadows are perceived in the field of view as soft obstacles that need to be avoided whenever possible but visible obstacles always have a higher avoidance priority. The blue trajectory shows an agent using the numerical flow control loop and we can see that it steps a bit into the shadow as the dark region avoidance is balanced with avoidance of the tree. The red trajectory shows an agent using synthetic optical flow unaware of the lighting conditions. It traverses the shadow as he only increases the distance between the trees and itself.

The next example in Figure 4.17 shows an agent in a street with two possible ways, one is lit and the other is dark. The agent using numerical optical flow (blue trajectory) decides to completely avoid the dark region by turning to to the bright street. An agent using synthetic optical flow (red trajectory) decides to move towards the dark street as it is the shortest path. This example illustrates how taking into account lighting can change the trajectory of the agent as the initial position of both agent models is the same yet they make different choices.

Figure 4.18 shows another example of a poorly lit street. In this scene, the only light

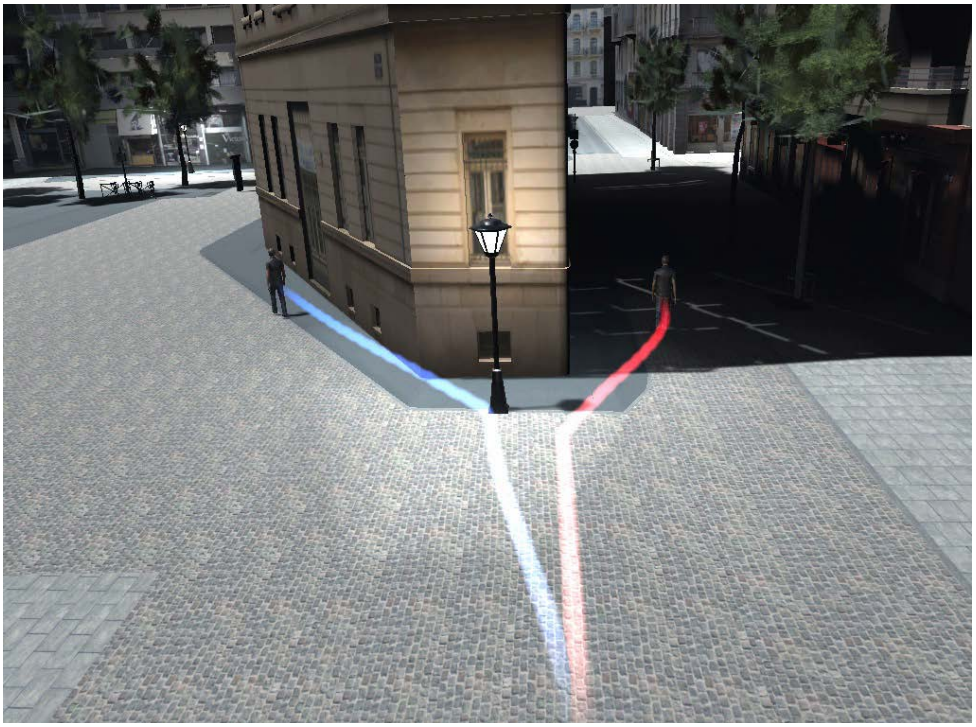


Figure 4.17 – Lit and dark street. The path of the agent forks in two streets, one is lit and the other one is in darkness. The agent takes the lit path to avoid dark regions. The blue trajectory is an agent using numerical flow navigation model, the red trajectory is an agent using the synthetic optical flow model.

sources are a few lamps that illuminate a reduced area. The agent needs to traverse this scene using our numerical optical flow model. The result is the agent moving towards the lamps as it avoids the dark regions. Again, as shadows are a low priority obstacle, the agent may step a bit into them.

4.2.2 Dynamic Lighting

This section presents the results of using numerical optical flow in scenes where the light condition change over time. We show that our agent can adapt to different lighting conditions and still reach the goal and avoid any obstacles in the way.

The first example is shown in Figure 4.19 where an agent walks through the street. In this scene the intensity of the ambient light changes over time, it starts with sufficient brightness to perceive the whole scene as visible. After a short while the light is reduced until the areas in the shadows are completely dark. The result is the agent moving

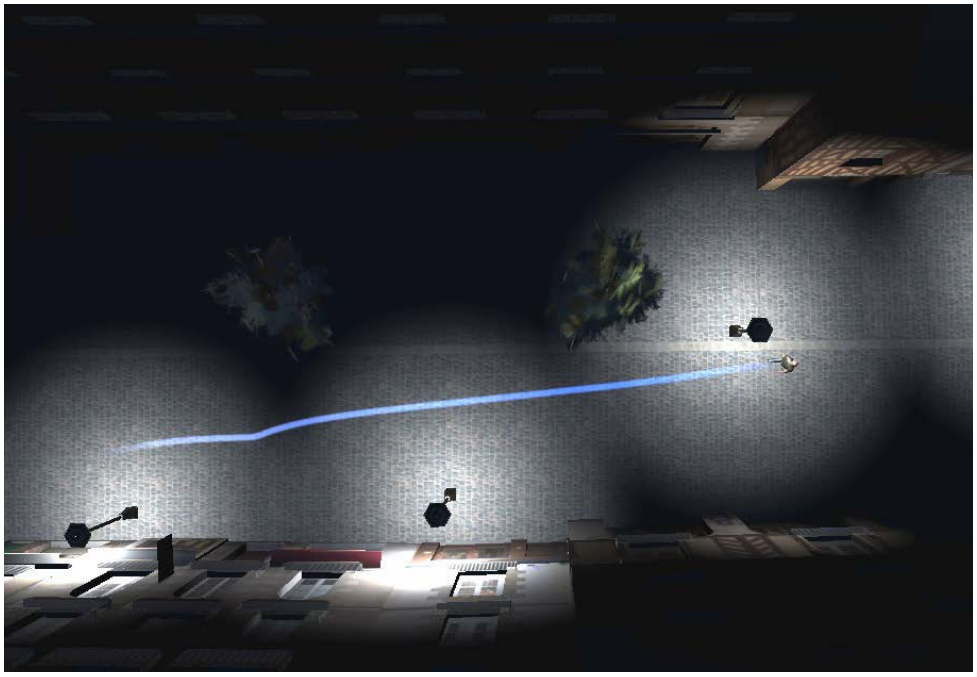


Figure 4.18 – Street with lamps. An agent is tasked to walk along the street. In order to avoid shadows, the agents changes its trajectory to move closer to the lamps that illuminate portions of the scene.

through the shadow at first and changing direction in the middle of the run as the street becomes too dark and searches for the bright areas.

The second example is shown in Figure 4.20. An agent walks in the same street but there is no moonlight and everything is in dark. The agent has a small torch that illuminates a limited area around him. We see the resulting trajectory of the agent passing close to the obstacles as they are only detected when they are inside the range of the torch.

Figure 4.21 shows a second torch example but with two agents moving in opposite directions. In this scene there are two competing factor in the navigation functions. First, agents perform avoidance with each other to prevent a collision. At the same time, the dark region avoidance causes them to move closer as they both have a light source. As a result the trajectories are closer to each other than it would in normal lighting conditions.

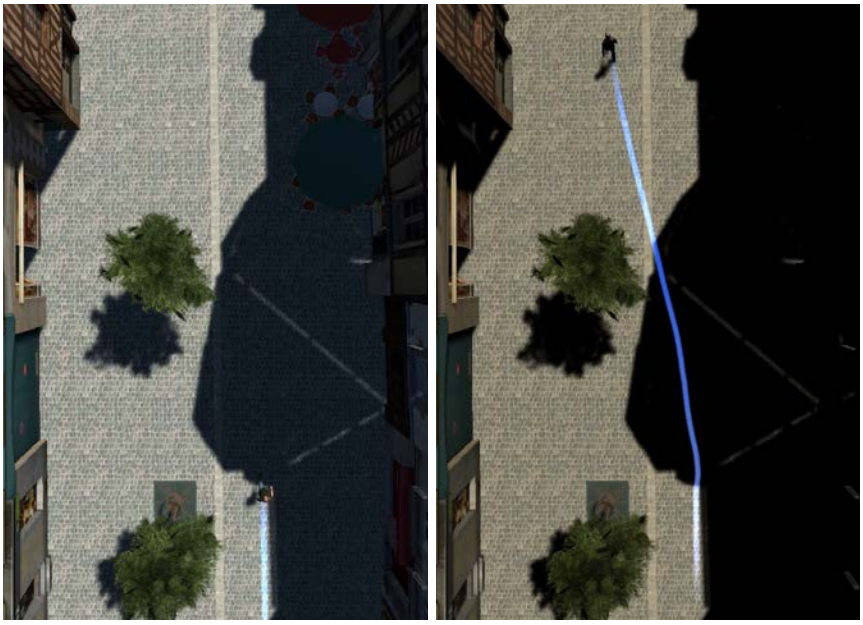


Figure 4.19 – Agent trajectory in a street with varying light. The light in the scene diminishes over time. The shadows eventually become completely dark.

Figure 4.20 – Agent trajectory in a street with varying light. The agent carries a light source which is the only light source in the scene. The agent navigates in the dark and reacts to obstacles as soon as they become visible.

Figure 4.21 – Multiple agents with torches. The street is dark and they move in opposite directions.

4.3 Comparisons with Other Models

Our algorithm is not aimed to compete with other crowd simulation methods as it has a high computation time per frame and per agent. However we still can compare our synthetic optical flow model with other steering algorithms for collision avoidance. We focus on two algorithms, RVO [vdBGLM11] and Dutra's model [DMCN+17]. We chose RVO as it is a well-known technique for character navigation in real-time and it has been used for many applications. The comparison with Dutra's model is more relevant as it is a synthetic vision algorithm but this model perceives geometrical information instead of visual information in our approach.

We provide comparisons of our two control loops presented in this chapter. Synthetic and numeric optical flow are similar yet it is worth comparing how the noise and reduced quality of numerical optical flow impacts the resulting trajectories of the agents.

4.3.1 RVO and Dutra's Model

We now provide a qualitative comparison of synthetic optical flow control loop, and more specifically the collision avoidance function (Eq. 3.18), with RVO [vdBGLM11] and the model proposed by Dutra et al [DMCN+17]. We chose those two approaches because they are good representative of two categories of approaches: velocity-based and vision-based approach. These algorithms share many similarities with ours. All of them evaluate the future risk of collision between the character and surrounding obstacles and adjust the characters motion accordingly. RVO selects at every frame a velocity for every agent that ensures no collision in the near future. Dutra on the other hand, uses a synthetic vision model to optimize time to closest approach and distance to closest approach. The main difference with our method is that our approach relies on the evaluation of visual features whilst previous approaches exploit geometrical relations.

The first example presented in Figure 4.22 illustrates a circle scenario compared with RVO and Dutra's model. The first difference between the models is how much agents deviate from the straight path to avoid other agents. RVO shows the minimum deviation possible as it searches the path that reaches the goal with the minimum effort. As a result it does not take into account personal space. Dutra's model and our control loop maintain more distance between the agents. The reaction of our model

starts early and increases progressively while Dutra reacts a bit later and leaving more personal space between the agents.

Figure 4.23 shows groups of agents crossing each other with a 90 degree angle. We can observe that Dutra's model tends to exhibit a sudden reaction in some situations rapidly increasing the speed of an agent. RVO shows a strong reaction in the sides that have the first interaction and then this propagates to the rest of the group, breaking the formation. Our model preserves better the formation with small adaptations as it finds enough room for the agents to move.

Figures 4.24 and 4.25 shows the same scenario with different agent density in the initial position. Two groups of agents need to traverse the scene in opposite directions.RVO2 works well in low-density scenarios however, the distance between agents of different groups is just the minimum to prevent collisions. In Dutra's model and our model, the minimum distance between agents is not fixed but rather it adapts to the density of the crowd. Dutra's model also favors many variations of speed for avoidance while in our algorithm avoidance favors a change in orientation rather than a change in the speed of an agent as the influence of the rotation is usually greater in the FOE's motion that acceleration. Our model favors following groups of agents with the same direction while in other approaches agents may follow very different paths from one another. RVO2 presents congestion issues when the initial agent-to-agent spacing is much smaller than the size of the agents. Dutra's model and ours allows a continuous rearrangement of the formation over time to minimize the congestion issue.

Figure 4.26 shows the comparison of the adaptations performed by the three models. The histograms shows how intense are the variation of direction of the agents in the example shown above them. We can see that Dutra's model favors sudden strong changes in direction while RVO balances between strong reactions and small ones. Our model shows small reactions and almost no strong reaction yet in this particular example exhibits the larger personal distance between agents. Our model reacts earlier than other models with a smoother adaptation.

4.3.2 Numerical VS Synthetic Optical Flow

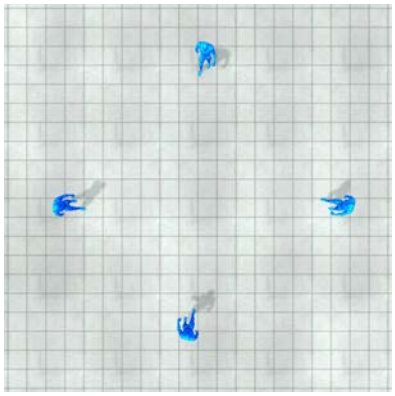
The main distinction between synthetic and numerical optical flow is the lower accuracy in the numeric method. As optical flow solvers are far from perfect the resulting images contain noise and errors. We proved that numerical optical flow is sufficient to

allow our agents to complete navigation tasks. In this section we compare the results of using synthetic and numerical optical flow in the same scenarios.

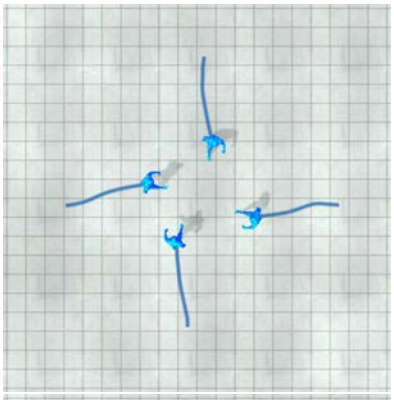
Figure 4.27 shows an agent navigating in a street scenario with static obstacles. The results are very similar in both situations as for static obstacles the numerical optical flow is capable of giving a good estimation of the flow. As expected the trajectory of agents using synthetic optical flow is a smoother curve compared to the trajectory of the agent using numeric optical flow.

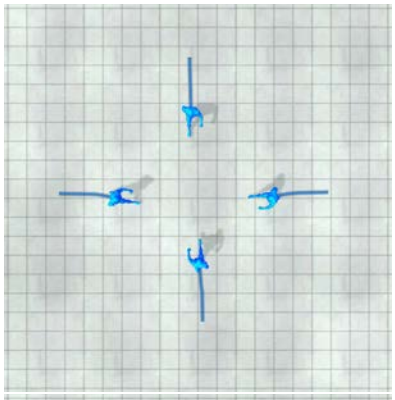
Figure 4.28 shows four agents in a circle scenario that need to reach the opposite side of the scene. When using synthetic optical flow, agents behave in a very organized way and reach the goal a bit faster. When using numerical flow, agents decelerate and accelerate much more resulting in stronger reactions. Some agents slow down to almost full stop to prevent the collision. This is due to the fact optical flow is better estimated when the displacement from frame to frame is superior to the pixel sizes. Synthetic optical flow does not show this problem as it has infinite sub-pixel precision. Still they manage to reach the goal without collisions.

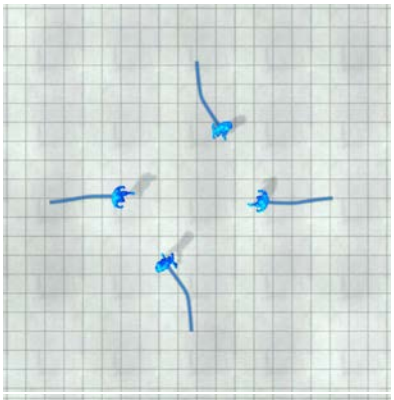
Figure 4.29 shows two groups of agents walking in opposite directions. Once again we observe the same effect, synthetic flow produces more organized groups and smoother trajectories while numerical optical flow produces stronger adaptations. In the end, both reach the goal without any collisions.

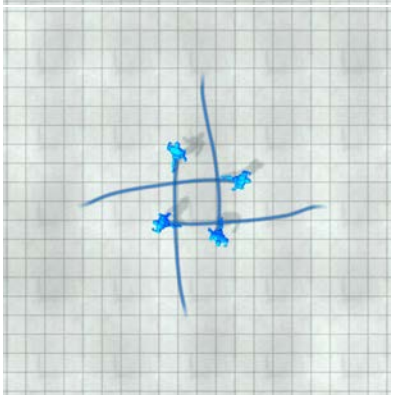


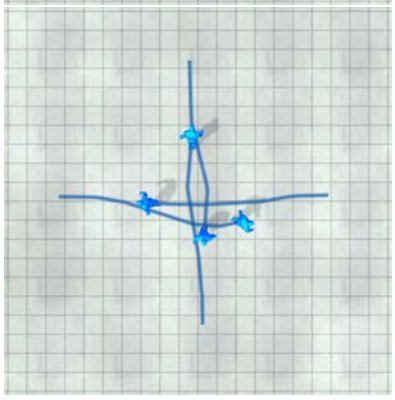
(a) Initial Position

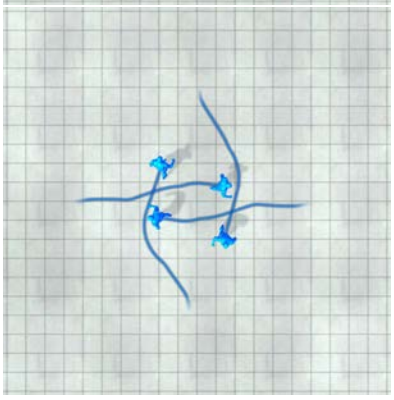






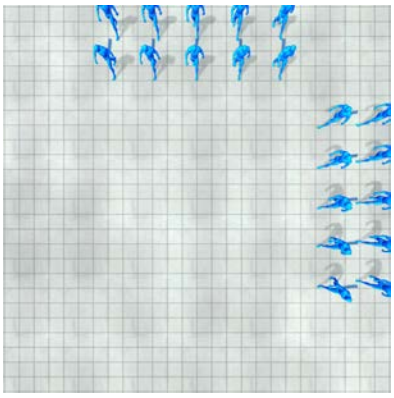




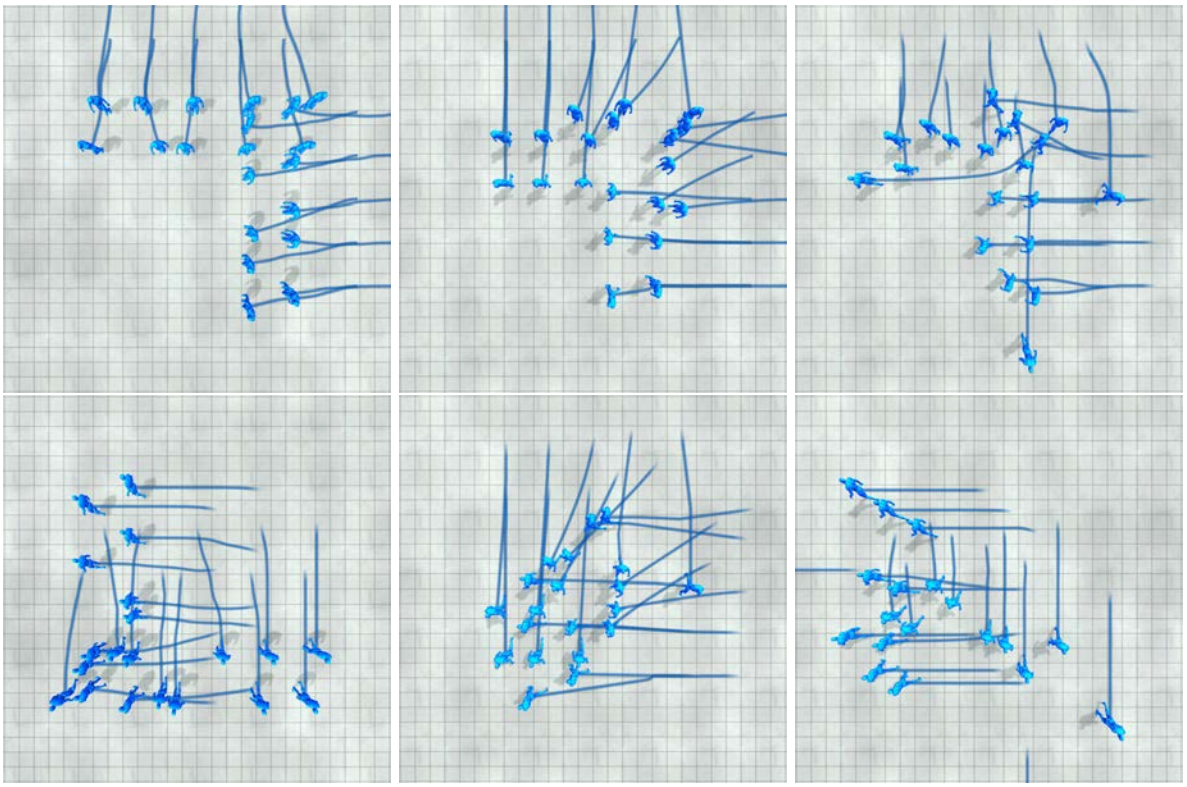


(b) Our model (c) RVO (d) Dutra

Figure 4.22 – Comparison of different character steering models in a circle scenario. Agents need to reach the opposite side of the scene.

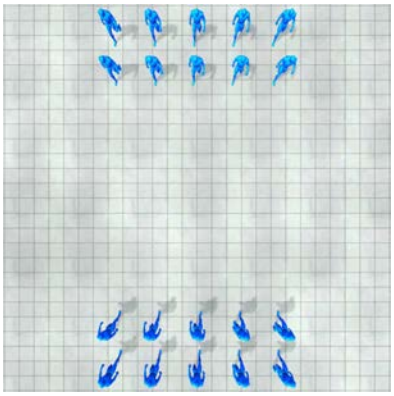


(a) Initial Position

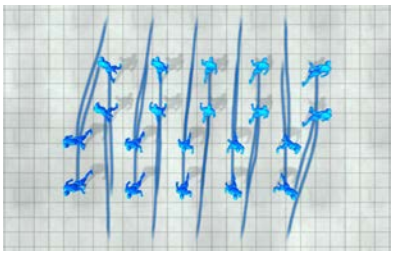


(b) Our model (c) RVO (d) Dutra

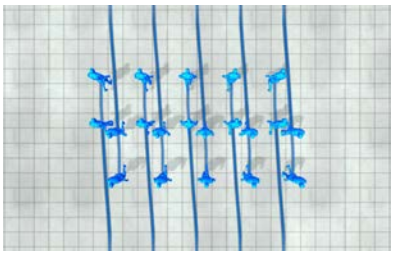
Figure 4.23 – Comparison of different character steering models in a lane crossing scenario. Two groups traverse the scene while crossing each other with an angle of 90 degrees.



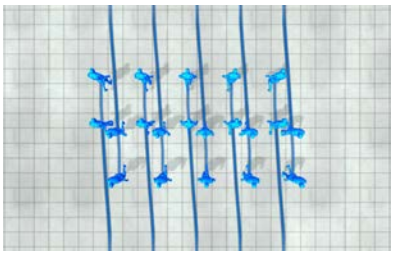
(a) Initial Position



(b) Our model

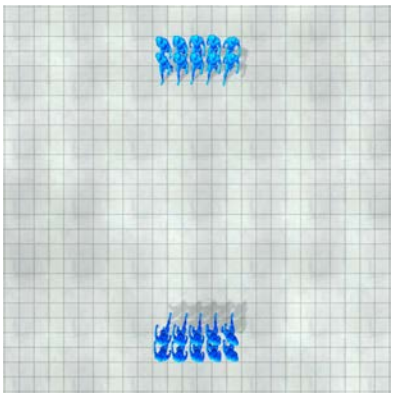


(c) RVO

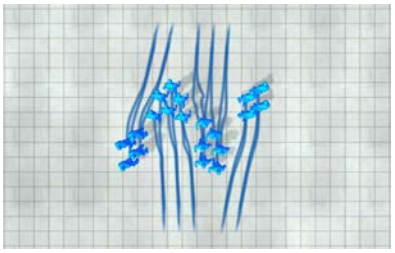


(d) Dutra

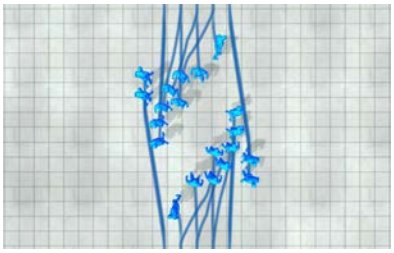
Figure 4.24 – Comparison of different character steering models in a lanes scenario with low density. Two groups traverse the scene in opposite directions.



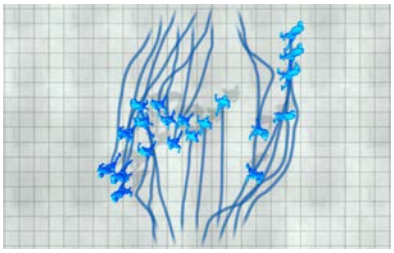
(a) Initial Position



(b) Our model

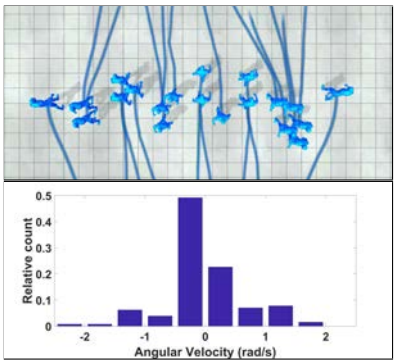


(c) RVO

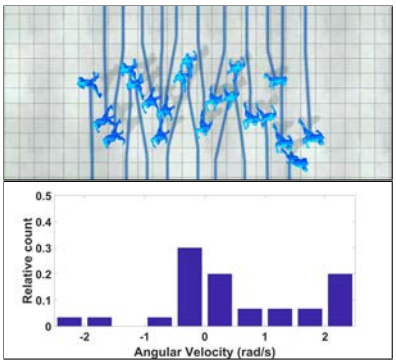


(d) Dutra

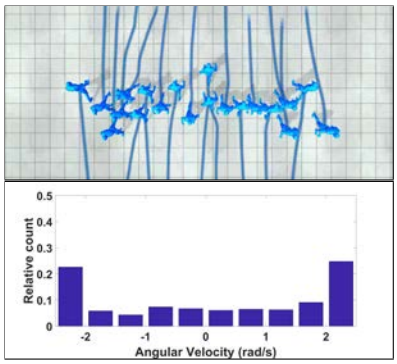
Figure 4.25 – Comparison of different character steering models in a lanes scenario with hight density. Two groups traverse the scene in opposite directions.



(a) Our model



(b) RVO



(c) Dutra

Figure 4.26 – Comparison of histograms of the angular velocity of an agent for different models. Only non-zero adaptations are being displayed.

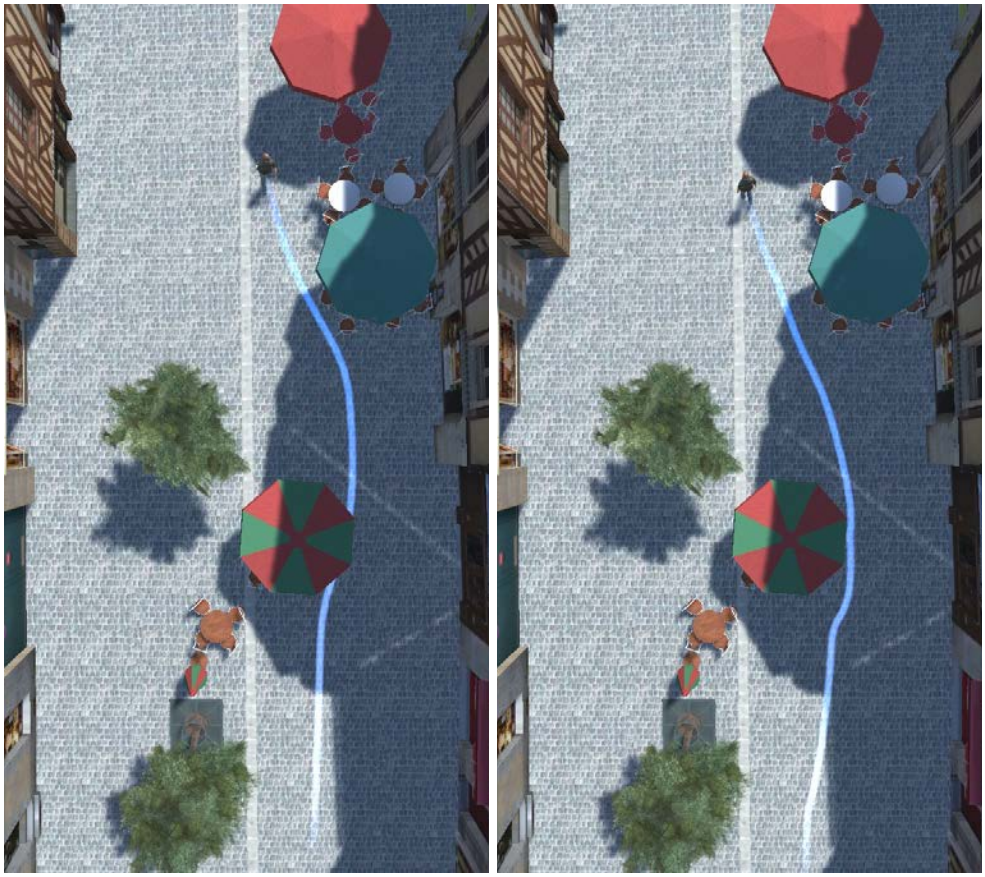


Figure 4.27 – Comparison of trajectories with different optical flow computation methods in a street with obstacles. (left) Synthetic optical flow and (right) numerical optical flow.

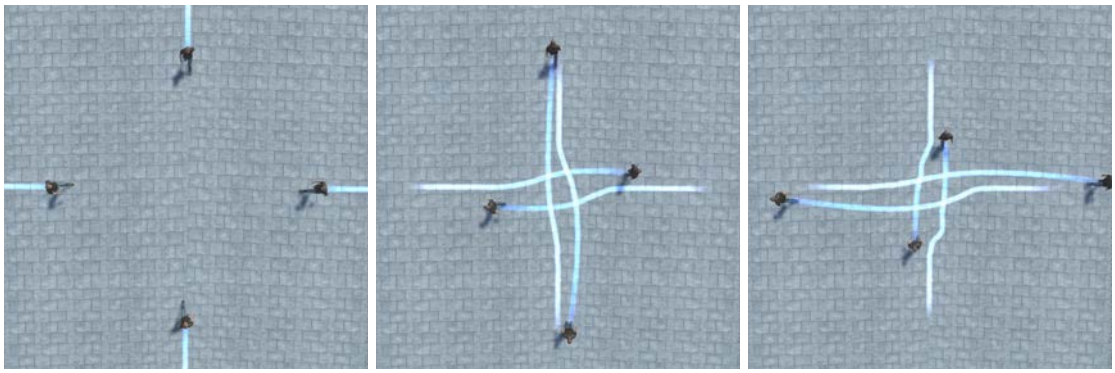


Figure 4.28 – Comparison of trajectories with different optical flow computation methods in a circle scenario. Agents are tasked to reach the opposite side of the circle. (left) Start position, (middle) synthetic optical flow and (right) numerical optical flow.

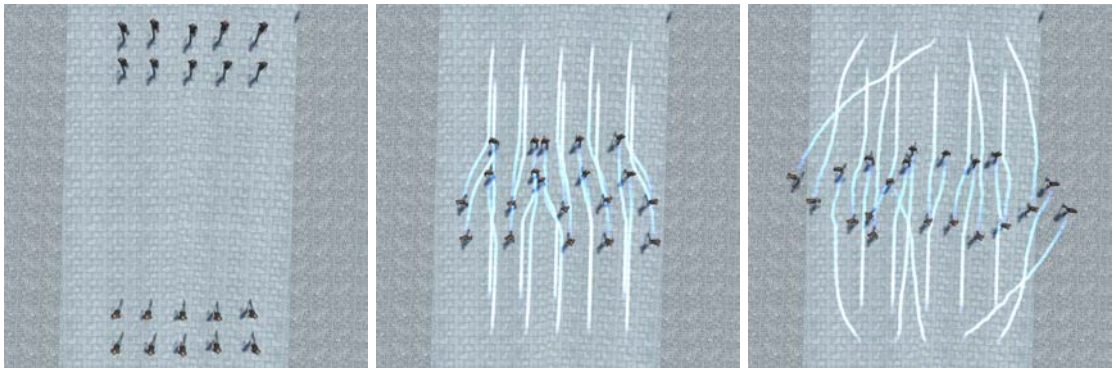


Figure 4.29 – Comparison of trajectories with different optical flow computation methods of lanes of agents. (left) Start position, (middle) synthetic optical flow and (right) numerical optical flow. Agents are tasked to reach the opposite side of the scene.

4.4 Control Function Weights

Control functions are weighted by a set of parameters $\alpha, \beta, \gamma, \delta$ in the navigation function. Different models use these parameters to achieve different tasks and their value impact how the agent behaves and interacts with other agents. In this section we analyze the contribution of the control function by changing the value of their corresponding parameter on its own or relative to another control function.

Target Reaching versus Collision Avoidance

Here we see the relative weight of Target reaching $\mathcal{L}_t$ versus the effect of collision avoidance $\mathcal{L}_{av}$. These two effects are contradictory, target reaching tries to align the agent with the direction of the goal while collision avoidance attempts to deviate the agent from this path in order to prevent collisions with objects in the scene. To compare the effect of these functions we use the navigation function in Equation (3.18) and fix the values $\alpha = 1, \beta = 1$ and change the value of γ.

Figure 4.30 shows two groups of agents tasked to reach the opposite side of the room. This scenario shows the effect of different values for γ. When it is too low (b) the agents barely adapt to other agents and some collisions occur. As γ becomes greater agents with the same goal form groups and increases the distance between them.

From this result we can interpret the relative contribution of $\mathcal{L}_t$ and $\mathcal{L}_{av}$ as a measure of personal space between opposing groups. A high value of γ causes agents to have a greater reaction to any detected obstacle and avoids it earlier.

Velocity Alignment

The control function velocity alignment $\mathcal{L}_{min}$ causes the velocity of an agent to match the velocity of a leader or another object. This control function is used in the navigation function of Equation (3.19) with the aim of allowing an agent to follow an object in the field of view. The corresponding parameter of this control function is α and in this situation its value affects the behavior of the agent by itself.

Figure 4.31 shows an agent following another one. The red agent goes through a set of way-points. Every time the red agent changes its direction to reach the next way-point the blue agent adapts its velocity to align with the one of the red agent. For higher values of α the adaptation is faster and the trajectory of the blue agent becomes

similar to the trajectory of the red agent. For lower values of α the adaptation is much slower producing a different trajectory.

From this result we interpret the value of α as the reaction time of the agent to adapt to changes in velocity of the leader. The velocity alignment parameter still interacts with collision avoidance γ in the same way as with target reaching in the previous example. Therefore, designing the reactions of an agent following a leader requires fine tuning of both parameters.

Speed Control and Collision Avoidance

The control function speed control $\mathcal{L}_v$ in Equation (3.13) defines a comfort speed for the agent and is controlled by the parameter β. The purpose of this is two-fold, first it allows the agent to keep a constant speed when no other behavior are active and second it limits how much other control functions are allowed to change the velocity of the agent.

Figure 4.32 shows a group of agents in a circle scenario with different values of β for all agents. The trajectories of the agents result very similar for all values of β however the completion time for all agents is reduced for higher values.

This result shows that β is a measure of how much an agent slows down to allow other agents to pass. It can be interpreted as how much priority has the navigation task for a certain agent as it is more or less permissive in changing its speed. For higher values even collisions between agent can occur as they would not slow down to prevent it.

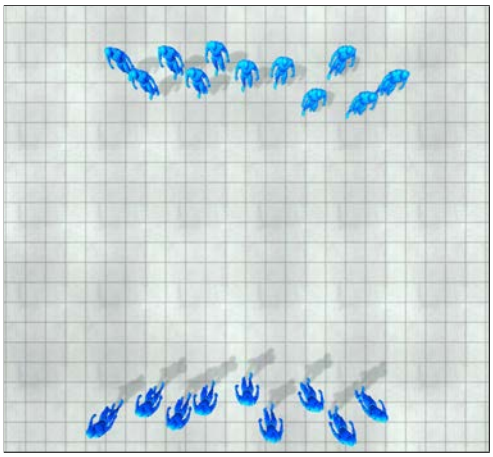
Dark Object Avoidance and Collision Avoidance

The control function for Dark Object Avoidance $\mathcal{L}_{dark}$ is responsible of moving the agent away from dark regions where numerical optical flow is unreliable. It is relevant to evaluate the effect of the relative value of its parameter δ and collision avoidance of visible obstacles γ. The value of δ is usually lower than the value of γ as avoiding visible obstacles has a higher priority than avoiding shadows that may or may not contain an obstacle. We fix the value of $\gamma = 1$ and change the value of δ.

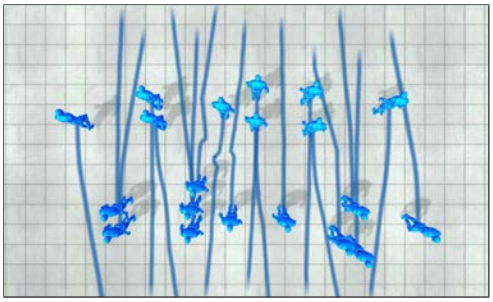
Figure 4.33 shows examples of different values of δ. For lower values agents almost ignore the shadow and step through it with minimal reaction as the presence of the tree is prioritized. For higher values the agent even moves farther away from the shadow

changing completely the trajectory. It still goes through the small shadow as the effect of collision avoidance with the walls still competes with the dark avoidance.

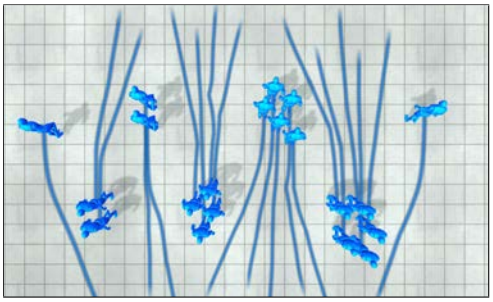
With this result we interpret the value of δ as a measure of danger in shadows. This allows to define different models of agents with different sensitivity or phobia to darkness. Values of δ larger than γ generate strong repulsive reactions to the presence of shadows.



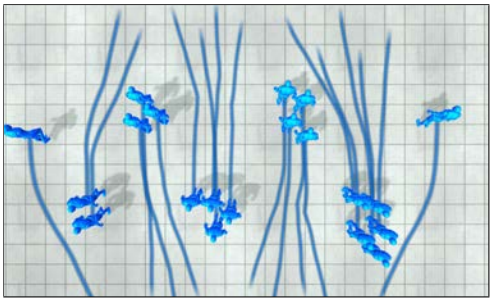
(a) Initial position.



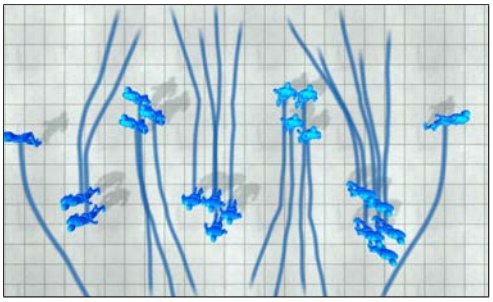
(b) $\gamma = 0.1$



(c) $\gamma = 0.5$

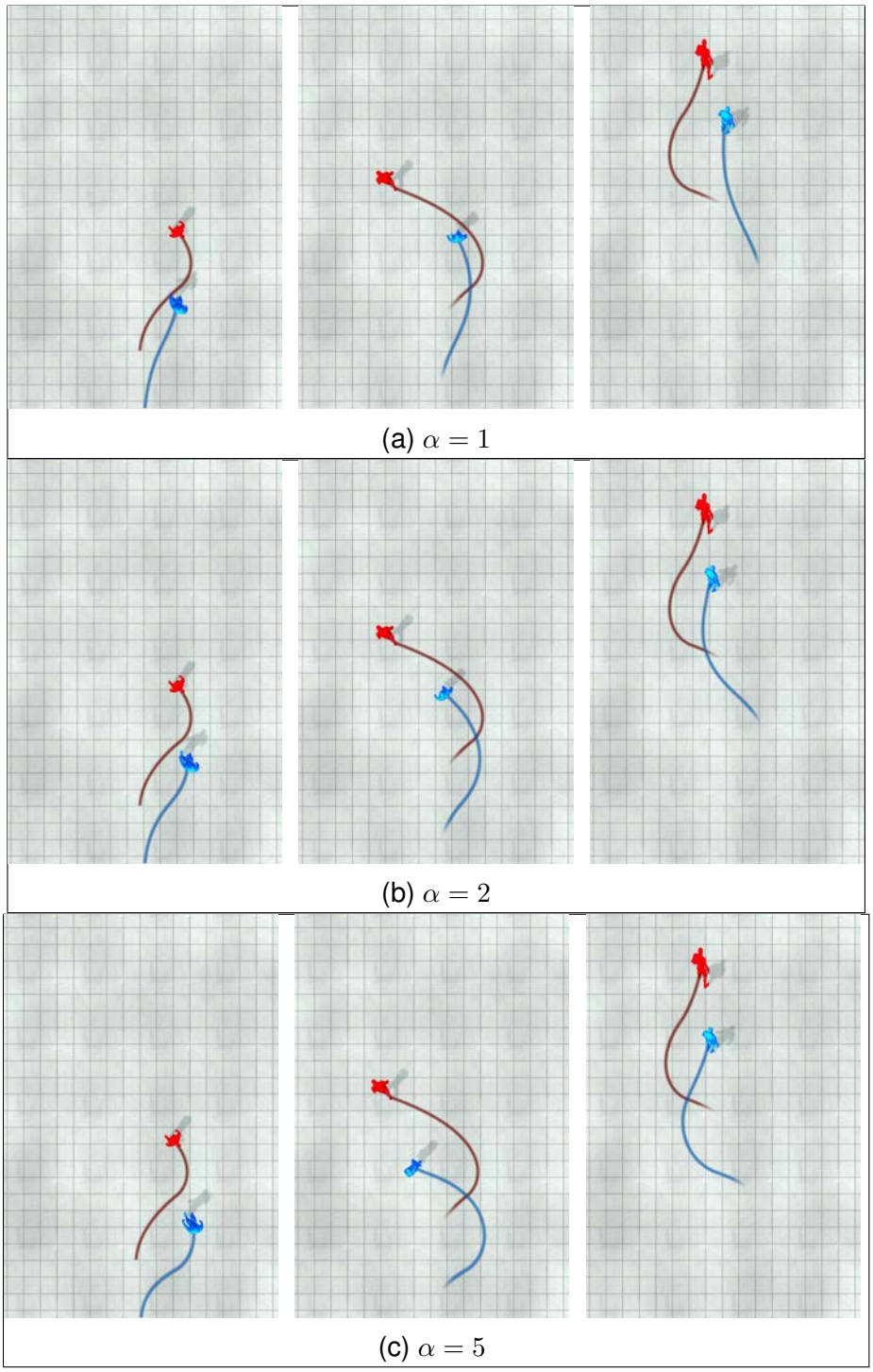


(d) $\gamma = 1.0$



(e) $\gamma = 1.5$

Figure 4.30 – Two rows of agents navigate towards the opposite side of the room avoiding each other. As γ increases, the distance between agent with different goals increases.

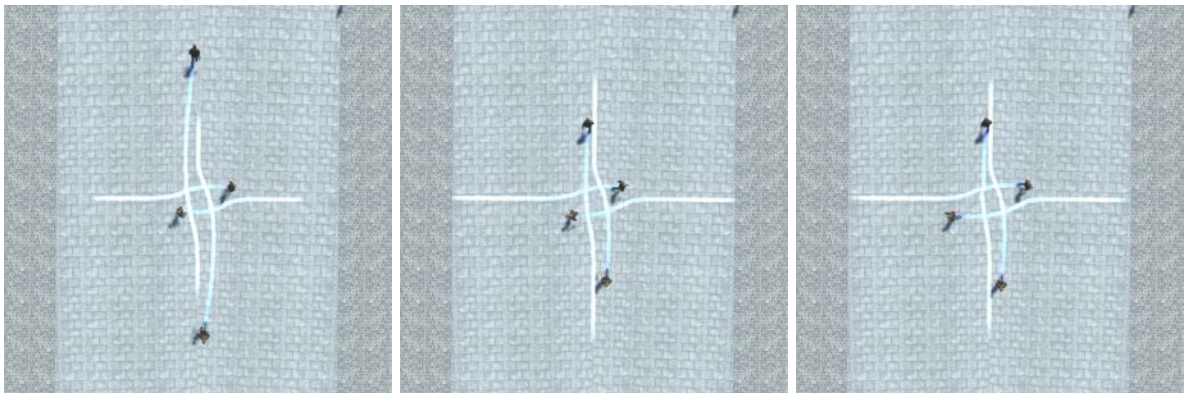


(a) $\alpha = 1$

(b) $\alpha = 2$

(c) $\alpha = 5$

Figure 4.31 – Following a leader. The blue agent must follow the red one, which goes through a predefined set of way-points. We show the resulting trajectory for different values of α.



(a) $\beta = 0.1$ (b) $\beta = 1$ (c) $\beta = 2$

Figure 4.32 – Trajectory comparison of an agent for different values of β. Agents are in a circular formation and need to reach the opposite side of the scene. The completion time for all the agents is reduced for larger values of β.

(a) $\delta = 0.1$ (b) $\delta = 1$ (c) $\delta = 1.5$

Figure 4.33 – Trajectory comparison of an agent for different values of δ. The agent is in a dark street and needs to traverse it while avoiding the shadows and other obstacles.

4.5 Performance

In this section we provide details regarding the performance of our algorithm. We describe the hardware used for each control loop and give details about the computation time and the resolution perceived by the agents. As the computation of numeric optical flow is an expensive GPU operation, we decided to use an improved GPU to test this control loop.

4.5.1 Synthetic Optical Flow Performance

In this section, we give indicative numbers about the performance of the synthetic optical flow control loop. Simulations ran on a 2.17GHz Intel Core i7 processor, 16 GB of RAM, Nvidia Quadro M2000M using Unreal Engine 4 as the graphics engine.

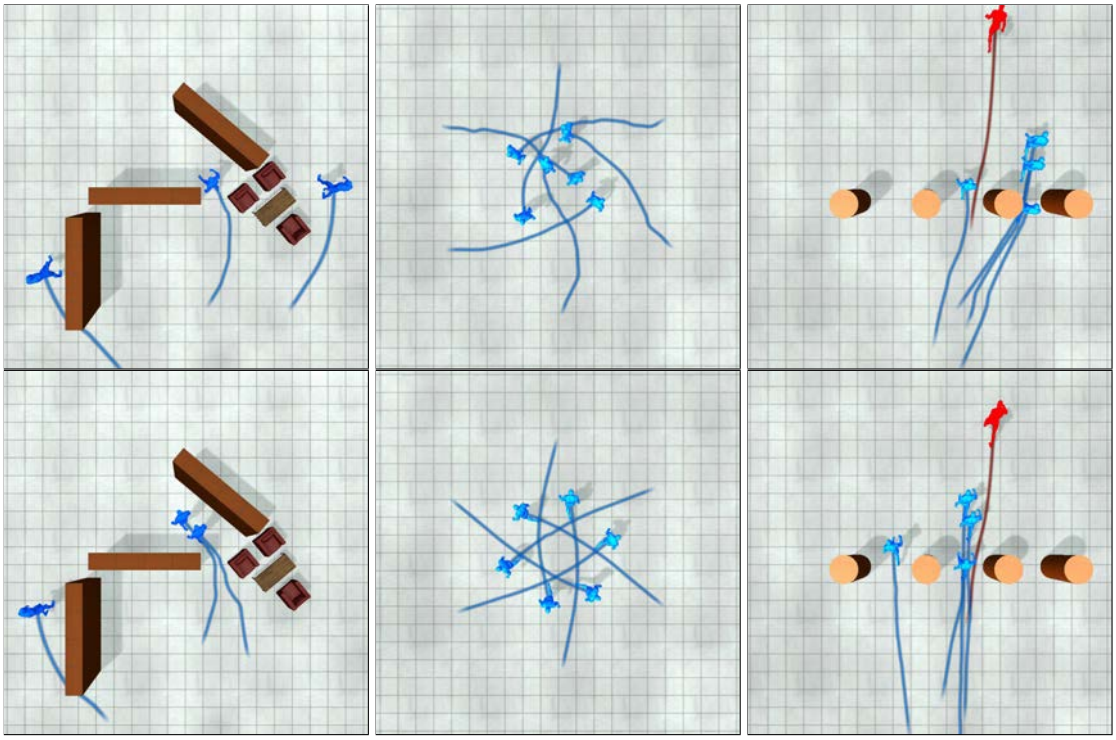
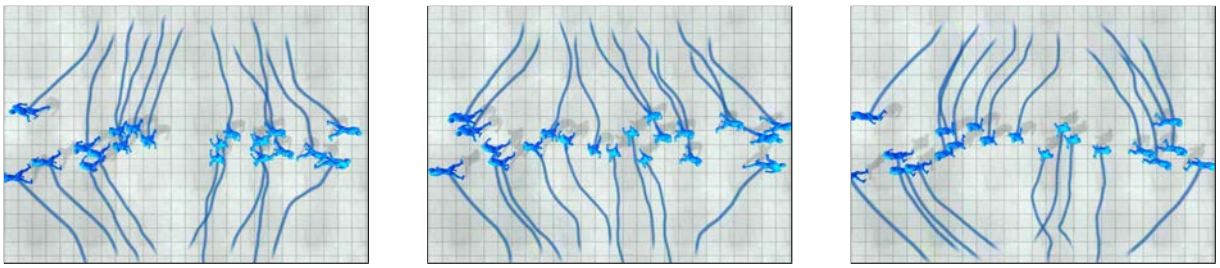


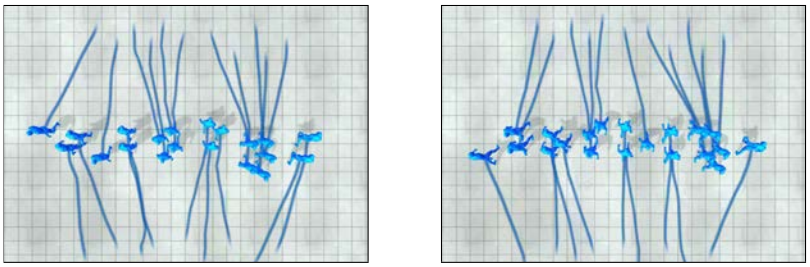
Figure 4.34 – Variation of trajectories depending on optical flow resolution. The first row shows the resulting trajectories of a 300x300 flow map resolution and the second row shows the same scenario with a 1024x1024 resolution.

The performance of our algorithm depends on the chosen resolution of the images. The GPU renders the flow image perceived by the agent and then they are downloaded

to the CPU. The data transfer takes a lot of time and thus we believe performance could be greatly improved with a full GPU implementation. We ran most of the simulations using a high-resolution camera 1024x1024. Under these conditions, every agent takes on average 0.48s to update. We can lower the resolution to 300x300 increasing performance to 0.03s per agent. Figure 4.34 shows the same scenarios shown through the paper with high and low flow resolution. High-resolution flow results in a quite straight trajectory while low resolution flows produces trajectories with higher curvature and noise due to the reduced precision in object position. In the first comparison, an agent even takes a very different path.



(a) (64x64) 0.003s (b) (128x128) 0.006s (c) (256x256) 0.027s



(d) (512x512) 0.111s (e) (1024x1024) 0.48s

Figure 4.35 – Trajectories of the scenario shown in Figure 4.30 for various resolutions. Computation time is shown for every resolution.

Figure 4.35 shows a comparison of trajectories of the scenario shown in Figure 4.30 for different resolutions and the computation time per-agent and per frame. For lower resolutions, collisions occur more often due to the reduced spatial precision, causing object segmentation to be less accurate.

Our model can be extended to simulate large crowds at the expense of higher computational time per frame. Our agents are updated concurrently, therefore, only one optical flow image is necessary at the same time. The memory requirements of this model is a 3-channel image encoding optical flow vectors (2 components) and the depth map. We use 16 bit per channel in order to obtain a high quality optical flow map.

The final memory requirement for our approach at any time is a 6 bytes per pixel times the resolution of the image which scales well for a high number of agents.

4.5.2 Numeric Optical Flow Performance

Computing optical flow using a sequence of images is an expensive operation. We decided to use FlowNet2.0 [IMS+17] to compute it as we believe it to be a good enough compromise between accuracy and performance. This is a deep learning approach and works best in powerful GPUs.

For the numeric optical flow control loop, simulations run on a 3.19GHz Intel Xeon processor, 32GB of RAM, a GTX 1080 Ti and the Unity engine to render the scene. The implementation of FlowNet2.0 used for this control loops is limited to a fixed resolution 448x320. Color images are scaled to match this resolution, then the resulting optical flow image is scaled back to the original resolution. Still, the resolution of the perceived images by the perception scheme affects the computation as object segmentation and dark region detection are CPU operation that depend on the size of the images. In most of the simulations our agents perceive images of 512x512 pixels. At this resolution, every iteration of the control loop takes on average 200ms. At a resolution of 256x256 the control loop takes 100ms. At higher resolutions, 1024x1024 it takes 750ms.

The memory requirements in this control loop are higher than for the synthetic optical flow control loop. First, as optical flow is computed using a pair of consecutive images, every agent needs to keep the previous color frame in local memory which increases the amount of memory used for large crowds. Then, the usage of FlowNet2.0 to capture optical flow requires a large amount of GPU memory as most deep learning approaches. However this does not increase with the number of agents as all agents are updated sequentially. Finally, the semantic image is a simple 1-channel 8 bit image as the number of objects present in the scene does not exceed 256 in our examples.

CHAPTER 5

DISCUSSION AND FUTURE WORK

This thesis has presented a new control algorithm for virtual human simulation. In the previous chapters we described a new perception model based on visual information and how it can be used to design multiple navigation models. The last chapter presented the trajectories generated by these models and evaluated them against other approaches. This chapter discusses the results of this navigation scheme its applications and new directions of research from this thesis.

5.1 Low Level Control

Virtual human simulation usually involves multiple layers that encodes how an agent moves through an environment. Figure 5.1 shows the multiple levels involved in agent control. The higher levels involve the motivations of the agent, what is his goal and what paths are available to reach this point of the scene. This involves path planning techniques to process the configuration of the scene with the general information about it. Then, it needs to communicate with the low level control to assign the agent model with a goal or set of way-points for the low level control scheme to follow. This thesis contributes to the low level control, providing a new agent model based on a novel perception scheme.

Our model consists in several layers, all of them contributing towards the final behavior of the agent. First, the expressions of the control functions defines the task is accomplished and how the agent interacts with the environment. Then, navigation functions combine the different tasks into a higher level behavior, that weight the different actions the agent need to perform. Finally, the optimization model dictates how the control variables of the agent are updated to reach the goal.

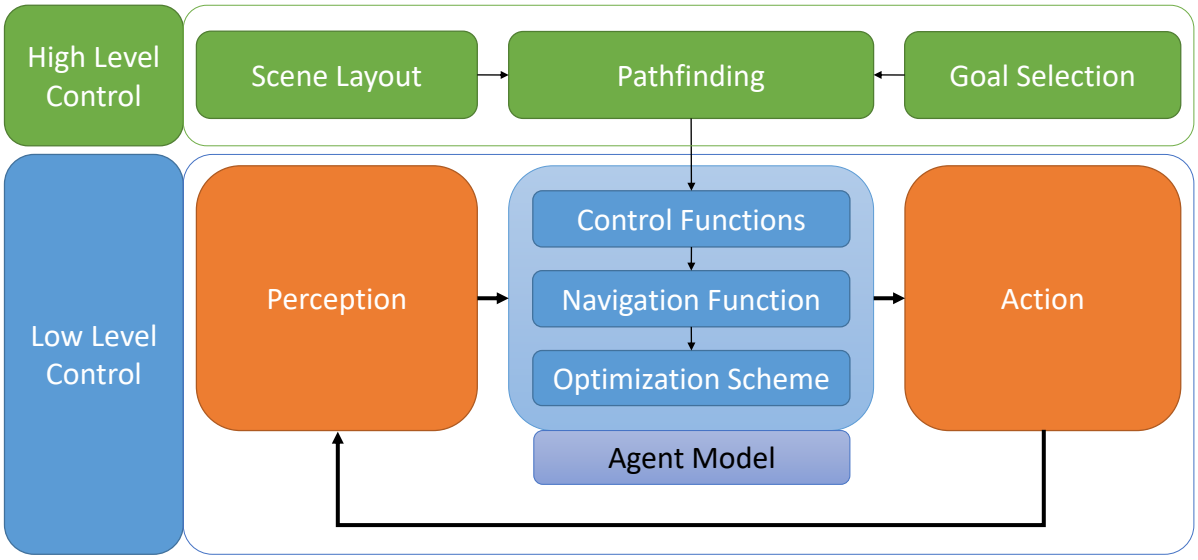


Figure 5.1 – Multi-layer control of virtual humans.

Future Work

There are multiple future directions from this work. First, this thesis provides a framework and the building blocks to define agent behavior based on visual information. To extend this work, additional control variables and control functions may be defined to enable more complex behavior. New tasks such as keeping a certain formation, a constant distance between agents, selective collision avoidance functions as not all obstacles are considered in the same manner.

Most animals rely on visual information to perform navigation and optical flow is known to play an important role for them as well. Our control scheme can be adjusted to reproduce the motion of animal life such as birds or fishes. Their locomotion is completely different from the human model as they navigate in a 3D space rather than the ground plane. New control functions should be defined to take into account this behavior and properly model these kinds of animals.

Most models, ours included, do not mix the different control layers to improve navigation. An interesting line of research would be to allow the visual perception scheme to interact with the higher level control scheme to update the layout of the scene when detecting blocked doors or corridors.

Finally, we may deal with the problem of selecting the adequate navigation function for each situation. In our model the parameters that weight each control function are

fixed through the simulation. However, real humans exhibit different behaviors depending on the specific situation, the amount of individuals or obstacles may change the minimum distance allowed between them. Higher level control could adapt these values or use a different navigation function entirely to adapt agents to different situations.

5.2 Benefits of Optical Flow

Our model is based on the use of optical flow and other visual information to simulate virtual humans. We have designed a control system that is capable of recording this information, processing it and generate an action to move in virtual environments. This is one of the contributions of this thesis as we try to reproduce the interactions of the human perception and locomotion systems. Our virtual agents are capable of perceiving optical flow as humans do [WKZ+01] but also depth, color images, semantic information and perceive the lighting conditions of the scene. Humans are naturally capable of perceiving the depth of the environment using both eyes (stereo vision) or through the learned size of objects. They also perform a semantic segmentation as they are capable of identifying different types of objects. Therefore, the control loops presented in this thesis are a step forward in reproducing human perception and behavior.

From the simulation point of view, the use of visual information allows the simulation to naturally reproduce visibility, occlusions, detecting risk of collisions through optical flow and sensitivity to the luminosity of the scene to react to lighting conditions. Our model is capable of handling all these phenomenons without any prior knowledge of the scene.

Future Work

One of the benefits of using optical flow and visual information is to model and simulate the perception system of human for virtual agents. This opens a new ways of research for experimental science as the next step is to compare our control loops with real human data. This would allow further validation of our model and can improve the study of real human behavior. In particular, it would allow to further explore what kind of visual features are most relevant in human navigation.

In this thesis we only explored visual information however humans poses additional

sense that are equally important for navigation. To further reproduce the human perception system sound and tactile information could be added to the inputs of the control loop. This involves developing new control functions able to handle and use this information. Another characteristic of humans that is worth of study, is the use of spatial memory. Our model only takes into account the visible information in a particular frame and then it is discarded. Adding memory of visual information would improve the performance of agents and better reproduce human behavior.

5.3 Dark Regions and Uncertainty

In this thesis we have demonstrate a novel approach to consider lighting conditions. Our agents are capable of navigate a virtual environment and can process the light perceived by their virtual sensor to produce a basic reaction to the differences in luminosity of the scene. The main motivation for for this behavior is the reliability of the visual sensors in regions with little visual information. Dark areas may contain obstacles which are not visible in the numerical optical flow map and therefore our agents would be unable to avoid them. The main benefit of our method is that the agent does not need any prior information about the luminosity of the scene.

Future Work

Our model provides a basic reaction to lighting conditions as agents perform avoidance on those regions where optical flow is unreliable due to the lack of light. To further improve this behavior more complex analysis of the perceived light and optical flow of the scene could be made. Optical flow could be given a reliability score to further evaluate how good the estimation is in different regions depending on the contrast of the image. This would improve general optical flow estimation and navigation but also allow to consider different lighting conditions.

When considering lighting conditions additional behaviors could be implemented. In situations of extreme darkness, humans tend to reduce their speed and use other sensors such as sound or tactile. This could be emulated by creating a new control function to slow down agents when darkness covers most of the visual sensors.

Our model avoids dark areas, however, humans can also do the opposite. In regions of very high temperatures humans tend to search darker zones and avoid the sunlight

to prevent sunburns. This behavior could be easily modeled by performing avoidance with lit regions attraction to dark regions.

Finally, our model does not distinguish dynamic from static lighting conditions as it does not have any concept of memory. In dynamic lighting conditions an agent holding a flashlight could memorize the environment and have a lit reconstruction of the scene as humans do.

5.4 Impact of Local Optimization

The agent models presented in this thesis rely on the use of a local minimization. The control functions are designed to update the agent's control variable using a gradient descent approach. The benefit of this approach is that the resulting trajectories tend to be smooth as the visible information usually does not change dramatically from frame to frame. Still, objects can enter or exit the field of view or being temporarily occluded by other objects which changes the control function values significantly. The main danger of local optimization models are local minimum that may render our agent stuck between obstacles. Our model does not check if the minimum value of the navigation function is a local minimum that would result in a collision. This is specially true in a high density situation such as the example shown in Figure 5.2 where agents collide with each other and get stuck.

Future Work

Even though we have demonstrated in several situations that the local optimization approach is sufficient to perform navigation in virtual environments the risk of facing local minimum remains. To solve this problem we could use a different solver to optimize the navigation function to find a global minimum. This would result in a different behavior of agents as the model depends on the optimization scheme.

Gradient descent is a simple method for optimization that allows agents to converge towards minimums in the control functions. To improve the behavior of the agent a second order minimization solver could be used to enable faster convergence. This would not change dramatically the behavior of the agent but it would allow a more stable control.

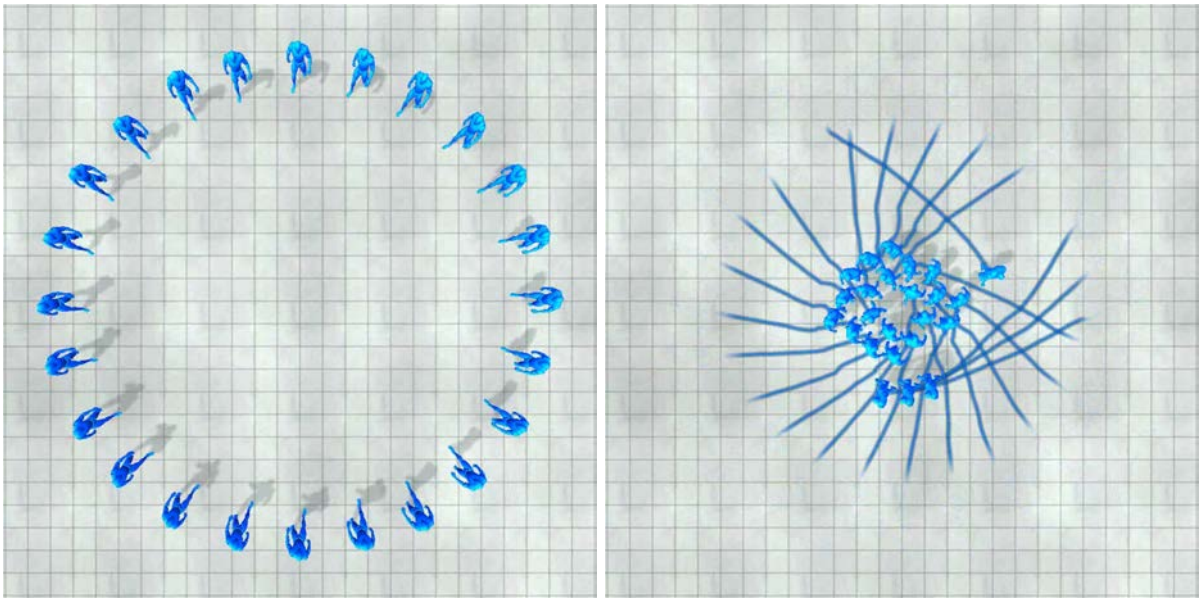


Figure 5.2 – Failure in high density situations.

5.5 Performance

As shown in Section 4.5 our models are computationally expensive and would not be suited for large crowd simulation in the current implementation. However, we believe there is a lot of room for optimization. In both synthetic and numeric optical flow one of the main bottlenecks is the transfer of information from GPU to CPU. Most of the visual information is generated using the render engines in GPU but they are processed in the CPU. The larger the size of these images, the higher the overhead of this transfer is. Then, every pixel of this images is analyzed in the CPU to perform image segmentation, extraction of objects and visual features. This adds more work to be done in the CPU. An implementation using the resources of the GPU for images processing could improve the performance of our algorithm as well as the compression of the images that still need to be downloaded could result in an increased performance.

5.6 Machine Learning

In the recent years machine learning has been applied to an increasingly array of different problems. In this thesis we work with visual information as images. In fact, in our model numerical optical flow is computed using a deep learning approach. There-

fore, a neural network could process the visual information and produce an output to the control variables of the agent to steer them and enable navigation.

In order to train a neural network to allow navigation a very large data-set of trajectories and images is required in order to generate an agent model capable of reliably traverse an environment. Our models do not require any prior training as we explicitly define the visual features required for navigation. This allows us to better understand the nature of navigation using visual information and in particular optical flow. Our method also maintains full control over the reaction of the agents to the perceived visual features while the machine learning approach hides this information. The main benefit of a machine learning approach is that there is no necessity of studying the complex interactions of the visual features as the model would learn them. Regardless, there is no proof to this date that it is actually possible to obtain a reliable perception-action control loop using visual data with machine learning.

5.7 Realism

The goal of this thesis is to simulate human behavior by reproducing their perceptual system. Limiting our agents to use only visual information greatly limits their perception of the environment but it also allows new information to be processed such as light conditions. We have presented a framework to use visual information to define navigation tasks and presented two control loops, one using synthetic optical flow and a second one using numerical optical flow.

The first control loop uses synthetic optical flow and depth information. This provides a precise optical flow map that can be segmented simply by finding discontinuities. This proves again that optical flow contains sufficient information and that it is a feasible source of information for autonomous agents. Humans can easily capture depth information using stereoscopic vision for short distances and even at large distances they can give a good estimation. This makes our model to be closer to real human than other models using geometrical information.

Then, with the aim of obtaining greater realism proposed a second model using numerical optical flow instead of synthetic. This means that, our agents have only access to color images and semantic information. As numerical optical flow is noisy and prone to have errors, the semantic image is necessary to separate objects from one another. As our agent can only access color images, we need to deal with missing information

or darkness. Light conditions change the way humans navigate an environment as they need to account for lack of information in their field of view. Our model can take into account these conditions and changes the behavior of the agent accordingly. This results in the first navigation algorithm capable of taking into account lighting conditions without any previous knowledge of the scene.

Future Work

Despite our model being a closer representation of human perception that most other models, we have not validated it with trajectories of real humans. This requires experiments with real humans that navigate in real environment while recording their trajectories and compare this with the results of our model. Furthermore, the use of virtual reality environments allows full control on the scene and what humans perceive. This could be exploited to further explore what visual features are most relevant for navigation and adjust our model accordingly.

5.8 Robotics

The robotics community has studied the problem of robot navigation in real environments. This thesis and the robot navigation problem have several common points. We try to reproduce the human perception system to allow agents navigate virtual environments. This means using visual information or images and process them to achieve an autonomous virtual human. In robotics they try to design a robot that can safely navigate with real humans using limited on board optic sensors. This makes our model suitable to be implemented in a robot. Our numerical optical flow control loop uses color images to perceive the environment similar to how a robot would capture an image of the environment. A future direction of research would be to adapt our model to enable safe robot navigation with humans.

The main problem to achieve this goal is semantic images. Our numerical optical flow model generates semantic images to perform image segmentation using the simulation data. A robot does not have access to this kind of information and therefore, need to obtain this information in a different way. Semantic segmentation is an open problem for the computer vision community and many solutions have been proposed. A deep learning solution to this problem [ZQS+17] may be able to provide a robot the

semantic images with sufficient accuracy to apply our algorithm in the real world. An alternative to this would be a reliable method to detect and segment obstacles using the optical flow image.

CHAPTER 6

CONCLUSION

The goal of this thesis is to reproduce the human visual perception system to virtual agent simulation. The problem is two-fold, what visual information humans use to navigate and how it is exploited to build a perception action loop for virtual environments. To address these questions this thesis describes a new perception and control framework for virtual humans that allows the acquisition of visual information to perform navigation tasks.

This work introduces the use of optical flow as the main source of information of the scene. Optical flow encodes the apparent motion of objects perceived in the field of view and it is known to play an important role in human navigation [WKZ+01]. It is a dense map and therefore it needs to be processed to extract the relevant information to complete tasks such as collision avoidance or following. Knowing the importance of this map, we built a new perception system that enables agents to perceive visual information as opposed to geometrical information in previous models. Two ways of computing optical flow have been proposed, synthetic and numeric optical flow. Synthetic optical flow encodes the projected motion of objects in the field of view an is obtained by querying the velocity of objects to the simulation. This results in an optical flow map with high accuracy that can be used to detect objects in the field of view. Numerical optical flow on the other hand is computed using a numerical solver on a pair of consecutive frames. The resulting optical flow is noisy and less accurate. Regardless, it can be used to perform navigation in a similar manner as for synthetic optical flow.

Then, we designed a new control model to exploit the information encoded in optical flow and other visual information. Our agents are capable of reaching a goal while avoiding obstacles, following a leader and even react to different lighting conditions. We designed two control loops that exploit the information in each of the optical flow maps.

We tested the control loops proposed in a variety of situation to demonstrate their effectiveness. The control loop using synthetic optical flow showed goal reaching with

collision avoidance and following with collision avoidance. We demonstrated that our agents can navigate static and dynamic environments with multiple agents and obstacles. As our model uses synthetic vision it can handle objects of arbitrary shapes and sizes without geometrical information of them.

The second control loop uses numeric optical flow which results in less smooth trajectories overall with respect to synthetic optical flow. Still, this model has demonstrated that it can navigate various environments and it can even react to light conditions. We designed agents with a repulsive behavior to dark areas making them avoid shadows and follow the lit path.

The results shown in this thesis demonstrate the benefits of using optical flow-based control loops and provides means to better reproduce human behavior.

Contributions

This thesis brings three contributions to the virtual human simulation problem.

The first contribution is detailed in the first part of Chapter 2. It describes a new perception system for virtual humans that is capable of recording visual information. Agents are capable of perceiving Optical flow, color, depth and semantic information as images. To use these images in a navigation problem, they need to be analyzed and processed. Obstacles in the field of view need to be segmented from the background to compute a set of visual features, their position in the field of view, the Focus of Expansion, time-to-collision and the optical flow vectors generated by the object.

The second contribution is the study of the visual features and their relation with the agent motion. The Focus of Expansion contains the information of the relative velocity between the observer agent and the obstacle. Therefore, it is the most important feature for collision avoidance. This point needs to be separated from the object to ensure a safe path. Time-to-collision of an object evaluates the remaining time until the object crosses the image plane of the agent. Therefore, this feature is used to identify the relevant obstacles that need to be avoided and discard those that are far away or moving in the same direction as the agent. Optical flow encodes the relative motion of the observer and the object and it is null when the two move in the same direction and with the same speed. Therefore, it can be used to perform following.

The last contribution is separated in two parts and is the definition of agent behavior using the visual features extracted from the perception system. Simple task are

expressed as mathematical expressions of the visual features in the control functions. The control functions described in this thesis involve target reaching, collision avoidance, dark obstacle avoidance, velocity alignment and speed control. This set of control functions are combined in navigation functions that describes how an agent navigates its environment. An important novelty in the behavior of agents is the analysis and reaction to lighting conditions without any previous knowledge of the scene.

PUBLICATIONS

1. **Character navigation in dynamic environments based on optical flow**
   A. Lopez, J. Pettré, E. Marchand, F. Chaumette. Computer Graphics Forum, (Eurographics 2019 Proceeding), 38(2):181-192, Mai 2019
2. **Attracted by light: vision-based steering virtual characters among dark and light obstacles**
   A. Lopez, J. Pettré, E. Marchand, F. Chaumette. ACM Motion, Interaction and Games (MIG '19), Newcastle upon Tyne, United Kingdom, Octobre 2019.

BIBLIOGRAPHY

[AGS+11] Junghyun Ahn, Stéphane Gobron, Quentin Silvestre, Horesh Ben Shitrit, Mirko Raca, Julien Pettré, Daniel Thalmann, Pascal Fua, and Ronan Boulic. Long term real trajectory reuse through region goal satisfaction. In Jan M. Allbeck and Petros Faloutsos, editors, *Motion in Games*, pages 412–423, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[ALHB08] G. Arechavaleta, J. Laumond, H. Hicheur, and A. Berthoz. An optimality principle governing human walking. *IEEE Transactions on Robotics*, 24(1):5–14, Feb 2008.

[Ang04] Dora E. Angelaki. Eyes on target: What neurons must do for the vestibuloocular reflex during linear motion. *Journal of Neurophysiology*, 92(1):20–35, 2004.

[ANMS13] Saad Ali, Ko Nishino, Dinesh Manocha, and Mubarak Shah. Modeling, simulation and visual analysis of crowds: a multidisciplinary perspective. In *Modeling, simulation and visual analysis of crowds*, pages 1–19. Springer, 2013.

[AWTB12] Junghyun Ahn, Nan Wang, Daniel Thalmann, and Ronan Boulic. Within-crowd immersive evaluation of collision avoidance behaviors. In *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '12, pages 231–238, New York, NY, USA, 2012. ACM.

[BB04] Paolo Baerlocher and Ronan Boulic. An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 20(6):402–417, Aug 2004.

[BBPW04] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In Tomás Pajdla and Jiří Matas, editors, *Computer Vision - ECCV 2004*, pages 25–36, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[BC89] A. Bruderlin and T. W. Calvert. Goal-directed, dynamic animation of human walking. *SIGGRAPH Comput. Graph.*, 23(3):233–242, July 1989.

[BPCL06] C. Braillon, C. Pradalier, J. L. Crowley, and C. Laugier. Real-time moving obstacle detection using optical flow models. In *2006 IEEE Intelligent Vehicles Symposium*, pages 466–471, June 2006.

[BTT90] Ronan Boulic, Nadia Magnenat Thalmann, and Daniel Thalmann. A global human walking model with real-time kinematic personification. *The Visual Computer*, 6(6):344–358, Nov 1990.

[CC01] Armel Crétual and François Chaumette. Visual servoing based on image motion. *The International Journal of Robotics Research*, 20(11):857–877, 2001.

[CH06] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics Automation Magazine*, 13(4):82–90, Dec 2006.

[CLC+05] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade. Footstep planning for the honda asimo humanoid. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 629–634, April 2005.

[CLS03] Min Gyu Choi, Jehee Lee, and Sung Yong Shin. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Trans. Graph.*, 22(2):182–203, April 2003.

[CMA03] N. Courty, E. Marchand, and B. Arnaldi. A new application for saliency maps: synthetic vision of autonomous actors. In *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, volume 3, pages III–1065, Sep. 2003.

[DDH13] Dorine C. Duives, Winnie Daamen, and Serge P. Hoogendoorn. State-of-the-art crowd motion simulation models. *Transportation Research Part C: Emerging Technologies*, 37:193 – 209, 2013.

[DMCN+17] T. B. Dutra, R. Marques, J.B. Cavalcante-Neto, C. A. Vidal, and J. Pettré. Gradient-based steering for vision-based crowd simulation algorithms. *Computer Graphics Forum*, 36(2):337–348, 2017.

[EAPL06] Claudia Esteves, Gustavo Arechavaleta, Julien Pettré, and Jean-Paul Laumond. Animation planning for virtual characters cooperation. *ACM Trans. Graph.*, 25(2):319–339, April 2006.

[EB97] Laurent Najman Eric Bouvier, Eyal Cohen. From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation. *Journal of Electronic Imaging*, 6:6 – 6 – 14, 1997.

[Far03] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, pages 363–370, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[FCCMCMT10] Antonio Fernández-Caballero, José Carlos Castillo, Javier Martínez-Cantos, and Rafael Martínez-Tomás. Optical flow or image subtraction in human detection from infrared camera on mobile robot. *Robotics and Autonomous Systems*, 58(12):1273 – 1281, 2010. Intelligent Robotics and Neuroscience.

[FH04] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, Sep 2004.

[FK02] Brett R Fajen and Nam-Gyoon Kim. Perceiving curvilinear heading in the presence of moving objects. *Journal of Experimental Psychology: Human Perception and Performance*, 28(5):1100, 2002.

[FOPV13] A. Faragasso, G. Oriolo, A. Paolillo, and M. Vendittelli. Vision-based corridor navigation for humanoid robots. In *2013 IEEE International Conference on Robotics and Automation*, pages 3190–3195, May 2013.

[FPCV16] M. Ferro, A. Paolillo, A. Cherubini, and M. Vendittelli. Omnidirectional humanoid navigation in cluttered environments based on optical flow information. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 75–80, Nov 2016.

[GBR12] V. Grabe, H. H. Bülthoff, and P. Robuffo Giordano. Robust optical-flow based self-motion estimation for a quadrotor uav. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2153–2159, Oct 2012.

[GBT04] P. Glardon, R. Boulic, and D. Thalmann. Pca-based walking engine using motion capture data. In *Proceedings Computer Graphics International, 2004.*, pages 292–298, June 2004.

[Gib86] J. J. Gibson. The ecological approach to visual perception. *Lawrence Erlbaum Associates Inc*, 1986.

[GLRLR+07] Martin Gérin-Lajoie, Janet L Ronsky, Barbara Loitz-Ramage, Ion Robu, Carol L Richards, and Bradford J McFadyen. Navigational strategies during fast walking: A comparison between trained athletes and non-athletes. *Gait & posture*, 26(4):539–545, 2007.

[GLRM05] Martin Gérin-Lajoie, Carol L Richards, and Bradford J McFadyen. The negotiation of stationary and moving obstructions during walking: anticipatory locomotor adaptations and preservation of personal space. *Motor control*, 9(3):242–269, 2005.

[GSC+06] S. Griffiths, J. Saunders, A. Curtis, B. Barber, T. McLain, and R. Beard. Maximizing miniature aerial vehicles. *IEEE Robotics Automation Magazine*, 13(3):34–43, Sep. 2006.

[HB93] F. Heitz and P. Bouthemy. Multimodal estimation of discontinuous optical flow using markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1217–1232, Dec 1993.

[HFV00] D Helbing, I Farkas, and T Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487—490, September 2000.

[HHMR12] B. Herissé, T. Hamel, R. Mahony, and F. Russotto. Landing a vtol unmanned aerial vehicle on a moving platform using optical flow. *IEEE Transactions on Robotics*, 28(1):77–89, Feb 2012.

[HM95] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, May 1995.

[HPA+07] Halim Hicheur, Quang-Cuong Pham, Gustavo Arechavaleta, Jean-Paul Laumond, and Alain Berthoz. The formation of trajectories during goal-oriented locomotion in humans. i. a stereotyped behaviour. *European Journal of Neuroscience*, 26(8):2376–2390, 2007.

[HSC+05] S. Hrabar, G. S. Sukhatme, P. Corke, K. Usher, and J. Roberts. Combined optic-flow and stereo-based navigation of urban canyons for a

uav. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3309–3316, Aug 2005.

[IMS+17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[JCP+10] Eunjung Ju, Myung Geol Choi, Minji Park, Jehee Lee, Kang Hoon Lee, and Shigeo Takahashi. Morphable crowds. In *ACM SIGGRAPH Asia 2010 Papers*, SIGGRAPH ASIA '10, pages 140:1–140:10, New York, NY, USA, 2010. ACM.

[KHvBO09] Ioannis Karamouzas, Peter Heil, Pascal van Beek, and Mark H. Overmars. A predictive collision avoidance model for pedestrian simulation. In Arjan Egges, Roland Geraerts, and Mark Overmars, editors, *Motion in Games*, pages 41–52, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[KL99] J. J. Kuffner and J. . Latombe. Fast synthetic vision, memory, and learning models for virtual humans. In *Proceedings Computer Animation 1999*, pages 118–127, May 1999.

[KL00] James J. Kuffner and Steven M Lavalle. Rrt-connect: an efficient approach to single-query path planning. *Proceedings - IEEE International Conference on Robotics and Automation*, 2:995–1001, 12 2000.

[KO12] I. Karamouzas and M. Overmars. Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):394–406, March 2012.

[KSG14] Ioannis Karamouzas, Brian Skinner, and Stephen J. Guy. Universal power law governing pedestrian interactions. *Phys. Rev. Lett.*, 113:238701, Dec 2014.

[KSH+12] Mubbasir Kapadia, Shawn Singh, William Hewlett, Glenn Reinman, and Petros Faloutsos. Parallelized egocentric fields for autonomous navigation. *The Visual Computer*, 28(12):1209–1227, Dec 2012.

[KSHF09] Mubbasir Kapadia, Shawn Singh, William Hewlett, and Petros Faloutsos. Egocentric affordance fields in pedestrian steering. In *Proceed-*

ings of the 2009 Symposium on Interactive 3D Graphics and Games, I3D '09, pages 215–223, New York, NY, USA, 2009. ACM.

[KTDVG16] Till Kroeger, Radu Timofte, Dengxin Dai, and Luc Van Gool. Fast optical flow using dense inverse search. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 471–488, Cham, 2016. Springer International Publishing.

[LCL07] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. *Computer Graphics Forum*, 26(3):655–664, 2007.

[LK05] Manfred Lau and James J. Kuffner. Behavior planning for character animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, pages 271–280, New York, NY, USA, 2005. ACM.

[LUC81] B LUCAS. An iterative image registratio technique with an application to stereo vision. In *Proceedings of 7th International Joint Conference on Artificial Intelligence, 1981*, pages 674–679, 1981.

[MHT11] Mehdi Moussaïd, Dirk Helbing, and Guy Theraulaz. How simple rules determine pedestrian behavior and crowd disasters. *Proceedings of the National Academy of Sciences*, 108(17):6884–6888, 2011.

[NA89] R. C. Nelson and J. Aloimonos. Obstacle avoidance using flow field divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1102–1106, Oct 1989.

[NGCL09] Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C. Lin. Aggregate dynamics for dense crowd simulation. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 122:1–122:8, New York, NY, USA, 2009. ACM.

[NRTT95] Hansrudi Noser, Olivier Renault, Daniel Thalmann, and Nadia Magnenat Thalmann. Navigation for digital actors based on synthetic vision, memory, and learning. *Computers & Graphics*, 19(1):7 – 19, 1995. Visual Computing.

[OB94] J. M. Odobez and P. Bouthemy. Detection of multiple moving objects using multiscale mrf with camera motion compensation. In *Proceedings of 1st International Conference on Image Processing*, volume 2, pages 257–261 vol.2, Nov 1994.

[OB98] Jean-Marc Odobez and Patrick Bouthemy. Direct incremental model-based image motion segmentation for video analysis. *Signal Processing*, 66(2):143 – 155, 1998.

[OPOD10] Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph.*, 29(4):123:1–123:9, July 2010.

[PAB07] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 99–108, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[POO+09] Julien Pettré, Jan Ondřej, Anne-Hélène Olivier, Armel Cretual, and Stéphane Donikian. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 189–198, New York, NY, USA, 2009. ACM.

[PPD07] Sébastien Paris, Julien Pettré, and Stéphane Donikian. Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum*, 26(3):665–674, 2007.

[PTI04] Aftab E Patla, Sebastian S Tomescu, and Milad GA Ishac. What visual information is used for navigation around obstacles in a cluttered environment? *Canadian journal of physiology and pharmacology*, 82(8-9):682–692, 2004.

[PvC+11] S. Patil, J. van den Berg, S. Curtis, M. C. Lin, and D. Manocha. Directing crowd simulations using navigation fields. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):244–254, Feb 2011.

[Rey87] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA, 1987. ACM.

[Rey99] C. Reynolds. Steering behaviors for autonomous characters. *Game Developers Conference*, pages 763–782, 1999.

[RH96] Constance S. Royden and Ellen C. Hildreth. Human heading judgments in the presence of moving objects. *Perception & Psychophysics*, 58(6):836–856, Jan 1996.

[RN13] Florian Raudies and Heiko Neumann. Modeling heading and path perception from optic flow in the case of independently moving objects. *Frontiers in Behavioral Neuroscience*, 7:23, 2013.

[RTT90] Olivier Renault, Nadia Magnenat Thalmann, and Daniel Thalmann. A vision-based approach to behavioural animation. *The Journal of Visualization and Computer Animation*, 1(1):18–21, 1990.

[SBC96] V. Sundareswaran, P. Bouthemy, and F. Chaumette. Exploiting image motion for active vision in a visual servoing framework. *Int. J. Rob. Res.*, 15(6):629–645, December 1996.

[SH07] Alla Safonova and Jessica K. Hodgins. Construction and optimal search of interpolated motion graphs. In *ACM SIGGRAPH 2007 Papers*, New York, NY, USA, 2007. ACM.

[SJ04] Sooyong Lee and Jae-Bok Song. Robust mobile robot localization using optical flow sensors and encoders. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 1, pages 1039–1044 Vol.1, April 2004.

[SK07] Kahlouche Souhila and Achour Karim. Optical flow based robot obstacle avoidance. *International Journal of Advanced Robotic Systems*, 4(1):2, 2007.

[SKH+11] Shawn Singh, Mubbasir Kapadia, Billy Hewlett, Glenn Reinman, and Petros Faloutsos. A modular framework for adaptive agent-based steering. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 141–150 PAGE@9, New York, NY, USA, 2011. ACM.

[SLC10] Alessandro Ribeiro Da Silva, Wallace Santos Lages, and Luiz Chaimowicz. Boids that see: Using self-occlusion for simulating large groups on gpus. *Comput. Entertain.*, 7(4):51:1–51:20, January 2010.

[ST05] Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, pages 19–28, New York, NY, USA, 2005. ACM.

[TCP06] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 1160–1168, New York, NY, USA, 2006. ACM.

[TGS91] M. Tistarelli, E. Grosso, and G. Sandini. Dynamic stereo in visual navigation. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 186–193, Jun 1991.

[TR95] D. Terzopoulos and T. F. Rabie. Animat vision: Active vision in artificial animals. In *Proceedings of IEEE International Conference on Computer Vision*, pages 801–808, June 1995.

[TT94] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 43–50, New York, NY, USA, 1994. ACM.

[vdBGLM11] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger, editors, *Robotics Research*, pages 3–19, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[vdBLM08] J. van den Berg, Ming Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE Int. Conf. on Robotics and Automation*, pages 1928–1935, May 2008.

[WJDL13] Qianqian Wu, Qingge Ji, Jinghong Du, and Xiaolian Li. Simulating the local behavior of small pedestrian groups using synthetic-vision based steering approach. In *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '13, pages 41–50, New York, NY, USA, 2013. ACM.

[WJS95] William H Warren Jr and Jeffrey A Saunders. Perceiving heading in the presence of moving objects. *Perception*, 24(3):315–331, 1995.

[WKZ+01] WH Warren, BA Kay, WD Zosh, AP Duchon, and S Sahuc. Optic flow is used to control human walking. *Nature neuroscience*, 4(2):213—216, February 2001.

[WMBM91] William H Warren, Daniel R Mestre, Arshavir W Blackwell, and Michael W Morris. Perception of circular heading from optical flow.

Journal of Experimental Psychology: Human Perception and Performance, 17(1):28, 1991.

[WRHS13] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[ZPB07] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In Fred A. Hamprecht, Christoph Schnörr, and Bernd Jähne, editors, *Pattern Recognition*, pages 214–223, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[ZQS+17] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. Icnet for real-time semantic segmentation on high-resolution images. *arXiv preprint arXiv:1704.08545*, 2017.

[ZSWS10] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart. Mav navigation through indoor corridors using optical flow. In *2010 IEEE International Conference on Robotics and Automation*, pages 3361–3368, May 2010.

UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Algorithmes de navigation basés sur des flux optiques pour les personnages virtuels

Mot clés : Animation, pilotage de personnage, flux optique

Résumé : La simulation de vrais humains dans un environnement virtuel est un problème ouvert avec de nombreuses applications, dès l'industrie du divertissement à la sécurité dans la planification architecturale. Les humains constituent un système visuel complexe capable de naviguer dans des environnements très dynamiques.

Cette thèse tente de reproduire le comportement humain en imitant le système visuel humain. Nous avons conçu un modèle d'agent capable de percevoir des informations visuelles et d'en définir un modèle de contrôle. En particulier, nous nous concentrons sur l'utilisation du flux optique c'est-à-dire le mouvement apparent des objets.

Deux boucles de contrôle sont proposées, la première fonctionne avec un flux optique synthétique. C'est une approximation disponible dans les simulations virtuelles qui permet à nos agents de percevoir cette flux avec une grande précision. Le deuxième boucle de contrôle calcule le flux optique à partir d'une séquence d'images couleur. Cela reproduit mieux le système de perception humaine mais le flux résultant perd en précision. Une nouvelle caractéristique du dernièr boucle de régulation est la possibilité de réagir à différentes conditions d'éclairage.

Enfin, nous évaluons notre modèle en testant nos agents dans de nombreux scénarios avec des environnements statiques et dynamiques.

Title: Optical flow-based navigation algorithms for virtual characters

Keywords: Animation, Character steering, optical flow

Abstract: Simulation of real humans in virtual environment is an open problem with many applications, from the entertaining industry to safety in architectural planning. Humans poses a complex visual system capable of navigation in highly dynamic environments.

This thesis attempts to reproduce human behavior by imitating the human visual system. We designed an agent model capable of perceiving visual information and define a control model from it. In particular, we focus on the usage of optical flow, in other words, the apparent motion of objects.

Two control loops are proposed, the first one operates with synthetic optical flow. It is an approximation available in virtual simulations that allows our agents to perceive this map with high precision. The second control loop computes optical flow from a sequence of color images using a numerical solver. This reproduces better the human perception system but the resulting flow loses accuracy. A novel characteristic of the latest control loop is the possibility of reacting to different lighting conditions.

Finally, we evaluate our model by testing our agents in many scenarios with static and dynamic environments.