Low Cost Perception of Dense Moving Crowd Clusters for Appropriate Navigation*

Ishani Chatterjee and Aaron Steinfeld, *Member, IEEE*

Abstract— **This paper describes an algorithm for rapidly and cheaply clustering humans moving in clusters during dense crowd conditions. The algorithm differs from other methods due to a focus on low cost, use of a single robot-mounted RGB-D sensor, and design choices driven by human perception and expectations of appropriate behavior. Use of a human-centered algorithm design process should lead to actions and decisions that are more aligned with human expectations since robots will mimic human perception and guesses during dense crowd motion.**

I. Introduction

A. 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 very 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). For example, a robot using this strategy while trying to rendezvous with a transit user who is blind [1] 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.

B. 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 can be very effective at tracking crowd motion but this constrains robot deployments to locations where this is feasible economically, politically, and geometrically. The latter is especially relevant in long hallways that have low ceilings, which is common in certain architectural settings (e.g., transit stations).

C. 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 have chosen the latter for this effort since this creates an inherently more scalable and generalizable solution. 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.

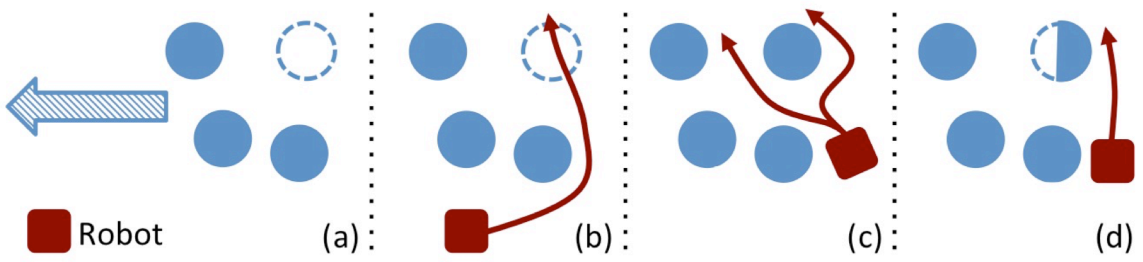
When humans need to move through a crowd they utilize certain strategies to reduce complexity and cognitive load. First, it is infeasible for humans to track every person simultaneously. Instead, humans watch group motion and utilize *gestalt* effects to abstract crowd dynamics into a

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

I. Chatterjee is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: ichatter@andrew.cmu.edu).

A. Steinfeld is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA (phone: 412-268-6346; fax: 412-268-6436; e-mail: steinfeld@cmu.edu).

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).



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 approximate state-based behaviors. We (a) look for a seam or gap in the crowd, (b) enter or cross it, and (b) follow the person in front of us, when needed. 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, humans do not bother with trying to be perfect. They estimate, guess, and gamble when moving around crowds. 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.

II. 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 of our computer vision 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 tracking more than 10 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 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 (e.g., Figure 1c). This space should be proportional to the robot's size and ability to brake or swerve.

III. Prior 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. They detect groups by first finding individual people, 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. This is not to say the authors approach is incorrect. It is just a different strategy that attacks the problem from a high-resource direction that can afford to track as many people as possible. Instead, we are exploring a low-cost, Gestalt-oriented perspective since some robots will not have the necessary sensors or computational resources required for the described approach.

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 teleoperation 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 tracking was done using motion models with nearest neighbor being used for data association. 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 needs since (a) our observation is limited to local frontal view with frequent occlusion, rather than a static global overhead view, and (b) we would like to account for more than five individuals.

The use of optical flow and density-based clustering to detect and analyze crowd movements has also been used [7]. However this work was limited to the clustering of 2D displacement vectors of pixels found in terms of 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; with a closer camera the thresholds have to increase 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 between tracked points 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 clusters having similar motion features like velocity and density. Instead, the authors extracted values of these features for each grid-cell, the weights of which were learned using Inverse Reinforcement Learning. They then assigned a cost to each cell based on the values of features in each grid cell and the planner performed a minimum-cost graph search. This may lead to problems when groups of people split or join, which can be common in 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.

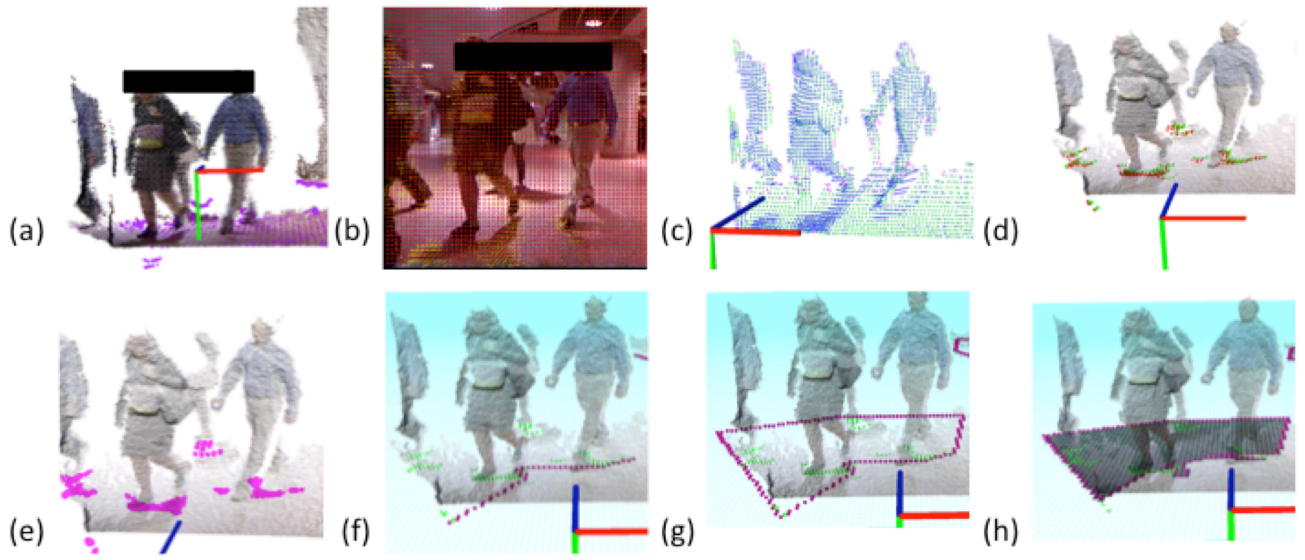
IV. System Design And Pipeline

Our system is implemented as ROS nodes that subscribe to and publish on various topics. The primary source comes from the RGB-D Kinect driver, which publishes a 3D depth-registered point cloud formed by combining the RGB and depth information of points, that contains the 3D x, y, z distances of points in meters as measured in the sensor coordinate system. The inputs to the perception system are the RGB image messages and the depth-registered point cloud messages that are matched to their respective RGB images. The overall pipeline of the system is shown in the flow diagram of Figure 2.

A. Ground Plane Estimation

One of the first steps performed necessary for building a 2D occupancy grid and planning in a planar stretch is detection of the ground plane (Figure 2a). A commonly used algorithm to find inliers belonging to a plane is RANSAC. The current input cloud is downsampled to a 3D grid of voxels to decrease computation while performing RANSAC and filtered by a height that is approximately a little less than the height of the physical ground plane measured vertically down from the sensor coordinate frame so that a majority of the points remaining in the cloud to be considered for plane-finding are those belonging to the ground plane. The algorithm finds the major plane whose normal is within a

Figure 2. System pipeline



certain threshold angle made with the Kinect y (vertical) axis which is set considering the maximum possible angle of incline of wedge-shaped walks found commonly in public places like malls. If the Kinect is placed with perfect zero roll (rotation about its x-axis) then this can be replaced by an equation of the form y=k in Kinect coordinates. However, on-line estimation can be helpful if the Kinect observes a plane in advance that has some incline. Current implementation takes the average of the first 10 valid plane coefficients for reducing errors in the plane equation. This can be repeated for every few frames to keep updating the ground plane. Figure 2a shows the points in the 3D point cloud belonging to the detected ground plane in purple.

B. 3D Flow Vector Computation

To detect and track crowd motion we first compute the 2D optical flow, which finds correspondences between image pixels of two consecutive frames, and then extract the corresponding 3D points from the previous and current point clouds to store them in new clouds known as optical flow clouds (current and previous). In Figure 2b, the 2D optical flow vectors can be seen as yellow arrows. Subtracting the 3D x, y, z coordinates of previous cloud from that of the current cloud can give the 3D flow vectors. Figure 2c shows the 3D flow vectors in bright blue for the captured people.

In order to overcome accuracy limitations that accompany the Lucas-Kanade Method and Horn and Schunck's 12, 13] method for 2D optical flow, 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 coordinates, e.g., in depth values at boundaries and edges.

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 when run on a GPU and displayed reasonable real-time speed with robustness to noise. To decrease computation, the 3D points are picked for every 'nth' image pixel instead of all pixels where n is a parameter; therefore the computed 3D flow vectors are partially dense.

However, quite a few of these points do not have valid depth values due to various factors, which is represented in the point cloud as *Not a Number* (NaN) for all the 3 dimensions x, y, z. Dropping these points straightaway results in decreased number of valid points for velocity calculation. Therefore, a NaN point is replaced by a valid neighbor with the aim of getting increased range and quantity of velocity vectors. This valid neighbor search cannot be performed among 3D points because all the three dimensions are NaN. However, the image pixel corresponding to a 3D NaN point always has a valid RGB value and replacing this NaN point by the 3D x, y, z of a similar image pixel could solve the purpose. The following algorithm is used to address the NaN issue:

- Find the pixel in the RGB image that generated the NaN point.
- Find all the neighboring pixels around it in a small k *x* k window centered on this pixel, where k is a parameter. The constraint of spatial closeness while deciding similarity is taken care of by the small window size.
- Find the most similar pixel to the point by choosing the neighbor that is the closest in terms of Euclidean distance between the L*, a*, b* values. The L*a*b* color space is a 3D color space defined by CIE, with 'L*' axis representing Lightness or Luminance from black (bottom) to white (top), 'a*' axis representing values from green (-a) to red (+a) and 'b*' axis from blue (-b) to yellow (+b). The 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.

- If all the neighboring pixels in the window have NaN 3D values indicating that the point is actually out of range (too close or too far) then do not replace and just omit the point.

Figure 2c shows the replaced points in the 3D cloud in purple that were originally NaN.

C. Filtering Out Points Above Robot Height

Once the 3D optical flow clouds (current and previous) have been formed, we discard the points above the robot height from both of these since anything above the robot height should not be considered as an obstacle. The remaining points are projected on the ground plane to find the projected flow vector (velocity) in 3D associated with every point in the optical flow cloud. Figure 2d shows the projected points on the ground plane, with red being those from the current point cloud and green from the previous. Each green point is associated with a red point, thereby marking the start and end points of the projected velocities.

D. Density Based Clustering of 3D Flow Vectors

Our work differs from Kim, et al [8] since the former treats points having similar projected velocities as one group. This grouping 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 features: (1) 3D Euclidean distance between their originating points, (2) absolute difference in angle between their directions, and (3) absolute difference between their magnitudes. All the three features have equal weighting in deciding similarity. The physical distance threshold is considered to be the robot width, as there is no use of treating points with similar velocity magnitude and direction 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. Figure 2e shows four people being grouped into a single cluster, as shown by the pink velocity vectors.

E. Mapping to a 2D Occupancy Grid

Once the projected flow cloud has been divided into separate smaller clouds, each containing one cluster, the next step is to utilize this clustering information to indicate occupancy in the obstacle map and to define the boundary of the cluster. The overall velocity of the cluster is the average of all flow vectors.

In order to prepare the occupancy grid, a coordinate frame is fixed on the ground plane to serve as the global frame. (The transform between the Kinect frame and the global frame is handled in ROS.) The ground plane is divided into square grids with each grid cell being identified by its center. Each of the clustered points is transformed to global coordinates from sensor coordinates using the published transform and mapped to the grid cell it falls within. Upon completion we have the grid centers in world coordinates representing these clusters.

F. Estimating the Shape of Each Projected Cluster

To define the boundary of the cluster, we find the leading visible edge and extend the extremes along the shadow rays of the Kinect to occupy some space behind the leading edge to manifest the design principle illustrated in Figure 1. The extremes of the leading visible edge are found by finding the minimum and maximum x/z ratio in sensor coordinates, which is an indicator of the angle formed by a point with the Kinect z-axis. Since the horizontal FoV of Kinect is about 60°, the following special case need not be considered which can occur at the boundary of -180° and 180° measured from the Kinect's positive x-axis.

The leading edge is determined as described. First, the concave hull of the grid cell centers is found to preserve boundaries, then the concave polygon is made continuous by ray-tracing the consecutive grid cell centers that form the line segment between 2 vertices. In our case, the ray-tracing algorithm used is Bresenham's line algorithm. For each x coordinate of grid centers involved, the minimum z coordinate is found to estimate the leading visible edge from Kinect. This is shown in Figure 2f as red points that are consecutive cell centers.

Out of the two extremes, the one with greater distance from sensor origin is considered and the edge is extended by 1.5 times the diagonal width of a walking human. This parameter was initially chosen to provide personal space and reflect CoBot's size and maneuvering capability. For example, to accommodate one row of humans immediately behind the leading edge, the extension should be performed such that any orientation of the human is accommodated behind. Therefore the extension is decided to be of the form close to a pixelized arc. The entire boundary is formed by joining the concave hull representing the leading edge and the convex extension behind (Figure 2g). This is then filled in to obtain all grid centers lying inside the shape, which are then marked as obstacles in the occupancy grid (Figure 2h).

V. Discussion

A. Summary

In this paper we describe an algorithm for rapidly and cheaply clustering humans moving in clusters during crowd conditions. The algorithm differs from other methods due to a focus on low cost, use of only an RGB-D sensor, and design choices driven by human perception and expectations of appropriate behavior. This should lead to actions and decisions that are more aligned with human expectations since robots would be mimicking human perception and guesses during dense crowd motion.

B. Limitations

The low-cost Gestalt approach described here relies on numerous simplifications in order to lower computational cost and sensor requirements. Any degree of abstraction in computer perception can create problems during edge cases and unexpected events. Since we do not track each person independently, like prior efforts [5, 6], our approach is incapable of using human detection for other HRI tasks. Also, our method is similar to a detection-by-tracking method [16], and thus will likely encounter similar challenges and limitations. By simplifying our approach we specialize the system and limit opportunities for richer interaction. This is acceptable for our overall effort since we are focused on techniques for robots that have a narrow set of tasks due to their cost or intended mission.

There are also ways to improve our system design. For example, it is possible to foresee crowd configurations where concave boundaries may be filled in by this algorithm, thereby marking small open regions as occupied. It is not clear yet whether this is a significant problem. Future data collection will help determine whether modifications to this part of the algorithm are necessary. Analysis of how this manifests in naturalistic crowd data, if it does, will guide which approach to use.

C. Future Work

The next steps for this project are collection of naturalistic dense crowd motion in real-world settings (e.g., transit stations), evaluation of perception performance, incorporation of navigation planning, and implementation on an autonomous robot. We will use this data to develop methods for predicting how groups in crowds will move, split, and merge. It is not clear what the correct method is for these challenges, but we expect some degree of machine learning will be needed. The work by Linder and Arras [5] on tracking individual people within crowds suggests this will be an effective and useful approach. Likewise, Kitani, et al [17] have been able to demonstrate excellent performance with predicting human movement with only one camera using machine learning techniques.

We are also expecting to extend our architecture to draw on the aforementioned human behavior and perception strategies for crowd navigation, like state control of crossing versus following, for reducing computational load while moving. Path planning and following increase computational demand above the initial task of finding a seam or gap, so finding ways to lower computation after identifying a good path may become necessary.

REFERENCES

[1] Dias, M. B.; Steinfeld, A.; Dias, M. B. "Future Directions in Indoor Navigation Technology for Blind Travelers," in Hassan A. Karimi (Ed.) *Indoor Wayfinding and Navigation*, Boca Raton, FL: CRC Press, pp. 203-226, 2015.

[2] Mutlu, B.; Forlizzi, J., "Robots in organizations: The role of workflow, social, and environmental factors in human-robot interaction," in *3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 287-294, 12-15 March, 2008.

[3] Veloso, M.; Biswas, J.; Coltin, B.; Rosenthal, S.; Kollar, T.; Mericli, C.; Samadi, M.; Brandao, S.; Ventura, R., "CoBots: Collaborative robots servicing multi-floor buildings," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5446-5447, 7-12 October, 2012.

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

[5] Linder, T.; Arras, K. O., "Multi-model hypothesis tracking of groups of people in RGB-D data," in *17th International Conference on Information Fusion (FUSION)*, pp. 1–7, 2014.

[6] Trautman, P.; Ma, J.; Murray, R.; Krause, A., "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *International Journal of Robotics Research*, 2015.

[7] Santoro, F.; Pedro, S.; Tan, Z.; Moeslund, T., "Crowd analysis by using optical flow and density based clustering," in *18th European Signal Processing Conference (EUSIPCO-2010)*, 2010.

[8] Kim, B.; Pineau, P. "Socially adaptive path planning in human environments using inverse reinforcement learning," *International Journal of Social Robotics*, 2015. http://dx.doi.org/10.1007/s12369-015-0310-2

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

[10] Vázquez, M.; Steinfeld, A., "An assisted photography method for street scenes," in *IEEE Workshop on Applications of Computer Vision (WACV)*, pp.89-94, 5-7 January, 2011.

[11] Vázquez, M.; Steinfeld, A. "An assisted photography framework to help visually impaired users properly aim a camera," *ACM Transactions on Computer-Human Interaction*, vol. 21, no. 5, Article 25, 2014.

[12] Lucas, B. D.; Kanade, T., "An iterative image registration technique with an application in stereo vision," in *Proc. Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, Vancouver, Canada, August 24-28. 1981, pp. 121-130.

[13] Horn, B. K. P.; Schunk, B. G., "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185-203, March 1981.

[14] Farneback, G., "Two-Frame Motion Estimation Based on Polynomial Expansion," *Lecture Notes in Computer Science*, vol. 2749, pp. 363-370, 2003.

[15] Ester, M.; Kriegel, J.; Sander, J.; Xu, X., "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 2-4 August, 1996.

[16] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2008.

[17] Kitani, K. M.; Ziebart, B. D.; Bagnell, J. A.; Hebert, M., "Activity forecasting," in *Proc. 12th European Conference on Computer Vision - Volume Part IV (ECCV'12*), 2012.