

Coupling camera-tracked humans with a simulated virtual crowd

Jorge Ivan Rivalcoba Rivas¹, Oriam De Gyves¹, Isaac Rudomín² and Nuria Pelechano³

¹Tecnológico de Monterrey, México

²Barcelona Supercomputing Center, Spain

³Universitat Politècnica de Catalunya, Spain

{ivan.rivalcoba, odegives, rudomin.isaac}@gmail.com, npelechano@lsi.upc.edu

Keywords: Crowd simulation, Computer vision, Semantic Local Binary Patterns (S-LBP), Computer graphics, Motion control

Abstract: Our objective with this paper is to show how we can couple a group of real people and a simulated crowd of virtual humans. We attach group behaviors to the simulated humans to get a plausible reaction to real people. We use a two stage system: in the first stage, a group of people are segmented from a live video, then a human detector algorithm extracts the positions of the people in the video, which are finally used to feed the second stage, the simulation system. The positions obtained by this process allow the second module to render the real humans as avatars in the scene, while the behavior of additional virtual humans is determined by using a simulation based on a social forces model. Developing the method required three specific contributions: a GPU implementation of the codebook algorithm that includes an auxiliary codebook to improve the background subtraction against illumination changes; the use of semantic local binary patterns as a human descriptor; the parallelization of a social forces model, in which we solve a case of agents merging with each other. The experimental results show how a large virtual crowd reacts to over a dozen humans in a real environment.

1 INTRODUCTION

Real-time crowd simulation is a research area that is growing rapidly and has become one of the main research directions in computer games, movies and virtual reality (Wang et al., 2012). Applications include urban planning, security and entertainment, among others (De Gyves et al., 2013). It has been studied extensively but only in a few cases the computer generated crowds react to real humans. Most research is focused on the behavior of virtual humans, that may be data driven, rule based or socially inspired. The end result is almost every time a crowd interacting only with its virtual environment. In the human computer interaction literature there is some research on the interaction between virtual humans and a specific kind of real humans: end-users, but most are limited to one or two users and a small number of virtual humans.

A simulation system which includes some real humans (pedestrians, for example) in crowds of agents, that react to the real humans would have several potential applications in special-effects for films and video games. One goal of this research is to visual-

ize an authoring tool that would use a camera looking down on a group of actors to reduce the time of animation production in the context of crowd generation. One could also conceive of games where several real players lead teams of virtual team-mates. Other applications would allow researchers to evaluate their behavior algorithms by directly comparing them with real human behaviors.

The aim of the present work is to perform the coupling between a group of real people and a simulated crowd of virtual humans. To achieve this, we propose a two-stage interactive system. In the first stage, named the vision system, pedestrians are segmented and tracked from a live video. The positions of the captured pedestrians are used in the second module, the simulation system. The simulation system is then responsible for the reaction of the virtual crowd to the tracked real humans.

2 RELATED WORK

A crucial part of making the system presented in this article suitable for interaction is the time con-

sumed by the recognition, simulation and rendering tasks. To get a deeper understanding of the system, this section describes the challenges and the state of the art related to computer vision and crowd simulation.

2.1 Human detection

As mentioned above, human detection is the first step of the system. This is a challenging problem due to several factors, which are listed below:

- Illumination variance.
- Non uniform colors.
- Different scales of object of interest.
- Cluttered backgrounds.
- Noise due to the small movements in the video sequence.

There has been a large amount of research using vision for human detection. Viola et al. described a machine learning approach for visual object recognition and also introduced the “*integral images*”, a popular technique to accelerate the calculus of many descriptors (Viola and Jones, 2001), this algorithm was used to detect faces in static images by combining complex classifiers in a cascade, which allowed background regions of the image to be quickly discarded while spending more computational time on regions with a high probability of being the interest objects. The same authors adapted the technique to detect pedestrians in a video sequence by taking advantage of the motion appearance of a walking person and using *AdaBoost* for the training process and constructing of the classifier (Viola et al., 2003). Dalal et al. performed a complete study of *Histogram of Oriented Gradients* (HOG) applied to the representation of humans (Dalal and Triggs, 2005), this method offers good results for pedestrian detection by evaluating local histograms of image gradient orientations over a dense normalized overlapping grid, avoiding the common problem of illumination variance. Tuzel et al. detected pedestrians by representing an image region as covariance matrices of spatial locations, intensities and derivatives, to name a few (Tuzel et al., 2007). This work follows the trend of using logic boost classifiers: the authors reported that their results were better than the work by Dalal when using the INRIA database.

In most cases the detection systems include a pre-processing phase, which consists of the background removal of the scene. This is the case of the work presented by Banerjee et al., which combined an adaptive background modeling with HOG features (Banerjee

and Sengupta, 2008). They proposed adapts itself to long lasting changes in the background over time by learning only the stationary parts of the scene and ignoring the moving foreground. The moving regions are then considered as a *Region of Interest* (ROI) and the HOG is applied to classify whether such region is a human or not. Another approach was presented by Bhuvaneswari et al. called edgelet features, which are short segments belonging to a line or a curve (Bhuvaneswari and Rauf, 2009). The detection was performed with a module that was composed of several classifiers arranged in cascade; the authors reported a detection rate up to 70%. The aforementioned works dealt with side perspective scenes, which is desirable for surveillance applications, however they are not recommended when dealing with crowd analysis, due to the occlusion presented when the crowd size increases, causing the accuracy of the recognition systems to decrease.

Lengvenis et al. presented the passenger counter system created and installed in the Kaunas public city transport (Lengvenis et al., 2013); unlike other work based on edge features, this paper uses the vertical and horizontal projection of the scene using a top-view perspective, also known as bird’s eye view. The results reported an accuracy of 86% for a single passenger getting on or off the bus, however it could not detect people passing each other or getting on a bus together and presented problems with lighting changes. Ozturk et al. proposed a system to determine the head and body orientation of humans in top-view scenes (Ozturk et al., 2009), but it has the same problems that the work by Lengvenis. Their system used a shape context based approach to detect basic body orientation and proposed an optical flow based on *Scaled Invariant Feature Transform* (SIFT) features, however the occlusions were left for future work, making this algorithm unsuitable for crowded scenes.

We decided to use a bird’s eye perspective in order to minimize the occlusion presented in crowded scenes. Another important issue to deal with is having a good background segmentation process to focus the computational effort in the classification tasks, as well as to determine a good representation model to recognize humans from aerial footage, taking into account the deformation present when a human moves away from the camera.

2.2 Simulation

There is extensive research that focuses on modeling accurate behaviors for virtual agents. Researchers have studied crowd simulations in both microscopic and macroscopic levels; macroscopic simulations are

concerned about the realistic movement of the crowd as a mass while microscopic simulations focus on the realistic movement of the individuals. One of the first research works focused on autonomous virtual agents and their behaviors are the steering behaviors by (Reynolds, 1987). According to Reynolds' research, a bird is aware of three aspects during flocking, itself, two or three nearest neighbors, and the rest of the flock.

Helbing et al. presented the social forces model, where each agent tries to move at a desired velocity while applying friction and repulsion forces against obstacles and other agents (Helbing and Molnár, 1995). According to the authors, social forces are well suited for pedestrians in normal, or known, situations. Helbing extended his research and studied the behavior of pedestrians in normal and evacuation situations (Helbing et al., 2002). A *nervousness* parameter causes pedestrians to move faster, while pushing other individuals, and exhibit a herding behavior during evacuations. The social forces model has been extended to support groups of related pedestrians (Moussaïd et al., 2010).

Treuille et al. presented the Continuum Crowds model, which unifies the path planning and collision evasion stages by using velocity and potential fields (Treuille et al., 2006). This model can simulate thousands of virtual agents in interactive applications, however, pedestrians do not have individual goals. Millan et al. used attraction and repulsion forces codified in textures to steer the agents through the environment (Millan et al., 2006). In this approach every individual agent has its own set of goals depending on which agents were nearby, but in some cases the forces may cancel each other, leaving the agents *stuck* in the environment.

Fiorini and Shiller introduced the term *Velocity Obstacles* (VO), a set of velocities that will lead into collision between agents (Fiorini and Shiller, 1998). Agents must choose velocities that are not elements of the VO set to move in collision-free routes. The GAMMA group extended this idea into the *Reciprocal Velocity Obstacles* (RVO) (Van den Berg and Manocha, 2008). The RVO assumes that every agent in the simulation is steered using the same rules, which eliminates the oscillations in the paths of the VOs. Bleiweiss parallelized the RVO algorithm and implemented it on CUDA, obtaining a 4.8x speedup over the original implementation (Bleiweiss, 2009). Van den Berg et al. further extended RVO into *Reciprocal n-body Collision Avoidance* (Van den Berg et al., 2011). The algorithm reduced the problem to solving a low-dimensional linear program. By using this algorithm, the authors were able to simulate thou-

sands of pedestrians in real-time.

A vision-based collision avoidance approach was presented by Ondřej et al. (Ondřej et al., 2010). The authors use a synthetic vision approach to detect obstacles in the field of view of every agent and *time-to-interaction* values to steer the agents.

In our work we have modeled the behavior of virtual pedestrians using the algorithm presented by (Moussaïd et al., 2010); this model incorporates a group force that makes it possible to simulate group formation. In his paper the size of the groups is determined statistically according to observations from video sequences.

2.3 Virtual Environments and Interaction

Tsai-Yen et al. presented a computer simulation crowded by virtual users and virtual humans (Li et al., 2002). Each group of virtual humans is led by an intelligent group leader and a world manager can interactively assign a goal configuration to a group leader through a graphical user interface at the virtual environment server. Thalmann et al. presented an interactive navigation system in which a user is able to control an avatar in the crowd through a natural interface, but the interaction remains between a single user and the virtual characters (Wang et al., 2012). Pelechano proposed an experiment to closely study the behavior of people interacting with a virtual crowd (Pelechano and Stocker, 2008). A virtual scenario is created to simulate a cocktail party. The main goal of the experiment was to examine whether participants interacting with a virtual crowd would react to the virtual crowd as they would do in a similar real situation.

2.4 Computer Vision and Graphics for Crowd Simulation

There is some research that has merged computer vision with computer graphics in the context of crowd simulation is data driven models. In this case the motion of real people is used to model virtual crowds, for example Musse (Musse et al., 2007) simulated the movement of virtual humans based on trajectories captured from filmed video sequences, these trajectories are grouped into similar classes using an unsupervised clustering algorithm, Lee (Lee et al., 2007) used aerial footage in closed environments for a similar purpose. Lerner (Lerner, 2007) created a model for collision avoidance by learning from real-world examples generated from tracked video segments of real pedestrian crowds. Sun and Qin (Sun and Qin, 2011)

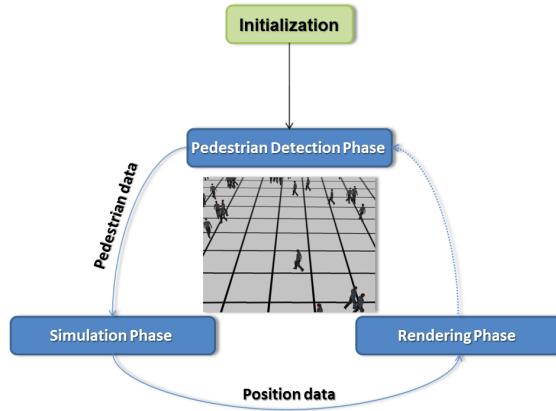


Figure 1: System overview. The human's data captured by the detection phase is used in the simulation phase. The simulated crowd is able to react to the real humans.

presented a model for solving collisions between humans, making use of motion capture data that allow a more natural and realistic motion of pedestrians compared with RVO model. Recently Ren (Ren et al., 2013) performed an integration of virtual pedestrians into a video of a scene consisting of real pedestrians. In that paper, to avoid segmentation issues due to occlusion, groups of pedestrians are taken as a unit instead of segmenting individuals.

3 SYSTEM ARCHITECTURE

The system we describe in this article is designed to be interactive even when using consumer grade hardware, while also being capable of capturing humans from video footage and simulating thousands of virtual agents. A diagram showing the application flow can be seen in Figure 1.

3.1 Vision system

The use of markers to analyze a crowd has been used successfully by (Wang and Sun, 2008): he employs colored hats to aid the detection system to extract information regarding to pedestrian dynamics. The system proposed in the present paper aims to perform a reliable non intrusive detection and tracking, thus the use of markers is not an option. In the same way, the computer vision algorithms must process the image in real time to be able to provide the positions of real pedestrians as an input to the simulation stage.

The system that performs the human detection consists of 5 stages (Figure 2). The first stage is the background subtraction which is responsible splitting the scene into background and foreground regions.

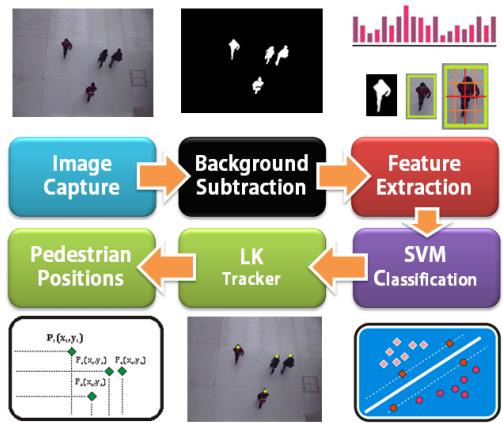


Figure 2: The vision stages that produce the set of positions that will be sent to the simulation system.



Figure 3: Head-shoulder part of a human that exhibits their omega-like shape.

Foreground regions are labeled as *Regions of interest* (ROI), and this labeling allows the system to concentrate the computational efforts in the human detection tasks. Once the set of ROI have been obtained, in the second stage ROIs are passed one by one to the feature extraction module which generates a vector of features that models the object detected. Each one of these vectors is then sent to a *support vector machine* (SVM) that performs the classification into human or not human. We decided to detect the head and shoulders section of the body due to the fact that the human head remains constant over all the scene (Mukherjee and Das, 2013), unlike other parts of the body. The head and shoulders section exhibit an Ω like shape (see Figure 3). This property makes the head and shoulders the most stable parts of the body to be detected and tracked.

In the rest of this section we explain each stage in more detail .

3.1.1 Stage 1: Background Subtraction

To perform background subtraction we propose a modified version of the codebook algorithm presented by (Kim et al., 2005). The codebook algorithm adopts a clustering and quantization technique to construct a background model from a long observation of sequences of pixel intensities. Each pixel owns its code-

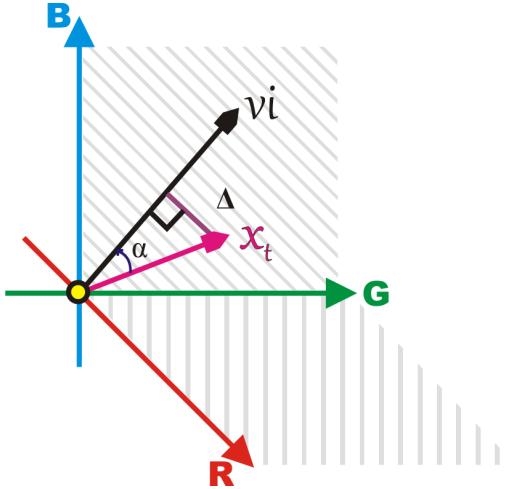


Figure 4: This figure shows the model used to calculate the color distortion Δ between two pixels x_t and v_i .

book, that is made of codewords produced on the training stage; these codewords are constructed based on two parameters:

- Brightness
- Color distortion

The brightness of a given pixel x at a certain time t , expressed as \vec{x}_t could be defined for this purpose as the norm of the vector in the **RGB** color space described by (1).

$$\|\vec{x}_t\|^2 = R^2 + G^2 + B^2 \quad (1)$$

The color distortion between two pixels, \vec{x}_t and \vec{v}_i , is the measure of the opposite side of the angle formed by the two color vectors, as seen in Figure 4. This property is described by the equation (2) and (3):

$$colordist(\vec{x}_t, \vec{v}_i) = \Delta = \|\vec{x}_t\|^2 - \|\vec{x}_t\|^2 \cos^2 \alpha \quad (2)$$

$$\|\vec{x}_t\|^2 \cos^2 \alpha = \frac{\langle \vec{x}_t, \vec{v}_i \rangle^2}{\|\vec{v}_i\|^2} \quad (3)$$

Each pixel owns its codebook, and each codebook is made of codewords. A codeword could be seen as a box that stores information about the pixel intensities; every codeword is generated through the observation of each pixel during a period of time, called training stage. The training may take one or two minutes, during which a new codeword is collected. If there is no codeword, already in the codebook, that meets the similarity measures described above, the codeword is stored in the codebook. Otherwise, the training algorithm counts every codeword match. At the end of the training process, codewords with few occurrences are discarded from the codebook.

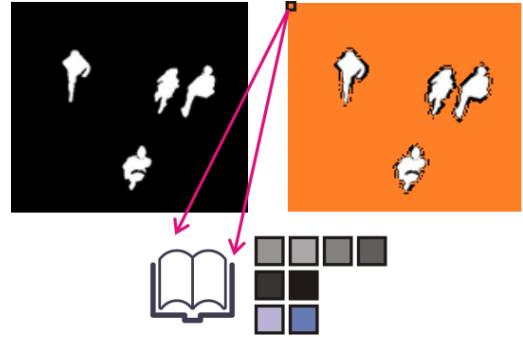


Figure 5: To maintain the background model and make it robust against intensity changes, a second code book is used and maintained by those regions with no movement. This image shows, in orange, all the regions that feed the auxiliary codebook, represented by the icon below the images.

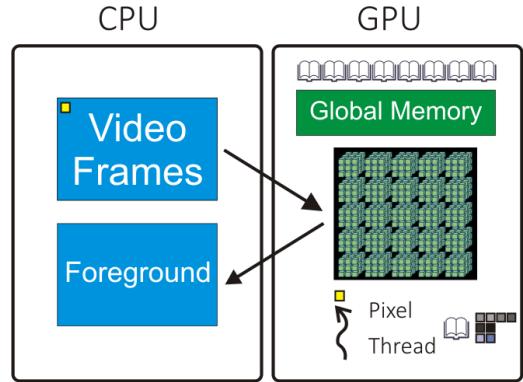


Figure 6: Once the image is captured, it is sent to the GPU to be processed and then returned to the CPU.

To make the algorithm more robust against illumination changes, the modification we propose consists of maintaining an auxiliary codebook that generates codewords only for regions with no movement. Once a certain time has elapsed, this codebook is cleaned of spurious data, and then is added to the main codebook. This allows the background model to be updated regularly, see Figure 5.

To perform this task within an acceptable frame rate, we implemented a parallel version of the codebook algorithm, using the graphics hardware. Each thread works over one pixel, as seen in Figure 6. Once a frame is captured, the image is processed on the GPU using the previous description.

3.1.2 Stage 2 & 3: ROI detection and feature extraction

Once the background has been removed, the next step is to extract the blobs from the foreground regions. These blobs are rectangular regions that contain the humans, and are converted from RGB color space to gray scale.

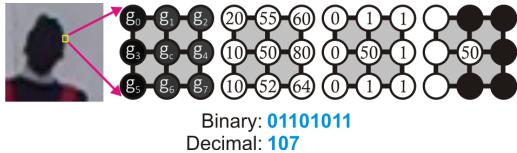


Figure 7: An illustration of the basic LBP operator.

To model a human we use a variant of *local binary patterns* (LBP) presented by (Ojala, 2002). The LBP operator assigns a label to every pixel of an image by thresholding the 3×3 neighborhood of each pixel with its own value and considering the result as a binary number. E.g., given a pixel (x_c, y_c) , the LBP of that pixel is calculated using equation 4, where P corresponds to the number of sampling points, R is the radius of the neighborhood to be sampled, g_p represents each sampling point and g_c represents the pixel (x_c, y_c) ; the histogram of the numbers obtained along all the image is used as the descriptor. Figure 7 shows an illustration of the basic LPB.

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p \quad (4)$$

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

Although LBPs have many advantages over other gradient-based features, the basic LBP is not robust enough to be used as human descriptor; however its strengths can be exploited using a variant called *semantic LBP* (S-LBP) proposed by (Huang, 2008). S-LBPs are made of uniform LBPs. An LBP is called uniform if only at most two bitwise transitions between 0 and 1 are presented, in other words several continuous “1” bits form an arc on the sampling circle; these arcs can be represented with its principle direction and arc length (see Figure 9); by using this strategy, the dimension of the feature vector is reduced. In our work we use a detection kernel of 32×32 pixels size which produces a 1160D vector; all the S-LBPs are used to compute an histogram, each histogram represents the feature vector of each region. To speed up the computation of S-LBPs, we propose the use of a look up table to classify the type of S-LBP.

As it has been demonstrated by (Tuzel et al., 2007) the ensemble of variable-size sub-windows can greatly promote detection efficiency. For this reason, we adopted a multi resolution approach similar to (Ahonen, 2006). We divided each detection kernel into local regions, then the S-LBPs descriptors are extracted for each region independently. The vector “ v ” obtained for each sub-window is normalized using the following equation:

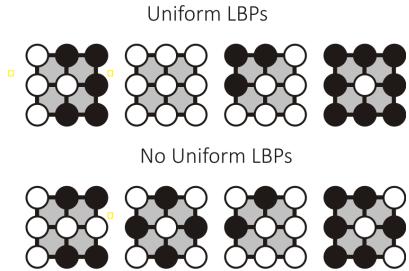


Figure 8: Uniform and non uniform LBPs.

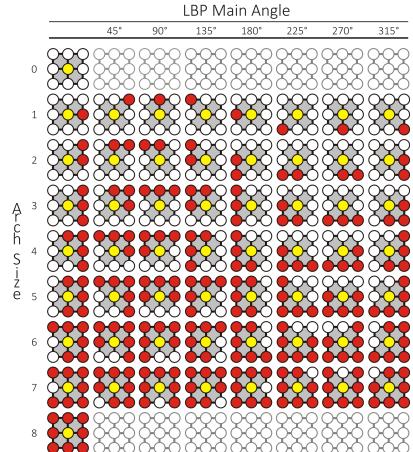


Figure 9: This figure shows the only 58 possible S-LBPs of a pixel.

$$L1 - \text{sqrt} : f = \sqrt{\frac{v}{\|v\|_1 + e}} \quad (6)$$

The descriptors are then concatenated to form a global descriptor vector, see Figure 10. As previously mentioned, a 1160D vector is generated using a detection kernel of 32×32 pixels, these vectors feed an SVM classifier that decide if the image of the kernel belongs to a person or not.

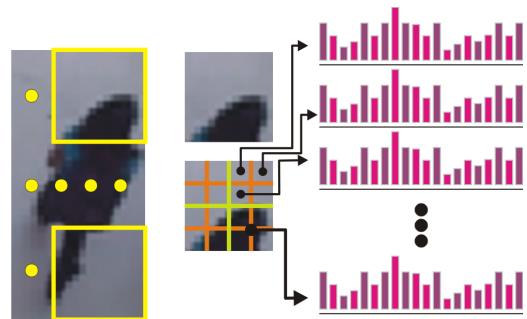


Figure 10: This figure shows the subdivision made to the detection kernel to compute the histogram.

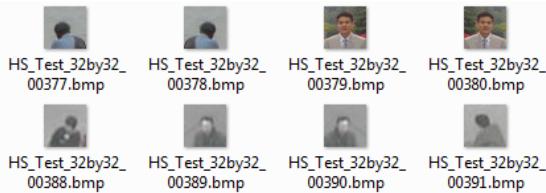


Figure 11: These are the type of images used to train the head and shoulder detector.

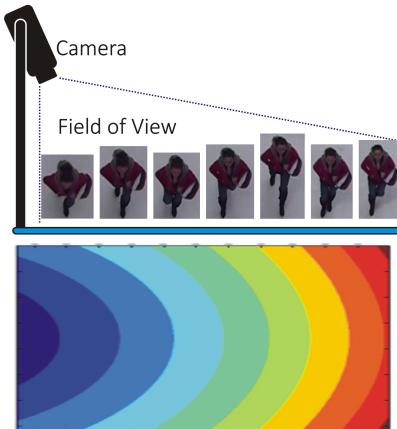


Figure 12: Segmentation of the scene in function of the distance between the camera and the human.

3.1.3 Stage 4: SVM classification

We use the OpenCV SVM library to train and classify the head and shoulders part of the human body, using the same dataset employed by (Li et al., 2009). This dataset uses top view shots of humans' heads and shoulders, as seen in Figure 11.

One important observation is that the pedestrians exhibit a deformation due to the camera perspective, making difficult to tune up the classifier (Rivalcoba and Rudomin, 2013). To overcome this issue, and improve the detection results, we divide the camera view on different segments in function by the distance between the camera and the human. Figure 12 shows the depth map and the deformation suffered by a group of humans as they move along the scene.

3.1.4 Stage 5: Lucas Kanade Tracker

Once a human head is detected, a point is associated with it. Each detected point is tracked over every frame using the pyramidal version of the Lucas Kanade tracker implemented on the OpenCV library. All the points detected are then stored into a texture to be used in the simulation stage; this step is detailed in the next section.

3.2 Simulation system

OpenGL 4.3 introduced *Compute Shaders* to its API, which enable its shading language to perform general computing in the graphics hardware. Compute Shaders provide an alternative approach to CUDA from Nvidia and OpenCL from the Khronos Group for General-Purpose Computing on Graphics Processing Units (GPGPU). Unlike CUDA and OpenCL, Compute Shaders are similar to other shader stages in the graphics pipeline and are able to read from textures, images and buffers and write to images and buffers without mapping to other data structures or switching between graphics and compute contexts. We have implemented several applications with graphics interoperability, including the technique presented in this Section, in both CUDA and Compute Shaders and the features of the latter allowed significantly faster development while maintaining a similar performance.

For our simulation system, we use Compute Shaders in order to simulate the agents and use a method based on social forces to steer the agents around the environment. All the logic related to agent control is performed in the graphics hardware, which allows us to perform large scale simulations with thousands of agents in real-time.

For the interoperability between rendering and simulation, we store all the data required for the simulation in texture memory. Data captured by the vision system (Section 3.1) is stored in the same textures that the simulated agents use.

To steer the simulated agents we developed a parallel implementation of the work by Moussaid et al. (Moussaïd et al., 2010). Three data textures are required for the simulation: position, velocity and target textures. Each texture element (*texel*) i holds the data of the agent or pedestrian i . To update the positions of the simulated agents, we use equations (7), (8) and (9).

$$\vec{f}_i = \vec{f}_i^d + \vec{f}_i^n + \vec{f}_i^o + \vec{f}_i^g \quad (7)$$

$$\vec{v}_{inew} = \vec{v}_i + \vec{f}_i \cdot \Delta t \quad (8)$$

$$\vec{p}_{inew} = \vec{p}_i + \vec{v}_{inew} \cdot \Delta t + \frac{\vec{f}_i \cdot \Delta t^2}{2} \quad (9)$$

Where \vec{f}_i represents the acceleration of the agent i in the current frame, and it is affected by its internal motivations \vec{f}_i^d , its neighbors \vec{f}_i^n , the nearest obstacles \vec{f}_i^o and its group dynamic \vec{f}_i^g . The new velocity \vec{v}_{inew} is obtained by adding the acceleration to the previous velocity \vec{v}_i of the agent. Finally, the position of the

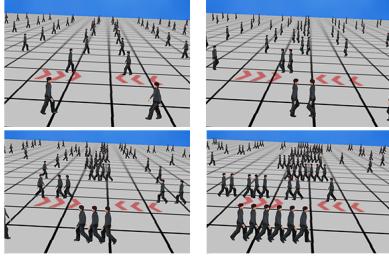


Figure 13: The original tangential velocity difference may be zero when agents face each other with the same velocity, leaving the agents stuck in a line.

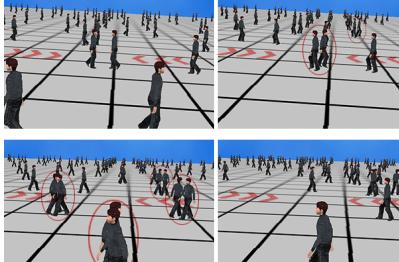


Figure 14: Bidirectional test performed with the modified tangential velocity difference.

agent is updated with both its acceleration and its velocity. In equations 8 and 9, Δt represents the time between two consecutive frames.

In his original research, Helbing describes a *body force* and a *sliding force* between agents for counteracting body compression and impeding relative tangential motion, respectively (Helbing et al., 2000). These forces are part of the force between agents, \vec{f}_i^n . Through experimentation, we found that the tangential velocity difference, $\Delta\vec{v}_{ji} = (\vec{v}_j - \vec{v}_i) \cdot t_{ij}$, used in the sliding force is $\Delta\vec{v}_{ji} = 0$ if the agents are facing each other in opposite directions with the same velocity. This leads to the agents getting stuck in tests like the bidirectional movement, as can be seen in Figure 13.

In case $\Delta\vec{v}_{ji} = \vec{v}_j - \vec{v}_i \leq \epsilon$ then we assign $\Delta\vec{v}_{ji} = \nu$, where ν represents a vector constant that satisfies $\|\nu\| \neq 0$. The magnitude of ν is proportional to the sliding force, and a greater sliding force makes the agents evade each other faster. A bidirectional test performed using this modification to the tangential velocity difference can be seen in Figure 14.

We detect neighbors based on their positions using a spatial hash-like algorithm. This is an image based technique that samples an area in an environment texture where each agent marks its position. The algorithm searches for the nearest neighbors using this environment texture; since both the simulated agents and the tracked humans store their positions in this texture, the simulated agents are aware of the tracked

humans, just as if they were other agents. By being aware of the position of the real humans, the agents are able to evade them.

4 RESULTS

We have implemented the techniques previously described using C++ for the CPU code, and OpenGL for the simulation and visualization, on a Windows 7 x64 operating system with an Intel i7-2630QM quad core processor at 2.00 GHz, 8GB of memory and an NVIDIA GT 540M graphics card. The GT 540M is a Fermi-family graphics card and is OpenGL 4.3 capable; it is considered a low to mid-range GPU. For the rendering and simulation systems we have only used OpenGL, specifically a 4.3 core profile context.

Results show that this system can perform a real-time coupling between a group of tracked humans and a computer simulated group of virtual characters; real pedestrians are rendered as red avatars while simulated pedestrian are rendered in gray color (Figure 15), in this case, we have tuned the group social forces to produce formations of up to 3 people, see Figure 16. The use of an auxiliary codebook for background modeling made the system capable to correctly segment the scene in the presence of illumination changes. The combination of semantic LBPs and the subdivision of the scene into segments allow handling the deformations suffered by the humans as they move away from the camera.

The usage of special cameras was not necessary, any standard web camera fully meets the minimum technical specifications to make the system reliable. The dimension of the processed images were 640 by 480 pixels. The top view perspective -also called bird's eye view- was used to capture as many pedestrians as possible, minimizing the probability of occurrence of occlusions. In our experiments, we were able to capture over a dozen pedestrians while keeping interactive frame rates.

We achieved a frame rate above 60 frames per second, enough to be considered a real-time system. Using the system, we were able to simulate thousands of autonomous agents. The use of the social forces model on the simulation side, produces a plausible effect of integration between the two worlds.



Figure 15: Left: Several tracked real humans (red) are coupled with a large virtual crowd (gray). Right: The captured image.



Figure 16: This image shows the formations produced using group social forces, the group formations are up to 3 people.

5 CONCLUSIONS AND FUTURE WORK

The computer interaction powered by camera technology is becoming more popular over the years. In our case, there are several challenges to overcome, such as background subtraction against illumination changes, human detection, tracking groups in crowded scenes, and fitting behaviors on virtual characters, to name a few.

Taking into account the above, we have presented a system that allows a large virtual crowd to react to real humans by tracking them on video using only consumer grade hardware. This system is capable of detecting and tracking over a dozen humans in a scene, and simulating thousands of virtual agents. It allows coupling between the tracked humans and the simulated agents while maintaining real-time frame rates.

The system presented in this paper could lead to a system that could test new behavior algorithms for virtual agents, whether they are data driven or rule based, because it allows a direct visual comparison with real humans. It may also allow more accurate urban planning and security simulations since some part of the population in the crowd is constituted of real people. In the entertainment industry, this system provides a contribution to the workspace of applications that seek the coupling of the real world and a virtual world.

We are working in making both systems, vision and simulation, totally asynchronous from each other. When the vision data is not ready to be processed by the simulation system, the captured pedestrians can be simulated using the same behaviors of the agents. Allowing the systems to work independently has two main advantages, the first one is that each system can be executed in a different platform -this would allow the vision system to be implemented in an embedded device such as a security camera while the simulation system is running in a remote computer. The second advantage of this setup is the ability to detect errors in the behaviors used in the simulation mode when the vision system updates the data of the tracked humans. Having a direct visual feedback between errors in simulated and real data means a better benchmark for a new simulation behavior.

REFERENCES

- Ahonen, T. (2006). Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence*, 28(12):2037–41.
- Banerjee, P. and Sengupta, S. (2008). Human motion detection and tracking for video surveillance. *Proceedings of the national Conference of tracking and video surveillance activity analysis*, pages 88–92.
- Bhuvaneswari, K. and Rauf, H. A. (2009). Edgelet based human detection and tracking by combined segmentation and soft decision. *Control, Automation, Communication and Energy Conservation*, (June):4–9.
- Bleiweiss, A. (2009). Multi agent navigation on GPU. *White paper, GDC*.
- Dalal, N. and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1:886–893.
- De Gyves, O., Toledo, L., and Rudomín, I. (2013). Comportamientos en simulación de multitudes : revisión del estado del arte. *Research in Computer Science.*, 62(Special Issue: Avances en Inteligencia Artificial.):319–334.

- Fiorini, P. and Shiller, Z. (1998). Motion Planning in Dynamic Environments Using Velocity Obstacles. *The International Journal of Robotics Research*, 17(7):760–772.
- Helbing, D., Farkas, I., and Vicsek, T. (2000). Simulating dynamical features of escape panic. *Nature*, 407(6803):487–90.
- Helbing, D., Farkas, I. J., Molnár, P., and Vicsek, T. (2002). Simulation of Pedestrian Crowds in Normal and Evacuation Situations. *Pedestrian and evacuation dynamics*, 21.
- Helbing, D. and Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286.
- Huang, T. (2008). Discriminative local binary patterns for human detection in personal album. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.
- Kim, K., Chalidabhongse, T. H., Harwood, D., and Davis, L. (2005). Real-time foreground–background segmentation using codebook model. *Real-Time Imaging*, 11(3):172–185.
- Lee, K., Choi, M., Hong, Q., and Lee, J. (2007). Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM ...*, pages 109–118, San Diego, California.
- Lengvenis, P., Simutis, R., Vaitkus, V., and Maskeliunas, R. (2013). Application Of Computer Vision Systems For Passenger Counting In Public Transport. *Electronics and Electrical Engineering*, 19(3):69–72.
- Lerner, A. (2007). Crowds by example. *Computer Graphics Forum*, 26(3):655–664.
- Li, M., Zhang, Z., Huang, K., and Tan, T. (2009). Rapid and robust human detection and tracking based on omega-shape features. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 2545–2548. IEEE.
- Li, T.-Y., wen Lin, J., Liu, Y.-L., and ming Hsu, C. (2002). Interactively Directing Virtual Crowds in a Virtual Environment. *Conf Artif Real Telexistence*, 10.
- Millan, E., Hernandez, B., and Rudomin, I. (2006). Large Crowds of Autonomous Animated Characters Using Fragment Shaders and Level of Detail. In Wolfgang Engel, editor, *ShaderX5: Advanced Rendering Techniques*, chapter Beyond Pix, pages 501—510. Charles River Media.
- Moussaïd, M., Perozo, N., Garnier, S., Helbing, D., and Theraulaz, G. (2010). The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PloS one*, 5(4):e10047.
- Mukherjee, S. and Das, K. (2013). Omega Model for Human Detection and Counting for application in Smart Surveillance System. *arXiv preprint arXiv:1303.0633*, 4(2):167–172.
- Musse, S. R., Jung, C. R., Jacques, J. C. S., and Braun, A. (2007). Using computer vision to simulate the motion of virtual agents. *Computer Animation and Virtual Worlds*, 18(2):83–93.
- Ojala, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence*, 24(7):971–987.
- Ondřej, J., Pettré, J., Olivier, A.-H., and Donikian, S. (2010). A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics*, 29(4):1.
- Ozturk, O., Yamasaki, T., and Aizawa, K. (2009). Tracking of humans and estimation of body/head orientation from top-view single camera for visual focus of attention analysis. *Computer Vision Workshops (ICCV Workshops)*, pages 1020–1027.
- Pelechano, N. and Stocker, C. (2008). Being a part of the crowd: towards validating VR crowds using presence. *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, (Aamas):12–16.
- Ren, Z., Gai, W., Zhong, F., Pettré, J., and Peng, Q. (2013). Inserting virtual pedestrians into pedestrian groups video with behavior consistency. *The Visual Computer*.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34.
- Rivalcoba, I. J. and Rudomin, I. (2013). Segmentación de peatones a partir de vistas aéreas. In *Research in Computing Science*, volume 62, pages 129–230.
- Sun, L. and Qin, W. (2011). A Data-Driven Approach for Simulating Pedestrian Collision Avoidance in Crossroads. *2011 Workshop on Digital Media and Digital Content Management*, pages 83–85.
- Treuville, A., Cooper, S., and Popović, Z. (2006). Continuum crowds. *ACM Transactions on Graphics*, 25(3):1160.
- Tuzel, O., Porikli, F., and Meer, P. (2007). Human Detection via Classification on Riemannian Manifolds. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.
- Van den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. *Robotics Research*, 70:3–19.
- Van den Berg, J. and Manocha, D. (2008). Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935. IEEE.
- Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 1:I-511–I-518.
- Viola, P., Jones, M., and Snow, D. (2003). Detecting pedestrians using patterns of motion and appearance. *International Conference on Computer Vision*, 63(2):153–161.
- Wang, X. and Sun, S. (2008). Data-Driven Macroscopic Crowd Animation Synthesis Method using Velocity Fields. *2008 International Symposium on Computational Intelligence and Design*, pages 157–160.
- Wang, Y., Dubey, R., Magnenat-Thalmann, N., and Thalmann, D. (2012). An immersive multi-agent system for interactive applications. *The Visual Computer*, 29(5):323–332.