A Low-Cost, Human-Inspired Perception Approach for Dense Moving Crowd Navigation

Ishani Chatterjee

CMU-RI-TR-16-47

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

August 2016

Abstract

Robot travel through dense moving crowds is necessary for timely execution of navigation tasks since humans and other robots depend on reliable arrival times. Therefore, robots need socially appropriate methods for finding and entering openings in free-flowing crowds. We describe a method for low-cost awareness of group formation, personal space approximation, and occlusion compensation for use in crowd navigation. Our approach attempts to incorporate social expectations and mimic human techniques when traversing dense, moving crowds. Humans usually cluster people into groups of similar speed and direction, instead of tracking individual motion, and take into account partially obscured people. Using a similar approach avoids the challenges of detecting and tracking a large number of people simultaneously. Another human technique is to estimate and gamble to reduce perceptual and cognitive load.

Our approach uses a single RGB-D sensor and (1) clusters all moving objects into groups and detects splits and merges in these groups, (2) applies a 2D polygon projection in obscured regions to reduce inappropriate motion and collisions due to unexpected concealed objects, and (3) defines a dynamic group personal space modeled using asymmetric Gaussians in order to inhibit certain socially inappropriate robot paths. This approach trades off detection of individual people for higher coverage and lower cost, while preserving high speed processing. A real-world quantitative evaluation of this approach showed good performance in comparison to an existing people detection approach. The projected polygon step captures significantly more people in the scene (77% vs. 80%) and supports group clustering in dense, complex scenarios. In addition to quantitative findings, there were multiple interesting examples cases of group characteristics and behaviors where other approaches typically encounter difficulty.

Acknowledgments

I would like to thank my adviser, Prof Aaron Steinfeld, for his continuous support and guidance for the development of this research. I was introduced to interesting research topics by him and was always encouraged to explore multiple ways of solving a problem. His valuable advice has been useful throughout my Masters and he has always kept me motivated.

In addition, I would also like to thank Prof Illah Nourbakhsh and Marynel Vázquez for their guidance and feedback. I thank Luis Ernesto Navarro-Serment for discussions that helped me gain a better understanding of some concepts.

I thank the Port Authority of Allegheny County for their permission to collect the data used in this study. Thanks are extended to Dhruv Saksena for his assistance in collecting data during part of this study. Thanks are also extended to Sankalp Arora and Subhagato Dutta for their knowledge that helped in solving some of the implementation challenges.

This work has been supported in part by the National Science Foundation (IIS-1012733).

Contents

Introduction 1

1.1 Motivation 1

1.2 System Considerations 2

1.3 Humans as Inspiration 2

1.4 Design Principles 3

Related Work 7

Finding Groups of Coherent Motion 10

3.1 Ground Plane Estimation 11

3.2 3D Flow Vector Computation 12

3.3 Density Based Clustering of 3D Flow Vectors 13

Group Shape for Occlusion Compensation 16

Dynamic Group Personal Space 20

Evaluation 24

6.1 Methods 24

6.2 Results 25

6.3 Other Observations 26

Conclusions 31

7.1 Discussion 31

7.2 Future work 32

References 34

Aspects of this work are also available in:

I. I. Chatterjee and A. Steinfeld, "Low Cost Perception of Dense Moving Crowd Clusters for Appropriate Navigation," in *Workshop on Social Norms in Robotics and HRI, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

II. I. Chatterjee and A. Steinfeld, "Performance of a Low-Cost, Human-Inspired Perception Approach for Dense Moving Crowd Navigation" in *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2016*. (To appear)

Chapter 1

Introduction

1.1 Motivation

As mobile robots become more common in society their novelty and entertainment value will reduce, thus creating new challenges for robot navigation. In particular, robots are currently so rare that independent travel through crowds often produces unnatural crowd dynamics and clearances. Bystanders unfamiliar with robots are usually careful, suspicious, and concerned for their safety, thereby creating large spaces for robots to move through. Similarly, inappropriate or rude behavior by the robot is tolerated due to novelty and the chance for a good laugh.

Simple heuristics based on occupancy and asocial methods like linear robot motion can be effective for these scenarios but will likely be viewed as inappropriate in future deployments where robots are more common. One common approach currently utilized by robot developers is to have a robot wait until a moving crowd disperses. This works fine in locations where moving crowds are transient (e.g., school hallways during class changes) but can freeze a robot indefinitely in locations that are continuously busy (e.g., large transportation hubs, sports and entertainment complexes, etc.) and thus hamper timely execution of navigation tasks that would be expected by humans or other robots. For example, a robot trying to rendezvous with a transit user who is blind [1] cannot wait due to schedule demands. If the robot uses this strategy of waiting until the crowd disperses, it would be unable to complete its fundamental mission in many busy transit stations. Actively traversing dense, moving crowds is a necessary requirement for certain classes of robots.

When indefinite stopping is not an option, the typical approach is to have a robot establish its presence using some type of salient signal (e.g., beeping, spinning light, etc.) coupled with smooth, predictable motion. This is usually effective but creates a nuisance, leading to resentment and sometimes undesirable physical contact. An ethnographic study examining the long-term deployment of hospital delivery robot collected evidence of people being clipped by the robot and significant irritation among certain staff members [2]. In some cases, bystanders kicked the robot in anger. Therefore, if the robot wants to prevent waiting till dispersion and not use the signaling method, it has to seek an opening in a free flowing crowd which can be done by holding a static position while looking. This reduces risk of collision, decreases challenges in tracking moving objects, and lowers wheel motor battery drain.

1.2 System Considerations

This effort is part of the larger CoBot project focused on mobile service robot assistance for daily activities, delivery tasks, and guiding visitors in a typical white-collar office setting [3; http://www.cs.cmu.edu/~coral/projects/cobot/]. A key aspect of the CoBot project is that all robot autonomy is run locally within the robot using only a Kinect sensor, WiFi, and a laptop. Some CoBot variations have used laser scanners and other sensors, but system capability limited to a single RGB-D sensor and a laptop is very likely in future commercial systems. High cost sensors and extensive computational capability limit the economic viability of helper robots. All CoBots are similar in size to an older child or young teenager. The omnidirectional base is very responsive and quick.

Likewise, we are restricting perception to that which can be achieved locally on the robot. Overhead sensors giving a global view of the scene can be very effective at tracking crowd motion but this constrains robot deployments to locations where this is feasible economically, politically, and geometrically. Therefore, we are focusing on methods that do not require external sensing.

1.3 Humans as Inspiration

When trying to create socially appropriate robot behavior one can either utilize traditional robotics methods and layer modules and modifiers for meeting social norms, or the system can be designed from the ground up to incorporate human expectations and behaviors. We are also drawing inspiration from human perception to address sensor viewpoint and system computation restrictions. Combined, we hope that this approach leads to inherently appropriate and understandable motion behaviors. A robot that uses human-like perception is more likely to make human-like motions and fail in human-like ways. This should increase the ability of bystanders to model robot motion behaviors.

When humans need to move through a crowd they utilize certain strategies to reduce complexity and cognitive load. First, it is infeasible for humans to track the individual motion of all pedestrians within a group simultaneously. Instead, humans watch group motion and utilize *gestalt* effects to abstract crowd dynamics into a simpler problem space. Loosely speaking, people moving in the same direction are detected by our motion-sensitive peripheral vision (rods) and merged into a single moving blob. This is also effective at including people who are partially obscured, thus avoiding a common problem in computer vision based pedestrian detection.

Second, humans use scenario-based behaviors. Common behaviors include looking for a seam or gap in the crowd, entering or crossing it, following the person in front of us, when needed, etc. This allows us to expend the least needed cognitive and perceptual effort when moving with a crowd, as evidenced by humans' ability to read smartphones while walking on busy sidewalks. Most robots can do this latter task using a single forward looking sensor and limited computation since it approximates platooning and other well-established following tasks. Therefore, the open challenge is finding the seam or gap in a crowd and navigating into or across it in a socially appropriate manner.

Finally, another human technique is to estimate, guess, and gamble to reduce perceptual and cognitive load, which is especially useful when only the leading edge of the crowd is visible. Humans do not bother with trying to be perfect. Just as it is impossible to track every person, it is also often impossible to perceive all influences on a space. Instead, humans utilize personal space, the flow of the crowd, and knowledge of their personal reaction time to judge how much room to wait for and how close to blindly follow the person in front of them. This is one of the reasons why walking with a smartphone in crowds is often problematic. The user's reaction time is considerably longer due to distraction, thus leading to unsafe behaviors.

1.4 Design Principles

As mentioned, our approach is driven by several principles motivated by robot capabilities and typical human methods for traversing dense, moving crowds. These principles are (1) low cost with limited capability, (2) inspiration from human gestalt-based perception, and (3) inspiration from human guessing and gambling when perception is limited.

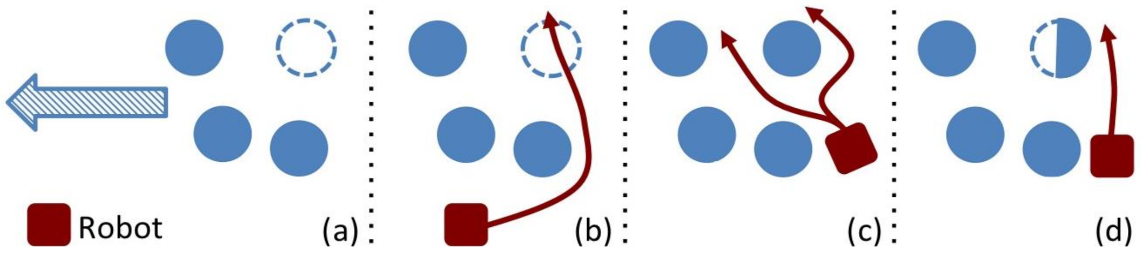
First, we strictly adhere to the principle of low cost and limited system capability. This is based on the likely economics of robots needing this capability and the current system characteristics of the CoBot. This leads to the selection of a regular Kinect RGB-D camera and a single 13” laptop. While the laptop is currently top of the line for the form factor (Intel Core i7 Processor and NVIDIA GeForce GTX 960M Graphics card), this computational capability is likely to be reasonable for economical robots 2-3 years from now.

The next principle is to draw on the way humans perceive dynamic obstacles while navigating through crowds. Humans usually do not have the ability to track individual motion of many pedestrians within a group. Instead, they fixate on various different points that represent the group to generate an overall gestalt of the motion features like bearing and speed of the collective entity. We use this principle of human perception of crowds to guide the system design pipeline. It is our belief that this approach of collecting points from groups into clusters having similar speed and direction will be fast and efficient for computers too and will eliminate the need to track individual people in crowds. This sidesteps the challenge of detecting and tracking a large number people concurrently while also capturing partially obscured people one or two layers deep in the group.

Another principle aligns with the earlier observation that humans estimate, guess, and gamble thereby reducing perceptual and cognitive load. Due to system characteristics, our system does not rely on an overhead camera for crowd perception but uses only front view obtained from the onboard RGB-D camera. This camera is fixed at the low end of human eye height, which is similar to the first-person view humans use while navigating. Thus, we can only observe the leading visible edge of the dynamic obstacles, while being uncertain about what lies behind this edge.

In steady crowds it is likely that there will be concealed people a small distance behind the row of persons perceived. This creates potential problems for path planners seeking optimal routes. For example, if there is a perceived L shaped group of 3 people (Figure 1a), and the robot assumes free space behind one of the persons forming the leading visible row, the robot could attempt to round the corner behind the leading row and encounter an unexpected occluded person (Figure 1b), leading to inappropriate reactions (Figure 1c). Assuming extra space behind the leading edge as occupied space decreases the chance of encroaching on a concealed person without adequate reaction time (Figure 1d). Guessing someone is in obscured space leads to path-

Figure 1: Due to obscured perception an L-shaped crowd (a) may lead to path plans that risk collisions with an unforeseen member of the group (b). Recovery when rounding a turn is possible but may lead to unpredictable paths (c). The approach described here limits this risk (d).



plans that are less likely to curl around the back of the leading row. This gives the robot an extra safety margin in most cases without having to resolve an extremely difficult perceptual challenge. In the worst case, the robot will just avoid paths near this group and opt to wait or take a different route.

Likewise, the robot should incorporate some level of personal space around the perceived and imagined members of a group, thereby maintaining social norms and also providing added room for last minute reactions.

This thesis describes the algorithms developed using these design principles, to extract characteristics of dense, moving crowds such as group formation, approximating the group personal space and compensating for occlusion due to local sensor view. An evaluation of the system in a real-world environment, in comparison with an existing people detection approach, was performed. Our main contributions are the following real-time algorithms:

1) Identify groups of coherent motion and detect splits and merges in these groups, using a single Kinect.
2) Define group shapes to reduce inappropriate motion and collisions due to unexpected concealed objects.
3) Appropriately combine individual personal spaces represented by asymmetric Gaussians to define a dynamic group personal space.

The output of these algorithms are various crowd motion features, which in the future can be used in planning socially appropriate paths.

Chapter 2

Related Work

Recent work in robot navigation in crowds is also underway within the SPENCER project [4], with the aim of making an assistive robot navigate through crowds in a busy airport. This approach uses a more capable system than CoBot, thus leveraging more sensors. For example, Linder and Arras [5] use multiple RGB-D sensors oriented vertically near each other. SPENCER tracks individual humans using HOG and upper body detectors, followed by group detection, then predicting social relation probabilities between them, and finally tracking groups. This is done by extending multi-model multi-hypotheses tracking to maintain stable group identifiers during sporadic change of tracks of people comprising the group. Adding sensors increases costs and computational overhead. Also, tracking each person individually leads to incomplete perception of all nearby people. Finding everyone can be problematic since obstructions are frequent in dense crowds. SPENCER adopts a strategy different than ours, they attack the problem from a high-resource direction that can afford to track as many people as possible. Instead, we (1) explore a low-cost, Gestalt-oriented perspective since some robots will not have the necessary sensors or computational resources required for the described approach, and (2) detect all moving objects for the task of navigation; our approach does not use trained templates for human detection. Classification of humans is left to other, higher-level functionality, if at all.

Similar work by Trautman, et al [6] tried to resolve the freezing robot problem by developing mathematical models for cooperative collision avoidance among a robot and nearby humans. This approach has shown comparable performance with human tele-operation in crowd densities near 0.8 humans/m^2. The pedestrian tracking system used for this work was a collection of three overhead stationary stereo cameras that tracked pedestrians using background subtraction. The input to the navigation algorithm were the tracks of five people with the highest probability of collision with the robot. This differs from our approach since (a) our observation is limited to local frontal view with frequent occlusion, rather than a static global overhead view, and (b) we support consideration of more than five individuals.

The use of optical flow and density-based clustering to detect and analyze crowd movements has also been used in [7]. However this work was limited to the clustering of 2D displacement vectors in pixel coordinates in image sequences, since the work was mainly focused on using video sequences to detect motion patterns and abnormal motions in crowds. When deploying on a robot, it is possible to work in 3D since the robot will have the real world coordinates and depth information from the RGB-D camera. Acceptance thresholds that define similarity between vectors are selected differently for the two cases. In the 2D flow case of [7], clustering takes place in the image plane, so these thresholds depend on the relative position between the camera and the crowded scene; the thresholds increase with a closer camera and decrease with a farther one. In our case, clustering takes place using real world 3D coordinates and spatial distances and angles. Therefore, the thresholds are independent of the distance between the crowd to be clustered and the camera. It is no longer necessary to keep changing parameters that now only depend on the robot width and an acceptable difference in headings and speeds for them to be considered as similar.

3D optical flow is analogous to human perception and thus a promising method to explore. Work by Kim, et al [8] used this approach in a feature extraction module for generating human-like trajectories in dynamic environments. However, this work did not group 3D points obtained from sensor data into blobs of coherent motion having similar motion features like velocity and density. Instead, the authors extracted values of these features for each grid-cell; the weights of these features were learned using Inverse Reinforcement Learning to come up with a cost at each grid-cell for navigation. Our method finds groups and captures group changes like splitting and merging. This could be used to extract features for predicting splits and merges. Identifying a split or merge in advance could be useful while navigating within dense crowds. Moussaïd, et al [9] found that 70% of pedestrians walk in groups, therefore finding and tracking the shape, size and motion characteristics of groups would be beneficial for robots navigating in crowds.

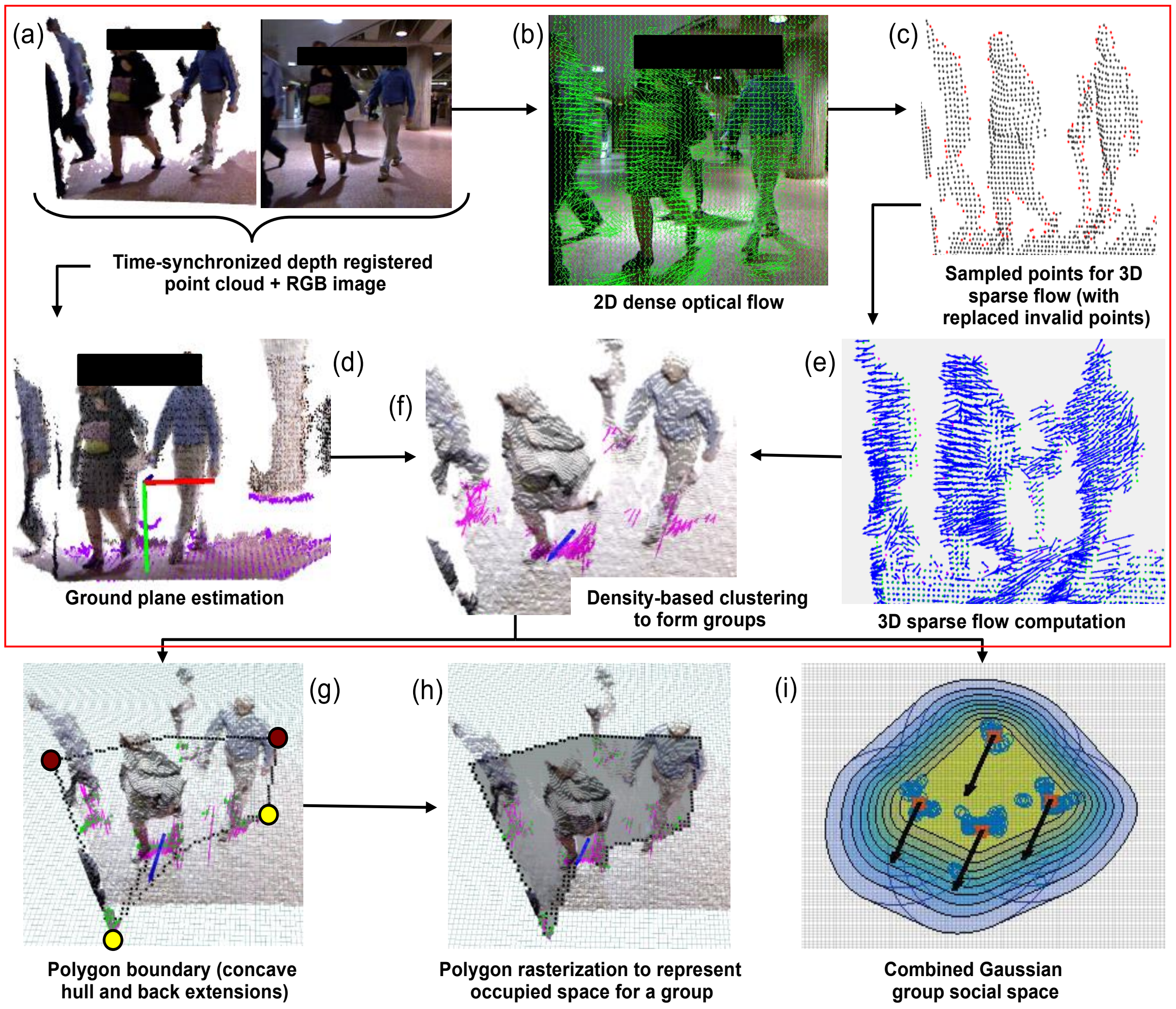
The use of Gestalt theories to abstract computer vision challenges into computationally tractable solutions is not new and has been proven effective in other domains. For example, past work by the team used regions of saliency to find regions of interest during assisted photography on smartphones, thus bypassing the need for object recognition and scene understanding [10, 11]. Experimental results with users who were sighted, low vision, and blind proved this approach works.

Chapter 3

Finding Groups of Coherent Motion

Our system, with the system pipeline shown in figure 2, was implemented in C++ as ROS nodes subscribing to various topics for input and publishing output in other topics. The inputs to the system at time = t were the 3D depth-registered point cloud **pc_curr** and the corresponding time-synchronized RGB image **frame_curr** from the Kinect (Figure 2a).

Figure 2 : System pipeline, screenshots taken on the same scene occuring at a randomly chosen time-instance. Steps inside the red rectangle are for extraction of blobs of coherent motion**.**


(a)
Time-synchronized depth registered point cloud + RGB image
(b)
2D dense optical flow
(c)
Sampled points for 3D sparse flow (with replaced invalid points)
(d)
Ground plane estimation
(f)
Density-based clustering to form groups
(e)
3D sparse flow computation
(g)
Polygon boundary (concave hull and back extensions)
(h)
Polygon rasterization to represent occupied space for a group
(i)
Combined Gaussian group social space

The system re-used 3D point cloud **pc_prev** and the corresponding RGB image **frame_prev** from the previous run performed at time = t-1. A depth-registered cloud contains RGB information along with the 3D coordinates of points in meters, as measured in the Kinect coordinate system. Algorithm 1 summarizes the steps for extraction of groups of coherent motion, as will be described below in sections 3.1, 3.2, and 3.3. Figure 2(b), (c), (d), (e) and
(f) of the system pipeline, as marked by the red-rectangle in Figure 2, illustrate the steps described in sections 3.1,3.2 and 3.3. The main steps involved are ground-plane estimation, computing 3D flow of obstacles, projection of flow vectors on ground plane, and using density-based clustering based on similar speeds, heading and spatial closeness of moving objects to identify groups.

3.1 Ground Plane Estimation

One of the first steps performed for building a 2D occupancy grid and planning in a planar stretch is detection of the ground plane to find inliers belonging to a plane using RANSAC. The current input cloud **pc_curr** is down-sampled to a 3D grid of voxels to decrease computation while performing RANSAC and filtered by a

Algorithm 1: Finding groups having coherent motion at time instant t

```
Inputs:  1) Depth-registered point cloud pc_curr at time t
         2) Rgb image frame_curr at time t
         3) Depth-registered point cloud pc_prev at time t-1
         4) Rgb image frame_prev at time t-1
Output: ROS message Cls_curr with array of clusters and metadata

// Ground plane estimation
Downsampled point cloud $pc\_curr_{ds}$ = voxelGrid (pc_curr, voxel size)
Ground plane coefficients [a,b,c,d] = groundPlane ($pc\_curr_{ds}$)

// 2D dense Optical flow computation
[flow_x, flow_y] = Farneback (frame_prev, frame_curr)

// 3D flow vector computation
for every nth row $n_x$ in frame_prev
    for every nth column $n_y$ in frame_prev
        image coordinate of pixel p at nth row and nth column = ($n_x$,$n_y$)
        ($q_x$,$q_y$) = ($n_x$ + flow_x at p, $n_y$ + flow_y at p)
        $point3D_{start}$ = 3Dpoint (pc_prev, linear index ($n_x$, $n_y$))
        $point3D_{end}$ = 3Dpoint (pc_curr, linear index ($q_x$, $q_y$))
        Store $point3D_{start}$ in a new point cloud $pc_{start}$
        Store $point3D_{end}$ in a new point cloud $pc_{end}$
    end for
end for
Replace invalid NaN points from $pc_{start}$ and $pc_{end}$ with valid points

// Obstacle velocity computation
$pc\_obs_{start}$ = filter ($pc_{start}$ ,robot_height) to remove flow vectors' start point above robot height
$pc\_obs_{end}$ = filter ($pc_{end}$, robot_height) to remove flow vectors' end points above robot height
$pc\_proj_{start}$ = project($pc\_obs_{start}$, [a b c d])
$pc\_proj_{end}$ = project($pc\_obs_{end}$, [a b c d])

// Cluster formation
for each point p_start in $pc\_proj_{start}$
    p_end = corresponding end of the projected flow vector in $pc\_proj_{end}$
    velocity = p_end − p_start
    Feature vector formed by concatenating point and velocity Fv = [p_end; velocity]
    Store the feature vector Fv at each point in $Fv_{array}$
end for
    set clustering parameters $D_{euclidean}$ , $Diff_{heading}$, $Diff_{speed}$
    [clusters] = DBSCANclustering ($Fv_{array}$, $D_{euclidean}$ , $Diff_{heading}$, $Diff_{speed}$)

// Publish as Custom ROS message
Frame _id = frameID (frame_curr)
Timestamp = timestamp (frame_curr)
Cls_curr = CustomROSmsgs( [clusters], Frame_id, Timestamp)
Publish(Cls_curr)
```

height that is a little less than the height of the physical ground plane. The algorithm finds the major plane whose normal is within a certain threshold angle made with the Kinect y (vertical) axis. This is set using the maximum possible angle of incline of ramps found commonly in public places like malls. To reduce errors, the unit normal and the distance of the plane from the origin are assigned the average of the first 10 valid unit normals and distances from the origin. This could be repeated for every few frames to keep updating the ground plane. Figure 2d shows points in the 3D point cloud in the detected ground plane as purple.

3.2 3D Flow Vector Computation

To detect crowd motion features we first compute the 2D optical flow, which finds correspondences between image pixels of two consecutive frames **frame_prev** and **frame_curr**. Figure 2b shows the 2D optical flow vectors. In order to overcome accuracy limitations that accompany the optical flow methods of Lucas-Kanade and Horn- Schunck's [12], [13], Santoro et. al [7] used the pyramidal Lucas-Kanade sparse optical flow, with the features to be tracked being determined using Shi-Tomasi corner detection algorithm. However, in our case the best possible accuracy is needed because even very small errors at image pixel levels can still mean large errors in 3D flow. Therefore, the Farneback [14] dense optical flow method was chosen which approximates local neighborhoods in an image by quadratic polynomial and finds displacement by equating coefficients. This showed good results for two-frame motion estimation and displayed real-time speed when run on a GPU, with robustness to noise.

To compute the 3D optical flow at a pixel location (**nx,ny**) in **frame_prev**, we first add the x and y components of 2d flow at that location to (**nx,ny**). We then pick the corresponding 3D flow vector from the 3D point-cloud space. To decrease computation and obtain a down-sampled cloud, the 3D points are picked from every 'nth' column in every nth row in frame_prev, where 'n' is the step-size. So a *step size* of 1 means to pick the 3D points corresponding to every pixel in the image, resulting in purely dense flow comprising every point. The distance between 2 pixels selected this way remains same in image coordinates, but the real 3D distance between the 3D points corresponding to those pixels keeps increasing as we go farther away from the sensor as measured along the Kinect's z axis. Therefore, for a very large step size it possible that, as we move away, the gap between the two consecutive 3D points selected this way is so large that a person is completely missed and not a single 3D point on the person makes it to the down-sampled 3D flow cloud. However, a very small step size means more flow vectors as inputs to further processing steps, thus increasing computation. We want a step size that is just large enough to give us a sufficient number of flow vectors on distant people so that they are not missed. Figure 2c shows the sampled point cloud in grey. An analysis of how step size impacts performance is shown in section 6.2 of Chapter 6.

Quite a few of the 3D points within Kinect's range, that are sampled using the step-size, may have invalid depth values. If an invalid point makes its way into the down-sampled cloud, we replace it by a valid-depth neighbor, if any exists, to get maximum possible valid velocity vectors and avoid drop outs. To find a valid neighbor, we first move to the image space. The image pixel corresponding to an invalid depth point always has a valid RGB value. Replacing the 3D coordinates of the invalid depth point by the depth coordinates of a similar pixel in the image space solves the purpose. The most similar neighboring image pixel is found as the one that has minimum Euclidean distance in terms of the commonly used L*a* b* color space values to approximate the representation of perceived color difference. L*a*b* space is designed such that a change in color values represented in this space represents the same change in color perceived by human eye, so the Euclidean distance between 2 points in L*, a*, b* space is an approximate representation of perceived difference. Spatial closeness is imposed by restricting search in a small neighborhood of size k around each invalid-depth pixel. Figure 2c shows the replaced invalid points in red. Figure 2e shows the final 3D velocity vectors.

Once the point clouds $\mathbf{pc}_{\mathbf{start}}$ and $\mathbf{pc}_{\mathbf{end}}$ containing the start and endpoints of the 3D velocity vectors respectively have been formed, we discard the points above the robot height to prevent them from being considered as obstacles and project the remaining points on the ground plane to find the projected flow clouds $\mathbf{pc\_proj}_{\mathbf{start}}$ and $\mathbf{pc\_proj}_{\mathbf{end}}$ representing the 3D velocity vectors projected on ground plane.

3.3 Density Based Clustering of 3D Flow Vectors

Finding groups may help to (1) detect when a split is about to occur by tracking the velocity directions and inter-member distance in the group, (2) detect when a gap between two groups is about to close due to merging, and (3) predict the future occupancy state of grid-cells. Groups of persons do not have a fixed shape and there can be any number of groups in the scene at any instant of time.

The Density Based Clustering with Noise (DBSCAN) clustering algorithm [15] is well suited for this type of clustering task. Similarity of two vectors is defined based on threshold values for the three equally weighted features: (1) 3D Euclidean distance between their originating points $\mathbf{D_{euclidean}}$, (2) absolute difference in angle between their directions, $\mathbf{Diff_{heading}}$ and (3) absolute difference between their magnitudes $\mathbf{Diff_{speed}}$. The physical distance threshold is considered to be the robot width, as there is no use of treating points with similar velocity as those belonging to separate clusters if their distance is lesser than the robot width, since the robot cannot enter into that space. The threshold value of the difference in angles and magnitudes were decided experimentally from observing clustering behavior in the dataset. The computationally expensive step in clustering is computing the similarity of one vector to all others which is of $O(N^2)$, where N is the number of points in the projected flow cloud. This function is parallelized using CPU threads. Figure 2f shows an identified group and the projected velocity vectors of members in pink. The overall velocity of the group is the mean of all vectors, shown in Figure 2f with the blue arrow at the group centroid.

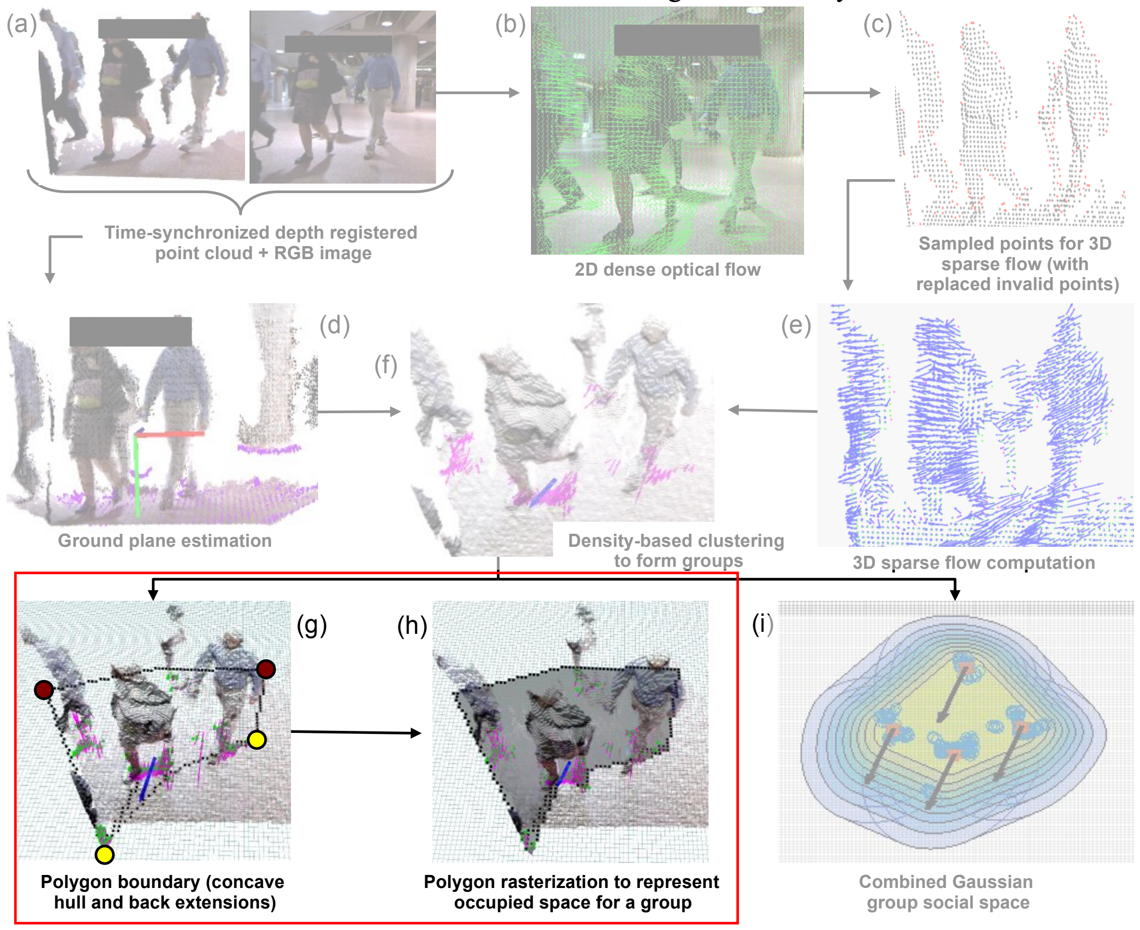
Chapter 4

Group Shape for Occlusion Compensation

Chapter 3 described extraction of groups with coherent motion. This chapter describes how the shape of each of these groups is estimated, especially in occluded regions, to reduce inappropriate motion and collisions due to unexpected concealed objects.

The fixed global frame on ground plane is related to the sensor frame by the transformation matrix ${}^{W}T_{S}$. The ground plane is discretized into 2D grids. Steps shown within the red rectangle in the system pipeline (Figure 3) show the steps followed for group shape estimation. Algorithm 2 describes these steps in detail.

Figure 3: System pipeline, with steps for group shape estimation shown inside the red rectangle. Screenshots are taken on the same scene occurring at a randomly chosen time-instance.


(a)
Time-synchronized depth registered point cloud + RGB image
(b)
2D dense optical flow
(c)
Sampled points for 3D sparse flow (with replaced invalid points)
(d)
Ground plane estimation
(f)
Density-based clustering to form groups
(e)
3D sparse flow computation
(g)
Polygon boundary (concave hull and back extensions)
(h)
Polygon rasterization to represent occupied space for a group
(i)
Combined Gaussian group social space

A group or cluster consists of projected points and velocities. The front boundary of these points is found as shown in Figure 3g to know the demarcation on the costmap of the area representing “lethal obstacle”. ‘Front’ refers to the side of obstacles facing the Kinect. Extending the extremes points of this front boundary into the occluded region, to assume an extra space behind as occupied space, decreases the chance of encroaching on a concealed person without adequate reaction time, as previously explained and illustrated in Chapter 1 section 1.4 and Figure 1 respectively. This gives the robot an extra safety margin without having to resolve an extremely difficult perceptual challenge. It also prevents the planner from finding paths that go through the gaps between people forming the group, and thus mimics the human tendency to avoid breaking a group. With these motivations, we construct the polygon representing a group/cluster using the following steps: (1) Finding the cluster points that are a part of the front boundary and the left and right extreme points on the front boundary. Figure 3g shows the two extreme points in yellow. (2) Joining them to seal gaps between people to obtain a continuous front boundary (in

Algorithm 2: Cluster extension for Occlusion compensation

```
  Inputs: 1) Array of Clusters Cls_arr obtained from Algorithm 1,
  all cluster points in sensor coordinates
          2) Transformation $^{W}T_S$ from global reference frame to
          sensor coordinate frame
          3) Grid cell width $Width_{grid}$, Grid cell height $Height_{grid}$
  Output: The costmap Cost_map with marked occupied space
for each cluster in Cls_arr :
  // Find cluster points constituting the front boundary
     $EP1_{left}$ = Point in cluster with min $\frac{x}{z}$ ratio in sensor coordinates
     $EP1_{right}$ = Point in cluster with max $\frac{x}{z}$ ratio in sensor
     coordinates

     n = number of points in Cluster
     if (n >= 4)
        hull = concaveHull(Cluster) with appropriate alpha
        if unable to find concaveHull then hull = convexHull
          (Cluster)
     else if (n < 4) all cluster points are marked as hull
     end if

     if ($EP1_{left}$ was not included in hull while finding concave hull)
          Insert $EP1_{left}$ in hull
      end if (do for left and right)

      sortedHull = sort (hull) in CCW order with respect to Cluster
        centroid if not sorted
      LeadPts = Points in sortedHull beginning from $EP1_{left}$ and
        ending in $EP1_{right}$
  // Find the discretized line-segments forming the front boundary
      $Boundary_{front}$ = [ ]

      LP_map = sensorToGridCell (LeadPts, $^{W}T_S$, $Width_{grid}$,
        $Height_{grid}$)
      for subsequent points i and i+1 in LP_map
          update $Bound_{front}$ with raytrace (LP_map[i],
            LP_map[i+1]) to find the cost map cells forming the
            discretized joining line
      end for
  // Estimating side edges and back boundary in Kinect's shadow
  region
     $Shadow_{left}$ = Unit vector along the line joining sensor origin
       and $EP1_{left}$ (do for left and right)
     Ext = total width of two average-sized people walking
     $EP2_{left}$ = $EP1_{left}$ extended along the unit vector Shadow by
       magnitude Ext (do for left and right)
     $EP1\_map_{left}$ = sensorToGridCell (EP1, $^{W}T_S$, $Width_{grid}$,
       $Height_{grid}$) (do for left and right)
     $EP2\_map_{left}$ = sensorToGridCell (EP2, $^{W}T_S$ , $Width_{grid}$,
       $Height_{grid}$) (do for left and right)
     $Edge_{left}$ = discretized line as cost map cells joining $EP1\_map$
       $_{left}$ and $EP2\_map_{left}$ (do for left and right side edge)
     $Bound_{back}$ = discretized line as cost map cells joining
     $EP2\_map_{left}$ and $EP2\_map_{right}$
// Marking lethal obstacle region
      $Poly_{final} = Bound_{front} \cup Edge_{left} \cup Bound_{back} \cup Edge_{right}$
      Cell_centers in cost map inside $Polygon_{final}$ = Lethal
        obstacle
end for
```

terms of consecutive grid-cells), shown in Figure 3g. (3) Extending the extremes found in (1) along the shadow cast by Kinect's infrared laser when it hits the front boundary to find the left and right sides. The two extended extreme points are shown in red in Figure 3g. Joining the two extended points gives the back-boundary of the extended space. (4) Joining the front, sides and back boundaries to form the final polygon, and denoting the area within as occupied space, as shown in Figure 3g.

The steps are described in Algorithm 2. The extremes $\mathbf{EP1_{left}}$ and $\mathbf{EP1_{right}}$ of the front boundary are found by finding the minimum and maximum x/z ratio in sensor coordinates, which works for all cases given an acute-angled horizontal field of view of Kinect. Then we find the concave hull of the cluster, which is geometrically an alpha shape, alpha determining the extent to which the details of the boundary are preserved (alpha = 0 includes all points of the set in the boundary). An alpha value of 0.3 worked in our case for all cluster sizes for capturing the desired boundary shape. We sort the hull points in counter-clockwise order around the centroid, and include $\mathbf{EP1_{left}}$ and $\mathbf{EP1_{right}}$ in the hull if already not present. Points that are a part of the front boundary, denoted by **LeadPts,** are the points in the sorted hull starting from $\mathbf{EP1_{left}}$ and ending in $\mathbf{EP1_{right}}$. We project **LeadPts** on the grid to obtain the list of grid cells **LP_map.** We find the continuous front boundary $\mathbf{Bound_{front}}$ by iteratively running a ray-tracing algorithm between subsequent grid-cells to effectively seal gaps between them and obtain consecutive grid-cells approximating straight line-segments running between subsequent grid-cells in **LeadPts.** We estimate the sides to be of the length of two average-sized people walking one after another. The line $\mathbf{Shadow_{left}}$ defining the left edge of the shadow cast by Kinect upon hitting the front boundary is given by the line joining Kinect's position and $\mathbf{EP1_{left}}$. Extending $\mathbf{EP1_{left}}$ along this line gives the point $\mathbf{EP2_{left}}$ .We find the grid-cells defining the left edge $\mathbf{Edge_{left}}$ by projecting $\mathbf{EP1_{left}}$ and $\mathbf{EP2_{left}}$ on the grid-map and ray-tracing between them. $\mathbf{Edge_{right}}$ is found similarly. The back-boundary is the discretized line joining $\mathbf{EP2\_map_{left}}$ and $\mathbf{EP2\_map_{right}}$, the two grid cells representing the left and right extended points. We finally perform polygon rasterization on the group polygon $\mathbf{Poly_{final}}$ to denote occupancy inside the polygon (Figure 3h).

Chapter 5

Dynamic Group Personal Space

Chapter 4 described cluster extension to accommodate for rows of occluded people walking immediately behind the people forming the leading edge. Additionally, a person's intimate and personal space should be respected whenever possible while navigation. This chapter describes a method to define the group personal space. Hall [16] found out proxemic interpersonal distances for a static person. However, in case of moving persons, the shape and size of the personal space can vary with walking velocity and can be represented by 2D Gaussians. A 2D Gaussian centered on the person's centroid (x0, y0) and having variances of $\boldsymbol{\sigma_x}$ and $\boldsymbol{\sigma_y}$ in the x and y direction, is given by:

$$f(x,y) = e^{-\left(\frac{(x-x0)^2}{2\sigma x^2} + \frac{(y-y0)^2}{2\sigma y^2}\right)}$$

The extent of different levels within an individual's personal space is mathematically modelled by Kirby [17] as a 2D Asymmetric Gaussian function centered on the person and having a +y axis aligned with the velocity, as shown in Figure 4a. The variance $\boldsymbol{\sigma_h}$ along the direction aligned with the person's velocity **v** indicated by the black arrow**,** representing the front space, is given by:

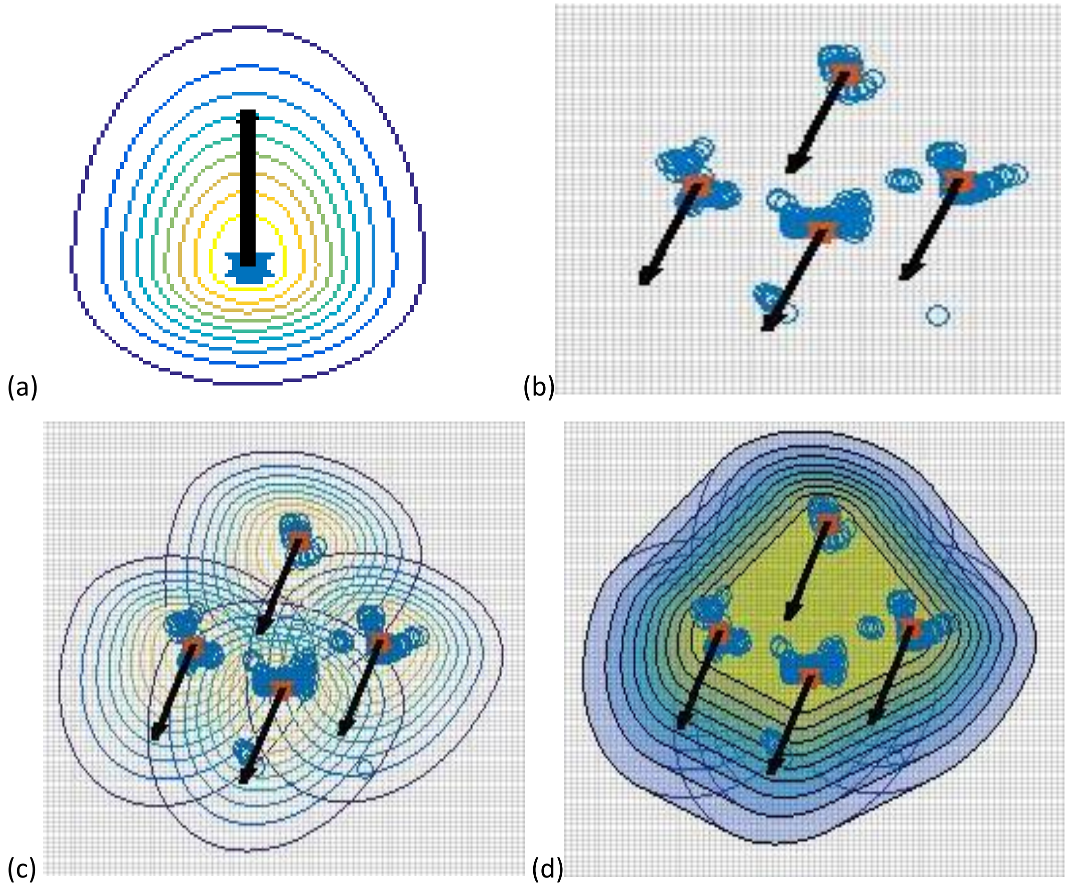
$$\sigma_h = \max(0.5, 2v)$$

The variances $\boldsymbol{\sigma_s}$ along the direction perpendicular to **v**, representing a person's sides, and $\boldsymbol{\sigma_r}$ along the direction to opposite to velocity **v** representing his rear, as found empirically by Kirby [17] are:

$$\sigma_s = 0.67\ \sigma_h \qquad \text{and} \quad \sigma_r = 0.5\ \sigma_h$$

The parameters were found to match the personal space respected in the United States while walking, but the shape remains the same across cultures.

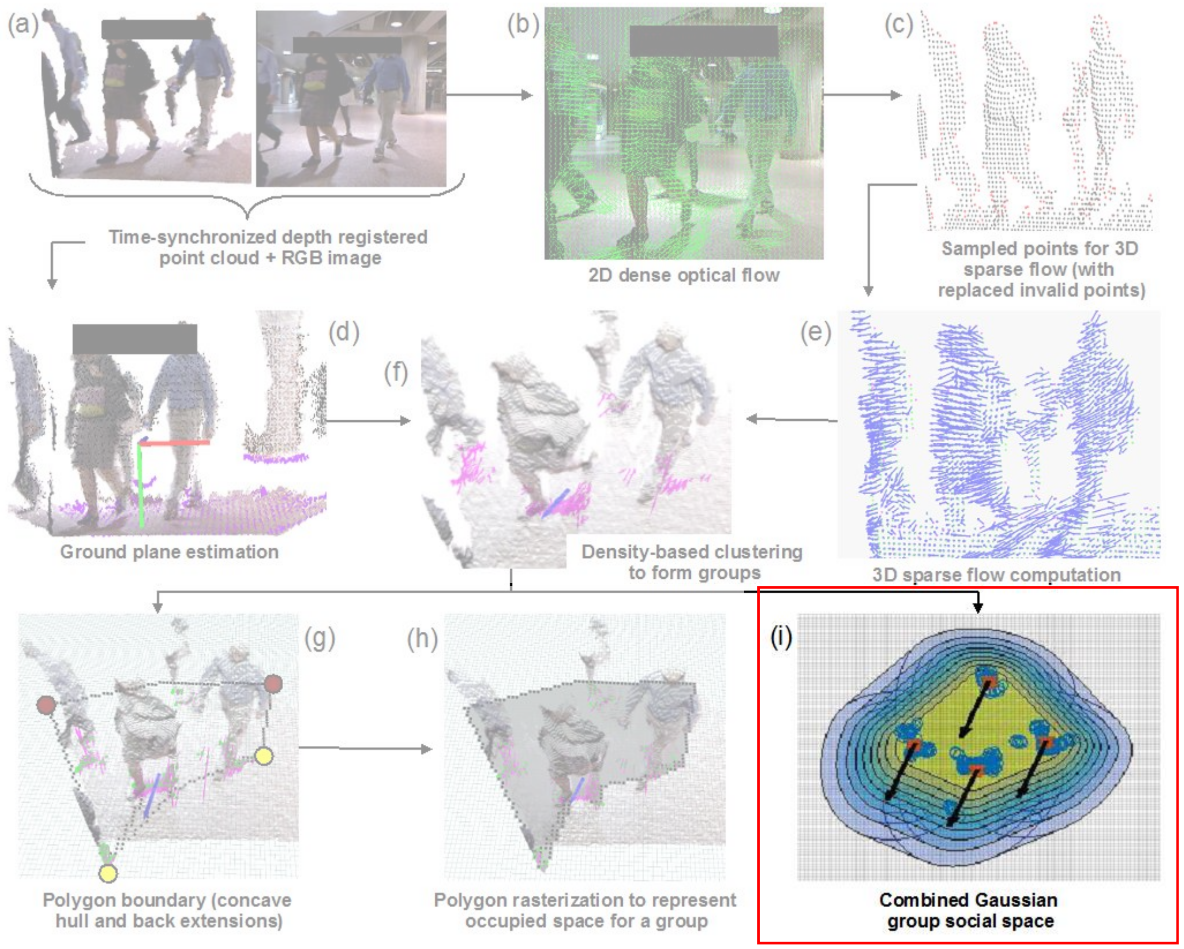
Figure 4: (a) Individual personal space bubble modeled as an asymmetric 2D Gaussian centered on the person (black arrow shows velocity direction) with decreasing costs from yellow to blue (b) Estimated centroids (red) and group average velocity assigned to people forming the group in the transit station data. (c) Individual personal space costs for the 4 people. (d) The group personal space as the convex hull of the iso-contours of the individual bubbles.



Our approach uses a *dynamic group personal space*. This is built on the idea that people in groups would not prefer the robot coming close to the spaces between them and their group members, just like they would not prefer intrusion in their own personal space.

To achieve this, we first find clusters within the group to locate individual positions, compute their centroids and assign the group mean velocity to each centroid, as shown in Figure 4b. We then construct the individual personal space iso-contours centered on each centroid and formed by joining grid-cells around the centroid having same cost, as shown in Figure 4c. We find the cost of the space in between iso-contours of a cost-level ‘n’ by constructing their convex hull. All grid points on the line joining the convex hull points are assigned the cost of the level ‘n’. This is iteratively done for all cost levels till the cost becomes negligible, marking the boundary of the personal space. Figure 4d shows the group iso-contours representing the final personal space of a group at each cost-level, with cost decreasing from yellow to blue. The step marked using a red rectangle in Figure 5 shows the position of the final group personal space in the system pipeline.

Figure 5: System pipeline, with the final group personal space shown inside the red rectangle. Screenshots are taken on the same scene occurring at a randomly chosen time-instance.


(a)
Time-synchronized depth registered point cloud + RGB image
(b)
2D dense optical flow
(c)
Sampled points for 3D sparse flow (with replaced invalid points)
(d)
Ground plane estimation
(f)
Density-based clustering to form groups
(e)
3D sparse flow computation
(g)
Polygon boundary (concave hull and back extensions)
(h)
Polygon rasterization to represent occupied space for a group
(i)
Combined Gaussian group social space

Chapter 6

Evaluation

6.1 Methods

The team arranged formal permission to collect Kinect data in an indoor light rail station. This station has heavy traffic during the morning rush hour commute due to the immediate proximity of several large white-collar businesses and the local courthouse. Commuters exited the station from two train platforms from either side of the selected location.

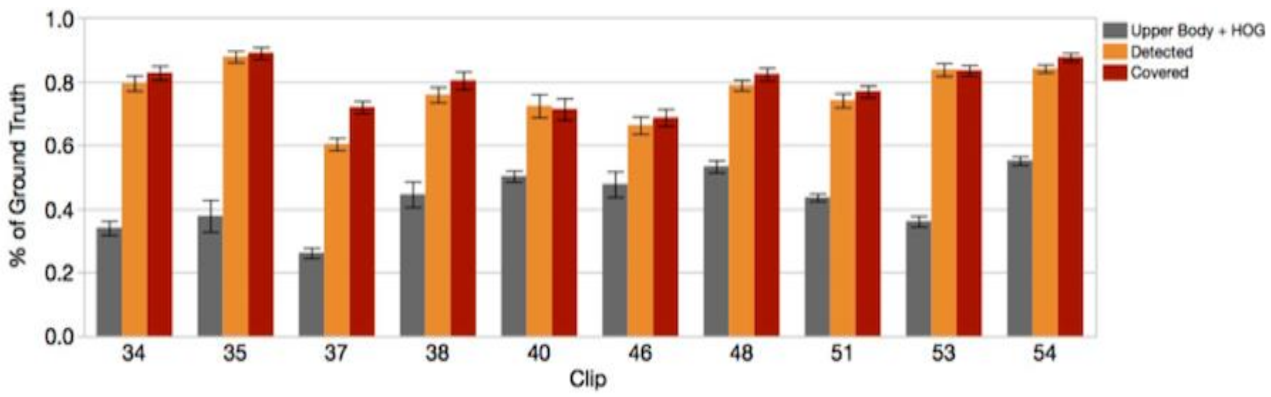
While our approach runs in real-time with average running time ~ 0.1 s per frame, we collected data with just a Kinect and tripod. This reduced the risk of a robot creating a novelty effect with the crowd. Our goal was to collect data that emulated the societal comfort and normalcy likely when robots are extensively deployed. While some bystanders were curious, only one altered their course to speak with us after passing the tripod. The team manually shifted the tripod to a variety of positions and rotations prior to each train arriving. Data was collected across two consecutive mornings. In keeping with our focus on low-cost methods, all sensing and analyses utilized a regular Kinect RGB-D camera and a single 13” laptop (Intel Core i7 Processor and NVIDIA GeForce GTX 960M Graphics card).

We selected ten short clips that contained a variety of views, angles, and crowd densities. We biased selection towards clips containing denser crowds and challenging conditions since these are the scenarios we seek to address. Performance was measured by how many people were in each frame of each clip (Ground Truth) and how many were captured by the group formation and occlusion compensation. We calculated the percentage of people captured by each system on a per frame basis. Parameters determining the extent of an individual’s personal space and its dependence on velocity were experimentally examined by [17]. For our evaluation we focused on assessing the ability of our system to capture groups of moving objects and the extent of coverage provided by the occlusion compensation step.

The three methods used to capture moving people were the upper body and HOG detectors from the SPENCER project (Upper Body + HOG; [4], [5]), our approach described above but without the projected polygon (Detected), and our complete approach with the projected polygon

(Covered). Our implementation of the SPENCER method was adapted to support more direct comparison of approaches. Specifically, we (a) only used the Kinect as input to the SPENCER process since we lack a laser sensor, (b) only compared to the upper body and HOG detector layer of the SPENCER system as we have not yet added a tracker layer to our approach, and (c) set the maximum z distance to match our method.

Figure 6: Performance over real-world samples (±1 standard error).



6.2 Results

Figure 6 shows the performance of the three system options. It is readily apparent that the approach proposed here is much better at capturing moving people when compared to the Upper Body + HOG method. The Detected and Covered results are visually close but a statistical analysis using a t-test reveals a significant difference between the two ($t(594) = 2.85$, $p < 0.01$). The mean number of people captured per frame for the two methods were 77% and 80%, respectively. Note that the clips showed a variety of differences, which is most likely due to specific grouping and movement patterns in the scene.

A key concern is whether the described approach can continue to capture moving people as crowds get denser. To test this we compared the number of people captured (Covered) to the ground truth for each frame (Figure 7) in the ten clips selected for evaluation. There are points in the plot that appear to be a single point but are actually multiple points placed on one another, from multiple frames that have the same coverage for a given ground truth. An analysis showed a strong, significant, positive correlation ($r(298) = 0.814$, $p < 0.0001$), suggesting good performance. Figure 7 shows this graphically and illustrates good performance for higher numbers of people per frame.

Figure 7: Coverage vs. Ground Truth.

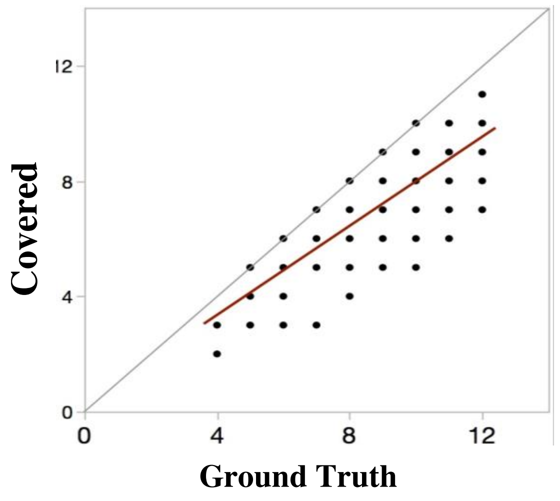
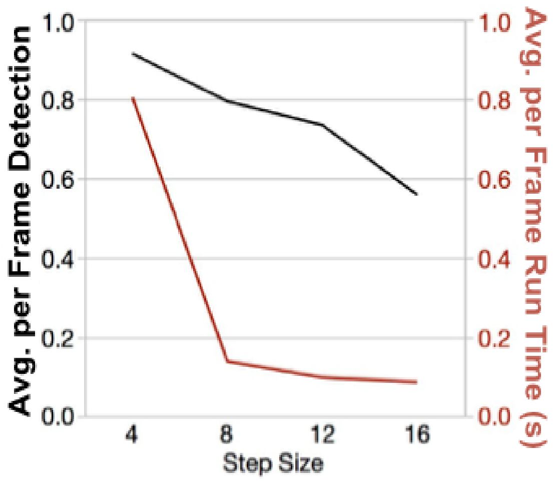


Figure 8: Performance vs. Step Size

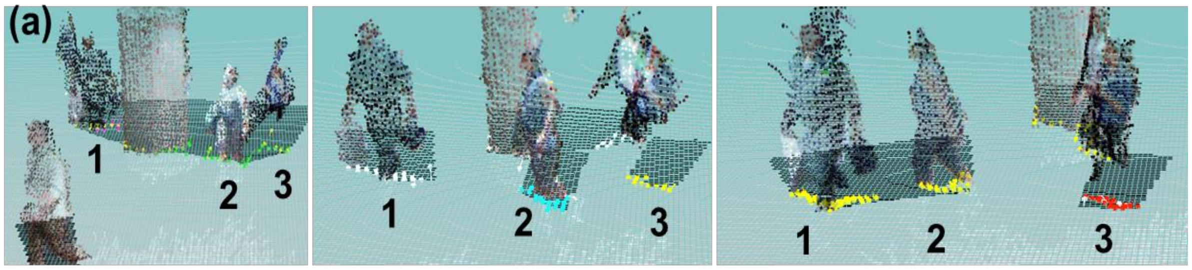


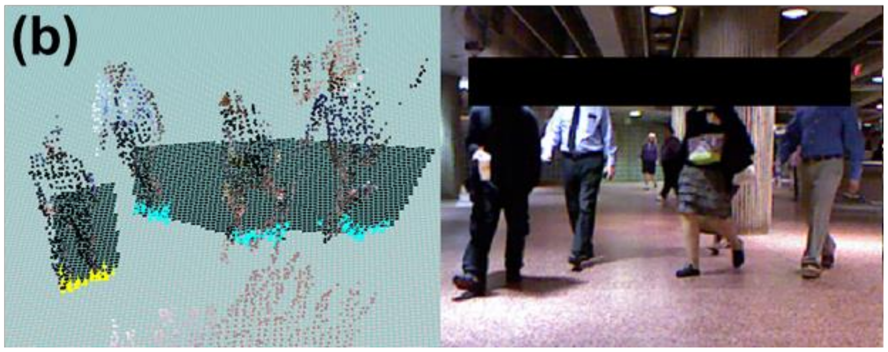
As mentioned earlier in section 3.2 Chapter 3, the step size used for 3D flow vector computation creates a tradeoff between detection and run time. A step size of 8 was used for the evaluation in this paper, but it appears values in the 8-12 range are reasonable (Figure 8). A step size 4 is likely reasonable with better hardware or further CPU and GPU parallelization. A step size below 4 may not be necessary since people missed at this value were typically those who were out of range.

6.3 Other Observations

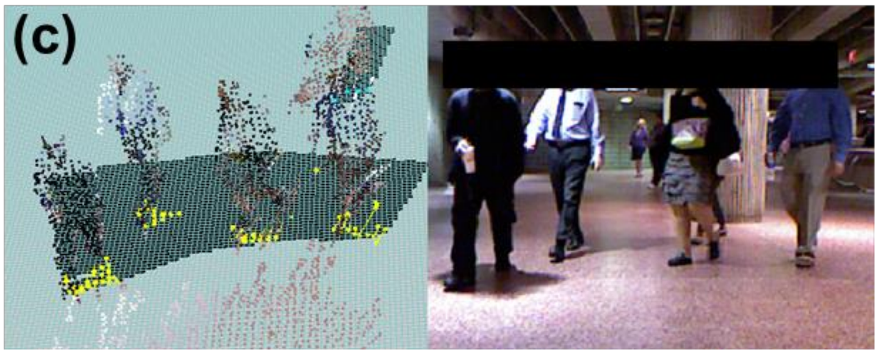
In addition to quantitative findings, there were multiple interesting examples that illustrated features of this approach. In the examples described here, each group has a common color for the leading edges, and the projected polygon (Covered) is shown as a grey shadowed region of the ground plane. Figure 9a shows 3 frames where person 2 split from person 3 and formed a group with person 1. Figure 9b and c show a lead person being caught by a group from behind. Figure 9d illustrates the ability to capture large groups of people, regardless of body angle, head angle, and layers in the group. Likewise, Figure 9e shows how the projected polygon captures the second layer of the group.

Figure 9a to d : Example Events showing qualitative performance on Transit Station Data

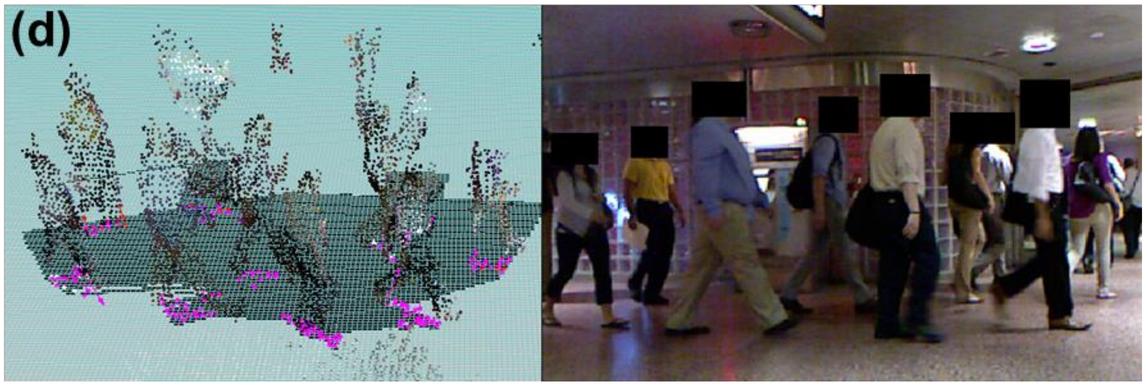




[redacted]

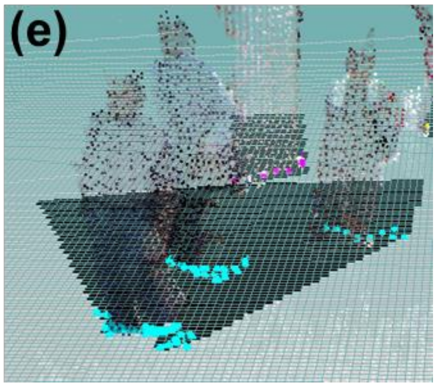


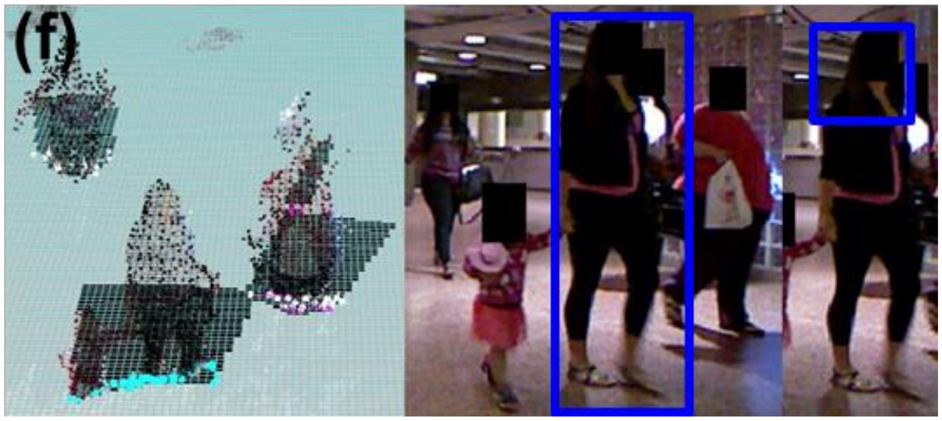
[redacted]

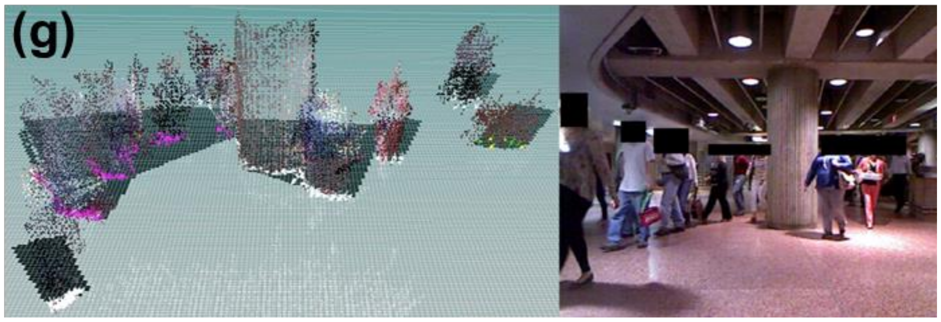


[redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]

Figure 9e to g: Example Events showing qualitative performance on Transit Station Data







Approaches that attempt to detect people before tracking (e.g., SPENCER) might have difficulty with atypical objects. Children often fall into this case since they are not average human-sized and typically absent from training data. Figure 9f shows an example where the SPENCER detectors captured an adult (blue boxes) but failed to capture a nearby child. Two adults in the background were also missed. Finally, Figure 9g illustrates the ability of the system to capture multiple groups and seal spaces within groups using the polygon extension method.

We observed that the SPENCER tracker could detect almost front facing people at far depth ranges, but had trouble detecting people who were densely packed and facing sideways or backward, even if the maximum allowable depth range was extended to accommodate all of the sensor data. This could be due to incomplete depth information for those people and the type of training template used. In contrast, our approach has no fixed template matching for capturing humans and it considers any moving set of 3D points seen by the sensor. Therefore, it is able to capture humans who are far away and within the field of view.

Chapter 7

Conclusions

7.1 Discussion

A. Algorithm Summary

In this paper we describe a method for low-cost perception of features of dense, moving crowds inspired by human perceptual processes. In contrast with other methods, this approach does not rely on prior training or attempt to capture individual humans in the robot's field of view (e.g., [4], [5]). Theoretically, this strategy trades off awareness of individual people for greater coverage, robustness to non-standard stimuli, and a variety of sensor configurations.

B. System Performance

The overall performance of the system was remarkably good, even in dense crowds where people were partially obscured behind the leading edge. Per frame performance for capturing moving people was around 77%.

Including the polygon shadow step successfully captured people obscured by the leading edge of the crowd and led to a statistically significant improvement in per frame performance (77% to 80%). Also, Coverage for each clip never dropped below Detection. In many cases, there were people who barely managed to be captured due to just 1 or 2 of their velocity vectors being seen. If the shadow extension was not applied, the plain cluster formation of leading edge points would not have captured many of these people.

Performance results for the approach described here compare favorably to the existing Upper Body + HOG steps of the SPENCER system [4], [5]. However, several scenario factors likely suppressed results for the SPENCER approach. As mentioned, laser data was not part of this evaluation and we only compared to a subset of the overall system. We also tested in just two parts of a single transit station. Results may change in other settings and room geometries. Also, our approach has no issues with moving objects that do not resemble adult humans (e.g., child in Figure 9f).

Discussions with the SPENCER team suggest the main differences in performance are due to their focus on having a high confidence on the detected object being a human, as is the case with many person detectors and trackers, whereas our approach does not require a strict differentiation between a moving human and any other moving object [18]. This aligns with our design principle of estimate, guess, and gamble to reduce perceptual and cognitive load. A major use case for the SPENCER project is escorting specific passengers within an airport. This requires detection and tracking of specific people. One SPENCER design decision was to suppress false positives, which is counter to our preference to suppress false negatives.

A related problem with our approach occurs when a person on the edge of one group is occluded by a person in another group ahead of them and is far enough to not be included in this person's polygon. The lack of semantic knowledge about moving objects makes it difficult to convert the partial perception of the trailing person into a full person-footprint. Approaches like SPENCER can handle this case. Another issue with using a non-semantic approach is that we cannot differentiate between any moving obstacle and humans. This works for crowd navigation but may not be well suited if use is extended to other HRI tasks.

7.2 Future work

The next step is to use learning to recover social reward functions while attempting to actually enter and move in the crowd. There are many possible influential environmental properties such as crowd density, relative velocity, and group factors like group width, group density, extent of the group's personal space etc. It will be necessary to perform short horizon predictions of likely positions and velocities of groups and simultaneous predictions of possible splitting/merging to estimate possible free/occupied space. Due to the potential complexity, we think a learning approach is the best strategy. This should help produce a smoother, more human-like path. Our long-term plans are to evaluate a fully implemented version of our approach on a publicly deployed robot. This evaluation used a static sensor to help disambiguate static objects from moving ones. Future versions of the system with moving camera might need to use map knowledge to differentiate static objects.

References

[1] M. B. Dias, A. Steinfeld, and M. B. Dias, “Future Directions in Indoor Navigation Technology for Blind Travelers,” in *Indoor Wayfinding and Navigation*, H. A. Karimi, Ed. Boca Raton, FL: CRC Press, 2015, pp. 203–226.

[2] B. Mutlu and J. Forlizzi, “Robots in Organizations: The Role of Workflow, Social, and Environmental Factors in Human-Robot Interaction,” in HRI ’08 Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction, 2008, pp. 287–294.

[3] M. Veloso, J. Biswas, B. Coltin, S. Rosenthal, T. Kollar, C. Mericli, M. Samadi, S. Brandao, and R. Ventura, “CoBots: Collaborative robots servicing multi-floor buildings,” in *IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 5446–5447.

[4] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore, H. Hung, O. A. I. Ramírez, M. Joosse, H. Khambhaita, T. Kucner, B. Leibe, A. J. Lilienthal, T. Linder, M. Lohse, M. Magnusson, B. Okal, L. Palmieri, U. Rafi, M. van Rooij, and L. Zhang, “SPENCER: A socially aware service robot for passenger guidance and help in busy airports,” in *Field and Service Robotics (FSR)*, 2015.

[5] T. Linder and K. O. Arras, “Multi-model hypothesis tracking of groups of people in RGB-D data,” in *International Conference on Information Fusion (FUSION)*, 2014.

[6] P. Trautman, J. Ma, R. M. Murray, and A. Krause, “Robot navigation in dense human crowds: Statistical models and experimental studies of human-robot cooperation,” *Int. J. Rob. Res.*, vol. 34, no. 3, pp. 335–356, Mar. 2015.

[7] F. Santoro, S. Pedro, Z.-H. Tan, and T. B. Moeslund, “Crowd Analysis by Using Optical Flow and Density Based Clustering,” in *European Signal Processing Conference (EUSIPCO)*, 2010, pp. 269–273.

[8] B. Kim and J. Pineau, “Socially Adaptive Path Planning in Human Environments Using Inverse Reinforcement Learning,” *Int. J. Soc. Robot.*, Jun. 2015.

[9] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, “The walking behaviour of pedestrian social groups and its impact on crowd dynamics,” *PLoS One*, vol. 5, no. 4, 2010.

[10] M. Vazquez and A. Steinfeld, “An assisted photography method for street scenes,” in *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV) 2011*, 2011.

[11] M. Vázquez and A. Steinfeld, “An Assisted Photography Framework to Help Visually Impaired Users Properly Aim a Camera,” *ACM Trans. Comput. Interact.*, vol. 21, no. 5, pp. 1–29, Nov. 2014.

[12] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” *Proc. 7th Int. Jt. Conf. Artif. Intell.*, no. x, pp. 674–679, 1981.

[13] B. K. P. Horn and B. G. Schunck, “Determining optical flow,” *Artificial Intelligence*, vol. 17, no. 1–3. pp. 185–203, 1981.

[14] G. Farnebäck, “Two-frame motion estimation based on polynomial expansion,” in *Image Analysis, Lecture Notes in Computer Science*, vol. 2749, 2003, pp. 363–370.

[15] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” in *Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.
[16] E. Hall, *The Hidden Dimension : man’s use of space in public and in private*. 1969.
[17] R. Kirby, “Social robot navigation,” Carnegie Mellon University, 2010.
[18] T. Linder, “Personal communication.” 2015.