Acknowledgements

In order to achieve this result I was offered guidance, hints and ideas by many different people. In the first place I would like to thank my promoter Dr. Davy Vanacken for providing me with detailed feedback and hints during the year I worked on this thesis. For creativity and feedback, I would like to thank my co-promoter, Prof. Dr. Kris Luyten. I would also like to thank Jo Vermeulen for coming up with useful ideas during the brainstorm sessions. Other people I would like to thank are my girlfriend, Dorien, and my family for supporting me during my time in college and university. I would also like to acknowledge all the professors and assistants who shared their knowledge and experiences in the last three years. At last I would like to thank my fellow students and friends for making me enjoy my time in university.

Abstract

In this work we present a rule-based group detection strategy. To achieve this we created a library with an event-driven architecture that facilitates the development of proxemic and group-aware applications. Our method to detect and track groups in a crowd uses a rule set based on basic metrics that are calculated using the frames captured by a Microsoft Kinect sensor. This rule set consists of 7 rules. The specific rules are minimum interpersonal distance, minimum difference in average speed, minimum difference in distance-to-screen, difference in initial and current direction, difference in the arrival times and the time all rules are met. To find the best values of the parameters we conducted observational tests of 9 different situations. We found that the best rule set is highly dependent on the situation and varies across different applications. We describe various research studies concerning group detection and tracking to provide a context of the research domain before we describe the rule set and the library we developed which enables developers to tune this rule set for any specific application. As an example we created a basic group-aware game that serves as a demo of how to use our library. We also note observations with regard to this game in this work.

Contents

List of Figures

List of Tables

Part I

Introduction

In a world driven by new technological innovations, human-computer interaction and more precisely computer vision, only covers a small part of the broader computer science domain. Computer vision itself has various topics, from people detection, tracking and recognition to surveillance purposes, for example to automatically aid guards while watching over a prison [1]. Computer vision capabilities are vastly improving not only because of the broad research community and different application purposes, but also the quickly evolving, easy accessible and cheaper hardware such as HD cameras and processors. A great example is the Microsoft Kinect which is able to detect and track multiple individuals using depth images. It was initially developed to capture gestures to interact with games for the Xbox 360 and came at a relatively low cost. Later Microsoft released a developer API which enabled developers to easily create applications using the Kinect sensor.

Besides detection and tracking of individuals, a fair amount of research has been done regarding group detection, tracking and more generally situation detection. Detection can be used for various applications, from crowd analysis to interaction with public displays. Crowd analysis is a broad domain itself containing crowd management (see figure 1) and surveillance appliances as the most important aspects of it. Classic surveillance techniques usually only consist of a camera monitoring the environment of interest. To extract useful information, to protect people and to prevent vandalism these video sequences have to be constantly monitored by a human specta-

tor. Crowd analysis can be used to extract events and behavioural patterns without the intervention of a human analyst [1, 2, 3]. This techniques should be assisted by a human to exclude false positives. Nevertheless the surveillance agent can be assisted using these methods.

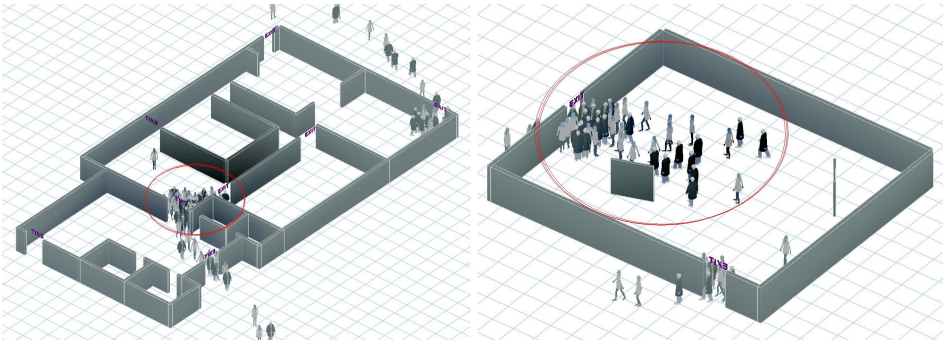

Figure 1: Crowd Simulation software enables event organizers and architects to get a fast representation of a possible emergency situation. [4]

Besides crowd analysis, public display interaction methods also use a fair amount of detection techniques to enhance the interaction techniques. Situation detection, for instance the detection of a group in front of a public display, is a very useful tool especially when context-aware applications are concerned. A great example of such a context-aware application is the work proposed by Vogel and Balakrishnan [5] where the application determines if private information can be shown on a public display. Using the difference in distance to the screen the application is able to determine if private information is appropriate or that the information should stay non-private as others might view the content of the public display. One can easily imagine an extension to this method that also includes the number of people, or even better the structure of the crowd by identifying individuals and groups, in

the determination process. For example, it is not unimaginable that people are more likely able to look over the shoulder of an interacting user if they stand closer to each other due to a higher crowd density. Or it should be possible that if a group is standing in front of a display, this display should be able to show group-related content. In this work we propose a method to detect groups in real-time.

Research question

This master thesis is a feasibility study regarding the detection of groups using a rule set. This rule set can be constructed using the following rules: interpersonal distance, speed, initial and current direction, arrival time, distance from the screen and the duration of the similarity. To achieve this we constructed a library that uses the computer vision capabilities of the Microsoft Kinect to detect and track small groups. The library uses the skeleton data that results from each frame the Kinect captures, to calculate the metrics needed for the rules. The research question is the following:

"Is it possible to detect groups using a basic rule set to check if two or more people belong to the same group? Which rules, that we can calculate using Microsoft's Kinect depth sensing capabilities, are needed to achieve the detection of groups?"

How did we achieve this?

First we studied other detection and tracking techniques regarding individuals and groups to identify to most important issues. Then we identified a set of rules that were used in these studies, to include them in our rule set. Then we developed our library that enables developers to easily create group-aware applications. We also used an application built using the library to observe the rules in recorded situations. During the observations of nine different situations we noticed that the ideal rule set is heavily dependent on the situation. To show the capabilities of the library we also created a group-aware game that is able to automatically divide a crowd into groups. We also note our observations that we recorded during game-play.

Part II

Related work

In this part a global view regarding detection and tracking is provided to establish the context and basic knowledge for this thesis. First we describe the detection of individuals and groups followed by various tracking techniques for individuals and groups. To conclude this part we will also provide a short list of toolkits that use proxemics to create proxemic-aware applications more easily.

1 Detection and tracking of individuals and groups

The main goal of this work is to recognize situations, such as groups, individual wanderers and passers-by in front of a public display using computer vision. To recognize these situations we first have to be able to detect and track people in front of a display. First we describe various detection techniques for individuals and groups followed by the tracking techniques of both categories. We finalize this chapter by describing two related proximity toolkits that enable developers to easily develop applications that use the detection and tracking of individuals.

1.1 Detection of individuals

Most of the time a system that is able to track an individual needs information about the scene (e.g. what are static objects?) and the detected individuals. Various computer vision techniques exist to do so, both for crowded as well as non-crowded environments. In this section techniques are described to detect individuals. Over the years many methods have been studied regarding detection and it would be out of the scope of this work to describe them all so only the most used and researched will be described in the following section.

1.1.1 Techniques to detect people using computer vision

There are various techniques to detect people in image and video data. First a description of sliding window techniques is provided followed by part-based model techniques. These two are the most used techniques. As an alternative a description of a technique that combines background subtraction and

face detection is provided.

Sliding window techniques

Sliding window techniques scan the image or frame when considering video sequences at all possible positions. While the image is scanned it is scaled at different levels to detect people at varying "distances". A sliding window technique consists of two major components: a *feature* and *classifier* component. The feature component encodes the visual appearance of each individual, whereas the classifier is the actual people detector. The latter determines if a human is present for each "sliding window", a region of the image or frame.

To further explain the feature and classifier components we will explain the Histograms of Oriented Gradients (HOG) technique by Dalal and Triggs [6] which is a type of sliding window technique. HOG was initially proposed as a human detector by Dalal and Triggs but can also be used for detection of other kinds objects in images. It is also included in the OpenCV library [1]. Using edge directions or the distribution of gradients the algorithm is capable of describing an object as an encoded object (see figure 2). To extract these encodings the image is divided into small uniformly shaped cells. Then a histogram of gradient directions or edge orientations is compiled for each pixel of each cell. All of these histograms combined result in the "descriptor" or in other words an encoded object.

[1] http://opencv.org/

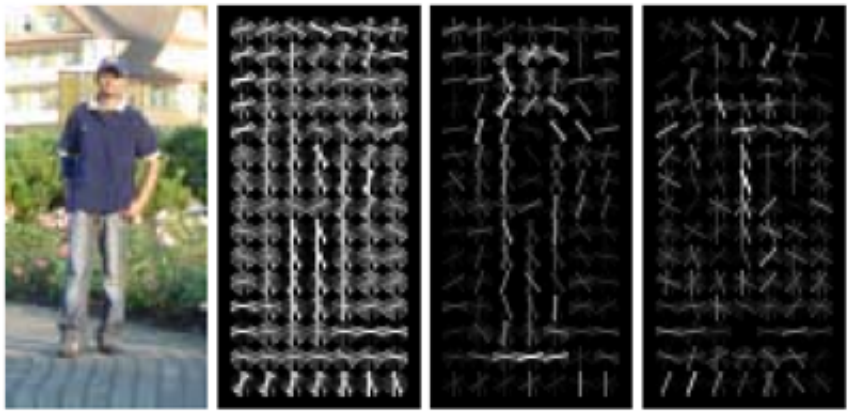

Figure 2: The Histograms of Oriented Gradients technique described by Dalal and Triggs [6] is able to encode an object using edge orientations and distribution of intensity gradients. Using this encoding, deduced from a fair amount training data, the proposed algorithm is able to deduce similar object from images. In this case a human encoding is generated. The second image from the left illustrates a HOG descriptor generated from the image on the left. The subsequent images illustrate the weighted positive (left) and negative (right) SVM weights.

Now the algorithm can use the object descriptor as a "feature" to easily detect similar objects in an image using a linear support vector machine (SVM). This SVM is the "classifier" for the HOG technique. To describe how a SVM works would be out of the scope of this work but the only thing that is necessary to understand the classification process is that it is a machine learning technique that is able to classify vector data into two classes (for more info [7]). In the case of human detection these classifications would be "person" or "non-person". This also indicates that a proper amount of training data is required to properly detect human beings in images knowing that a human can perform different postures and can have different orientations.

A main drawback of this technique is that it can not handle occlusions. Another disadvantage is that sliding window techniques are computationally expensive and are therefore not favourable for object and human detection in videos because computations could not be accomplished in real-time. But thanks to recent developments using parallel GPU computations this drawback is eliminated and sliding window techniques can now be applied in real-time applications [8].

Figure 3: An example of a part-based model technique for human detection. Purple rectangles indicate torsos, red rectangles indicate the head and shoulders, blue rectangles indicate the legs and yellow rectangles indicate full-body detections.

Part-based models

Part-based models are comparable with sliding window techniques in that they also use features and classifiers. This is however the only similarity. First we will describe the part-based model at a high level followed by a part-based implementation by Wu and Nevatia [9].

Part-based models first model individual parts of a human body. The number of parts and which parts may vary, meaning every technique can use other parts or regions of the human body. These parts are then encoded like what happened with the full-human body silhouettes in the previously explained sliding window technique. To encode all relevant body parts a fair amount of training data is needed to find limbs and other body parts in different positions and viewpoints. The second component of a part-based model models the composition of the human body using the parts which were encoded in the first phase. This way a genuine body structure can be extracted from various detected body parts. This technique is used more than sliding window techniques, even Microsofts Kinect sensor uses this technique to detect humans although it uses depth sensing technology to create a depth map instead of a 2D image.

Wu and Nevatia [9] implemented a part-based model which used so-called "edgelet" features. These shape features where developed keeping the human silhouette in mind. An edgelet is a short segment of a line or curve which can be found on the edge of a human silhouette as illustrated in figure 4. Using these edgelets a descriptor can be composed to describe possible body parts. The body parts considered by Wu and Nevatia are (1)

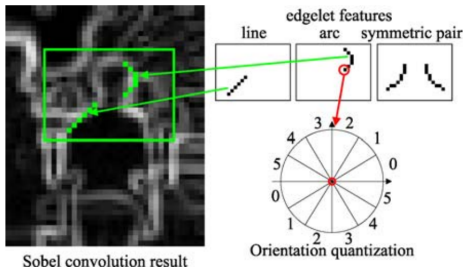

Figure 4: The edgelet features which were used to detect humans in images by Wu and Nevatia [9]. They use these features to reconstruct a human silhouette for detection purposes.

head and shoulders, (2) torso, (2) legs and (4) the full body. Using these descriptors (edgelet features) the classifier can detect body parts.

When the image is scanned for body parts and all responses are collected, Wu and Nevatia create a human object (vector) for each head-shoulder or full body response. They note that these detected humans already give a high detection rate by themselves, but to allocate all parts to a human they use Bayesian probability calculations to match remaining parts with already found humans. These calculations make use of a hypothesis, the expected position of a body part, and compare this with the parts found when the image was scanned. If the probability is high enough the part is allocated to the respective human. If not, it will not be assigned to any human. When the algorithm is completed all partially visible (head-shoulder) and completely visible humans are detected.

Part-based models are able to deal with partial occlusions, which is not the case for sliding windows techniques. Another great advantage is that part-based models are more tolerant to changes in viewpoint or positions

of parts, resulting in a higher success rate. A downside of this technique is that higher resolution images are needed than the previously described sliding window techniques.

Background subtraction using stereovision in combination with face detection

Muñoz-Salinas et al. [10] and Darrell et al.[11] tackle the human detection problem from another perspective. They first apply background subtraction followed by a face detection algorithm to deduce humans from the remaining stereo image.

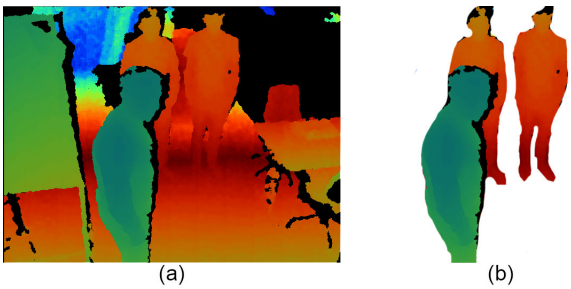

Figure 5: (a) a depth image from the Microsoft's Kinect sensor before background subtraction and (b) after background subtraction.

The technique proposed by Muñoz-Salinas et al. is an extension of a study by Darrell et al. [11] which also used a stereo camera. Using these stereo images they are able to built a disparity map that illustrates motion and can be used to derive a perception of distances. This disparity map is build using the two images recorded by a stereo camera using the differences of the images and the known distance between the two cameras. Using this

map they are able extract independent objects (motion blobs) using background subtraction and they are able to only derive new objects and people in the scene. Now that the motion blobs are extracted we only have the foreground left. The remaining motion blobs on the foreground are then analysed by a skin detector that only selects the regions that might possibly contain a face. Next a face detector is applied to these regions to identify possible people. Muñoz-Salinas et al. note two main drawbacks for this approach. First of all the detection of skin is relative to the illumination changes in the environment and therefore not always produces good results. As a second drawback they note that the method always needs a face to detect possible individuals which is not the case if people are facing away from the stereo camera.

Muñoz-Salinas et al. [10] almost implemented the same method but try to eliminate the drawbacks especially when it comes to tracking. They try to accomplish this by merging two approaches for people detection using stereo images. The first approach is already explained in the previous paragraph and makes use of a face detector to qualify an object as a person. The second approach checks if an object has sufficient weight in an occupancy map. This map shows the occupancy of an object in the image. If an object has sufficient weight it is regarded as a person, this threshold can be set to the weight of a human body using training. Note that this might regard objects with a similar occupancy than a human as a person which leads to false positives. But using the two approaches combined they are able to decrease the error rate of false positives. Note that a person still has to face the camera to be detected by Muñoz-Salinas et al. . There is one difference between the approach by Darrell et al. and Muñoz-Salinas et al. but it only

applies to tracking. Darrell et al. constantly need a face as Muñoz-Salinas et al. only need a face for the first detection and are then able to track the person without the person constantly facing the camera. This however only concerns tracking (see section 1.3).

Note that both methods use stereo vision to build a disparity map. Both techniques also use face detection and as a result are unable to detect people if they are facing away from the camera. After describing this technique we can conclude that it is therefore necessary for a good people detection technique to be able to detect people even if they are not facing the camera.

1.1.2 General detection challenges

The above techniques require a specific number of static elements to be successful, such as non-occluded faces while applying a face detection technique. This leads to a number of challenges. We note the most common challenges regarding detection and how they can be overcome.

Occlusions/Dense regions

In crowded and dense areas occlusions are common, thereby increasing the difficulty to detect (and track) people. In many cases a huge *motion blob* results from analysing images, indicating possible people merged into each other and making it harder to extract individuals.

A lot of work has been done to solve this problem. Part detectors (e.g. [9]) as described earlier are able to handle motion blobs more accurately. Due to the fact that only specific parts of a human body have to be de-

tected these methods are able to deduce humans with a higher success rate than the before mentioned sliding window techniques. Rodriguez et al. [12] try to tackle the problem using a density-aware head detection system in combination with an energy function which results in a heat map (see figure 6). Using this energy function they are able to get much more accurate results for person detection in dense regions. The energy function helps to determine if a detected human is an actual human by combining the probability that the detected area is a human with the crowd density at that place. By extending a classic object detector (in this case people) with a heat map they are able to exclude false positive detections thereby decreasing the error rate and even detect new true positives (e.g. drop a detection of a human in an open space because it is unlikely).

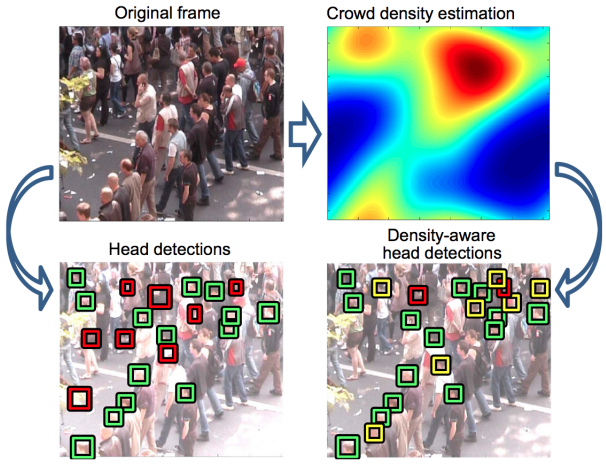

Figure 6: In the bottom-left corner the detections of the object detector are shown without heat map improvement and at the bottom-right with the heat map improvement by Rodriguez et al. [12]. (Red rectangles: false positives, Green: true positives, Yellow: new true positives)

To overcome occlusion problems Muñoz-Salinas et al. [10] approach the problem from a different angle by adjusting the setup. They note that the position of the camera is a key to success and note that a camera mounted to the ceiling is ideal to prevent occlusions. But this comes with the downside that proper silhouette detection and as a result gesture recognition are impossible. This drawback is not a big problem for the detection itself but it excludes interaction techniques using gestures when considering public display environments.

Resolution

The image resolution plays a major role regarding the detection rates. Sliding window techniques generally do not need a high resolution and already have high detection rates. Part-based models on the other hand need a higher resolution to obtain a sufficient detection rate. Paleček et al. [13] note that techniques can also be affected by resolution changes. They note that time critical application can be slowed down if higher resolution images are used for people detection because of the increased number of iterations.

1.2 Detection of groups

Detection of humans is one challenge but the detection of groups is an even bigger challenge due to all factors that have to be considered. Are they closely related or just asking directions for instance, such a question is hard to answer using computer vision techniques. First of all a clear definition of the term "group" is needed. Next an overview considering the most prevailing methods is provided followed by two possible classifications and a conclusion that describes how our algorithm can be classified.

1.2.1 How to define a group

A clear definition of a group is mandatory before starting detection of groups. Therefore we provide two group definitions from different perspectives: the social psychology and the computer vision perspective.

Definition of a group in the field of social psychology

The human definition of a group states *people that know each other, interact which each other, share a common frame of reference and their behaviour is influenced by one another* [14]. As an extension McGrath [15] described "Groupiness". Groupiness is a term to indicate the closeness of group members. A family for instance has a higher groupiness than a crowd. McGrath notes that a group is "groupier" if past interaction occured, they still interact on a regular basis and are expected to interact in the future.

Definition of a group in the field of computer vision

Zaidenberg et al. [2] conclude that interactions, conversations and so-called "groupiness" are hard to deduce from video and image footage and therefore modify the previous definition using only observable properties and define a group as *"Two or more people who are spatially and temporally close to each other and have similar direction and speed of movement for a minimum duration"*.

Inter-personal distance, speed and direction are often used during group detection [1, 16, 17]. McPhail and Wohlstein [16] derived objective static thresholds using social science studies. They found that a inter-personal distance of **7 feet (2.13m)** as an ideal threshold to divide a crowd into groups.

The ideal speed difference is **0.5ft/s (0.15m/s)** and the direction difference should not be exceeding **3 degrees**. These findings are also used by Ge et al. [17] and Zanotto et al. [18] to detect small group structures in a crowd.

Keeping these findings in mind we will not be using interaction as a key factor to group people. We will only consider the factors that are derivable from computer vision images (see section 3).

1.2.2 Behaviour of groups

People that move in groups usually have different moving patterns and behaviour than individuals. These behavioural patterns are useful to deduce groups from a crowd and therefore we provided a dedicated section to them. First we describe the behaviour of people by using f-formations, which originated in the field of psychology, followed by a study by Azad et al. [19] that describes the territoriality and behaviour of groups on and around public displays.

F-formations

F-formations or *face-formations* were first described by Kendon [20] and were used by Marshall et al. [21] to analyse spatial patterns during interaction or by Marquardt et al. [22] to detect small groups. They are a way to describe the structure of a group if they are interacting or collaborating using three different regions: *o-space*, *p-space* and the *r-space*.

The o-space denotes the space inside the group. This space does not necessarily have to contain an object that the members of the group collaborate

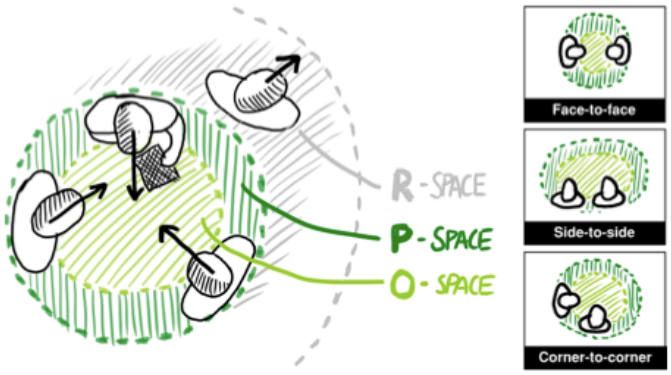

Figure 7: F-formations are a way to describe the structure of a group. The o-space denotes the region where the joint activity is located, the p-space where members of the group are located and the r-space that denotes the region outside of the group.

with but can also be a public display around which the users are grouped, or a more obvious subject: a regular conversation. The p-space is the ring surrounding the o-space, this is the space where people that belong to the group of the f-formation are located. Former two spaces are surrounded by the r-space which can also be described as the outside world (see figure 7 for clarification). People which are located in the latter are not considered as members of the group. This however does not mean that people inside the f-formation do not monitor the r-space to see others that want to join the group for example.

An f-formation can have many possible shapes and do not have to be necessarily circular. For example a group of two people can have an L-shape (or corner-to-corner), they can stand side-by-side while watching a public display or face-to-face while interacting. Marquardt et al. [22] note that these kind of shapes are most of the time related to the tasks a group is

executing. If the f-formations is L-shaped the group is more likely to be working on a communicative task. Side-by-side on the other hand indicates a more collaborative task and a face-to-face shape suggests a more competitive task as people try to occlude their actions to spectators in the r-zone.

When we discuss the different group detection techniques in section 1.2.3 the technique by Marquardt et al. [22] will be described which uses these f-formations to detect small groups.

Behaviour in front of public displays

Azad et al. [19] dedicated a complete study on how people behave on and around a large public display. First they conducted a field study to observe the intra- and inter-group behaviour followed by a controlled study to examine how people manage the on-display territoriality and how they move in front of a vertical display. We will first note their findings regarding the field study followed by the results of the controlled study.

Field Study The result of their field study can be divided into intra- and inter-group behaviours. We only note the most relevant findings starting with the intra-group behaviour.

Intra-group behavioural results were divided into two stages: the initial *approach* and the actual *interaction*. The approach stage is considered as the period between the first movement towards the display by one member until the group arrives at the display. They note three primary movement

formations: *led*, *asynchronously delayed* and *simultaneous*. Led approaches are characterized by one or two leaders that lead the group to the public display. Async delayed approaches have one or more members that arrive later (1-10 minutes) and simultaneous approaches obviously require all members to arrive at a similar time.

The most relevant category of behaviours Azad et al. considered are inter-group behaviours. Inter-group behaviours consider all groups in the immediate surroundings of the public display and how they behave with respect to one another. This is really interesting because their findings also indicate certain properties of groups. These findings can be helpful to distinguish groups from one another. There where three remarkable observations noted. First they noted that members of the group stood closer to each other when the vicinity of the display was crowded. Density changes also resulted in formation morphs. Secondly they reported that if no screen real estate was available for new approaching groups, these groups automatically queued up. The already interacting groups continued the interaction process as other groups stacked up behind. The last group that approached the display is also stacked last in the queue. The third and last notable observation when considering inter-group behaviours describes the event when a new group positions itself besides another group's display (e.g. The photo kiosk alongside another that is already occupied). If one or more members of the first group infringe the area in front of the adjacent display the new group automatically squeezes itself in whereas the group that arrived earlier remains more or less the same when density is considered.

Controlled experiment During their controlled experiment Azad et al. [19] observed how concurrent individuals and groups share and use a large display. During this lab test they conducted tests regarding five different grades of collaboration from no collaboration to highly collaborative tasks. There where two types of user categories: Individuals (SINGLES) and groups that only consisted of two people (PAIRS). The users had to solve a puzzle using pieces that where located on a big whiteboard. The solution was shown above the area where they had to operate. The pieces where located different as the collaboration level changed. For instance, if the researchers wanted the users to collaborate more they placed puzzle pieces in the working region of another user/pair. The results found after conducting different tests are divided into three categories: on-display behaviour, off-display behaviour and on- and off-display behaviour all together. We will only describe the off-display results as they are only relevant for this work.

For off-display behaviour which considers the formations which are formed during interaction, they note two formation related observations. First of all they describe the possible initial and settled formations which are formed. They noticed that pairs were unlikely to change their right-to-left positions. Rotations rarely occurred. Secondly they note that a certain buffer zone is maintained between the two different groups while interaction occurred. The latter observation is useful to deduce groups from a crowd knowing that separate groups always preserve a buffer.

Using the findings in the studies by Marquardt et al. [22] that uses f-formations and Azad et al. [19] that observed the behaviour of groups

around a public display we are able to deduce useful data that is strongly related to group structures. For example we can keep in mind that the inter-group buffer as described by Azad et al. might be useful to divide a crowd in groups. Another finding that might be useful is the property that group members step closer to each other than normal in a crowded environment. The next section is devoted to techniques to detect groups and it also contains a study that used f-formations to detect small group structures.

1.2.3 Techniques to discover groups

Group discovery techniques mostly rely on the people detection techniques previously described in section 1.1.1. Clustering is a well established method in the field of group detection. There are a few approaches regarding clustering techniques. We discuss *agglomerative* clustering, *divisive* clustering and *k-means* clustering. A different approach uses social forces defined by the social force model to determine who knows who. Another recent study uses f-formations described in section 1.2.2 to detect small group structures.

Agglomerative Clustering

Agglomerative clustering (see figure 10) [17, 1, 23] is a bottom-up clustering technique starting with each detected person as a cluster. The algorithm then looks for clusters that can be merged using a distance measure (e.g. Euclidian, Manhattan, Mahalanobis or Hausdorff distance). Finding the best value for this distance measure is the hardest part of the technique to be successful. The algorithm stops when no new clusters are formed resulting in a set of clusters or in the case of group analysis: groups.

Chang et al. [1] use the spatial-temporal dissimilarity between trajectories of individuals as a distance measure to form clusters using Kruskals algorithm [24]. This algorithm starts with each individual as a node (or forests with one node). During the first iteration each possible pair distances are compared and the two nodes with the smallest distance are connected. The second iteration the previously found pair is expanded with their closest neighbour. This process goes on till all nodes are added to the minimum spanning tree that originates during the process or till all remaining nodes are further than a set threshold θ and therefore excluded from the spanning tree. The minimum spanning tree thus varies when the threshold θ is altered. This spanning tree (see figure 8) is a naive representation of the crowd because Kruskal uses a greedy approach, meaning only a local optimum is reached. As a result Chang et al. note that weaker connectivity between nodes is never considered and thereby "closer" edges than these represented in spanning tree may bee overseen. To illustrate this algorithm and its shortcomings figure 8 shows the process of creating the spanning tree and the hierarchical group structure as a result. Although the imaginary edge "ae" is a closer connection then crossing the complete spanning tree, it is not included. This indicates that Kruskals algorithm tends to be greedy.

Ge et al. [17] compare noisy trajectories of detected individuals (see figure 9). Each iteration they calculate the norm of the velocity difference vector based on terms defined by the different Gestalt principles: Common Fate (same outcome), Similarity (similar movement, direction), Proximity (inter-personal distance) and Patterning. Using this difference vector as a distance measure, they are able to find the similarity between two individu-

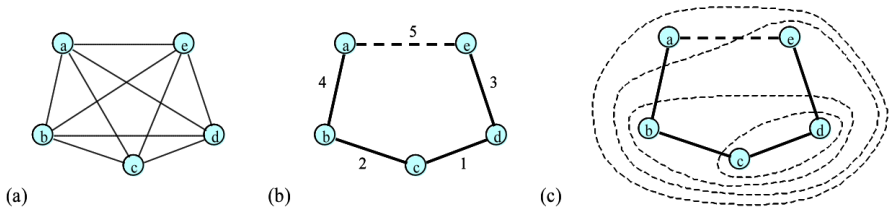

Figure 8: Agglomerative clustering using Kruskals algorithm to extract groups from a set of detected individuals as described by Chang et al. [1]. (a) represents all nodes and their inter-personal distances. (b) represents the minimum spanning tree that results after applying Kruskals algorithm. Note that the edge "ae" is imaginary. (c) shows the possible group structure with varying thresholds for the distance measure.

als using the Hausdorff distance, which is often used for trajectory analysis. The technique proposed by Ge et al. does come with the downside that it can't be used in real-time applications because assumptions are made using data of a longer time frame. Nevertheless this method has the benefit that assumptions are better because they are derived from a bigger set of data, excluding false positives.

We note that bottom-up clustering tends to be a greedy approach but on the upside it is more efficient when dealing with small groups in crowded scenes because less iterations are needed to conduct all groups from a crowd.

Divisive Clustering

Divisive clustering [1] is a top-down approach starting with the crowd as a big encompassing cluster. Recursively smaller clusters are found until no further splits are possible and people with strong connections are in the same group. A split occurs if an element or subcluster of the cluster has a greater

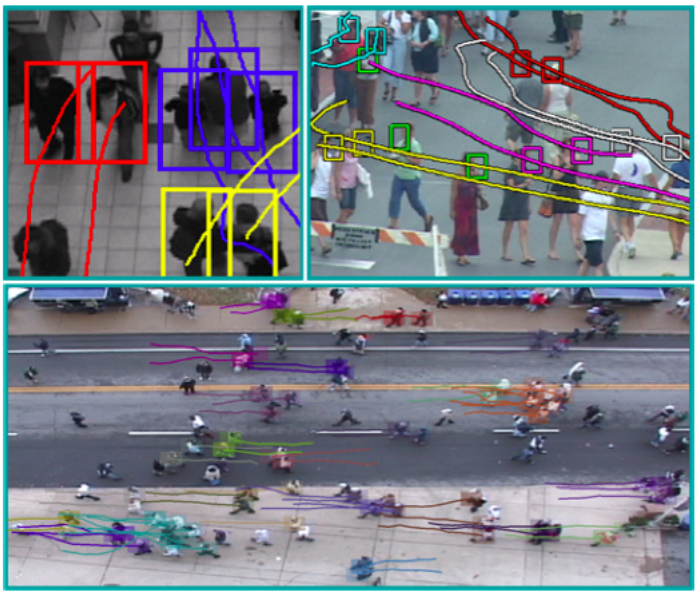

Figure 9: Ge et al. [17] compare noisy trajectories to group individuals.

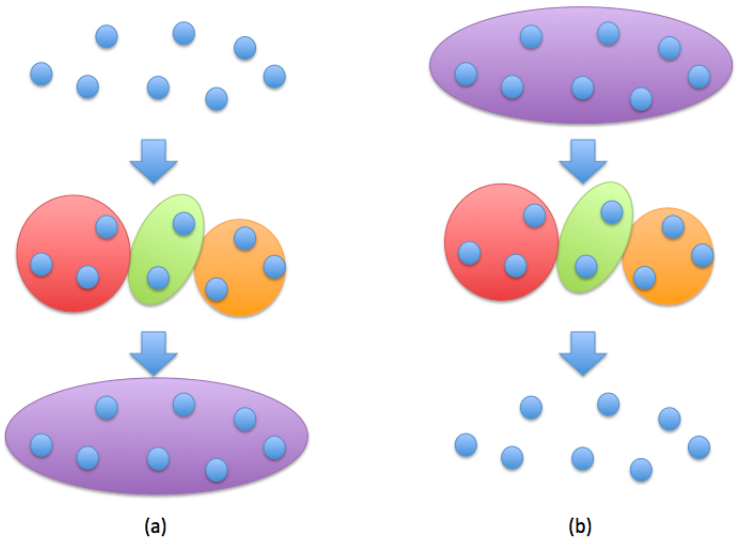

Figure 10: The comparison between agglomerative clustering (a) [1, 17, 23] and divisive clustering (b) [1].

distance than the set threshold (*cut size*). A split should result in stronger within-group connections, as between-group connections get weaker.

Chang et al. [1] use the same distance measure as in their agglomerative clustering algorithm, the spatial-temporal dissimilarity between tracks of people. They apply a slightly modified divisive clustering technique as they do not use a cut size but they adopt an approach proposed by Newman [25] to split clusters. Newman argued that the cut size is not intuitive regarding the concept of grouping. He proposed a method using the *modularity measure* which expresses the difference between expected and actual connections between individuals of a group. For instance, if a and b are strongly connected if their connection is stronger than with any other pair in the cluster. He stated that this measure should be maximized to find the optimal solution when performing a split procedure. Chang et al. note that finding the maximum modularity measure is NP-hard and therefore they use an approximation which Newman had proven to be a valid substitute [26]. The approximation can be derived using an eigen decomposition. The process continues recursively until all connections are equally strong or no divisions are possible (which is the case if only one person belongs to the group in question).

If a divisive clustering algorithm is used to detect large groups it is faster than agglomerative clustering because less iterations are needed to find the optimal solution.

K-means

Another approach that uses a clustering algorithm is proposed by Qin and Shelton [27]. They implement *k-means* with the similarity of the trajectories of each individual as the distance measure. The algorithm (the crosses in figure 11) starts with K initial centroids. Each iteration all individuals are added to the group closest to them. After each individual is associated with a group a new group average (centroid) is calculated. Then a new iteration starts using the new centroids to associate individuals with the centroid. The algorithm continuous until the centroids (averages) stop changing or a maximum number of iterations are executed. The local optimum is then reached and all individuals are associated with a group (see figure 11).

Choosing the initial centroids is a difficult task. First of all these centroids are key to detect the correct clusters. As an addition the algorithm takes longer to identify clusters if the clusters are poorly chosen in the first place, because more iteration are necessary due to the centroids that keep changing. Another and major drawback of these initial centroids is that the number of groups should be known beforehand, which is not an ideal solution considering public places where all kinds of group structures may occur.

Qin and Shelton try to overcome the latter drawback by iterating the algorithm with a variety of values for K to find the best local maximum. They argue that this can be done in parallel to shorten execution time. The method employed by Qin and Shelton however doesn't always yield for the best result as they still only iterate the algorithm with a small value for K and thereby exclude situations where more groups than K are present.

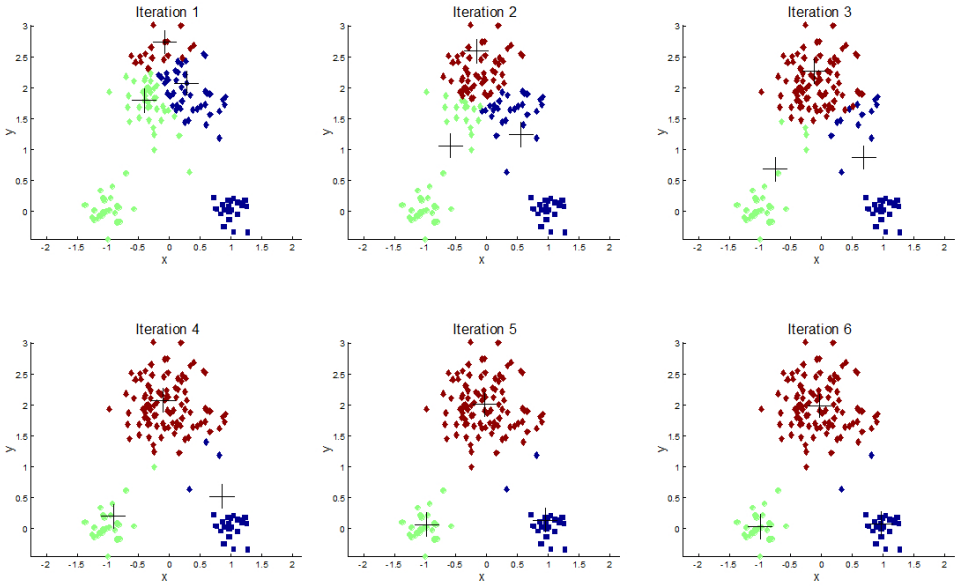

Figure 11: The K-means algorithm which is used by Qin and Shelton [27] to deduce groups from a crowd. Each iteration the result improves until the stopping criteria is reached. (figure from http://www.practicaldb.com/)

Using The "Social Force Model"

The Social Force Model (SFM) was first proposed by Helbing and Molnar [28] to enhance crowd simulations. The SFM works like an analogy of fluid dynamics, keeping in mind all forces that influence the movement of each individual. For instance, if two people belong to each other a certain group relation force is present, which tends to attract two people while moving (see figure 12).

Šochman and Hogg [29] use the SFM by inverting it. They start with the known trajectories of each individual and then try to find the social forces that caused that behaviour. Knowing the forces they are able to answer the

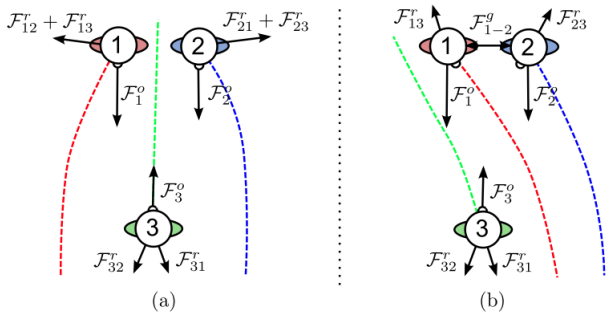

Figure 12: The Social Force Model uses the different forces on individuals. In (a) there is no significant group relation whereas in (b) a clear relationship is illustrated by the red and blue line that stick together. [28, 29]

question "Who knows who?" using the found group relation forces. They show that their method of inverting the SFM is especially useful in crowded scenes where other methods (clustering) are less applicable. This method is a *delayed* method, meaning that it makes assumptions based on known data, but the final group decision is taken with an offline error minimization process. This also means that the detection of a group is not instantaneously which makes this method less applicable in time-critical applications.

Mazzon et al. [30] extend the work by Šochman and Hogg [29] by adding extra rules to prevent false assumptions regarding group crossing and approaching. To prevent groups from falsely splitting they add a direction factor which keeps groups together even though a force from a crossing pedestrian might draw them apart. That way the group consisting of person 1 and 2 in figure 12 are grouped in both scenarios even if a person splits the group. If a group has a similar direction (within 180 degrees) they will not split. To avoid the case that one might be added to a group although he was just passing by, Mazzon et al. add a delay (# frames) before a person

is added to a group.

F-formations

Marquardt et al. [22] use f-formations to detect small group structures using the Kinects depth-sensing capabilities. Their set-up consists op two Kinects mounted to to ceiling. This way they prevent occlusions and they are able to detect individuals properly. Using the Kinect they are able to determine the inter-personal distances, body orientation and even the direction the individual is facing. These are the only necessary parameters that they have to know about an individual to check if he belongs to an f-formation or not.

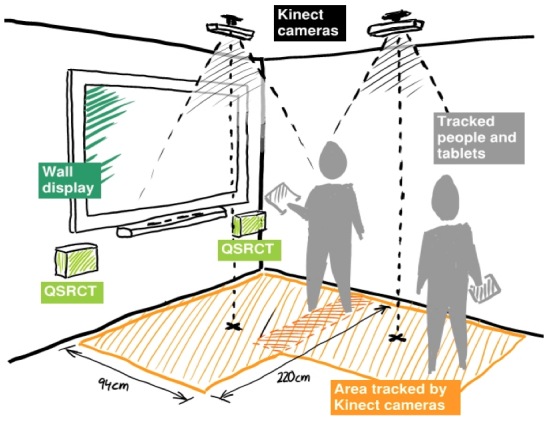

Figure 13: Marquardt et al. [22] use two Kinects with a top-down view to detect f-formations. This way no occlusion problems arise and they are able to deduce a person's body orientation and gaze direction. These parameters in combination with the interpersonal distance make the detection of f-formations possible.

The algorithm proposed by Marquardt et al. keeps in mind that two

people can only belong to an f-formation if:

- they do not stand behind each other.
- the angle between their orientation vectors is smaller than 180 degrees.
- their interpersonal distance is small enough to communicate and their o-spaces (see section 1.2.2) overlap.

The algorithm iterates through all pairs of individuals and calculates the above parameters. If all conditions are met an f-formation type (L-shape, face-to-face, side-by-side or none) is assigned to the pair. People outside any formation are also detected and tagged. Because the parameters have hard thresholds Marquardt et al. implemented hysteresis to prevent "flickering" from happening, this way the borders can be crossed for a short period without the person being removed from the f-formation.

This technique is able to detect groups of people in real-time using only distance and body orientation as parameters. Of course this does not always result in the optimal solution because they only guess relationships based on the f-formations structure and do not consider other parameters such as crowd density that can have an impact on the orientation and distance of one person to another (e.g. A higher density results in an unnatural orientation and closer distance because people of the space that is heavily reduced).

1.2.4 Classifications of group detection techniques

There are many ways to classify a group detection technique. There are techniques that need a (long) time window and techniques that can handle

single frames separately. There are techniques that use training data (supervised), whereas others do not need a priori data (unsupervised). In this section a distinction between these classifications is described.

Distinction between single frame and time-window approaches

A single frame method is able to detect groups without any knowledge about the trajectories of people. On the other hand there are methods that need a longer time window to detect groups by using the trajectories (e.g. Using the social force model [29] or the extension of this model by Mazzon et al. [30]). The latter often has to be able to process noisy data due to noisy trajectories but their greatest drawback is that time-window approaches are more time consuming and are therefore not applicable for time critical applications On the other hand they provide reliable assumptions of group structures because they use more data to come to these results. This way they can provide a great amount of information for applications that are not time-critical (e.g. monitoring a recorded video sequence).

Distinction between supervised and unsupervised methods

Supervised methods use classifiers or heuristics and use data from previous frames and a lot of training data to detect groups whereas unsupervised techniques do not need training data and no or little data of the previous frames. This makes unsupervised techniques hard to develop because they have to find group structures on a per frame basis without little or no knowledge of previous frames. Making estimates of who belongs to who is therefore harder. Zanotto et al. [18] argue that the variability of group

structures and crowd structures is hard to tackle for supervised techniques and therefore favour unsupervised techniques that do not need training data to operate.

Classification of previously described techniques

The techniques listed in the previous section were all unsupervised techniques, meaning that they do not use machine-learning techniques or heuristics to detect groups. There is only one technique that can be used in time-critical applications. The f-formations technique by Marquardt et al. [22] is capable to detect groups in real-time using three straightforward rules (see section 1.2.3 for their set of rules). We do not consider f-formations but try to group people using other assumptions (e.g. if they arrived at roughly the same time they might be related). All the other techniques compare trajectories resulting in a longer time until detection. Only the two social force model techniques try to make assumptions using the already known data. This however causes false assumptions and still there is still a short delay present. A summary of the presented techniques can be found in table 1.

	Single Frame	Time window	Supervised	Unsupervised
Agglomerative clustering				
Chang et al. [1]		X		X
Ge et al. [17]		X		X
Divisive clustering				
Chang et al. [1]		X		X
K-means				
Qin and Shelton [27]		X		X
Social Force Model				
Šochman and Hog [29]	(X)	(X)		X
Mazzon et al. [30]	(X)	(X)		X
F-formations				
Marquardt et al. [22]	X			X

Table 1: Classification of the group detection techniques described in section 1.2.3.

1.3 Tracking of individuals

Besides detection of individuals and groups we also describe the possibilities regarding tracking of individuals and groups. In this section we provide an overview of well established techniques to track individuals using computer vision. Note that some of these techniques can be applied to the former described detection techniques as an extension.

1.3.1 Tracking techniques

In this section we describe three techniques that can be used to track individuals while moving. First we will describe *color-based tracking* followed by *frame-to-frame tracking* and a tracking technique that uses *a combination of the previous techniques* for increased robustness.

Color-based and pattern-based tracking

Color based tracking uses detected colors to distinguish detected people from each other. McKenna et al. [31] note that this technique is able to readdress ambiguous detections that occur after occlusions to the correct person. They use this technique in combination with a background subtraction detection technique (see section 1.1.1) and also take gradient information into account to cope with illumination and shadow variations. Another advantage of this method is the ability to extract qualitative estimates of depth ordering and positions of detected persons that are very useful data to deduce groups as described in section 1.2. Off course people with identical clothing colors are indistinguishable and are therefore a hassle with regard to tracking algorithms based on color. The Microsoft Kinect uses a similar method to track humans. It detects users for each frame it captures individually. Afterwards the user gets a unique ID based on color data found in previous frames. This technique is also called *tracking-by-detection* and is used in numerous studies.

Thaler and Bailer [32] use a similar approach to track humans in a (panoramic) video sequence. They use a *tracking score* for each individual which is calculated based on a color histogram. The color histogram can be described as an image of the human that is divided into individual parts that all have a color code. If a tracking score is similar to a tracking score in a previously detected frame the persons are "matched". To improve this tracking score the method makes predictions regarding possible motion, this way a similar tracking score that can be allocated to two or more different people can be more accurately allocated to a person which motion is

comparable to the predicted motion and thereby avoid false allocations. The technique described by Thaler and Bailer is comparable with the color-based tracking technique by McKenna et al. described in the previous paragraph.

Color-based tracking is highly dependent on good illumination of the observed scene as change of perceived color might cause false assumptions. McKenna et al. [31] try to overcome this problem by using gradient information which is less sensitive to illumination changes. Another minor disadvantage is that a color tracker is not able to deal with occlusions. Thaler and Bailer [32] try to overcome the last drawback by including the 10 previous frames into the calculations and are thereby able to deal with short-term occlusions. Occlusions are however a general problem when tracking methods are concerned. Pattern-based tracking works in a similar manner as the former color-based techniques in that way that it looks for a similar visible image area to distinguish people from one another instead of a color.

Frame-to-frame tracking

Zaidenberg et al. [2] propose a frame-to-frame tracking technique (or F2F tracking) that combines several factors to calculate the similarity between an object and previously found objects which are linked if a threshold is reached. They use the 2D and 3D displacement distance, 2D area and shape ratio, a RGB color histogram, a histogram of oriented gradients (HOG, see 1.1.1), color covariance and the dominant color. This similarity measure is used in combination with a weighted graph that combines new links and temporal links which are discarded after n frames. Using previously found links the system is able to track objects even if they are not recognized for

several frames. This is particularly useful for occlusion handling.

Combination of previously described techniques

A combination of techniques is not always possible because of the compatibility of the previously described methods. Nevertheless a combination of techniques sometimes yields a better result because they can eliminate each others drawbacks.

Darrell et al. [11] use such a method. They use three different methods to distinguish humans and thereby are able to track them. They use *depth*, *color* and *pattern* information. Depth information is obtained using a stereo vision camera which can cause false positives as depth estimates may be incorrect. The RGB-camera looks for fleshlike colors. This however causes false positives to arise as different skin colors are prevalent. And even if this was not the case it would falsely detect flesh-like colors of inhuman objects in the environment. The pattern detector used is a face pattern detection technique. One might argue that this is not a good indicator on its own as a face should always be visible to the camera. All these techniques have their own drawbacks. But Darrell et al. argue that a combination of the three techniques to detect and thereby track humans yields better, robust and fast tracking capabilities.

The previously described computer vision tracking techniques have one thing in common: they all look for a frame of reference to recognize previously detected elements. This is not surprising if we compare the techniques with

our own methods to follow people in a crowd in our everyday life. If someone is wearing a red t-shirt we will be able to "track" them in a crowd because we look out for the red t-shirt which we use as a frame of reference.

1.4 Tracking groups of people

Besides detection of groups, tracking is also a valuable subject regarding crowd analysis, for instance to aid surveillance monitoring techniques [2, 33]. Another example can be found in the gaming industry where Microsoft's Xbox[2] uses tracking of users to enhance the gaming experience using the Kinect. Tracking is necessary in this case to distinguish multiple gamers. In the following section we will describe various techniques that are able to track groups. Note that these methods mostly first track individuals using previously described tracking methods for individuals before they try to form groups. Forming groups is already considered in section 1.2 therefore we only provide a brief description of 4 group tracking techniques.

1.4.1 Techniques to track groups of people

McKenna et al. [31] propose a color-based tracking algorithm (see figure 14). They note that their method should be able to cope with slow illumination changes, different shapes and rotations. Therefore they argue that color is a good reference to track humans. Their tracking algorithm is based on background subtraction detection of humanoid figures and use three levels of abstraction to track these figures: *regions*, *persons* and *groups*. If a humanoid figure is detected it is encompassed by a rectangle defining a tracking region. A region is defined as a connected component that has been

[2] http://www.xbox.com/

tracked for at least n frames. Each region has a bounding box (rectangle), a timestamp and a tracking status. Each person has an appearance model based on color. This appearance model can be used to match and thereby track persons in subsequent frames. McKenna et al. also found that matching overlapping bounding boxes was an effective method to track people. In addition the system is able to form groups using persons of whose bounding box (region) is the same. This can be compared with group formation using proximity described in section 1.2. A region can contain more persons over time as they are merged if they are in each others proximity. For a region to be seen as a person it has to satisfy a set of rules. First of all the size of the bounding box has to exceed a certain threshold and their position on the x-axis should be similar. In addition rules like a specified aspect ratio and skin color can be added. A person is seen as a group of one. When regions are matched with more than one person they are merged to form a group. If however their proximity is not sufficient the group is split. This method described by McKenna et al. does not compare routes of each individual but the only parameters they use are proximity and color to determine if two groups in subsequent frames are the same. In addition they are capable to cope with shading and slow illumination changes.

Zaidenberg et al. [2] base their algorithm on the following definition of a group: *"two or more people who are spatially and temporally close to each other and have similar direction and speed of movement for a minimum duration"*. Tracking is based on F2F-tracking explained in the previous section (1.3). This tracking technique tracks individuals but also produces a weighted graph representing the distances between all detected mobiles (or objects). This weighted graph can in turn be used to deduce groups. An

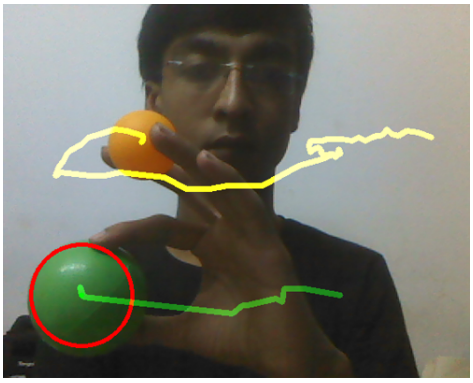

Figure 14: Color based tracking performed on a green and orange ball [34]. The same method can also be used to track people.

example of such a graph is presented in figure 15. The group tracking algorithm presented by Zaidenberg et al. includes 4 different parts: group creation, updating, split/merging of groups and group termination. People are added to a group using a group coherence measure. This measure is defined by the speed, distance and direction of an individual. Using this measure the algorithm is able to identify possible new members of a group or two individuals that can be merged to form a new group.

Mazzon et al. [30] also propose a graph-based approach comparable to Zaidenberg et al. [2]. Their approach is not applicable for time-critical approaches as it takes multiple frames to detect groups in the first place. If group detection is performed the detected groups are matched using their centroids (proximity measure similar to McKenna et al.), velocity and a short temporal buffer that enables the method to handle short occlusions. The method first tries to match groups in subsequent frames (short paths). If a group is not found in a frame (no match is found), the group will be added to a buffer for n frames and will be matched later on. The frames in

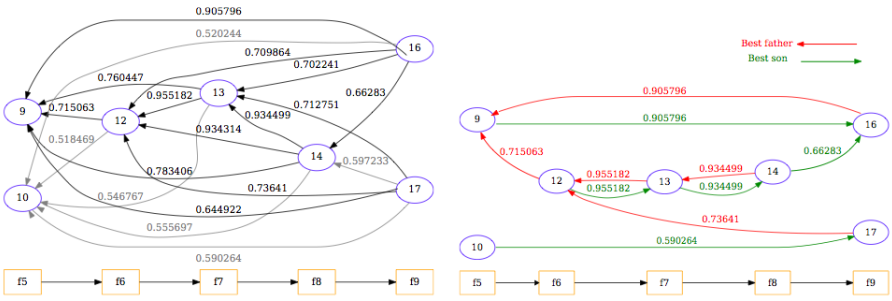

Figure 15: At the left side the complete graph shows all "distances" between detected objects. Note the timeline that shows the frames in which an object was detected. At the right side a visual representation is given if the graph is pruned and only the best father(older frame)/son(new frame) combinations are maintained. This graph is used during the F2F tracking technique used by Zaidenberg et al. [2].

between will be extended with an estimation of the group's position which is calculated using linear interpolation. This way people detected earlier can be linked to a group that was not detected in each frame yielding for a robust result.

Like the previously described methods by Mazzon et al. and Zaidenberg et al., Qin and Shelton [27] propose an "Offline" method (not useable in time critical applications). To link groups detected in different frames (using a K-means clustering approach) they do not only use position and velocity (motion smoothness) but also add a color histogram comparison to further enhance the group matching algorithm. These two parameters, motion and color, are then used to calculate the probability that a group is found again and therefore can be matched. To handle occlusions the system uses a similar method as Mazzon et al. which uses linear interpolation to estimate the position of a group that was not found in every frame.

	Color	Position	Velocity	Direction	Occlusion Resistant	Real-time
Mazzon et al. [30]		X	X		X	no
McKenna et al. [31]	X	X				yes
Zaidenberg et al. [2]		X	X	X		delayed
Qin and Shelton [27]	X	X	X		X	no

Table 2: Comparison between different group tracking approaches discussed in section 1.4.1.

1.4.2 Similarities between group tracking techniques

Now that we have described a few group tracking algorithms we can conclude that they all have similar approaches. McKenna et al. describe an "online" method whereas all other approaches need a longer time to track groups. For a better visualisation of the similarities table 2 gives a good overview of the different parameters used during the group tracking process.

1.5 Proximity toolkits

Recently multiple toolkits have emerged that use sensing hardware or the built-in sensors of Microsoft's Kinect. As we provide a library that offers developers to detect groups using a set of rules, we provide related toolkits in this section. Marquardt et al. [35] provide *the Proximity Toolkit* which enables developers to quickly prototype and explore new interaction techniques that use proximity information. Nebeling et al. [36] have build the *XDKinect* framework around Microsoft's Kinect to facilitate multi-device cross-platform development using multi-modal interaction techniques. In

this section we will briefly describe the capabilities of both frameworks starting with the Proximity Toolkit. To conclude this chapter we briefly list different proxemic aware toolkits.

1.5.1 The Proximity Toolkit

The Proximity Toolkit by Marquardt et al. [35] was created to speed up the development of proximity based applications and to enable fast prototyping. The toolkit acts as a mediator between various sensors (not only the Kinect) and the application by extracting proxemic information from the sensors and by making this information accessible in an understandable manner. Marquardt et al. note three key features. First of all it can be used as a prototyping tool for novel interaction techniques by providing developers with proxemic information: orientation, motion, distance, identity and location information between entities. This way the designers can concentrate on their research instead of experimenting with the sensor hardware. Secondly the toolkit has a visualisation monitoring tool that offers to possibility to record and observe different situations in a 3D environment. Last of all the architecture separates the hardware from the proxemics data and thereby offers the possibility to combine different sensor hardware.

1.5.2 XDKinect

The XDKinect framework by Nebeling et al. [36] is another framework that facilitates the development of multi-modal applications in combination with proximity information. Developers can use the client-server architecture where the server is connected to the Kinect to easily create applications using a web-based approach while using sensors of the Kinect. Both audio

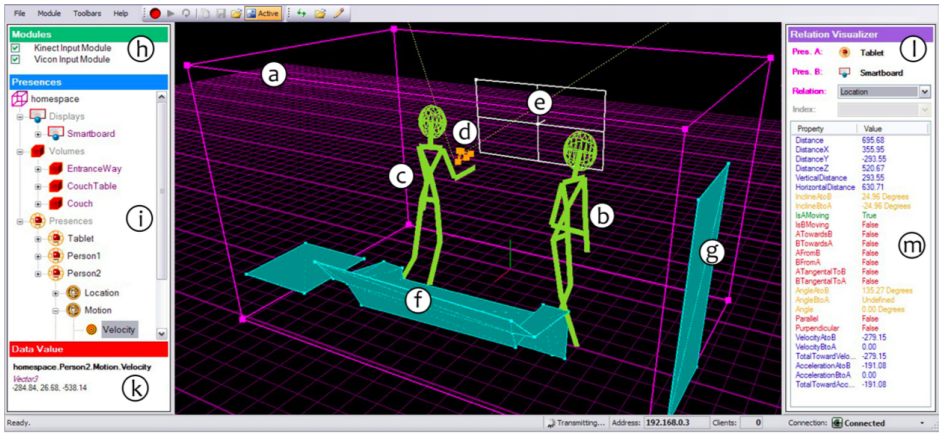

Figure 16: The visual monitoring tool of the Proximity Tookit by Marquardt et al. [35] provides developers and interaction designers with proxemic information in a 3D space. Situations can be visually observed and can be recorded.

input and gesture recognition are built-in to facilitate the development of multi-modal applications. A great advantage of the web-based approach is that applications can easily be created for multiple platforms at once and secondly developers should not learn a new programming language as the default web standards are used. Since the Kinect SDK is only available for Windows the XDKinect framework can be used as a middle-man to create a cross-platform applications. Another feature of the XDKinect framework are the provided APIs that provide developers with multiple abstractions instead of the "raw" data that is provided by the Kinect SDK. The different APIs are:

- **Time-based API:** enables the user to get and constrain the Kinects streaming data.
- **Multi-modal API:** enables the developer to capture gestures and speech input.

- **Proxemics API:** provides the developer with proxemic awareness information.
- **Cross-device communication API:** enables communication between multiple client for easier multi-device application development.
- **Settings API:** To quickly change the framework settings to the applications requirements.

Using these APIs a developer has a lot of flexibility throughout the design and implementation process yielding for a broad application domain. During their evaluation process they asked multiple application developers to create a basic photo gallery application using their event-driven framework. To evaluate the toolkit they used Olsens rules [37] to determine the usefulness. These rules are created specifically for evaluation of User Interface Systems such as the toolkit in question.

1.5.3 Other toolkits

Nebeling et al. [36] provide a great overview of the differences between existing frameworks that provide application developers with proxemic information. To describe all frameworks would be beyond the scope of this work but table 3 provides a good overview of the different functionalities.

Public displays come in different sizes, from palm-sized information signs to sky high billboards, from tabletops to ground projected ambient displays. All of these may have varying intentions, from providing information to entertainment. Different sizes, placements, expected audience and other specifications can be used to part public displays into groups but in this work the classification of screens is done using their spatial context because we try to provide application developers with information about this context.

	cross-device comm.	device-device aware.	Gesture	speech	distance	orientation	identity	motion	multi-user
XDKinect [36]	X	X	X	X	X	-	X	-	X
Kinected Browser [38]	-	-	X	X	-	-	X	-	X
jQMultiTouch [39]	-	-	X	-	-	-	X	-	X
Shared Substance [40]	X	X	X	-	-	-	X	-	X
GroupTogether [22]	X	X	-	-	X	X	X	-	X
Interact. Ambient Disp. [5]	-	-	X	-	X	X	X	X	X
Proximity Toolkit [35]	X	X	-	-	X	X	X	X	X
Proxemic Media Player [41]	-	X	-	-	X	X	X	X	X

Table 3: Comparison between different proximity aware toolkits provided by Nebeling et al. [36].

Part III

Our group detection approach

While exploring the various methods and techniques to deduce groups from a crowd we noticed that a group detection library would be a great solution to extend already existing toolkits that only consider individuals. To detect groups we wrote a library that in some way extended the capabilities of the Kinect for windows API from Microsoft. We use Kinects people detection technology in order to obtain the best possible location of each user. Various parameters could be altered to enhance the recognition rates and thereby resulting in a list of variables which are more or less suitable for detection of specific events, such as the formation of a group. In our approach we try to develop an unsupervised group detection method that is able to detect groups using a single frame of reference (see section 1.2.4). The library created for this purpose enables a developer to easily adapt the group "binding" properties for different application contexts.

2 The concept, architecture and hardware set-up of the solution

In this section we describe the concept, our hardware set-up as well as the software architecture behind the library.

2.1 Concept

First we will describe the idea behind this thesis before we will describe the different aspects of the library itself in subsequent sections.

The idea

The idea behind the library is to provide a flexible and easy to understand technique to detect groups in a crowd. We offer to ability to create a rule set using different rules that can be derived from the Kinect data we get from the Kinect for Windows API[3]. A good example of such a rule is the interpersonal distance. See section 3 for more information regarding all possible rules. The created rule set can be adjusted to match the application's purposes. If a developer for instance wants to divide a crowd into groups by only using the interpersonal distance he only has to enable this rule. The rules can have different weights to indicate the more important rules. If the rules included in the rule set are met or exceed a certain threshold the two detected persons will be grouped together. This can lead to *a chain reaction* if a person is grouped with multiple detected persons. Using the library we would like developers not only to be able to easily create a proximic aware application but also to develop a *group-aware* application for instance to

[3] http://www.microsoft.com/en-us/kinectforwindows/

improve screen estate management.

Our technique tries to identify groups using a single frame of reference and an unsupervised method using a simple rule set (see section 1.2.4). We do not compare trajectories although it is often used in the group detection domain [17, 27]. This makes it possible to use our method in time-critical applications.

Now that we clarified the purpose of this master thesis we go further in depth about how we developed the library.

2.2 Set-up

Our set-up contains two elements: the Kinect features as a key hardware piece that can not be excluded and a omni-directional projector that is able to provide feedback using ground projection (see figure 17). Although we use a Kinect in this set-up the same results can be achieved using similar hardware. The only requirement is the capability to detect humans and to make a good estimate of their positions.

2.2.1 The Kinect

As mentioned in section 1.1.2, occlusions can lead to lower detection rates. Therefore we try to overcome occlusions by putting the camera at a higher level. Using a bird-eye view, less occlusions will occur (see figure 18). This however is not a scalable solution and does not always has the expected results. Children for instance will be completely occluded when they are located behind an adult. Nevertheless this set-up is sufficient for our proof

Figure 17: The fish-eye projector (a) and the Kinect (b)

of concept. For real world usage a better set-up is needed. A good example is the set-up used by Marquardt et al. [22] that uses ceiling mounted Kinects that look down to the ground. This results in a reduced number of occlusions.

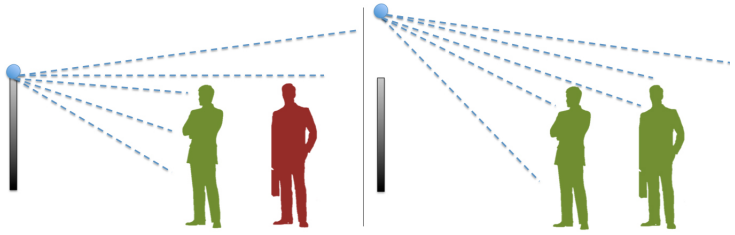

Figure 18: Our approach to handle occlusions in a non-software manner by placing the Kinect in a bird-eye view (right) instead of horizontal faced camera (left).

2.2.2 Fish-eye projector for feedback

Additionally we added a fish eye projector (figure 17a) that provides passing people with feedback on the ground. This only is an optional piece of hardware that we only used for our demo application.

2.3 The architecture of the library

The library that was developed to enable group detection and tracking can be viewed as the middleman between the Kinect for Windows API and the application that makes use of the library. First we will describe the the global architecture of our library followed by the connection between the library, the Kinect API and third-party applications.

2.3.1 Global overview of the architecture

First of all our library subscribe to events of the Kinect api. Secondly the applications using our library need to subscribe to event of the library. Our group detection library can therefore be seen as the middle man in between the group-aware application and the Kinect API providing higher level data then the original data provided by the Kinect API (see figure 19).

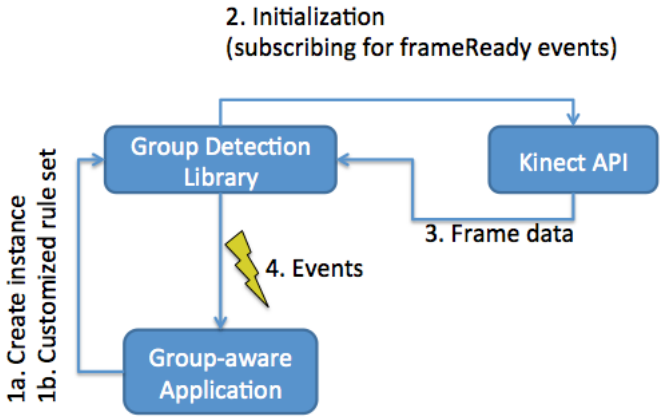

Figure 19: The application's architecture consists of three main "actors": The Kinect for Windows API, our Group Detection Library and the Group-aware Application that uses the events triggered by our library.

The visual representation of how our library works in figure 19 shows the 4 main steps that happen during the process:

1. Create an instance of the group detection library object and insert your customized rule set (see section 3).

2. Then the library will enable the Kinect and subscribe to its frameReady event.

3. During runtime the Kinect API will send frameReady events to the library that in turn processes these events.

4. The library triggers events to which the group-aware application can subscribe.

Now that the hardware set-up and global architecture of our library, we will describe the relation between our library, the Kinect API and the applications that will use our library.

2.3.2 Relation between Kinect API and our library

The Kinect for Windows API itself captures positional data at an estimated frame rate of 30 fps. This positional data contains the exact positions of detected humans in front of the Kinect sensor. Note that the Kinect API is only capable to detect and track six people at the same time. Our library uses this data to extract useful data with regard to detection and tracking of groups, like the position of each detected person. Besides the data for group detection, the library also stores information about detected individuals. To get a clear view of the information that is stored by the library we refer to section 2.3.3. This way the library can both be used to create proxemic aware

applications intended for individuals, as well as group-aware applications. First we will describe the events to which our library subscribes followed by how we extract and manage the data that we need for the group detection process.

Event subscriptions

Before we start the Kinect we enable the *skeleton stream*. The skeleton stream enables us to detect and track individuals by providing accurate positional data each frame. Next in the initialisation process we subscribe to the *SkeletonFrameReady* event of the Kinect Sensor that triggers a function each time a skeleton frame containing all skeleton data is ready. The function that handles these frames will be described in the next paragraph.

Data extraction and management

In this paragraph we will describe the function that extracts and stores the information that can be found in the "Skeleton" objects. Each Skeleton object contains the position (x,y,z) of the human associated with the skeleton. Because we only need to know the position and not the position of the joints (see figure 20) which can only be achieved for two persons using Kinect v1, Kinect v1 is a sufficient solution. If joints were needed the new Kinect v2 would be necessary as it is able to fully track the joints of six people.

First of all we extract the user identification number of each skeleton. This number is a random unique tracking number that the Kinect uses to

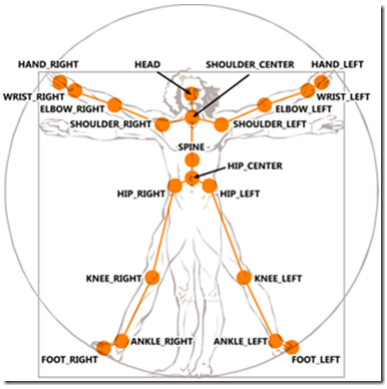

Figure 20: Kinect v1 is able to fully track the joints of two people but can also track body of four other people. As we only need the position because it is sufficient for our library Kinect v1 is sufficient.

identify the detected people. Using this numbers we check if a person is lost, found or tracked (previously found). If a person is found for the first time it is registered in a Human array that contains all detected persons. If it was already detected in a previous frame the skeleton is updated for the appropriate person in the afore mentioned human array. If a person is lost the associated Human object is deleted, this also means that if a human is occluded for one frame its object is deleted and a new object has to be created if it is detected again. There are possibilities to overcome this flaw, but it is not integrated in the Kinect API (see section 1.3). Because the Kinect API does not give specific tracking information per user, we are not able to apply such a technique to our solution. If a user is occluded for one frame the Kinect will automatically assign a new tracking ID to the "new" skeleton.

The position we get for each detected skeleton consists of an (x,y,z) coordinate. Because the z-coordinate is the distance from the mounted

Kinect we need to recalculate the z index as it is not correct due to its mounted position. This is achieved by applying Pythagoras. Using the z index of each skeleton provided by the Kinect API, we can recalculate the distance as in figure 21 using following formula with "adist" as the *actual distance*, "pdist" as the distance perceived by the Kinect camera, "kheight" as the height at which the Kinect is set up and "jheight" as the y index of the recognized humanoid which is provided by the Kinect API:

$$adist = \sqrt{pdist^2 - (kheight - jheight)^2} \tag{1}$$

Now that we have the actual position of each person in front of the screen we are able to derive the data that is needed to check the rules described in section 3.

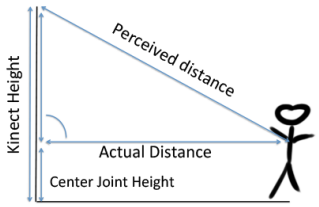

Figure 21: Calculating the actual distance using Pythagoras.

Calculations in between subsequent frames

In the previous paragraph we described how we store the raw data for each detected human in a human array. To check if a rule applies we need to calculate different measures in between subsequent frames. The calculations

are basic mathematical problems that do not cause the library to lower the original frame rate of the Kinect. The algorithm iterates each possible pair of humans. To enable a developer to change his specific rule set during runtime all calculations are performed for every new frame. If this was not the case not all rules would be correctly calculated (e.g. the initial direction of a person is calculated 1 second after the first detection). For more details about the specific calculations for each rule we refer to section 3.

2.3.3 Relation between our library and third-party applications

Now that we clearly explained the relation between our library and the original Kinect API we will also describe the relation between our library and the application that makes use of our library. First we will describe how to start and end using the library followed by the different possibilities regarding event subscriptions.

Using the library

To use the library a new group detection library object has to be created. This object is then used to initiate any further actions. Then the developer has to initialize the Kinect. When the Kinect is initialized, the developer can subscribe to events (see following paragraph) and create customized rule sets (see section 3) if necessary. To stop the library and in turn stop the Kinect it suffices to call the stopKinect function on the group detection library object.

Event subscriptions

The library works with an event-driven architecture. First of all the library subscribes to events of the Kinect API and secondly it triggers events to which a third-party application can subscribe to. In this section we describe the possibilities with regard to event subscriptions and the data that is offered to the eventhandlers.

There are three events with regard to individuals: when a person is detected, a person moved or disappeared. The events and the data that they offer for the eventhandlers are listed in table 4. These events offer developers the flexibility to create applications that solely handle individuals.

Event	Data
PersonFoundEvent	Human object Location List of other detected humans
PersonLostEvent	Human object Last known location List of other detected humans
PersonMovedEvent	Human object Previous location New location List of other detected humans

Table 4: Events regarding individuals and the data that is offered to the eventhandlers.

Off course these events can easily be combined with the events for groups. There are four different group events that are triggered by the library: detection, merging, splitting and dissolving (see table 5). These group events can

only be triggered if group detection is enabled. Last but certainly not least we offer the possibility to subscribe to the frame ready event (see table 6). This event is triggered when the frame captured by the Kinect is completely processed by our library. Subscribing to this event enables the developer to get a regular update of the situation in front of the Kinect detection area.

Event	Data
GroupFoundEvent	Group object Group ID List of all detected groups
GroupExtendedEvent	Group object Group ID New person ID
GroupReducedEvent	Group object Group ID Human object of the person that left
GroupDissolvedEvent	Group ID Last known location

Table 5: Events regarding groups and the data that is offered to the eventhandlers.

Event	Data
FrameReadyEvent	List of detected humans List of detected groups

Table 6: The frame ready event and the data offered to the eventhandler.

We also created two objects that are passed by the event arguments: the human and group object. A human object that contains all possible information about a human including the skeleton object that is offered to the library by the Kinect API. Secondly we also offer a group object that contains the different members (humans) and the center of gravity and size

of the group (see table 7).

Object	The data stored in these objects
Human	A unique identifier
	The current skeleton
	The previous skeleton
	The first detected skeleton
	His distance from the screen
	The distance that the human already covered
	His average speed
	A group ID if associated with a group
	His initial direction
Group	The group ID
	A list of members
	The center of gravity

Table 7: The objects that the library uses.

Using the previously described events and objects it is possible to create various proxemic-aware applications, not only using individual detection and tracking, but also using group detection and tracking.

2.4 Comparison with other proxemic toolkits

In section 1.5 we alread referred to a number of toolkits that also offer developers a library/toolkit to facilitate the design and developing process of proxemic-aware applications. Our application offers a few of the same capabilities but adds the extra feature to enable group detection. To provide an organized view of all the present libraries we redraw table 3 and add our library in table 8.

As you may notice the GroupTogether toolkit [22] by Marquardt et al. is also able to detect small group structures. They do this by using f-formations

	cross-device comm.	device-device aware.	gesture	speech	distance	orientation	identity	motion	multi-user	group detection
XDKinect [36]	X	X	X	X	X	-	X	-	X	-
Kinected Browser [38]	-	-	X	X	-	-	X	-	X	-
jQMultiTouch [39]	-	-	X	-	-	-	X	-	X	-
Shared Substance [40]	X	X	X	-	-	-	X	-	X	-
GroupTogether [22]	X	X	-	-	X	X	X	-	X	X
Interact. Ambient Disp. [5]	-	-	X	-	X	X	X	X	X	-
Proximity Toolkit [35]	X	X	-	-	X	X	X	X	X	-
Proxemic Media Player [41]	-	X	-	-	X	X	X	X	X	-
Group detection library	-	-	-	-	X	-	X	X	X	X

Table 8: Comparison between different proximity aware toolkits provided by Nebeling et al. [36] including our library.

as a starting point. Their method also suggests a set of rules to determine if two or more people are grouped together in an f-formation (see section 1.2.3). As group detection is our main purpose we do not include cross-device communication and neither provide device-device awareness. Nevertheless our method to detect groups can be easily adopted by the other methods as we only need the position of each human for each frame. Other features we do not support are gesture and speech recognition. We also do not register the orientation of detected people as it is not accurately detectable from a sideways view. The GroupTogether toolkit on the other hand is able to calculate the orientation because they use a ceiling mounted Kinect enabling them to see the position of the head on the body. We just used the regular Kinect API because it is easier to develop our test library instead of using the raw data captured by the Kinect and process this data. This raw data

processing is necessary when using an top-down view because this viewpoint is not supported by the Kinect API.

3 The rule set

In this section we will describe the key feature of this work: the rule set. The rule set is an adjustable set of rules that checks if two or more people belong to the same group. First we will define the rules separately with the needed calculations included, followed by a description of the usage of the rules.

3.1 Definition of the rules

First an explanation of all the different rules is provided in a clear overview to prevent ambiguities in the following subsections. Evaluation of the rules is done in section 4 where we evaluate the rules using nine different scenarios. Following rules are implemented:

- Interpersonal distance (cm)
- Maximum difference in distance from the screen/camera (cm)
- Maximum difference in speed (m/s)
- Maximum difference in initial direction (degrees)
- Maximum difference in current direction (degrees)
- Maximum difference in arrival time (seconds)
- Time that the similarity occurs (seconds)

Note that the represented rules are derived from the techniques described in section 1.2.3.

Interpersonal Distance

The most obvious parameter is the distance between two persons. Most of the work regarding group detection uses the interpersonal distance in combination with other factors to check if two or more people belong to each other. This distance is calculated using the center of each person's skeleton. To choose the most appropriate threshold for this rule one should keep the findings by Hall [42] in mind. In his study he found that the interpersonal space can be classified into four different distances: the Intimate zone (<45cm), personal distance (35cm - 1.21m), social distance (1.21cm - 3m), public distance (>3m). Critics, however, argue that the geographical relationship of these results is not taken into account [43]. For example they argue that people in Europe are far more intimate while talking than Americans. Nevertheless these findings are still commonly used in the HCI community although they are sometimes modified to enhance results [44].

McPhail and Wohlstein [30] also used the notion distance in combination with speed and direction. They note that during their study an interpersonal distance of 2.10 meters was ideal to separate a crowd into groups. Keeping Hall's classification in mind this seems a rather big distance and would mean that people at a social distance are related. We argue that a smaller distance like Hall's personal space could yield better results because 2.10 meters would mean that a person in the center of our detection region would always be connected to any other passer-by, resulting in false positives. Note that our detection region is rather small because we only use one Kinect sensor. The region can possibly be enlarged by using multiple Kinects. Azad et al. [19] also found that the interpersonal distance of group members decreased

if the area got more crowded and vice versa. Another observation they did during their study was that there always remained a buffer zone between two groups. A static threshold for the interpersonal distance therefore is unlikely to work.

Maximum difference in screen distance

Another distance-related rule calculates the distance from each user to the camera, and if the screen is located underneath the camera, as in our setup, to the screen (see figure 22).

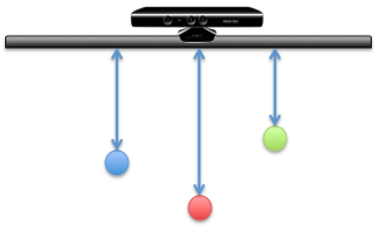

Figure 22: The blue arrows denote the distance from the screen. Individuals are indicated with the coloured dots.

We argue that the distance to screen is a good measure to identify related people because a group of people is most likely to stand close to each other at almost the same distance. Also if they face the screen they are likely to stand in a line at approximately the same distance from the screen.

Maximum difference in avarage speed

This rule enables us to identify two people that walk at the same pace, if that is the case there is a chance that they are related.

McPhail and Wohlstein [30] also use speed, in combination with interpersonal distance and travelling direction, to check if two or more people are related. They consider people to be related if their speed is similar using a threshold of 0.5 ft/s or 0.15 m/s.

Our implementation which keeps track of each individual's speed (m/s) can use these speeds in a rule set. To calculate the speed of an individual we rely on the tracking capabilities of the Kinect API. Each frame we calculate the distance between the new frame's skeleton and the previous skeleton using the center joint x-z coordinates. Adding this distance to the total distance covered by the human we can calculate the avarage speed using the time since first detection. The rule is met when the differences between two people's paces are below a set threshold.

Maximum difference in intitial direction

Another factor we included is the difference in initial direction. Note that this is not the same as the current direction described in section 1.2.1. We calculate the initial direction using the first two skeletons that are detected of each user and we never recalculate this value. The reason why we both use initial direction and current direction is when a group stops walking and someone turns around to face the group his current direction changes. As a result the person will be removed from the group because he has a different direction. As a result two groups coming from different directions will never merge if this rule is enabled.

A similar rule could be implemented by using the orientation. Note that orientation and direction are different things. Orientation refers to the gaze direction whereas direction refers to the walking direction. It is possible that a person has a different orientation and direction, for instance if the person walks backward. Marquardt et al. [22] register the orientation by analysing the posture of a detected human (head relative to the body). Deducing the orientation is impossible to achieve using the basic Kinect API. The API does not provide sufficient data like head and body positions to calculate the orientation. We therefore only consider the direction that is calculated during the arrival and the current direction.

Mazzon et al. [30] use the direction to extend the original social force model group detection technique. Using the direction they were able to deal with group crossing (e.g. two people walking to the north, one individual walks in between the two people to the south). Normally the two groups would split because of the crossing individual releases their group binding force, but by including the direction into the calculations this individual has no effect on the group binding force.

Maximum difference in current direction

We also implemented another direction related rule. We chose this rule based on the findings by McPhail and Wohlstein [16]. They also consider the current direction, meaning changes occur which is not the case for our initial direction. McPhail and Wohlstein consider two people that travel in the same direction to within 3 degrees to be members of the same group. We recalculate the current direction for each frame to respond to every change.

Maximum difference in arrival time

This rule makes it possible to distinguish groups or individuals, that arrived at different times. A threshold (in seconds) can be set. This threshold represents the maximum difference in arrival time. If the difference exceeds the threshold two individuals will not be addressed to the same group. We did not find any reference of this rule being used in related work, but we argue that it can be a good rule to divide a crowd into groups. Azad et al.[19] noted that group gatherings might be asynchronously, meaning that people may arrive at different times. We argue that this rule can be necessary to easily exclude passers by or divide groups arriving at different times (see section 4).

Minimum time that the similarity occurs

This rule is a bit special in that it can relate to all of the above rules. If a rule set is created, this rule checks if the complete rule set is met for a minimum time interval. Or in other words: the minimum time that the similarity occurs.

Mazzon et al. [30] also propose a similar rule as an extension to the inverted social force model proposed by Šochman and Hog [29]. They waited to add a person to a group for a couple of frames, preventing passers-by to be falsely linked to a group.

We argue that this rule is useful in that it prevents the system to make (false) assumptions too quickly (e.g. including a passer-by in a group). This rule has a downside as well. Due to the threshold, groups will only be detected after the time period of the threshold is exceeded. This causes the immediate detection capabilities of the library to be delayed.

3.2 Using the rule set to enable group detection

To provide a flexible library we enable developers to create customized rule sets for their applications (e.g. to only use interpersonal distance as a group detection rule). In this section we describe the different possibilities with regard to the creation of rule sets. During the creation process of a rule set the developer needs to iterate a few steps:

1. Select the rules that are needed to detect groups, adapted to the application's purposes.
2. Choose thresholds for each rule.
3. Create the rule set by combining rules and:
   (a) by choosing the boolean operations
   (b) or by adding weights to check if rule set complies with a static threshold
4. Evaluate the created rule set

In our rule set example we give an example of this process (section 3.2.4). Note that the rule sets can be combined in every possible way (e.g. rule set A and B can successfully be combined with every boolean operator). Off course the developer should be cautious not to produce contradictory

statements. In the following sections we describe the previously enumerated steps followed by an example.

3.2.1 Choosing the rules that are needed to construct your customized rule set

Before you can start constructing a customized rule set, a developer needs to choose the rules he would like to include in this rule set. As we discuss in section 4 the chosen rules heavily depend on the intented purpose of the application. A good example: if the application is intended to group more or less, stationary people. In this case it is not recommended to use the speed rule, knowing that the average speed of each individual gradually decreases over time. As a result all people in the detection region will be merged over time if only the speed rule is used to create the rule set. Creating an appropriate rule set needs evaluation during the development process. In section 4 we evaluate all rules individually for nine different scenarios. An evaluation for every possible combination is impossible to achieve because of the great number of possibilities. Additionally each rule can have a custom threshold, resulting in an even higher number of possibilities.

3.2.2 Setting the rule thresholds and getting the rule values

Using the values of the thresholds way we are able not only to offer yes/no rules, but are also able to return a percentage (e.g. What is the chance that two individuals are related?). The thresholds the are checked by each rule can be modified to the specific applications (e.g. crowded areas might need a smaller interpersonal distance threshold than non-crowded environments). These thresholds also have an effect on the returned percentages. If, for

instance, the speed rule threshold is set to 1 m/s it means that the returned percentage of the rule is calculated with the range 0 m/s to 1 m/s. If the threshold is then changed to 2 m/s the range broadens from 1 m/s to 2 m/s and also affects the returned percentage (see table 9).

Minimum (m/s)	Value (m/s)	Threshold (m/s)	Percentage
0	0.12	1	88%
0	0.12	2	94%
0	1.20	1	0%
0	1.20	2	40%

Table 9: How threshold changes affect the returned percentage of the rules (e.g. the speed rule).

All the rules presented in section 3 have a threshold property that can be set manually using the properties of the library listed in table 10.

Rule	Property
Interpersonal distance (cm)	double interpersonalDistanceThreshold
Distance from screen (cm)	double distanceFromScreenThreshold
Speed (m/s)	double speedThreshold
Arrival time (seconds)	int arrivalTimeThreshold
Initial direction (degrees)	int initialDirectionThreshold
Current direction (degrees)	int currentDirectionThreshold
Rule time threshold (seconds)	int ruleTimeThreshold

Table 10: The threshold properties of the rules.

The library enables developers to create rule sets themselves. This also means they have to be able to get the measured values of the rules to check if they comply to their desired threshold. Developers have to get these values themselves to construct the rule set (see section 3.2.4 and our demo application in section 5.2). If necessary the developer is also able to get the certainty percentage (e.g. the interpersonal distance threshold is set to 100

cm, and a pair is positioned 80 cm from each other the percentage will be 20%, if they would be at 30 cm from each other the percentage would be 70%).

To do this the developer can use the functions listed in table 11. All these functions need the index of the persons that have to be checked as a parameter. The index is needed for the function to get the intended individuals from the human array that is created by our library.

Rule	Function
Interpersonal distance	double IsDistanceOK(int i, int j);
Distance from screen	double IsDistanceFromScreenOK(int i, int j);
Speed	double IsSpeedOK(int i, int j);
Arrival time	double IsArrivalTimeOK(int i, int j);
Initial direction	double IsInitialDirectionOK(int i, int j);
Current direction	double IsCurrentDirectionOK(int i, int j);

Table 11: The functions that return the percentage of the rule certainty. Note that "rule time threshold" is not listed because it is checked by the system automatically if it is enabled.

3.2.3 Combining rules to create a customized rule set

Combining rules is the key feature of our method. There are two ways to combine rules: using booleans or using weights. Note that these two methods can be combined. We describe the two methods separately, but after this paragraphs it should be clear that rule sets can be combined. To conclude this section we also describe how a developer can use the percentages of each rule while creating a rule set.

Using boolean operations

The most natural way to see if groups are together can be done using booleans. For example, if two individuals have the same average speed AND initial direction AND in combination with an interpersonal distance that is smaller than a threshold, they are in the same group. In this particular example two AND-operators are used to create a rule set. The rules themselves return true if they are beneath a set threshold (e.g. an interpersonal distance threshold of 100 cm). The library provides a technique to create boolean operations with the rules. All boolean operations are possible, but the developer has to make sure he does not create contradictory statements. Note that it is not only possible to combine rules using boolean operations, but that it is also possible to combine rule sets. This also counts for the following method to create rule sets, using weights.

Using weights in combination with mathematical operations

The second possible method offers the possibility to add weights to each individual rule. This might be needed to indicate more important rules. A rule set using weights has a different structure than a rule set using booleans. Instead of using boolean operations the developer has to use mathematical operations. After the calculation of the rule set statement the result has to be compared with a predefined threshold. The example in the section should make this method clear.

Using the percentages that are returned from the rules' checking functions

As listed in table 11 the rules' checking functions, that check if two people comply to the rule, return a percentage. This percentage can also be used to check if a certain rule exceeds a set percentage. This can be done for every percentage individually. Subsequently, this rule can be inserted in a rule set. (e.g. interpersonal distance >20% =>true).
Now that we described the three main building blocks it should be clear that all methods can be used interchangeably. In the following rule set example we only use the method using weights. In our demo application we use boolean operations (see section 5).

3.2.4 Rule set example

A good rule set is customized for the destined application because all rules act different in different situations (see section 4). Therefore the developer that uses the library should consider all possible situations that have to be detected, before constructing the rule set.

Example application

A store manager want his customers that are queueing to pay their goods to know how long they will have to wait. To make an estimate about the remaining waiting time he uses the number of people in the queue. He argues that a better estimate can be made when groups are detected. They can be seen as an individual with regard to the handling time. He asks the IT manager if he is able to develop an application that is capable to detect

groups and accordingly tries to make an estimate of the remaining waiting time.

Choosing the rules for the rule set

The first question the developer has to answer is how he would define and detect a group. In this situation groups and individuals can be separated using 5 different decisive rules: interpersonal distance, distance from screen, arrival time, initial direction and the similarity threshold. Interpersonal distance and distance from the screen are good rules in this situation because groups are not likely to blend in a queue. The other two rules are also useful as groups usually come from the same direction at the same time, but these are less decisive. We also use the similarity threshold rule because passers-by can then by excluded. We do not use the average speed and current direction in this rule set. The average speed of decreases to 0 as everybody stands still so this rule can not be used. Same happens with the current direction that is the same for everybody as they are standing in line.

Secondly the developer is able to add weights to the rules to indicate which rules are more decisive than others. In this case we can conclude that interpersonal distance and distance from screen are more decisive than arrival time and direction because people might still by wandering around while others are already in the waiting line. Therefore we can add smaller weights to the direction and arrival time rules and higher weights to the two distance rules. The developer might have to alter the weights during the test process to raise the effectiveness of the application.

Creating the rule set

Using the 5 rules chosen in the previous step we can create a rule set with weights customized for the application. An example rule set might look like the following:

$$result = 0.3\alpha + 0.3\beta + 0.2\gamma + 0.2\delta \tag{2}$$

Where α represents the interpersonal distance, β is the distance from screen, γ is the initial direction and δ is the arrival time rule. The result of this rule set can be compared with a group detection threshold to exclude false detection. If the result is higher than a threshold a group is detected. Note that we do not include the similarity threshold rule. The reason why is that it can not be included in a rule set. It is just a threshold that has to be exceeded, before the rule set results count.

Inserting the rule set into the library

Now that the rule set is created it still has to be inserted into the algorithm that calculates all measures and checks if groups are formed. To do this the developer has to create a delegate for the original AreParametersOK function of the library. This can be represented with the following (pseudo-)code:

```
//Set the rule time threshold that prevents the formation of
    groups for a couple of seconds
ruleTimeThreshold = 3 seconds

//Check each rule
```

```
bool a = Is the interpersonal distance less than the set
    threshold
bool b = Is the distance from screen similar
bool c = Is the initial direction similar
bool d = Is the arrival time similar

//Add weights to the booleans
double result = 0.3*a + 0.3*b + 0.2*c + 0.2*d;
//Set the threshold
double threshold = 0.6;
//if the result is higher than the set threshold return true
return result > threshold and the ruleTimeThreshold is exceeded;
```

After coding the AreParametersOK replacement function the developer still has to insert it into the library by setting the delegate. Now that the group detection rule set is added to the library it can be successfully used for the purpose intended.

In the previous chapter we described the steps how a rule set can be created. The described technique offers improved flexibility towards the developer that uses the library. Rule sets can also be adapted during runtime (e.g. to cope with varying crowd densities).

4 Evaluating the chosen rules using different scenarios

To identify the pros and cons of the rules described in the previous section we did a pilot test by recording nine different situations that might appear in front of the Kinect camera.

4.1 Methodology

The scenarios described in the following paragraphs were recorded using Kinect Studio[4]. This tool enabled us to exactly rerun each scenario multiple times to test each rule with the exact same captured data. First we describe what happens for each scenario before we note the remarkable findings with regard to the rules described in the previous section. We also provide the error rate for each rule, for 1 to 3 different thresholds. This error rate was calculated by using a toggle button for the observer. Every time an error occurred, the observer had to click a button. If the error was gone the button was toggled notifying the system that everything was good. The error rate is calculated after the situation ended, using the time everything was right and the time an error was noticed. Note that this error rate has strange values for some rules, but when reading the explanation these values will make sense. All the scenarios are also illustrated for better understanding. At the end of this section we note our overall findings.

[4] http://msdn.microsoft.com/en-us/library/hh855389.aspx

4.2 Scenarios

In this section we describe each scenario that we recorded using the Kinect Studio tool. After we described the context we note our findings with regard to each rule.

Scenario 1 - The wanderer

The first scenario (illustrated in figure 23) is built around a wanderer that is positioned in the center of the detection region in front of the Kinect. While he stays wandering around one person passes by in front of him and later on a group of two people passes by behind him in the other direction.

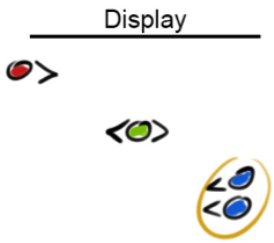

Figure 23: Scenario 1 - While one person stands in the middle of the detection region. One person passes by in front of him and a group of two people passes by behind him.

During this scenario we noticed that the arrival time and direction were good rules to keep the three "groups" separated. The distance from screen also worked good in this scenario because the groups walked a different distances from the screen. Here we also first noticed that the similarity time threshold also worked. Passers-by did not merge with another group because they were only similar for a time smaller than the time threshold. Interper-

sonal distance did not work as an individual rule because it automatically caused the wanderer in the middle to merge with the passing individual and the group. Speed would be a good rule in this scenario if no occlusions would occur but due to the wanderer in the middle the group that passes by behind him is occluded for a split second, causing the calculated pace to be incorrect. The current direction is not a good rule because it recalculates the direction for every frame. This results in "flickering" if people "stand still", meaning that the direction degree changes quickly due to small changes of a human's position.

Rule	Threshold	Error rate
Interpersonal distance (cm)	45	22,22%
	100	6,25%
	121	11,11%
Time (seconds)	1	17,65%
	2	29,41%
	3	16,67%
Distance from screen (cm)	0.7	0%
	1.1	26,67%
Arrival Time (seconds)	2	12,5%
	4	12,5%
	6	12,5%
Speed (m/s)	0.04	85,71%
	0.08	6,25%
	0.12	5,88%
Initial direction (degrees)	40	5,88%
Current direction (degrees)	40	100%

Table 12: Error rates for each rule for scenario 1

Scenario 2 - Merge and split

The second scenario (illustrated in figure 24) describes the situation where four people gather in front of a display. First a group of two and an individual merge into one group. Then a third individual comes up to join the group. At the end two people leave the group which leads to a split.

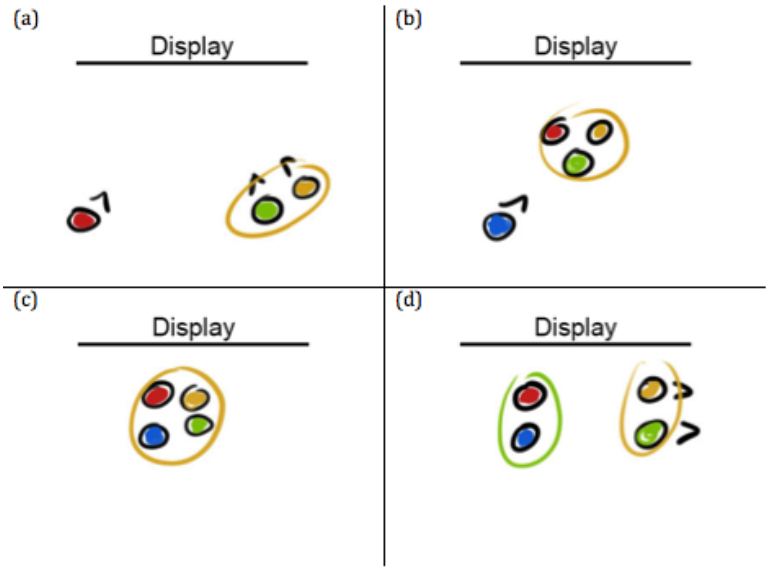

Figure 24: Scenario 2 - First a group of two and an individual merge into one group. Then a third individual comes up to join the group. At the end two people leave the group which leads to a split.

Interpersonal distance is a great rule for this situation because it can instantly merge individuals. But this scenario does not consider any passers-by that would be included if they crossed the interpersonal threshold border of one of the group members. Distance from screen is not a good rule in this

situation because the people that form the group are not aligned. This way they are not all merged into a group. Initial direction and arrival are no good rules either because the members of the group come from different directions at different times. The current direction changes to often, resulting in false groupings that change quickly.

Rule	Threshold	Error rate
Interpersonal distance (cm)	45	100%
	100	20%
	121	4,76%
Time (seconds)	1	18,18%
	2	14,29%
	3	25%
Distance from screen (cm)	O.7	9,09%
	1.1	21,74%
Arrival Time (seconds)	2	59,09%
	4	58,33%
	6	60,87%
Speed (m/s)	0.04	75%
	0.08	28,57%
	0.12	9,09%
Initial direction (degrees)	40	54,17%
Current direction (degrees)	40	100%

Table 13: Error rates for each rule for scenario 2

Scenario 3 - A mix of groups

The scenario illustrated in figure 25 consists of two groups of two and three people that arrive at a similar time in front of a public display. During "interaction" with the display the groups mix, but do not merge.

This scenario is a great example to show when the initial direction rule

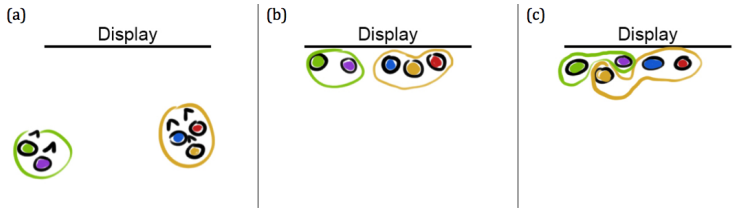

Figure 25: Scenario 3 - A mix of two groups

can keep two groups separated. Because the two groups mix during interaction interpersonal distance and distance from screen are no good rules. Same with the arrival time and speed. Because both groups arrive at a similar time with a similar speed they would be merged if only arrival time or speed rules would be used. The current direction rule results do not suffice.

Rule	Threshold	Error rate
Interpersonal distance (cm)	45	75%
	100	75%
	121	67,86%
Time (seconds)	1	80%
	2	80%
	3	68%
Distance from screen (cm)	O.7	62,96%
	1.1	62,96%
Arrival Time (seconds)	2	0%
	4	0%
	6	0%
Speed (m/s)	0.04	88,89%
	0.08	76,92%
	0.12	82,14%
Initial direction (degrees)	40	0%
Current direction (degrees)	40	100%

Table 14: Error rates for each rule for scenario 3

Scenario 4 - Three individuals

As a starting point for this scenario one person was located in front of a public display. After a while two people arrive from the same location although they do not belong to each other. One person approaches the screen while the other just passes by. The situation is illustrated in figure 26.

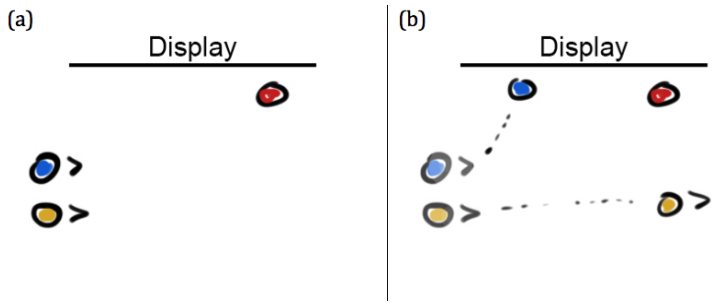

Figure 26: Scenario 4 - Three individuals

The interpersonal distance is not a good rule on itself for this scenario because it groups people to quickly. The similarity time threshold in combination with any other rule works great in this scenario because the actual grouping only occurs after a time threshold is exceeded that the similarity occurred. Distance from the screen is not a good rule because the two individuals in the front are at the exact same distance from the screen. The two individuals that arrive at the same time and come from the same direction are immediately grouped if their respective rules are enabled. Therefore a combination of rules is necessary in this situation. The current direction rule results are not good.

Rule	Threshold	Error rate
Interpersonal distance (cm)	45	0%
	100	37,5%
	121	44,44%
Time (seconds)	1	37,5%
	2	25%
	3	20%
Distance from screen (cm)	O.7	44,44%
	1.1	44,44%
Arrival Time (seconds)	2	11,11%
	4	11,11%
	6	50%
Speed (m/s)	0.04	8,33%
	0.08	11,11%
	0.12	27,27%
Initial direction (degrees)	40	25%
Current direction (degrees)	40	100%

Table 15: Error rates for each rule for scenario 4

Scenario 5 - Two groups, different directions

This scenario contains two groups of two individuals that approach the screen from different directions at a different time (see figure 27).

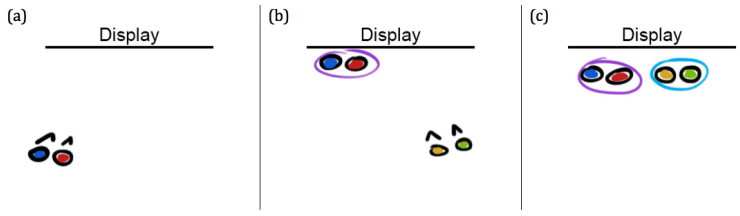

Figure 27: Scenario 5 - Two groups that come from different directions at a different time.

The interpersonal arriving time works good when the two groups are far

apart, but when the second group aligns with the first group the two groups are merged because the two individuals in the middle are close enough for the interpersonal distance rule to be met. The same thing happens for the distance from the screen rule. Initial direction and arrival time both work great in this example because the groups arrive at a different time from a different direction. The same thing seems to be occurring for the speed rule but due to the two groups standing still their average speeds gradually come together, resulting in false groupings. The current direction rule is not useful because the values constantly changes due to small changes in the position of each human. This causes "uncontrolled" group splitting and merging.

Rule	Threshold	Error rate
Interpersonal distance (cm)	45	79,17%
	100	23,53%
	121	47,83%
Time (seconds)	1	78,95%
	2	72,73%
	3	69,57%
Distance from screen (cm)	O.7	55,17%
	1.1	56,67%
Arrival Time (seconds)	2	0%
	4	0%
	6	0%
Speed (m/s)	0.04	100%
	0.08	75%
	0.12	66,66%
Initial direction (degrees)	40	0%
Current direction (degrees)	40	1OO%

Table 16: Error rates for each rule for scenario 5

Scenario 6 - Two groups, same direction

This scenario contains two groups of two individuals that approach the screen from different directions at a different time (see figure 28).

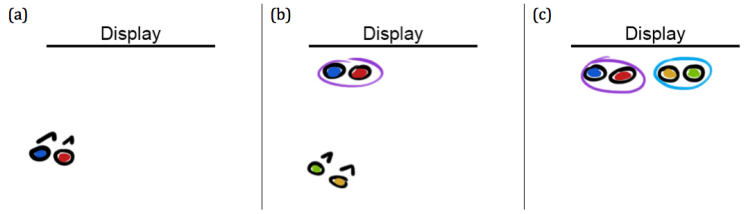

Figure 28: Scenario 6 - Two groups that come from the same direction at a different time.

During this scenario we almost made similar observations as in the previous situation. The only difference here is that the direction rule no longer works. When the second group arrives from the same direction they are immediately merged with the first group. The arrival time rule is the only rule resulting in correct assumptions.

Rule	Threshold	Error rate
Interpersonal distance (cm)	45	68%
	100	37,5%
	121	41,67%
Time (seconds)	1	52,17%
	2	48%
	3	46,15%
Distance from screen (cm)	O.7	40%
	1.1	42,31%
Arrival Time (seconds)	2	4,17%
	4	0%
	6	33,33%
Speed (m/s)	0.04	66,67%
	0.08	69,57%
	0.12	66,67%
Initial direction (degrees)	40	12,5%
Current direction (degrees)	40	100%

Table 17: Error rates for each rule for scenario 6

Scenario 7 - A group formation and a passer-by

A group of two and an individual gather in front of the Kinect. After a while a passer-by passes in front of the Kinect behind the group (see figure 29).

Interpersonal distance works great as a rule in this scenario because the passer-by walks by at a larger distance than the set threshold. The same counts for the distance from screen rule. All the members of the group are positioned at approximately the same distance from the screen resulting in a grouping. Arrival time also works great in this scenario because all the group members arrive at the same time, the passer-by passes by after the threshold is already exceeded and is thereby not included in the group.

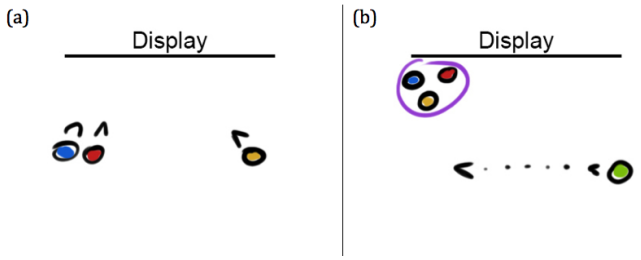

Figure 29: Scenario 7 - A group formation and an individual passes behind the group.

Rule	Threshold	Error rate
Interpersonal distance (cm)	45	100%
	100	0%
	121	0%
Time (seconds)	1	56,52%
	2	50%
	3	46,43%
Distance from screen (cm)	O.7	0%
	1.1	0%
Arrival Time (seconds)	2	79,17%
	4	0%
	6	0%
Speed (m/s)	0.04	91,3%
	0.08	21,74%
	0.12	5%
Initial direction (degrees)	40	66,67%
Current direction (degrees)	40	100%

Table 18: Error rates for each rule for scenario 7

Scenario 8 - The crossing

This scenario consists of two groups that cross each other in front of the Kinect camera (see figure 30).

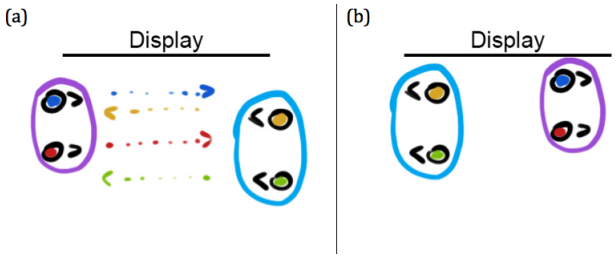

Figure 30: Scenario 8 - Two groups cross each other.

The only valid rule in this scenario is the direction rule as it correctly groups the individuals coming from the same direction. Both groups cross each other resulting in false assumptions for all the other rules.

Rule	Threshold	Error rate
Interpersonal distance (cm)	45	80%
	100	80%
	121	66,67%
Time (seconds)	1	66,67%
	2	40%
	3	80%
Distance from screen (cm)	O.7	100%
	1.1	100%
Arrival Time (seconds)	2	100%
	4	100%
	6	100%
Speed (m/s)	0.04	100%
	0.08	100%
	0.12	100%
Initial direction (degrees)	40	66,67%
Current direction (degrees)	40	100%

Table 19: Error rates for each rule for scenario 8

Scenario 9 - Merge and split 2

Scenario 9 is a longer scenario and starts with one individual that positions himself in front of the Kinect camera. After a few seconds a second individual joins the first person and they form a group. Again a few seconds later a third person comes in and joins the group. After a short time period the first person who walked into the detection area leaves the group causing a group split (see figure 31).

Because all the group members arrive at a different time from different direction this scenario can only be correctly found by the interpersonal distance rule. The distance from screen rule seems to work but does not identify the person that leaves the group because this individual keeps walking at

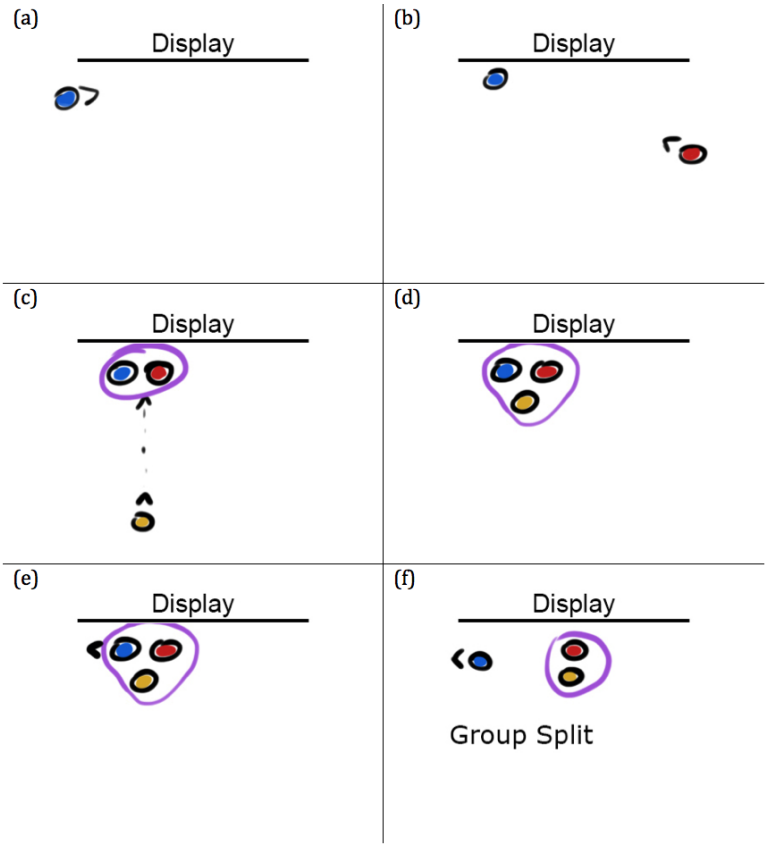

Figure 31: Scenario 9 - Individuals that come frome different direction at different times are merged. In the end the first person leaves the group.

the same distance from the screen. Like in the previous scenario the current direction rule was not useful.

Rule	Threshold	Error rate
Interpersonal distance (cm)	45	40%
	100	10,53%
	121	19,05%
Time (seconds)	1	6,25%
	2	16,67%
	3	17,65%
Distance from screen (cm)	O.7	11,76%
	1.1	10,53%
Arrival Time (seconds)	2	58,82%
	4	61,11%
	6	11,11%
Speed (m/s)	0.04	26,67%
	0.08	12,5%
	0.12	17,65%
Initial direction (degrees)	40	100%
Current direction (degrees)	40	100%

Table 20: Error rates for each rule for scenario 9

4.3 Overall findings

After running through all these different scenarios we can conclude that all rules are useful to cover all the possible scenarios. But when we look at the individual error rates we can conclude that it is necessary to combine several rules to reduce the error rate.

The similarity time threshold rule for instance is a good rule to exclude thresholds just as Mazzon et al. [30] already mentioned in their extension for the inverted social force model technique (see section 1.2.3). Just as the direction rule that keeps groups apart that are walking in different directions. The only rule that was not useful because of the continuous changes of the values, was the current direction rule. It did not suffice in every situation. The interpersonal distance is a great rule keeping in mind that all

other research presented in the related work section uses this rule to group people. Arrival time and speed are related because the speed is calculated using the time that an individual is present in the detection zone. Arrival time in itself is a good measure to group people if the scenario is more likely to consist of groups coming at the same time although this is certainly not always the case (see section 1.2.2). The initial direction rule can be used if the developer wants to group people that come from different direction (e.g. check who comes in and out at a shopping mall exit). We also noticed that the current direction was not a good rule. Due to the fact that people don not really stand still although it seems like that, the Kinect registers small changes. This causes the current direction to change rapidly, resulting in uncontrolled groupings and group splits. This also means that the current direction rule is not useful to include in a rule set.

A combination of the rules can be used to create a strong rule set although the composition of the rule set is heavily dependent on the intended situation. A good combination we found during a few pilot tests used two rules, the interpersonal distance rule and the similarity time threshold. This way it was possible to exclude passers-by. To find the best combination of rules our library offers the flexibility to alter rule sets during runtime, thereby offering the possibility to change the rule set and to find a best fit to the current situation in front of the camera. Off course a thorough evaluation of the rule set should be done to determine the best thresholds and weights for each specific rule. In the following section we try 6 different rule sets that use various combinations (see section 5.2).
The construction of a valuable rule set is a cumbersome task. But using the findings noted in the previous paragraphs the task can be accomplished

faster. Nevertheless, it should be noted that finding the best rule set needs a lot of testing.

5 Demo application - “Whac-a-Mole”

As an example of the usage of our group detection library we developed a small game that relates to the “Whac-A-Mole” arcade game (see figure 32). The game makes use of the library’s capabilities to group people with the similarities defined in the adapted rule set. In this section we describe the application, the implementation that makes use of our library and we finalize this chapter with our findings and results regarding the usage and the applicability of the library.

Figure 32: The original “Whac-a-Mole” arcade game. [45]

5.1 Description

The concept of "Whac-a-Mole" is widely known: try to hit the mole with a hammer when it pops out of the ground. Our game adds a few features. First of all we use groups instead of individuals. This brings us to the second feature which uses the center of gravity of a group as a virtual hammer to knock down the randomly appearing moles. Two groups are able to play against each other by beating the mole as fast a possible before the other group hits the mole with their center of gravity. Detected individuals can be assigned to a group using two different rules: arrival time difference and initial direction (see figure 33). We chose these rules because they do not change after the first detection, thereby the group layouts do not change while playing the game. We also use a combination of the rules to give an example of a rule set.

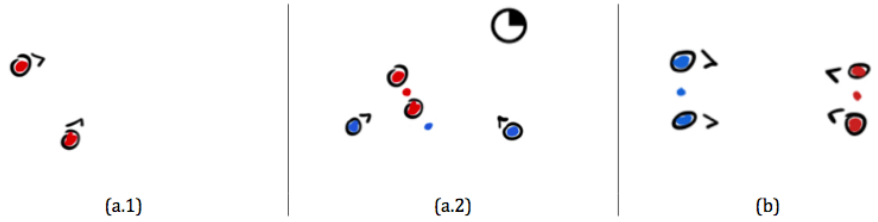

Figure 33: The situation illustrated in (a.1) and (a.2) shows how the arrival time can be used to divide two groups of people. Situation (b) illustrates the use of initial direction to divide groups. The small dots in between two individuals illustrate the center of gravity of each group.

The first assignment method uses the arrival time of the individuals. A group is formed if two or more people arrive within a certain time threshold

(we take 3 seconds but it does not matter in this case). The second group has to arrive after this threshold to be perfectly distinguishable from the other group as is shown in figure 33 (a.2). The second assignment method uses the initial direction rule to divide the crowd into groups. Figure 33 (b) illustrates this method. The third assignment method uses both rules.

The gamescreen is projected on the ground using a fisheye projector (see figure 34). A circle is projected underneath each player and a hammer illustrates the center of gravity for each group that is used as the virtual hammer of the group. Every time a "hammer" hits a mole the group connected to the hammer gets a point. The group that first reaches 5 points wins.

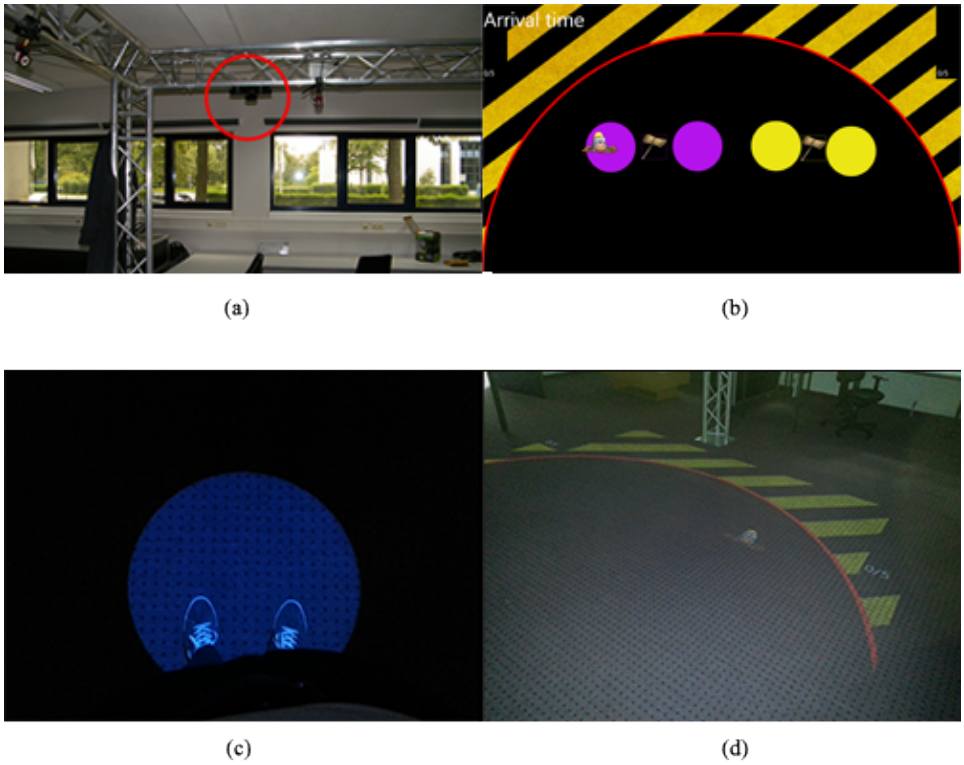

Figure 34: In (a) the Kinect is hanging at about 2 meters capturing what happens on the "game board". (c) shows how a circle is projected underneath a player. (b) and (d) show how the game field is projected on the ground using the fisheye projector. The gamefield is restricted to the yellow and black stripes which is the same as the detection area of the Kinect. The circles are are representations of detected people. Their colors are group dependent. Each group's center of gravity is shown with a hammer in the same color.

5.2 Implementation

The implementation uses the flow described in section 3 to create a rule set. The game described in the previous section uses two rules. First of all the arrival time rule and the initial direction are used and alternatively two rules are combined. We will first describe the three different rule sets using the group detection function delegates. Secondly we describe the event subscription procedure. To finalize this section we provide a short explanation of the data we use for our game.

Building the rule sets

First we describe the delegate function for the arrival time rule. We first set the threshold followed by the actual rule checking function (see pseudo code).

```
first set the difference in arrival time threshold;
if (is arrival time between person i and person j similar?)
  group the two individuals with index i and j;
else
  keep the individuals separated;
```

The second group division technique uses the initial direction rule. Analogous to the arrival time rule we first set the threshold (50 degrees) and subsequently check if the initial direction is similar (see pseudo-code).

```
first set the maximum difference in initial direction threshold;
if (is initial direction between person i and person j similar?)
  group the two individuals with index i and j;
else
  keep the individuals separated;
```

The third group division method uses the two previous methods combined. The pseudo code look like the following.

```
first set the maximum difference in initial direction threshold;
if (is initial direction between person i and person j similar?
     && is arrival time between person i and person j similar?)
  group the two individuals with index i and j;
else
  keep the individuals separated;
```

Using the three disclosed functions we are able to divide groups. Note that the chosen division technique can be changed by changing the delegate for the *AreParametersOK* function of the library. Also note that the

delegate can be changed at runtime by using the *setParametersOKDelegate* function of the library.

Subscribing for the events

Our game makes use of the event-driven architecture of our library. We therefore subscribe to the FrameReadyEvent to regularly get an accurate update of the players positions. The *FrameReadyEventHandler* uses the data provided by *FrameReadyEvent* to draw the visualisation and check if the mole is "whac'ed". This event is triggered when each frame that is captured by the Kinect is processed by our library. This yields for a high rate of events which is necessary for the smoothness of our game's visualisation. The data we use for the game is described in the next paragraph.

Getting additional data for the game

Besides the information needed for the visualisation we also need information to calculate game-related statistics. First we plot the circles that represent the different players on a "map" using a small offset to adjust the position on the floor. We do not use a calibration method because our setup had a very good approximation and a small offset was sufficient. Secondly we plot the center of gravity for each group. The center of gravity is provided by our library for each group in the list of groups that is provided by the FrameReadyEvent. We also use the center of gravity coordinates to check if it hits the mole.

5.3 Result

After developing the game described in the previous sections we had a few findings with regard to the library and a few problems that came up while playing the game. First we note our observations during the initial tests of our game.

Observations during gameplay

When starting the game people immediately notice the halos around their feet (see figure 35). They also noticed that these halos are representations of themselves without any explanation. When it comes to the group merging process they try to create a group but due to occlusions these formations are not maintained throughout the process. One of the major issues with the application is the lack of occlusion handling that causes groups to split during gameplay. At first people try to reform the group but they argue that is interrupts the game itself. Frustration rises if a lot of occlusions occur.

A remarkable comment that was given during a pilot test. A player asked: "How does it come that we are not merged? We are standing close to each other...". This was due to the the fact that only initial direction was used to group individuals.

Using the library

The library itself served as a middle man during the development process. This way the developer only had to deal with high level data that was already processed, resulting in a shorter developing period. Another advantage of

Figure 35: People playing our game

the group detection capabilities of the library is that it enabled us to quickly divide a crowd into groups or in this case playing teams.

Problems

A few problems occurred during the initial tests. A major problem are occlusions. Although we try to overcome the majority of occlusions by lifting the Kinect to a higher viewpoint, it is still not an optimal solution. A better solution is proposed by Marquardt et al. [22] that uses multiple ceiling mounted Kinects. Another problem occurred during gameplay when a person walked out the detection region. This led to group-splitting which in turn led to game disruption. This can be solved by using multiple Kinects to broaden the detection area. Nevertheless our demo application only used one Kinect which is sufficient to serve as a proof of concept.

5.4 Trying other rule sets

To identify valuable combinations with other rules, we did an observation of three user groups that had to play our game (see figure 36). In this section we describe the various rule sets we tested and how well they worked for our game. Note that we used the game to test the rule sets, even though they are not worthy to be used for our game. But it the game was just used as a medium to evaluate the rule sets.

Methodology

The evaluation starts by explaining the intention of the game to the individuals. They are not told that which rules are used to group people. If, however, they did not find a way to form a group, we told them what they had to for the system to recognize them as a group. During the sessions we recorded the various games using a video camera. Using these video recordings we were able to find remarks that we discuss in the following paragraphs. There were three groups of users, two groups of two people and one group of 6 people.

The set of rule sets

We choose rule sets that seemed useful to us. Note that we only used boolean operators to combine the rules.

1. Interpersonal distance (100 cm) AND arrival time (5 s)
2. Interpersonal distance (100 cm) AND arrival time (5 s) AND initial direction (40 degrees)

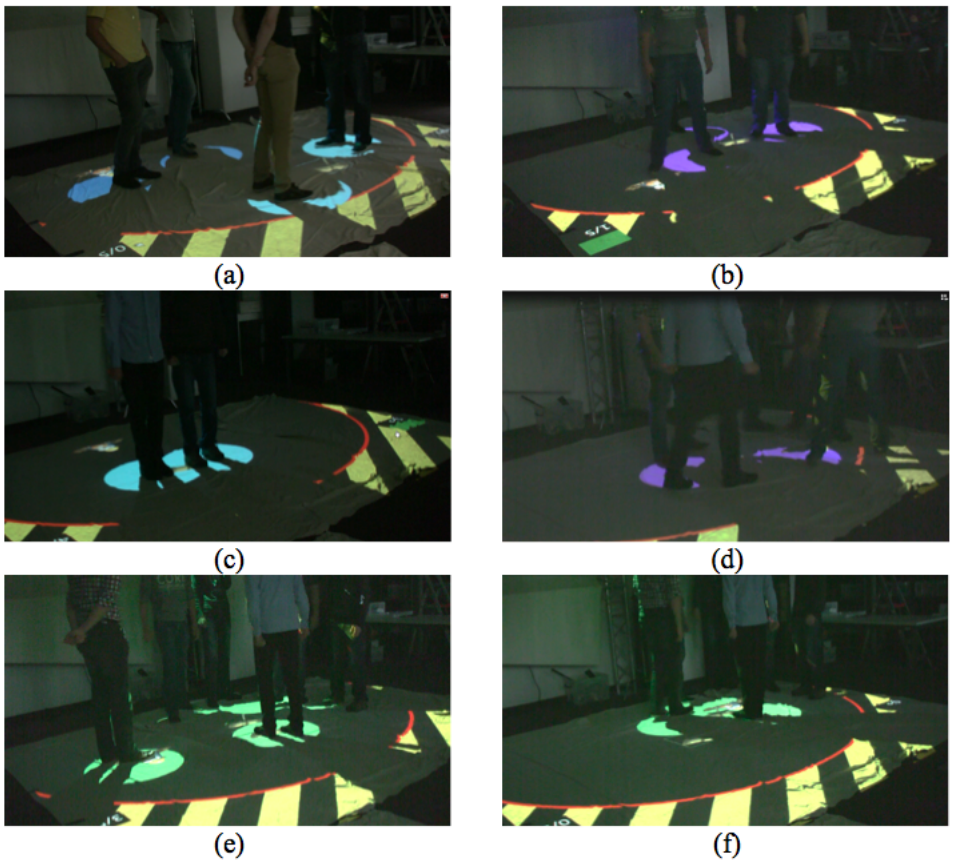

Figure 36: People playing our game. In (a), (b) and (c) all people are detected, whereas in (d),(e) and (f) some people are occluded or not detected. Note that (d), (e) and (f) contains six people. In the other cases less people are positioned in the detection region.

3. Interpersonal distance (121 cm) AND similarity time threshold (3 s)

4. Interpersonal distance (121 cm) AND average speed (0.1 m/s)

5. Arrival time (5 s)

6. Screen difference (40 cm)

The first four rules all have the interpersonal distance rule included. We included this rule because it is the most natural way to group people. Note that the first two interpersonal distance rules have a threshold of 100 cm whereas the other two have an interpersonal distance of 121 cm. We made combinations with the other rules by thinking of real world examples. The second rule set for instance applies if a group comes from the same direction, at the same time and the individuals stand close to each other. The third rule only groups people if they stand close to each other for a specific duration. The last two rule sets consist of one rule.

Overall findings

The users were excited to use the system and initially tried to form a group by standing close to each other. But because we used the interpersonal distance rule in combination with other rules, this approach did not always work. When they were not able to create a group, we told them what they had to do. When they knew what had to be done they were able to form a group, but due to occlusions and bad detection rates of the Kinect itself this was hard. In the beginning they were excited to play the game, but when detection failed numerous times due to our set-up, the users got a bit frustrated. Someone said: "Why aren't we a group? We are standing close to each other... ". This statement indicates the importance of the interpersonal distance rule. People expect to be grouped when standing close to each other. Another person thought he was detected, although their was no halo beneath him. After a short period he knew he only was detected if there was a halo underneath him.

Because we used different thresholds for the interpersonal distance in the first four rule sets, we were able to see how people reacted to this rule. We noticed that the threshold of 100 cm caused the individuals to feel uncomfortable when standing so close to each other. When we used the threshold of 121 cm this was not the case. When we solely used the screen distance rule, we immediately found that a chain reaction occurred, grouping all the individuals in front of the display. Speed seemed to be the worst rule: because people tend to move slowly or even stand still in front of the kinect, the rule merged the individuals together.

We also noticed that our set-up is not scalable for a high crowd density. When using two to four players the system had relatively high detection rates in contrast to the situation where six people where in front of the Kinect (see figure 36 (d-f)). This had two causes. First of all the Kinect does not always detect individuals (e.g because of a strange posture or illumination). And secondly there are a lot of occlusions when more people are positioned in the detection zone. This yields for a different set-up, perhaps the set-up proposed by Marquardt et al. [22] that uses ceiling mounted Kinects to avoid occlusion problems. Another disadvantage of our set-up is the fish-eye projector. Due to an increased number of users the projection of the game on the ground gets distorted and game elements will be invisible (see figure 36 (f)). A better solution could be an interactive illuminated floor.

5.5 Other applications

In this scenario we created a game. This however does not exclude other possible application domains. Using our library can speed up the development process of any application that benefits from group detection or crowd division that can be achieved using a customized rule set (e.g. the retail manager's wish to estimate the waiting time for the checkout queue). It is easy to think of other applications in the game area because there already exist a lot of games that work with teams or "groups".

Coming up with other applications outside the gaming domain is harder because group-aware applications are not omnipresent. However the development of this library might encourage developers to think of other applications that can use the group detection capabilities of our library. During a brainstorming session we for instance had the idea of a group-aware interactive display at the entrance of a restaurant. To automatically inform approaching groups if there is still a table left for their group we could use our group detection library. The library could provide the developers of the display with group information that is needed to check if an appropriate table is available.

Part IV

Conclusion and future work

In this section we summarize the findings of our work and try to give an answer to our research question if it is possible to detect if two or more people belong to the same group using a basic rule set and which rules are necessary to do so. At the end of this chapter we note what can be done in future work to improve our group detection strategy.

6 Conclusion

To achieve the result of this master thesis we did research to find the best rules to detect groups. First we will conclude with the rules in the rule set followed by the resulting library that enables developers to easily develop a group-aware application.

6.1 The rule set

Finding the best rules to be included in the rule set was one of the most important parts of this work. While reading related work about group detection strategies and by adding a few creative ideas we were able to compose a set of rules that can be used to create a rule set. Although we were not able to find one rule set that can be used for every situation, we argue that using a combination of the rules we present can be adapted for almost every situation. In section 4 we did an evaluation of each rule separately for nine different scenarios, resulting in a few overall findings (e.g. the grouping delay (rule similarity threshold) is useful to exclude passers-by). Nevertheless, we were not able to find an optimal rule set, consisting of one or more rules, that is capable to detect all groups without false positives and negatives.

6.2 The group detection library

To enable developers to easily develop group-aware applications, we created a library that has a built-in group detection method. This system uses our rule set strategy. The library we wrote is capable of changing the rule set during runtime offering the developer great flexibility. If a rule set for instance is built for an open non-crowded area and another rule set is built

for the complete opposite scenario the developer is able to swap the rule sets in order to improve group detections. Constructing a rule set is made fairly easy thanks to our library. For each rule a percentage can be calculated, the percentage is the likeability that two people are in the same group. Using the rules a developer can create a rule set by combining several rules and using weights and boolean operations. The rule set can this way be customized for the intended application.

6.3 Answer to our research question

As stated in the previous section it is impossible to create one rule set that is capable of detecting groups in all situations. Nevertheless, we found the most used rules while reading related work and were able to combine them to create a set of rules. These can be used to construct a customized rule set for the intended purpose. Nevertheless it is hard to find the a good rule set and an evaluation process is needed to check if the rule set is sufficient for the intended application. To prove that our concept is able to detect groups we created a game that automatically divided bystanders into groups using their original direction or arrival time. While playing the game it was clear that our set-up is not capable to achieve sufficient detection rates, resulting in an unpleasant gaming experience. Occlusions occurred often, resulting in unintended group splitting and merging. The higher the number of people, the more occlusions occurred (see section 5.4). This also means that the current system is not scalable for real world usage. Nevertheless, it is impossible to make completely correct group detections using computer vision if their social background is unknown. If an individual for instance is asking directions from another person, the system might group them,

although they are not a real group. Or if the crowd density rises in the detection region, people have to stand closer than normal, decreasing the overall interpersonal distance. This may also lead to false assumptions.

7 Future work

In this work we presented a rule set based strategy for group detection. Although we did initial tests and observations we noticed that a good evaluation of a rule set is difficult to accomplish. In the future a thorough evaluation of the combination of rules is necessary. Another disadvantage of our approach is the set-up. We used a bird-eye view to use the original Kinect API for human detections. It would be better to use a top-down view like Marquardt et al. [22] to prevent occlusion problems.

Perhaps the rule set can be expanded to enable developers to create an even more customized rule set although we argue that the most important rules are already included. A combination with the f-formations strategy by Marquardt et al. [22] for instance is not unthinkable. Both approaches have a rule set based approach and the ceiling mounted Kinect that is used in f-formations method is capable of dealing with occlusions. Another great advantage of their method is that they are able to deduce the orientation of the detected people which might improve our group detection decisions. Our library can be extended with this approach, but because we used Microsoft's Kinect API we did not include it in this work. Another extension might come from the field of social media relationships. If the system is able to identify people and is able to group them using their social background, the grouping of people should vastly improve.

Besides improvements for the rule set and the library itself it should be noted that only very few applications exist that use group detection strategies. Therefore it is necessary to think about how we can use group detection methods to improve interaction methods and create group-aware applications. In this work we created a game and gave an example application that tried to estimate the remaining waiting time at the checkout of a time. But it is necessary to think of other applications that can be improved using our library.

References

[1] M.-C. Chang, N. Krahnstoever, S. Lim, and T. Yu, "Group level activity recognition in crowded environments across multiple cameras," in *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pp. 56–63, Aug 2010.

[2] S. Zaidenberg, B. Boulay, C. Garate, D. P. Chau, E. Corvee, and F. Bremond, "Group interaction and group tracking for video-surveillance in underground railway stations," in *International Workshop on Behaviour Analysis and Video Understanding (ICVS 2011)*, (Sophia Antipolis, France), p. 10, Sept. 2011.

[3] M. Valera and S. Velastin, "Intelligent distributed surveillance systems: a review," *Vision, Image and Signal Processing, IEE Proceedings -*, vol. 152, pp. 192–204, April 2005.

[4] "Cife, seed, project, stanford, university." `http://eil.stanford.edu/egress/`, 2004-2006.

[5] D. Vogel and R. Balakrishnan, "Interactive public ambient displays: Transitioning from implicit to explicit, public to personal, interaction with multiple users," in *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, (New York, NY, USA), pp. 137–146, ACM, 2004.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893 vol. 1, June 2005.

[7] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. New York, NY, USA: Cambridge University Press, 2000.

[8] C. Wojek, G. Dorkó, A. Schulz, and B. Schiele, "Sliding-windows for rapid object class localization: A parallel technique," in *Proceedings of the 30th DAGM Symposium on Pattern Recognition*, (Berlin, Heidelberg), pp. 71–81, Springer-Verlag, 2008.

[9] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors," *Int. J. Comput. Vision*, vol. 75, pp. 247–266, Nov. 2007.

[10] R. Muoz-Salinas, E. Aguirre, and M. Garca-Silvente, "People detection and tracking using stereo vision and color," *Image and Vision Computing*, vol. 25, no. 6, pp. 995 – 1007, 2007.

[11] T. Darrell, G. Gordon, M. Harville, and J. Woodfill, "Integrated person tracking using stereo, color, and pattern detection," *Int. J. Comput. Vision*, vol. 37, pp. 175–185, June 2000.

[12] M. Rodriguez, J. Sivic, I. Laptev, and J.-Y. Audibert, "Density-aware person detection and tracking in crowds," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[13] K. Paleček, D. Gerónimo, and F. Lerasle, "Pre-attention cues for person detection," in *Proceedings of the 2011 International Conference on Cognitive Behavioural Systems*, COST'11, (Berlin, Heidelberg), pp. 225–235, Springer-Verlag, 2012.

[14] J. Levine and R. Moreland, *Small Groups: Key Readings*. Key Readings in Social Psychology, Taylor & Francis, 2004.

[15] J. E. McGrath, *Groups: Interaction and performance*, vol. 14. Prentice-Hall Englewood Cliffs, NJ, 1984.

[16] C. McPhail and R. T. Wohlstein, "Using film to analyze pedestrian behavior," *Sociological Methods and Research*, vol. 10, no. 3, pp. 347–375, 1982.

[17] W. Ge, R. Collins, and B. Ruback, "Automatically detecting the small group structure of a crowd," in *Applications of Computer Vision (WACV), 2009 Workshop on*, pp. 1–8, Dec 2009.

[18] M. Zanotto, L. Bazzani, M. Cristani, and V. Murino, "Online bayesian nonparametrics for group detection," 2012.

[19] A. Azad, J. Ruiz, D. Vogel, M. Hancock, and E. Lank, "Territoriality and behaviour on and around large vertical publicly-shared displays," in *Proceedings of the Designing Interactive Systems Conference*, DIS '12, (New York, NY, USA), pp. 468–477, 2012.

[20] A. Kendon, *Conducting Interaction: Patterns of Behavior in Focused Encounters*. Studies in Interactional Sociolinguistics, Cambridge University Press, 1990.

[21] P. Marshall, Y. Rogers, and N. Pantidi, "Using f-formations to analyse spatial patterns of interaction in physical environments," in *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, CSCW '11, (New York, NY, USA), pp. 445–454, 2011.

[22] N. Marquardt, K. Hinckley, and S. Greenberg, "Cross-device interaction via micro-mobility and f-formations," in *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, (New York, NY, USA), pp. 13–22, ACM, 2012.

[23] A. Leykin and M. Tuceryan, "Detecting shopper groups in video sequences," in *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pp. 417–422, Sept 2007.

[24] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," in *Proceedings of the American Mathematical Society, 7*, 1956.

[25] M. E. Newman, "Modularity and community structure in networks," *Proc Natl Acad Sci U S A*, vol. 103, pp. 8577–8582, June 2006.

[26] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, 2006.

[27] Z. Qin, "Improving multi-target tracking via social grouping," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, (Washington, DC, USA), pp. 1972–1978, IEEE Computer Society, 2012.

[28] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, pp. 4282–4286, May 1995.

[29] J. Šochman and D. Hogg, "Who knows who - inverting the social force model for finding groups," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 830–837, Nov 2011.

[30] R. Mazzon, F. Poiesi, and A. Cavallaro, "Detection and tracking of groups in crowd," in *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*, pp. 202–207, Aug 2013.

[31] S. Mckenna, S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld, "Tracking groups of people," *Computer Vision and Image Understanding*, vol. 80, pp. 42–56, 2000.

[32] M. Thaler and W. Bailer, "Real-time person detection and tracking in panoramic video," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2013.

[33] F. Cupillard, F. Bremond, and M. Thonnat, "Tracking groups of people for video surveillance," in *Video-Based Surveillance Systems* (P. Remagnino, G. Jones, N. Paragios, and C. Regazzoni, eds.), pp. 89–100, Springer US, 2002.

[34] B. Jain, "Multiple colored ball tracking using opencv python and threading." `http://bipuljain.wordpress.com/2011/12/20/multiple-color-tracking-using-opencv-python/`, 2011. (Visited on 08/21/2014).

[35] N. Marquardt, R. Diaz-Marino, S. Boring, and S. Greenberg, "The proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, (New York, NY, USA), pp. 315–326, ACM, 2011.

[36] M. Nebeling, E. Teunissen, M. Husmann, and M. C. Norrie, "Xdkinect: Development framework for cross-device interaction using kinect," in *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '14, (New York, NY, USA), pp. 65–74, ACM, 2014.

[37] D. R. Olsen, Jr., "Evaluating user interface systems research," in *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, (New York, NY, USA), pp. 251–258, ACM, 2007.

[38] D. Liebling and M. R. Morris, "Kinected browser: Depth camera interaction for the web," in *Proc. 2012 ACM Conference on Interactive Tabletops and Surfaces*, ACM, November 2012.

[39] M. Nebeling and M. Norrie, "jqmultitouch: Lightweight toolkit and development framework for multi-touch/multi-device web interfaces," in

Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS '12, (New York, NY, USA), pp. 61–70, ACM, 2012.

[40] T. Gjerlufsen, C. N. Klokmose, J. Eagan, C. Pillias, and M. Beaudouin-Lafon, "Shared substance: Developing flexible multi-surface applications," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, (New York, NY, USA), pp. 3383–3392, ACM, 2011.

[41] T. Ballendat, N. Marquardt, and S. Greenberg, "Proxemic interaction: Designing for a proximity and orientation-aware environment," in *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, (New York, NY, USA), pp. 121–130, ACM, 2010.

[42] E. Hall, *The hidden dimension*. A Doubleday anchor book, Doubleday, 1966.

[43] E. Griffin, *A First Look at Communication Theory*. McGraw-Hill Companies,Incorporated, 2008.

[44] A. Azad, J. Ruiz, M. Hancock, and L. E., "Group behaviours around public displays," 2011.

[45] "Whac-a-mole." `http://www.bobsspaceracers.com/whac-a-mole.html`. (Visited on 08/23/2014).

Appendices

Appendix A - Code snippets

Example code for the AreParametersOK replacement function. This code snippet can be used for the example application in section 3.2.4.

```
/*
function that replaces the original
AreParametersOK function of the library
*/
private Boolean AreParametersOKNew(int i , int j){
  //First check if the pair exists (Mandatory!)
  if (!grapi.checkPairExistence(i, j))
    return false;
  /* get the rule percentages or if necessary convert to 1 or 0
    in this example we use 0/1 results */
  int alpha = grapi.IsDistanceOK(i,j)>0.0?1:0;
  int beta = grapi.IsDistanceFromScreenOK(i,j)>0.0?1:0;
  int gamma = grapi.IsInitialDirectionOK(i,j)>0.0?1 : 0;
  int delta = grapi.IsArrivalTimeOK(i,j)>0.0?1:0;

  //set the rule time threshold to 3 seconds
  grapi.ruleTimeThreshold = 3;

  double result = 0.3*alpha + 0.3*beta + 0.2*gamma + 0.2*delta;
  double threshold = 0.6;
  //if the result is higher than the set threshold return true
  return result > threshold;
}
```

Appendix B - Dutch summary

Een regelgebaseerde aanpak om groepen te herkennen door gebruik te maken van een diepte camera

Inleiding

In dit onderzoek trachten we een methode te ontwikkelen die gebruikt kan worden om groepen te herkennen in verschillende situaties. De methode gebruikt een verzameling van regels die gebruikt kan worden om te bepalen of mensen bij elkaar horen (bijvoorbeeld: “is de afstand t.o.v. elkaar klein genoeg?”). De techniek de we in dit artikel bespreken maakt gebruik van de detectiemogelijkheden van Microsofts Kinect sensor. Allereerst stellen we ons de vraag welke regels nuttig kunnen zijn om groepen te detecteren. Hierbij houden we in het achterhoofd dat deze regels afleidbaar moeten zijn van de data die door de Kinect wordt geobserveerd. Ten tweede ontwikkelden we ook een bibliotheek die ontwikkelaars de mogelijkheid biedt om zelf snel een applicatie te schrijven die in staat is om groepen te detecteren. Als laatste moeten we ons zeker de vraag stellen of het detecteren van groepen wel mogelijk is door gebruik te maken van regels. Om een duidelijke context te bieden omschrijven we kort een aantal gerelateerde technieken alvorens onze eigen aanpak toe te lichten. Om af te sluiten bespreken we kort een voorbeeldapplicatie die gebruik maakt van onze bibliotheek.

Gerelateerde technieken

De gerelateerde technieken kunnen we opdelen door gebruik te maken van twee classificaties. Enerzijds kunnen we de reeds onderzochte methoden verdelen in real-time of vertraagde/uitgestelde technieken. Bij de eerste groep wordt op basis van 1 beeld een veronderstelling gemaakt van de geachte groepsindeling. Bij de tweede groep worden meerdere beelden (of "film") gebruikt om veronderstellingen te maken. Bij de laatste groep is het dus mogelijk om meer accurate resultaten te verkrijgen omdat er meer data beschikbaar is om deze groepsverdelingen te ondersteunen. Anderzijds kunnen we groepdetectietechnieken ook verdelen in "toezicht-technieken" (supervised) en "zonder toezicht" technieken (unsupervised). Supervised technieken zijn over het algemeen beter in staat om juiste groepsverdelingen te maken omdat ze gebruik maken van machine learning waarbij een trainingset wordt gebruikt. Deze technieken vereisen dus wel veel data om goede verdelingen op te leveren. Unsupervised technieken hebben dan weer geen data nodig van eerdere frames of training data en verdeelt individuen op basis van regels die vooropgesteld werden bijvoorbeeld door te kijken naar de afstand tussen de individuen.

Onze aanpak

Na het doornemen van verwant onderzoek kwamen we tot een verzameling van zeven regels die het mogelijk maken om mensen te groeperen in real-time. De belangrijkste en meeste logische regel is die van de interpersoonlijke afstand die kijkt of twee individuen zich kort genoeg bij elkaar bevinden. Ten tweede hebben we ook een regel toegevoegd die kijkt of in-

dividuen zich op ongeveer gelijke afstand van de camera bevinden. Hier vonden we geen voorbeelden van in de literatuur maar we beargumenteren deze keuze met het feit dat mensen die bij elkaar horen meestal op ongeveer dezelfde afstand van de camera staan (als de camera zich boven het scherm bevindt). De derde regel kijkt of de gemiddelde snelheid van individuen gelijkaardig is, McPhail en Wohlstein gebruiken bijvoorbeeld een grens van 0.15 m/s. Verder hebben we ook nog een regel die kijkt of mensen initieel uit dezelfde richting komen en een regel die kijkt of mensen in dezelfde richting wandelen. Ten vijfde is er een regel die kijkt of twee individuen op een gelijkaardig tijdstip voor het eerst gedetecteerd werden. De laatste regel die we toevoegde is een speciale regel omdat deze ervoor kan zorgen dat onze methode plots niet meer real-time werkt maar kan vertragen naarmate de grens wordt ingesteld. De regel kijkt namelijk hoe lang twee individuen door de verzameling regels als een groep wordt gezien. Als dit lang genoeg is worden ze ook effectief als groep gerapporteerd door onze methode.

Onze aanpak voorziet ontwikkelaars met de nodige flexibiliteit om zelf een set van regels samen te stellen. Deze ruleset kan dus aangepast worden zodat ze als gewenst beslissingen kan maken in verschillende scenarios. Om deze regels te evalueren en te testen ontwikkelden we een bibliotheek die ons in staat stelde om enerzijds snel een testapplicatie en anderzijds snel een voorbeeldapplicatie te programmeren. Tijdens deze initiële testen werd al snel duidelijk dat een optimale ruleset die werkt in elke situatie zeer moeilijk of zelfs onmogelijk te vinden is. Hierdoor hebben we besloten om de ruleset dynamisch te maken om zo de ontwikkelaar de mogelijkheid te bieden om zijn set van regels aan te passen aan zijn situatie.

Om de mogelijkheden van onze bibliotheek te evalueren ontwikkelden we een spel dat gebruik maakt van de eerder besproken groepsdetectie techniek om spelers te verdelen in twee groepen. Het spel is gebaseerd op het welbekende Mollen meppen maar in plaats van één speler moeten de twee teams zo snel mogelijk een mol proberen te vangen met het zwaartepunt van hun groep. Het scherm wordt op de grond geprojecteerd om het spel een extra dimensie te geven. De Kinect werd op een hoogte van twee meter bevestigd om zo het speelveld te observeren (zie afbeelding 37). Na enkele observaties werd al snel duidelijk dat de detectie van spelers onvoldoende was wanneer iemand tijdelijk onzichtbaar is voor de Kinect. Dit zorgt er voor dat groepen hervormt worden tijdens het spelen, wat voor frustratie leidt bij de spelers. Een betere oplossing zou zijn om de Kinect van boven uit te laten observeren waardoor occlusies niet kunnen plaatsvinden.

Conclusie

Allereerst kunnen we concluderen dat het mogelijk is om groepen te detecteren aan de hand van een verzameling regels. Al moeten we deze uitspraak wel nuanceren door het feit dat deze verzameling regels zeer situatiegebonden is en dat een optimale set van regels die voor alle situaties perfect werkt onmogelijk te vinden is. De bibliotheek die we ontwikkelden houdt hier rekening mee en biedt ontwikkelaars een makkelijke manier aan om mensen te groeperen die gedetecteerd worden door de Kinect. Buiten het feit dat onze bibliotheek kan dienen voor groepsdetectie kan hij ook gezien worden als een tool die ontwikkelaars voorziet van evenementen die plaatsvinden voor de Kinect enerzijds, en minder abstracte data over de scene anderzijds.

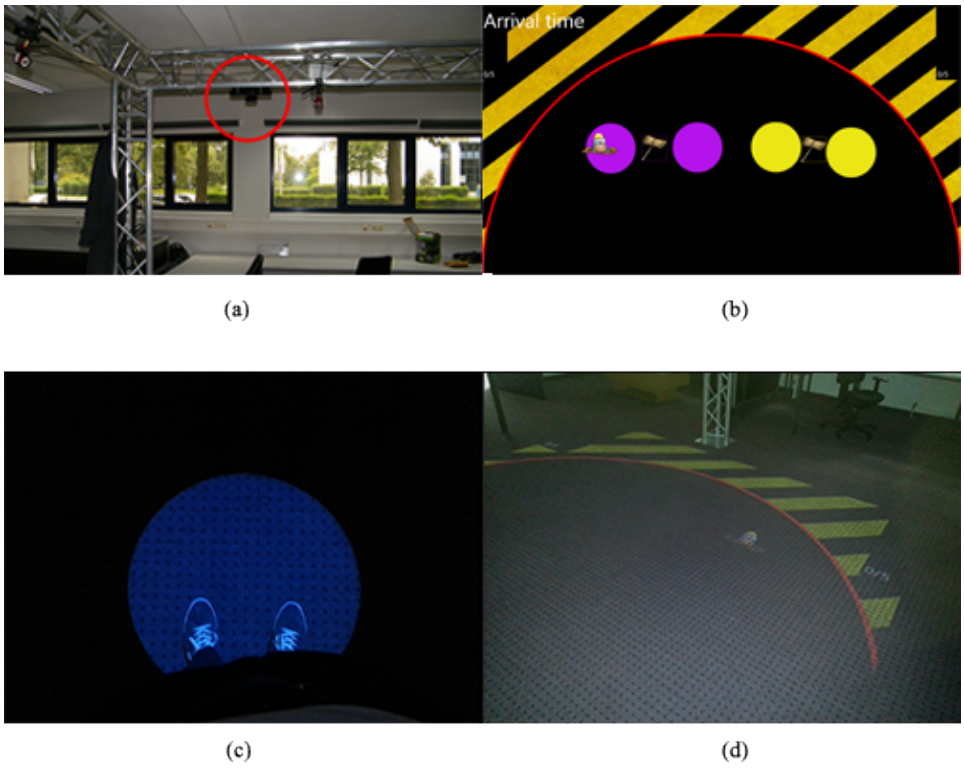

Figure 37: De set-up van onze applicatie en enkele beelden van het spel.

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
A rule-based approach to detect groups using a depth sensing camera

Richting: **master in de informatica-Human-Computer Interaction**
Jaar: **2014**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Winters, Yannick

Datum: **2/09/2014**