STALKERBOT: Learning to Navigate Dynamic Human Environments by Following People

Liz Murphy and Peter Corke
CyPhy Lab, Queensland University of Technology, Australia
firstname.lastname@qut.edu.au

Abstract

Service robots that operate in human environments will accomplish tasks most efficiently and least disruptively if they have the capability to mimic and understand the motion patterns of the people in their workspace. This work demonstrates how a robot can create a human-centric navigational map online, and that this map reflects changes in the environment that trigger altered motion patterns of people. An RGBD sensor mounted on the robot is used to detect and track people moving through the environment. The trajectories are clustered online and organised into a tree-like probabilistic data structure which can be used to detect anomalous trajectories. A costmap is reverse-engineered from the clustered trajectories that can then inform the robot's onboard planning process. Results show that the resultant paths taken by the robot mimic expected human behaviour and can allow the robot to respond to altered human motion behaviours in the environment.

1 INTRODUCTION

A robot navigating in an environment densely populated by people can benefit from an awareness of human behaviour that is not easily gleaned from standard sensors. It is easy to envisage situations in which people learn by observing those around them and alter their behaviour accordingly; if something is spilt on the floor, the crowd will divert around it, or if the flow of human traffic is down the left hand side of a corridor it would be a foolish person who chooses a path down the right. Incorporating this sort of information into navigational maps used by robots is near impossible when relying on sensors like vision and laser alone; instead what is most often used is an occupancy grid representation where all open space is viewed as equally good. In this paper we propose an online method of creating human-centric navigational maps by following people through the environment.

The problem of interaction between humans and robots in cluttered environs is well studied [Thrun *et al.*, 1997] [Burgard *et al.*, 1999] [Nourbakhsh *et al.*, 2003] and the ability of mobile robots to navigate and localize in areas populated by humans is a mature capability. In recent times, the focus has shifted to equipping robots with 'socially acceptable' behaviours [Bennewitz *et al.*, 2005] [Ziebart *et al.*, 2009] [Müller *et al.*, 2010] [Tipaldi and Arras, 2011] so as to impinge less on the human environment. Both humans and robots can benefit from a robot's increased awareness of human behaviours; the prospect of a near-collision with a robot is a startling experience for a human — the incidence of which we'd like to reduce, and for the robot it is suboptimal as it usually requires recovery behaviours and diversions from its planned path. However, learning about the environment from humans presents a complex challenge — human behaviour is highly dynamic, the representation of trajectories through an area potentially involves large amounts of data, and any solution needs to be flexible enough to incorporate both deviations in individuals' behaviours as well adapt to temporal changes of average behaviours. In this work we equip a robot with the tools to take human-like paths, and to adapt to changes in human behaviour online. The robot builds up a representation of human trajectories online by detecting, tracking and physically following people through the environment. It can then use this information to generate a navigational map, which can then be used by a standard planning algorithm (such as A*) to quickly plan paths online. What differentiates our approach most from existing approaches is the online learning aspect, as well as the trajectory gathering using a moving platform.

This paper makes 3 key contributions to the problem of learning human preferred paths through the environment. The first is a *People Tracker* ROS package that leverages the OpenNI [Ope, 2011] library's open source Skeleton Tracker to allow a mobile base to safely and per-

Figure 1: The experimental platform, a Mobile Robots Guiabot with Kinect sensor mounted on top of the touch screen

sistently follow a person at normal walking pace. The second is an online clustering technique that builds a skeletal representation of the trajectories taken by people through an area. Its probabilistic foundation means that it requires limited tuning, and can incorporate individuals different walking speeds easily. The final contribution is a map creation tool that takes the trajectory clusters and builds a navigational map that is heavily biased towards locations that people walk on in an environment. Coupled with the clusterer, we have a fast online map generation process that can quickly adapt to changed behaviours of people in the environment. Both the clustering and map creation processes have been implemented under ROS and the code is available from our lab's ROS repository at `http://www.ros.org/wiki/cyphy`.

2 Related Work

The process of detecting and tracking humans has received much attention in both the robotics and computer vision literature. It has been done using a variety of sensors such as LADAR [Arras *et al.*, 2007] [Mozos *et al.*, 2010] [Navarro-Serment *et al.*, 2010], vision [Siebel and Maybank, 2006] [Schlegel *et al.*, 1998], and more recently using RGBD sensors such as the Microsoft Kinect or PrimeSense PrimeSensor [Shotton *et al.*, 2011] [Spinello and Arras, 2011]. Tracking methods include Kalman filters [Azarbayejani and Pentland, 1996], multi-hypothesis tracking (MHT) [Luber *et al.*, 2009] and joint probabilistic data association filters (JPDAF) [Schulz *et al.*, 2003], which rely on using blob detection to determine the human torso in images or laser scans, or by detecting and tracking legs. More recently, skeleton tracking has been implemented on the Kinect using a machine learning algorithm trained on hundreds of thousands of human poses [Shotton *et al.*, 2011] which tracks the 3D positions of human joints from frame-to-frame using a depth image sequence.

In this work we utilize the OpenNI framework, a set of open source APIs which constitute a combined person detector and tracker compatible with the Microsoft Kinect [Ope, 2011].

People following for mobile robots has been implemented in various ways. In [Kirby *et al.*, 2007] two methods of following people were compared: *direction following*: where the robot drives directly towards the person's location; and *path following* where the robot attempts to replicate exactly the path taken by the person. Qualitative survey results found that human subjects in the experiment found *direction following* to be a more human-like behaviour. Direction following is a form of pure pursuit tracking [Coulter, 1992], and was also implemented in [Hemachandra *et al.*, 2011] to follow a tour guide in an office environment.

A number of previous approaches to creating navigational maps from people tracking exist in the literature. In [Bennewitz *et al.*, 2005] a collection of trajectories is learned by observing motion patterns between places that people stop for long periods of time. These trajectories clustered into motion patterns using an expectation maximization technique. Hidden Markov Models derived from these learned patterns are used to maintain a belief over the location of people. In [Ziebart *et al.*, 2009], maximum entropy inverse optimal control uses the goal-directed behaviour of pedestrians to learn a cost function that best explains their previous trajectories. Because the cost function maps features computed from the environment to cost, it exhibits resilience to changing configurations of obstacles in the environment. In [Tipaldi and Arras, 2011] a *spatial affordance map* uses a non-homogenous spatial Poisson process to represent human activity and uses this to plan paths in time and space that maximize the likelihood of encountering people. A navigational map is built in [O'Callaghan *et al.*, 2011] using Gaussian Processes to learn a function that describes how peoples' motion deviates from a shortest path prior. Distinct from these approaches, where trajectories are learned using fixed cameras or laser range finders or simulation; our approach seeks to identify trajectories on board the robot.

A critical part of this work is the notion of clustering similar trajectories together in order to make the map creation process computationally tractable. An experi-

mental evaluation of similarity measures and clustering methodologies used in the computer vision community is provided in [Morris and Trivedi, 2009]. In this work we build on the work of [Piciarelli and Foresti, 2006], chosen primarily because it employs a distance measure that allows existing clusters to be compared with incomplete trajectories, as is the case when the robot begins to follow a person.

3 Method

Creating a navigational map is broken up into 3 subtasks; *people following*, *trajectory clustering* and *map creation*.

3.1 People Following

The first step is to obtain a set of trajectories by detecting people walking through the environment and then following them to obtain a trajectory. To detect and track people, we use the API provided by OpenNI [Ope, 2011] to interface to the *User Generator* middleware that generates a representation of a body in the 3D scene. This allows us to pick out the location of a given user's torso on a frame-to-frame basis. Although early versions of the OpenNI skeleton tracker required users to 'surrender' to the kinect in order for tracking to begin, recent versions allow the saving and loading of user calibration files. We have found loading a configuration file for a single user at the start of operation to (anecdotally) work well in detecting other people in the environment.

We then use a pure pursuit approach [Coulter, 1992] to follow the person. It operates by calculating an error term

$$e = \sqrt{(x^* - x)^2 + (y^* - y)^2} - d^* \tag{1}$$

which is the difference between the desired following distance d^* and the current distance of the robot from the person at offset (x, y) in the robot frame.

From this, we use a basic Proportional-Integral controller with gain terms K_i, K_e to set the robot's desired forward velocity

$$v^* = K_v e + K_i \int e dt \tag{2}$$

The bearing of the person relative to the robot is

$$\theta^* = \tan^{-1} \frac{y^* - y}{x^* - x} \tag{3}$$

and the difference between that and the robot's current heading θ is used to set the angular velocity

$$\alpha = K_h(\theta^* \ominus \theta) \tag{4}$$

with a proportional controller gain $K_h > 0$ and where $\ominus$ denotes the smallest difference on $\mathbb{S}$. Given the limitations of the Kinect sensor, which has a tracking range of 0.8–3.5 metres and a horizontal angular field of view of view of 57° [Pri, 2011], we chose a set point of 1.5 metres behind the person being tracked. The gain K_h on the angular correction term is set to 2.0 — relatively high compared to K_v and K_i which are 1.0 — as the relatively narrow horizontal field of view of the Kinect means the robot needs to be able to turn quickly to keep the person in the frame and maintain tracking. We capped forward velocity at 1.2 ms^{-1} and angular velocity at 0.8 rad s^{-1} for safety reasons, and implemented the people follower as a ROS package.

3.2 Trajectory Clustering

The trajectory clustering algorithm is based on [Piciarelli and Foresti, 2006], but modified to deal with the trajectories being sourced from a mobile platform rather than from fixed downward-looking overhead cameras as in the original paper. We recap the basis of the algorithm here.

Central to the algorithm is the notion of raw *trajectories*, which embody the instantaneous locations (t_x^i, t_y^i) of the person being followed at time i, and *clusters* which aggregate together similar trajectories in a probabilistic representation $(c_x^j, c_y^j, c_{\sigma^2}^j)$ at time j.

The algorithm has two parts: *tree building* and a *tree maintenance* phase. The former is depicted as a state machine in Figure 2.

- A **New Trajectory** is considered to appear on start up, or when a significant discontinuity appears in the input to the clusterer (we assume trajectories are continuous in space, and not necessarily time, and that the robot will move to find a new person to follow after following a person to the endpoint of a previous trajectory). We allow a new trajectory T to reach a minimum size l_{new}, and then compare it to existing branches C in the cluster tree using a distance measure

$$D(T, C) = \frac{1}{n} \sum_{i=1}^{n} d(t_i, C) \tag{5}$$

where

$$\begin{aligned} d(t_i, C) =& \frac{dist(t_i, \hat{c})}{\sigma_j^2} \\ \hat{c} = c_j \text{ s.t. } j =& \underset{j=1}{\overset{n}{\operatorname{argmin}}} dist(t_i, C_j) \end{aligned} \tag{6}$$

and $dist(t_i, c_j)$ is the Euclidean distance from point t_i on the trajectory to a point c_j on the cluster. Piciarelli [Piciarelli and Foresti, 2006] used a sliding temporal window to allow the most recent point in the trajectory to be fitted to the closest point in the cluster given that walking speeds vary between people. Due to the cluster pruning process we employ

here, we found there was negligible computational penalty incurred by continually finding the closest point on the cluster instead — and that this proved more robust. If the distance between the new trajectory and the closest existing cluster is found to be less than some threshold level Dtc_{Thresh} then we begin updating the matching cluster. Otherwise, we start creating a new cluster.

- In the **Creating** state, points from the person's trajectory are continually added to a new, temporary cluster. The distance between the last point added to the cluster and the penultimate point is continually monitored, and if it exceeds a threshold level $Step_{Thresh}$ we assume a new trajectory has begun. New clusters are thus added to the cluster tree in a *delayed* fashion. Once we have a complete trajectory we prune it using a Mahalanobis distance between the current point under evaluation and the last point added to the cluster, and a Chi-Squared threshold test that ensures new points are only added to the cluster if there is a less than 90% chance they were derived from a distribution of variance σ_o^2 centred on the last point added.

$$\begin{bmatrix} d_x^i \\ d_y^i \end{bmatrix}^T \begin{bmatrix} \sigma_o^2 & 0 \\ 0 & \sigma_o^2 \end{bmatrix} \begin{bmatrix} d_x^i \\ d_y^i \end{bmatrix} < \chi^2_{p=90\%} \quad (7)$$

where (d_x^i, d_y^i) is the difference between point i on the trajectory currently under evaluation and the last point added to the newly-formed cluster. The pruned trajectory is then added to the cluster tree, and all variances are initialized to a set value σ_o^2.

- While **Updating** an existing cluster the incoming trajectory is used to update, in a weighted average, the closest point $\hat{c} = (\hat{c}_x, \hat{c}_y)$ on the existing cluster

$$\begin{aligned} \hat{c}_x &= (1-\alpha)\hat{c}_x + \alpha t_x^i \\ \hat{c}_y &= (1-\alpha)\hat{c}_y + \alpha t_y^i \\ \hat{c}_{\sigma^2} &= (1-\alpha)\hat{c}_{\sigma^2} + \alpha(dist(t_i, \hat{c}))^2 \end{aligned} \quad (8)$$

The parameter $\alpha \in [0, 1]$ allows the rate at which clusters fit to newly detected data to be moderated. The trajectory-cluster distance of Equation 5 is continually monitored. If it exceeds a threshold level $Drift_{thresh}$, or the end of the trajectory is reached, we clear the trajectory and move to the **Splitting** state.

- In **Splitting** we check to see if there are any child nodes of the previously-matched cluster. If not, we immediately create a new child cluster and transition to the **Creating** state. Otherwise, we delay comparing the offshoot of the newly split trajectory with existing child nodes until the new trajectory reaches size l_{new}. At this point the trajectory-cluster distance of Equation 5 is again employed and the closest matching child cluster less than threshold Dtc_{Thresh} is selected for **Updating**. If no child clusters are matched, a new child cluster is created and we enter the **Creating** state.

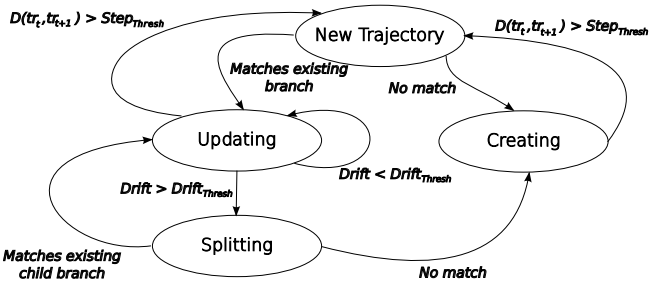


Figure 2: Tree building state machine

$\mathbf{Dtc_{Thresh}}$	10.0
$\mathbf{Step_{Thresh}}$	1.0
$\mathbf{Drift_{Thresh}}$	5.0
$\sigma_\mathbf{o}$	0.3

Table 1: Threshold values used in trajectory clustering

While *tree building* operates on a frame-by-frame basis, *tree maintenance* occurs only periodically. It involves 3 operations:

- **Merging** traverses levels of the tree and uses a cluster-cluster variant of Equation 5 to compare the distance between sibling clusters. Should it be less than a threshold d_{sib}, then a weighted average of the two clusters c_1 and c_2 is taken

$$\begin{aligned} [c_x^{merged}, c_y^{merged}] &= \mu[c_{x1}, c_{y1}] + \nu[c_{x2}, c_{y2}] \\ c_{\sigma^2}^{merged} &= \mu^2\sigma_1 + \nu^2\sigma_2 \end{aligned} \quad (9)$$

where

$$\mu = \frac{k}{\sigma_1}; \quad \nu = \frac{k}{\sigma_2}; \quad \text{and } k = \frac{\sigma_1\sigma_2}{\sigma_1 + \sigma_2}.$$

Any child nodes of the merged clusters are reparented to the new merged cluster.

- **Concatenation** joins clusters together in the case where a parent node has only one child cluster.
- **Pruning** gives us the option to remove clusters from the tree that have not been recently updated.

The clustering algorithm has been implemented in C++ under ROS. Threshold values used in the generation of the results for this paper are given in Table 1.

3.3 Map Creation

The map creation process is akin to an inverse Occupancy Grid building process, and is outlined in Algorithm 1. It is a fast, online technique. Each time

a new cluster tree arrives the existing map is cleared. Each node of a cluster c^n is a 2D probability distribution $\mathcal{N}([c^n_{x_i}, c^n_{y_i}], c^n_{\sigma_i^2})$ describing the likelihood of people traversing the location centered at $[c^n_{x_i}, c^n_{y_i}]$. Clusters that represent popular trajectories will exhibit low variance in the nodes.

We want the people-centric costmap to place low costs on areas of the map commonly traversed, and high costs elsewhere. Our observations of people correspond to areas in which we want low cost. This is diametrically opposite to the standard occupancy grid mapping problem where observations (eg laser returns) are indicative of high cost regions, and it is the unobserved areas along the line-of-sight to the obstacle that have their *costs* (proportional to likelihood of occupancy) reduced. Essentially, what algorithm 1 implements is half of the occupancy grid mapping process that results in areas in which our observations fall having their costs reduced. We generate observations by drawing n samples from each cluster node distribution. The map creation process is also implemented in C++ as a ROS process.

Algorithm 1 People Map Creation

```
for all clusters c do
  for all nodes in cluster n do
    Generate k samples from N(μ_n, σ_n)
    for all s=1 to k do
      (x_k, y_k) ←QUANTIZE(s_x, s_y)
      Map[x_k, y_k] ← Map[x_k, y_k] + m_lfree − m_lo
    end for
  end for
end for
```

4 Experiments and Results

Experiments were carried out using a MobileRobots GuiaBot shown in Figure 1. A set of 16 different trajectories were gathered, this raw data is overlaid on a floor plan of the experimental area in Figure 3. All trajectories emanate from roughly the same point, an area of 2 metres diameter at the exit to the lift shaft. They are uni-directional, radiating away from this point to 6 different locations on the floor. Figure 4 shows how the costmap and clusters are incrementally built up with the arrival of new trajectories. Although this is a small dataset, our final map in Figure 4(h) already shows significant areas corresponding to high foot-traffic. Also notable is that the raw dataset comprises 5929 location points, but the final set of 9 clusters has a total of 261 points. It is this skeletal representation of the trajectory data that means costmap creation can be done on-the-fly.

Figure 5 compares the results of planning paths from the lift exit to 3 different locations on the floor using

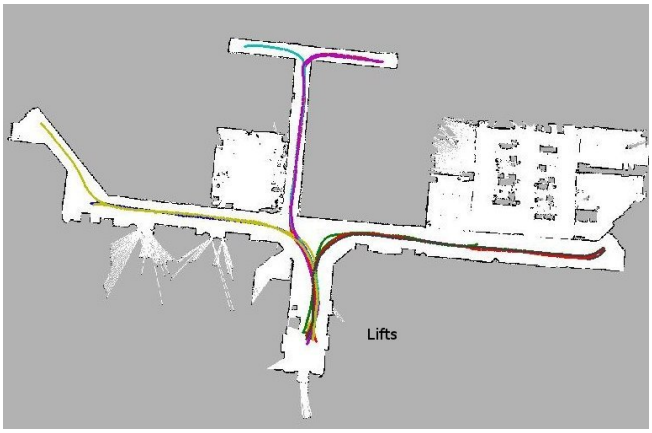
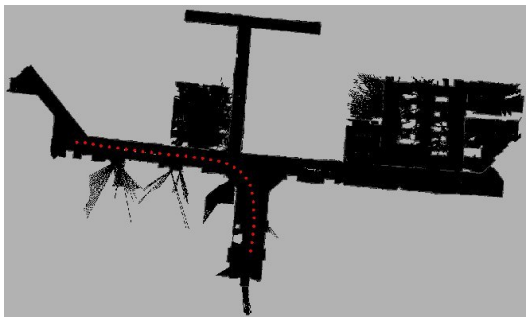


Figure 3: The raw trajectories taken by people

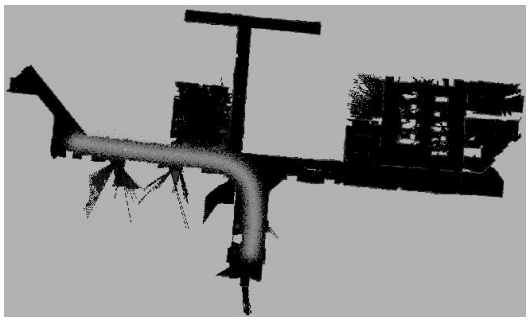
(a) our people-centric map, shown in green, and (b) the default navigation stack in ROS that makes use of Occupancy Grids and inflated obstacles, shown in red. There are notable differences in the paths, our costmap has successfully captured the common 'channels' that people walk in on the floor, and this reflected in the plans.

5 Conclusions and Future Work

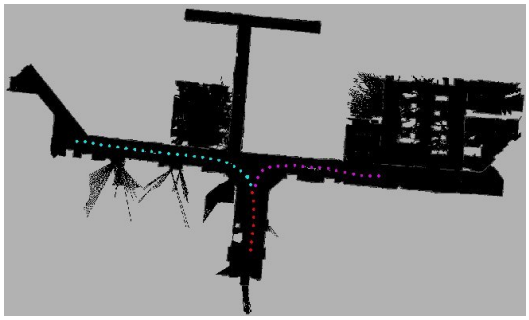
There are a number of obvious ways in which the work presented here could be extended. The small dataset shows that this is a viable method, but more data needs to be gathered to test the robustness of the algorithm and to converge on suitable threshold parameters as described in section 3.2. The variance (σ^2) associated with the clusters is presently 1-dimensional, meaning that the associated pose distributions are circular. This is not necessarily realistic as there is likely to be more variance in the direction of forward motion along the trajectory — as the individual nodes embody a temporal averaging of people's walking speeds – rather than laterally. It is assumed the position of the trajectories is known in the global frame with some certainty, however this is an abstraction of the truth as the person's pose is calculated from the robot's position derived from running adaptive Monte Carlo localization (AMCL) under ROS. This means that pose uncertainty data is easily available and could be integrated into trajectory pruning and matching. To date, only one-way trajectories emanating from a single location have been clustered. Another obvious extension is to build a *forest* of trajectory clusters with individual trees rooted at common entry points in the environment such as doorways, staircases and lifts. Currently, trajectories that double back on themselves pose problems for the clustering technique, so a method of detecting when a person is turning around, creating a new trajectory and matching it amongst branches of the forest will need to be developed. The tree structure



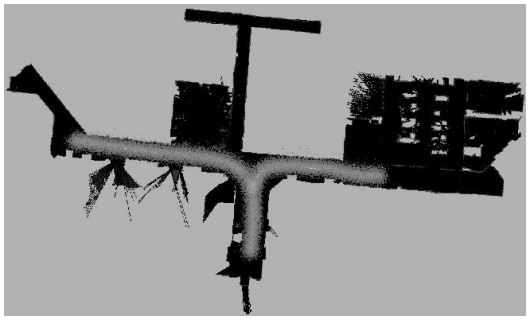
(a) Clusters, After 1 Trajectory



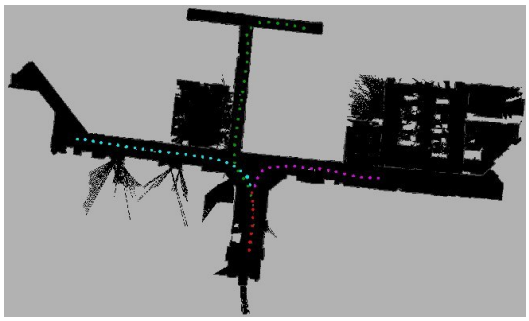
(b) Map, After 1 Trajectory



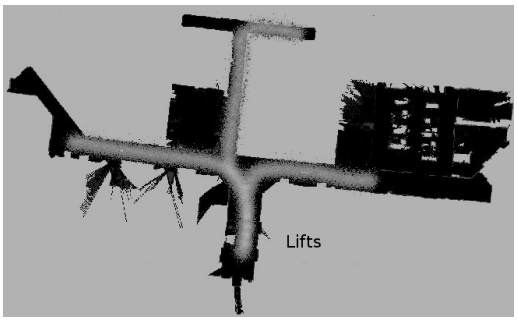
(c) Cluster, After 2 trajectories. NB the red trajectory of the previous image has split in two.



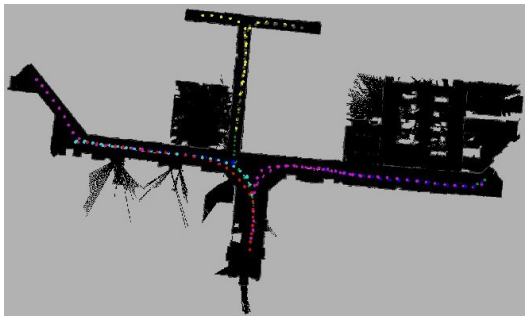
(d) Map, after 2 trajectories



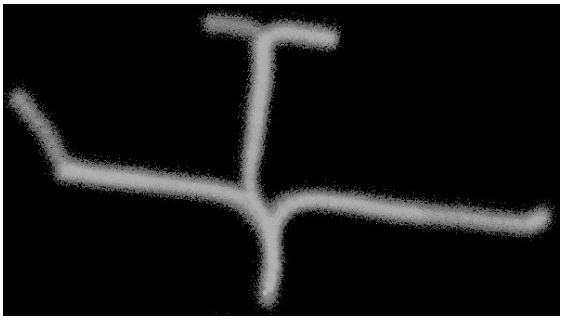
(e) Clusters, after 6 trajectories



(f) Map, after 6 trajectories. NB, the well trodden section emerging from the lifts is already noticeably lower cost than the rest of the map.



(g) Clusters, after 16 trajectories. The final cluster tree has 9 nodes, the end result of various creation, splitting, merging and concatenation operations. Each cluster is plotted in a different colour here.



(h) Map, after 16 trajectories. This is our final people-centric costmap, shown without the underlying floorplan. Black indicates high cost, white is low cost.

Figure 4: The evolution of the cluster tree (with the mean of individual cluster points plotted as coloured dots, each cluster has a different colour associated with it), shown together with the evolution of the costmap after the addition of the stated number of trajectories. The underlying geometry of the floor is shown in black shadow, and the costmap itself is high cost everywhere except where you see the gaussian blur emerging. Lighter colours indicate lower cost. Although this is a small dataset, even after only 16 trajectories well-trodden paths are easily visible in the costmap.

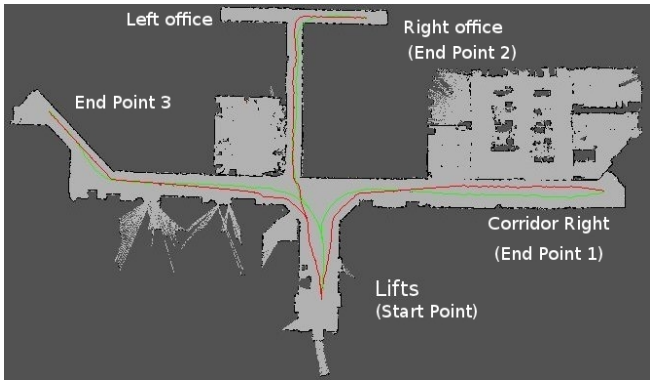


Figure 5: Planning paths over the people centric costmap (green paths) compared with the red paths generated by planning over the default Costmap2D maps which the ROS navigation stack creates from a map created by *gmapping*. Note that because people tend to take the same paths from the lift, the green paths are co-located as they emerge from the lift shaft in the bottom right of the figure. The people centric path also tracks closer to the wall on the path to the *right office* - reflecting the fact that more tracks in our dataset were destined for the *right office* than the left and that people begin to plan their paths early. The path to *corridor right* is also located in the center of the corridor when using the people-centric map. The standard costmap cuts the first corner, takes an extended diagonal track and hugs the left-hand wall of the corridor a lot more than people do.

produced by the trajectory clustering algorithm could be used in collision detection, as it gives a prior on the likely future path of pedestrians at critical points in the environment. Finally, the costmap produced in the map creation process could be used to implement a sensory filter for localization algorithms like AMCL to increase robustness around people; as the map highlights areas where dynamic obstacles are common.

References

[Arras *et al.*, 2007] K.O. Arras, O.M. Mozos, and W. Burgard. Using boosted features for the detection of people in 2d range data. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3402 –3407, april 2007.

[Azarbayejani and Pentland, 1996] A. Azarbayejani and A. Pentland. Real-time self-calibrating stereo person tracking using 3-d shape estimation from blob features. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 627 –632 vol.3, aug 1996.

[Bennewitz *et al.*, 2005] Maren Bennewitz, Wolfram Burgard, Grzegorz Cielniak, and Sebastian Thrun. Learning motion patterns of people for compliant robot motion. *Internationl Journal of Robotics Research*, 24:31–48, 2005.

[Burgard *et al.*, 1999] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk Hhnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(12):3 – 55, 1999.

[Coulter, 1992] R. Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Robotics Institute, Pittsburgh, PA, January 1992.

[Hemachandra *et al.*, 2011] Sachithra Hemachandra, Thomas Kollar, Nicholas Roy, and Seth J. Teller. Following and interpreting narrated guided tours. In *ICRA*, pages 2574–2579, 2011.

[Kirby *et al.*, 2007] Rachel Kirby, Jodi Forlizzi, and Reid Simmons. Natural person-following behavior for social robots. In *Proceedings of Human-Robot Interaction*, pages 17–24, March 2007.

[Luber *et al.*, 2009] Matthias Luber, Gian Diego Tipaldi, and Kai O. Arras. Place-dependent people tracking. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, Lucerne, CH, 2009.

[Morris and Trivedi, 2009] Brendan Morris and Mohan Trivedi. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2009.

[Mozos *et al.*, 2010] Oscar Martinez Mozos, Ryo Kurazume, and Tsutomu Hasegawa. Multi-part people detection using 2d range data. *International Journal of Social Robotics*, 2(1):31–40, 2010.

[Müller *et al.*, 2010] Jörg Müller, Cyrill Stachniss, Kai O. Arras, and Wolfram Burgard. *Cognitive Systems*, chapter Socially Inspired Motion Planning for Mobile Robots in Populated Environments. Cognitive Systems Monographs. Springer, 2010. In press.

[Navarro-Serment *et al.*, 2010] Luis Ernesto Navarro-Serment, Christoph Mertz, and Martial Hebert. Pedestrian detection and tracking using three-dimensional ladar data. *The International Journal of Robotics Research, Special Issue on the Seventh International Conference on Field and Service Robots*, 29(1):1516 – 1528, October 2010.

[Nourbakhsh *et al.*, 2003] I.R. Nourbakhsh, C. Kunz, and T. Willeke. The mobot museum robot installations: a five year experiment. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 4, pages 3636 – 3641 vol.3, oct. 2003.

[O'Callaghan *et al.*, 2011] S.T. O'Callaghan, S.P.N. Singh, A. Alempijevic, and F.T. Ramos. Learning navigational maps by observing human motion patterns. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4333 –4340, may 2011.

[Ope, 2011] OpenNI organization. *OpenNI Reference Guide*, December 2011. Last viewed 08-12-2011 14:10.

[Piciarelli and Foresti, 2006] C. Piciarelli and G. L. Foresti. On-line trajectory clustering for anomalous events detection. *Pattern Recognit. Lett*, pages 1835–1842, 2006.

[Pri, 2011] The primesensor - the primesense 3d sensing solution. Online Datasheet, 2011.

[Schlegel *et al.*, 1998] Christian Schlegel, Jorg Illmann, Heiko Jaberg, Matthias Schuster, and Robert Worz. Vision based person tracking with a mobile robot. In *British Machine Vision Conference (BMVC), Proceedings of*, 1998.

[Schulz *et al.*, 2003] Dirk Schulz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. People tracking with mobile robots using sample-based joint probabilistic data association filters. *The International Journal of Robotics Research*, 22(2):99–116, February 2003.

[Shotton *et al.*, 2011] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition*, 2011.

[Siebel and Maybank, 2006] Nils Siebel and Steve Maybank. Fusion of multiple tracking algorithms for robust people tracking. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision ECCV 2002*, volume 2353 of *Lecture Notes in Computer Science*, pages 373–387. Springer Berlin / Heidelberg, 2006.

[Spinello and Arras, 2011] Luciano Spinello and Kai Oliver Arras. People detection in rgb-d data. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3838–3843, 2011.

[Thrun *et al.*, 1997] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In David Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *Artificial Intelligenceand MobileRobots*, Cambridge,MA, 1997. MIT/AAAI Press.

[Tipaldi and Arras, 2011] Gian Diego Tipaldi and Kai Oliver Arras. I want my coffee hot! learning to find people under spatio-temporal constraints. In *ICRA*, pages 1217–1222. IEEE, 2011.

[Ziebart *et al.*, 2009] Brian D. Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J. Andrew (Drew) Bagnell, Martial Hebert, Anind Dey, and Siddhartha Srinivasa. Planning-based pre-

diction for pedestrians. In *Proc. IROS 2009*, October 2009.