Unsupervised Robot Learning to Predict Person Motion

Shuang Xiao, Zhan Wang and John Folkesson

Abstract— **Socially interacting robots will need to understand the intentions and recognize the behaviors of people they come in contact with. In this paper we look at how a robot can learn to recognize and predict people's intended path based on its own observations of people over time. Our approach uses people tracking on the robot from either RGBD cameras or LIDAR. The tracks are separated into homogeneous motion classes using a pre-trained SVM. Then the individual classes are clustered and prototypes are extracted from each cluster. These are then used to predict a person's future motion based on matching to a partial prototype and using the rest of the prototype as the predicted motion. Results from experiments in a kitchen environment in our lab demonstrate the capabilities of the proposed method.**

I. Introduction

It has often been pointed out that mobile robots operating in areas occupied by people will benefit from identification of the human behavior and using this to interact or to avoid interfering with the humans [1], [2], [3]. Ideally the robot will while operating in a new environment be able to learn to make these predictions based on its own observation of the people going about their normal business. Over time these predictions should both improve and adapt to changes in motion patterns.

For example in [4], a complete theoretical framework is presented for robot motion planning using RRT that incorporates predictions of person motion along with a set of rules such as 'collision free rule' and 'interference free rule'. This is the sort of framework in which our work could be applied.

In this paper we propose a solution for learning to predict the future motions of people. Our approach has the robot collect short trajectories by tracking people using a RGBD camera [5] or using LIDAR. The trajectories are then clustered to find prototypical patterns of motion and these are then used to predict likely future motions from partial trajectories.

Our first contribution is the introduction of a modified distance function for the clustering algorithm. This distance is an improvement over the previously used edit distance in that it can measure similarity of non-overlapping but nearby trajectories especially ones with little or no movement.

A second contribution is the on-line adaptive prediction of likely future trajectories from partial observation of a person track. This includes developing a pipeline for on-line unsupervised learning of prototypical trajectories from the person tracks collected by the robot while it carries out tasks in some environment. The prediction can help the robot avoid moving into the person's intended path. The main difference in this pipeline from previous work is the use of the pre-trained SVM as a classifier to separate the on-line training data into more homogeneous classes before learning subpatterns within each class.

[1] The authors are all with the Centre for Autonomous Systems at KTH Royal Institute of Technology, Stockholm, SE-100 44, Sweden. {shuangx,zhanw,johnf}@kth.se

II. Motivation and overall strategy

Our goal is to have the robot predict where a person might be moving so as to better plan its own motion. In addition we want our robot to improve and adapt its predictions over time by using observations of the people that it tracks in the course of doing its tasks in an environment. In order to do this we will learn prototypical trajectories in an unsupervised manner from the tracking data.

A limitations that has a strong influence on our choices of methods is the relatively small amount of data we have to learn from. As very inhomogeneous data is harder to learn from. We split the data into homogenous classes for learning. That is done to learn the prototypes that presumably could be learned from the non-split data if we had more of it. The use of the learned protypes then does not require first determining the class then the prototype but rather all the protypes are matched to and the class is then not explicit anymore. For the same reason we learn in an absolute frame as it is easier to learn the specific environment of the test data than a general behaviour in the normalized frame that is used to do the initial spliting into subclasses.

Our predictions will be the result of matching currently observed starting sections of people tracks with the starting sections of the prototypes. The closest match will then give a prediction of person's unknown planned path as the continuation of the known prototype trajectory. If several prototypes match similarly well we can form multiple hypotheses over the future paths with the similarity to the prototype giving some ranking of the hypotheses. Either way it enables the robot to move in a way less likely to interfere with the person.

Our approach to the unsupervised learning of paths is to use a method similar to K-means clustering on the trajectory data represented as vectors of cell labels, where 25 cm square cells form a uniform grid over the environment. The clustering method, partitioning around medoids *PAM* [6], minimizes a sum of general pairwise dissimilarities in contrast to the K-means algorithm which is restricted to the euclidean distance as a measure of the dissimilarities.

This learning is aided by having training data that all belong to the same general sort of human activity. For example separating people standing still from people moving in a very decided direction from A to B will make learning easier. This was confirmed in our experiments as illustrated in Fig. (1).

In order to separate the incoming learning data into these more homogeneous classes we train a SVM in a supervised manner to do the separations. Note that this SVM will not be further trained on-line but will rather be a prefiltering stage for the classifier which could then be trained on-line. The classification boundaries of these activity classes will not change even if motion patterns differ. This is supported by the results of [7]. We can also expect this because of the nature of the features and the classes that we will present later. The more specific prototypes of the trajectories, on the other hand, will benefit from long term learning that can adjust to changes in the environment and the behaviors of the people.

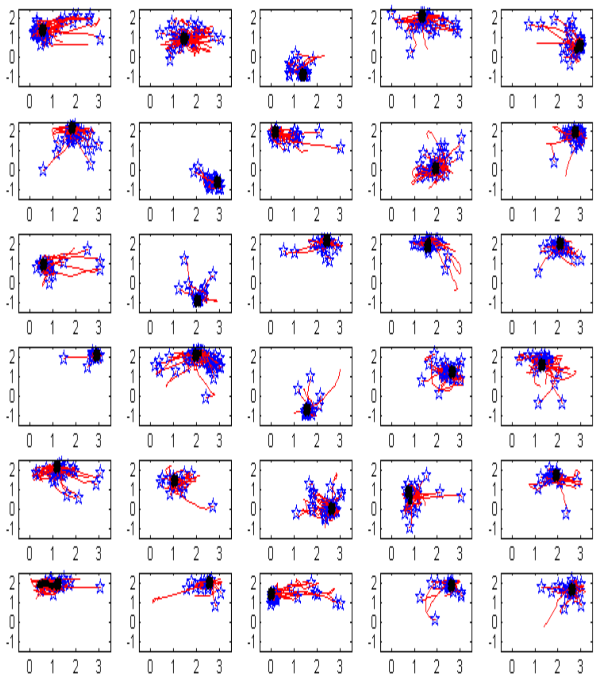


Fig. 1: PAM clustering with k=30, ie. w=1 eq. (2), on the unclassified data. The stars are cluster centers, solid black and path start, open blue. Most of the 'prototypes' end up being very short or stopped trajectories. These then are not very useful to predict future motion. That is why we learn different prototypes for different activity classes as in Fig. (9)

III. Related Work

Learning of motion patterns has been done using combination of clustering followed by a Hidden Markov Model HMM [8] and IOHMM [9]. These methods were shown to produce good results. As compared to our method they required a more careful modeling of the trajectories for the environment with certain resting places being defined for behaviors. In [8] they learn a probabilistic model of the trajectories using EM which is roughly comparable to our learning the centers of clusters of trajectories using partitioning around medoids *PAM*. They then derive the HMM to allow prediction while we match to partial prototypes to form a hypothesis over future trajectories.

In [7] several different approaches are used with the aim of motion prediction over short times. First they use a SVM classifier to learn motion classes in a supervised manner. This learning used certain features in the trajectories to describe them. Our SVM classifier uses a similar approach. They then used the trained SVM to classify a much larger data set allowing them to extract how certain areas tended to have certain classes of motion patterns. So for example 'busy walk' in hallways. We however use the trained SVM to help the unsupervised learning by making the training data more homogeneous.

To do so called anticipation in [7] they perform what they call DP matching for clustering using an edit distance between state chain vectors similar to the edit distance we describe in our method. They also then generate prototypes that can then be matched to using the partial trajectories to the partial prototypes. The main difference compared to our method is in the distance or similarity measure used and in the fact that we used the SVM to separate the training data into more homogeneous classes before clustering. Both of these differences can be considered incremental improvements on their method. Similar work using unsupervised approach is shown in [10], [11].

In another recent work [12] they take a completely different approach to the same problem. The idea of an ego graph of possible future trajectories is used to predict future movement. The ego graph is similar to the lattice graph that is often used for path planning rather than prediction and for similar reasons. In particular it is possible to build in dynamic constraints into the set of graphs.

IV. Method

Here we will describe our approach in more detail. We will first describe the learning methods SVM and PAM as applied to the pre-processed person tracks collected by the robot. These are lists of timestamped xy coordinates along the trajectory. There was considerable effort put into collection of person trajectories removing outliers, and smoothing the data. We will discuss those parts briefly in the experiment section.

A. SVM Classification

As SVM is a supervised method we need to first decide how to break the trajectory set into classes that have similar characteristics. With reference to the classes defined in [7] we defined four classes of normalized trajectories. That is trajectories that have been translated and rotated to start from the same point and have their longest distance from that start along the x-axis. This normalization removes some of the dependence on the specific environment. In particular the global starting point and direction are lost. Our classed are illustrated in Fig. (2). We also built on the work of [7] when selecting our set of feature types. Four of these are illustrated in Fig. (3). In addition to those four we also used five features based on speed between trajectory points, V_{seg}. These were

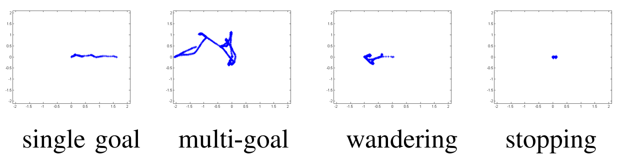


Fig. 2: The classes that were labeled in our data.

the max, min average, variance and an efficiency E defined as:

$$E = \frac{d_{se}}{t \sum V_{seg}} \qquad (1)$$

where d_{se} is the total distance between start and end points and t is the total time of the trajectory.

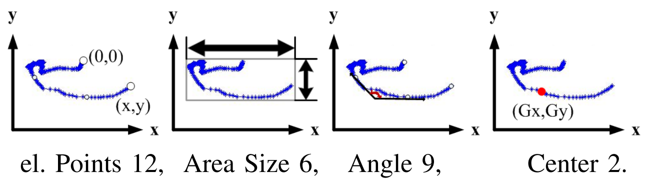


Fig. 3: 4 of the 5 feature types used for SVM classification.

For relative points three sample points are taken at equal time intervals along the trajectory. At each sampled point, x coordinate, y coordinate, arctan(y/x), the distance from this (x,y) point to the origin are calculated. For area size we took the max, min and average of all x y coordinates along the trajectory. For angles we computed angles for uniformly sampled segments along the trajectory along with the variance in these and the max angle relative to the origin.

B. PAM Clustering

The partitioning around medoids (PAM) algorithm resembles the K-means algorithm. The main difference is that instead of the mean being computed from the cluster as the new cluster center, one cluster element is selected as the representative of the cluster, known as the medoid. The medoids are determined by minimizing the summation of the distance from each cluster member to the medoid.

The PAM clustering method requires a distance function be defined between the training data examples. To define our distance we first form an uniform grid of square cells over the environment. The trajectories are represented by vectors of integers of length L. Each integer represent the index of the grid cell that the trajectory was in at evenly space times along the path. The time interval is adjusted for each trajectory to be $\frac{T}{L-1}$, where T is the entire time span of the trajectory. So the first element is the starting cell index and so on. The edit distance, d_{edit} is defined by the minimum number of insertions, deletions and value changes are needed to turn one vector into the other as illustrated in Fig. (4).

Using the edit distance to measure the similarity of two trajectories has the property that paths through the same cells in the same order but with slightly different timing will be similar while paths with no such overlap will be dissimilar. This turns out to be good for clustering the trajectories with longer extent such as the single goal class but it works very poorly (see Fig. (5)) for the stopping class as these often do not overlap at all giving an edit distance of the vector lengths (change all the elements of the vector). We could

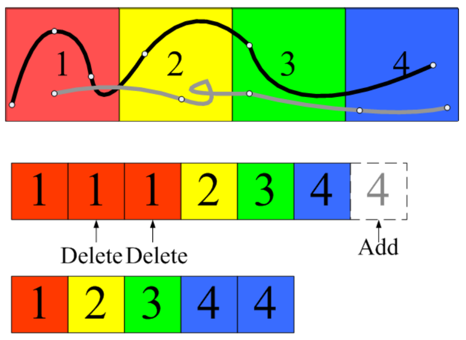


Fig. 4: 2 trajectories at top are converted into the 2 indexes vectors below. The edit distance is 3 in this example.

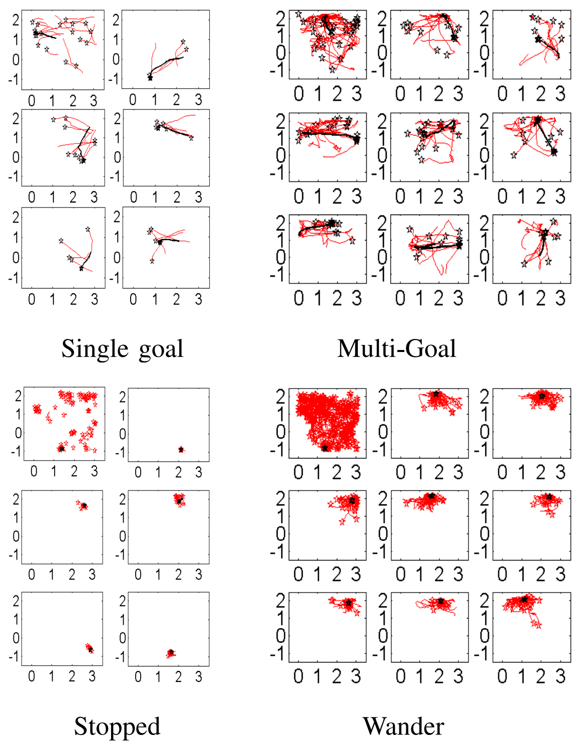


Fig. 5: Pure edit distance, ie. setting w=0 in eq.(2), results in poor clustering. The stars are cluster centers, solid, and path start, open. The prototypes are less representative of the cluster members than what we will show later in Fig. (9).

use a different distance for the stopping class to avoid this problem but our goal is to predict future motion by matching to prototype trajectories based on partial trajectories. As the beginning part of a trajectory can not be classified by the SVM we must have the same similarity measure for all classes and be able to apply it to unclassified partial trajectories.

We found that such a measure is given by:

$$d = \frac{d_{edit} + w \sum_i d_i}{L}. \quad (2)$$

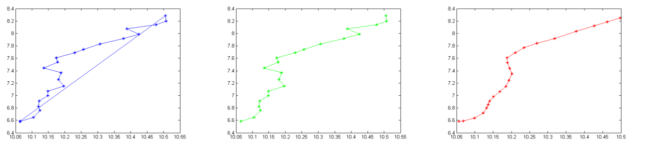
where w is a relative weight parameter and d_i are the euclidean distances between the cells that the i^{th} elements of the two vectors point to. If we use units of the grid spacing then the d_i will have values of order 1 for nearby cells. So that both distance terms will be of order 1 for nearby trajectories and we expect a weight near 1.0 would count each distance about equally. Also we see that this distance does not depend strongly on L as both the edit distance and sum scale linearly with L and this is then offset by the L in the denominator.

Having defined the distance between two trajectories, the PAM algorithm can iteratively improve the clustering given an initial set of k cluster centers in the same way that K-means clustering does. With our modified distance we expect that both single goal and stopping trajectories can be clustered and later matched to.

V. Experiments

A. Data Collection and Pre-Processing

Our data consisted of tracks collected in a common kitchen and dining area of our lab, shown in Fig (7). We used the robot's RGBD camera to collect people tracks over a 10 day period for 4 hours each day with a tracking method described in [5]. We also collected laser scanner data over one 8 hour period and applied a simple detection of dynamics (ie people) by comparison to the static scene. The tracks were all between 2 and 17 seconds long. The average of the RGBD tracks was about 7.8 seconds while the laser was 13.2 seconds. We then smooth this data using a Kalman filter while also removing the outliers defined by a threshold on the mahalanobis distance. This is illustrated in Fig (6).



Detections | Outliers Removed | Kalman Filtered

Fig. 6: The data pre-processing. On the left are raw detections center without outliers and the right Kalman filter smoothed.

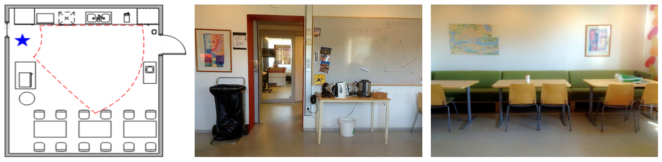


Fig. 7: The kitchen/dinning area of our lab. The robot is shown as the star in the corner, its RGBD camera FOV in red. The hot water cooker and the dinning area are shown.

Class Distribution	Laser	RGBD
Single Goal	101	431
Multi-goal	225	445
Wandering	551	637
Stopping	274	548

TABLE I: Data Summary

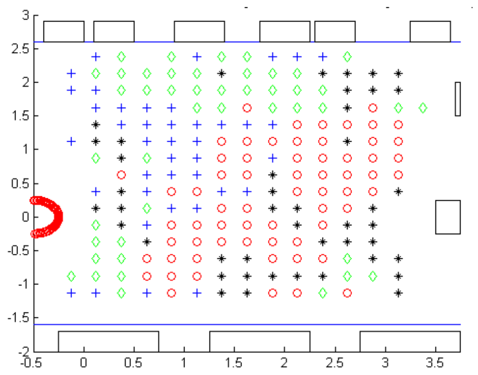
RGBD Data	Single Goal	Multi Goal	Wandering	Stopping
Single Goal	0.826	0.097	0.077	0.000
Multi-goal	0.061	0.602	0.328	0.009
Wandering	0.022	0.027	0.846	0.105
Stopping	0.000	0.000	0.062	0.938
Laser Data	Single Goal	Multi Goal	Wandering	Stopping
Single Goal	0.673	0.109	0.218	0.000
Multi-goal	0.009	0.782	0.201	0.004
Wandering	0.000	0.005	0.951	0.044
Stopping	0.000	0.000	0.245	0.755

TABLE II: SVM Confusion Matrix

B. SVM Results

For the SVM implementation, we used the toolbox LibSVM in Matlab. The one-against-one method combined with the RBF kernel was used for the multi-class classification. After training, we tested the model by the leave-one-out method. This is a cross validation method that tests each element by using the model trained by the remaining elements. Table I shows how our two data sets were distributed between the different classes.

The classified data can also be used to map the environment in terms of which motion pattern is most prevalent in each area of the environment. This is shown in Fig. (8) The resulting confusion matrix is shown in Table II. One can see that the classification is not perfect but then the hand labeling of trajectories was also often ambiguous.



Red o : Single goal Blue + : Multi-Goal, Green diamond: Wandering, Black * Stopping.

Fig. 8: Each symbol indicates the highest frequency pattern.

C. PAM Clustering

The SVM classified laser scanner data is separated into training (80%) and test (20%) sets and used to train four sets of cluster centers using PAM. This is to show that the SVM can be used as a pre-processing step for the unsupervised online learning of prototype from data collected by the robot. As new complete tracks arrive the robot will be able to pass them through the SVM and then run the PAM clustering to recompute the cluster centers. The test set will be used to test the predictive power of the resulting prototypes.

The resulting clusters and cluster centers for each activity class are shown in Fig. (9). As one can see both the very short non-overlapping stopping and the longer often overlapping single goal trajectories were well clustered and representative prototypes have emerged. In particular we have captured the patterns of the longer trajectories which were lost when all the training data was lumped together as in Fig. (1).

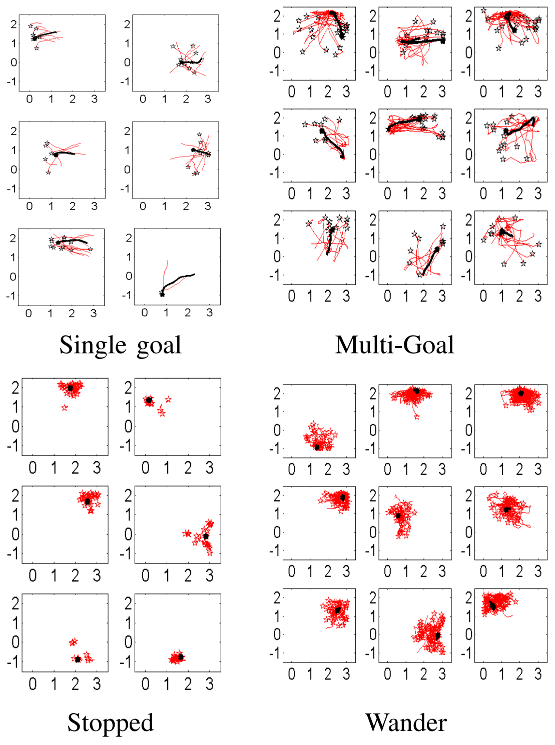


Fig. 9: Examples of (w=1) clustered trajectories for the single goal, multi-goal, stopping and wandering classes. The prototypes or cluster centers are shown in a contrasting color.

D. Motion Prediction

In order to verify the capability of our prototypes to predict future human motion, we match the first n points of the test trajectories with the first n points of each of the prototypes. In that way we could find the most similar prototype as a function of n. We also know the entire test trajectory which gives the ground truth of the most similar trajectory. We know that as n approaches L, ie. the length of the prototypes, the best match will equal the ground truth. This is shown in Fig. (10). This figure shows for example that if we only have the first 10 points of a trajectory we can be 70% confident that the future trajectory is near one of the three most similar. This figure can also be compared to the corresponding Fig. 14 in [7] although direct comparison is not possible as the data and environment are different.

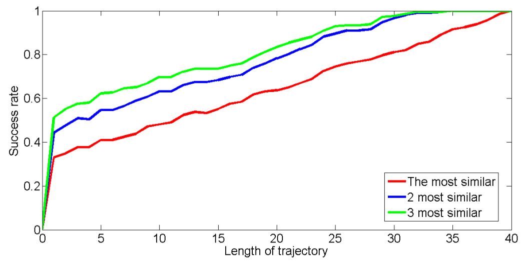


Fig. 10: The percentage correctly matched partial trajectory of n points is plotted vs n. The upper, middle and bottom curves show when the correct match was one of the 3, 2 or 1 most similar respectively, for the 40 point trajectories.

We also considered the question what distance can be considered a good match. In Fig. (11) we show the plots of correctly matching the partial trajectory for 5, 10, and 20 when matching is defined as distance below a threshold. The plots are as a function of this threshold. Also shown is the number of incorrect prototypes that were also below the threshold, ie. false positives. This figure shows, for example, that if we only have the first 10 points of a trajectory we can be 95% confident that the correct prototypes is closer than 3 but on average there will be 2 others also closer than 3.

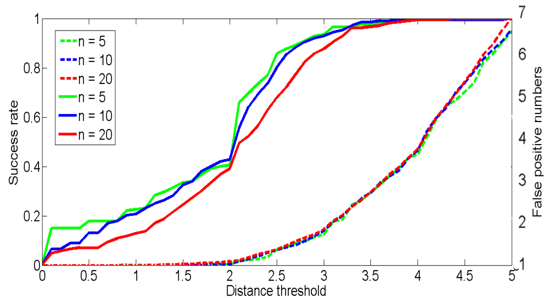


Fig. 11: The percentage correctly matched partial trajectory of n points is plotted vs the matching threshold. Also shown, as the dashed lines, is the number of incorrect matches found using the threshold. With w=1, a threshold of 2 corresponds to, for example, having all n samples of the trajectory in cells adjacent to the prototype. Cell size is 25 cm.

Finally we present in Fig. (12) four examples of how the robot might use these predictions in practice. We plot a map of the kitchen grid that is based on the robot observing the first n=5, 10, and 20 points of the 40 point trajectory and for the closest two prototypes we shade each future cell the prototype passes through. We see that although the predictions work well for the first three of the examples we also show one where it did not work so well. It is naturally impossible to always get a good prediction in this open space using only 30 prototypes. As we collect more training data we will be able to increase the number of prototypes.

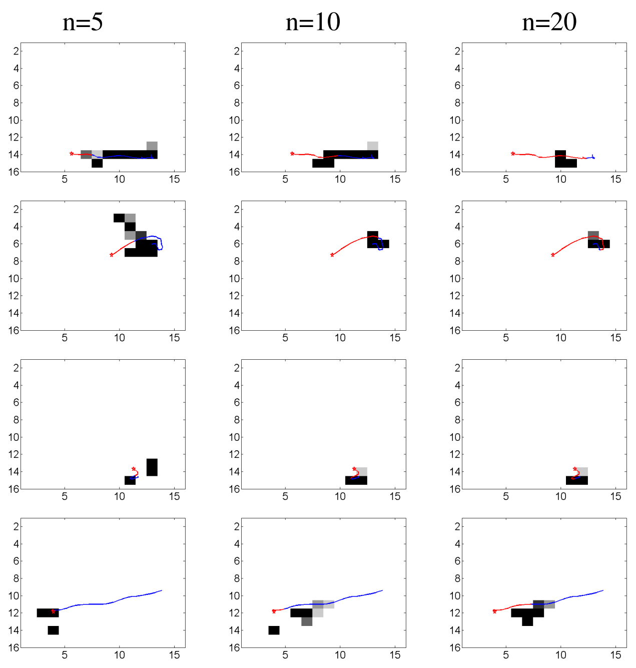


Fig. 12: These map shows the predicted possible futures of the person who's past, n=5, 10 or 20 time intervals were matched. The 'observed' track is shown in red and actual future is shown in blue. The future trajectories of the two closest prototypes are shown as filled cells. The light gray cells are the second closest, the closest prototype is dark gray and when the same cell is covered by both it is black.

VI. Discussion

Generally we would expect that the edit distance can capture certain types of similarity and the metric distance another. The edit distance works well for long trajectories that are constrained such as in a corridor. The metric distance, on the other hand, works well for shorter trajectories and in more open areas.

In our experiments we had a relatively small amount of data in a relatively open area. Our approach was able to find some patterns in this data. The predictions are strongly influenced by the number of prototypes. If there are too many prototypes there will be too many matches and prediction will eventually only enumerate physically possible paths. If there are too few prototypes the prediction will miss significantly likely alternatives. Our data was not enough to see any problems with too many prototypes but as the robot continues to collect data over weeks and months it will be able to add prototypes and get a finer and finer model of the motion patterns. Eventually it will not be advantageous to add more prototypes.

The choice of number of prototypes for each class was made by trial and error until we found what looked like good prototypes. This work was mainly interested in the existence of such a solution but one could investigate further the criteria for setting k on each class so that this could become automatic. This is direction for future work. We also looked at other values for w and found that for values within a factor of at least ten of 1 there was not any significant change in the results.

It will be interesting to see both how much our prediction can improve over time and to begin to use these in control of the robot behavior. Besides adding prototypes more data would allow us to incorporate the statistics of the clusters in our predictions. The frequency of a cluster in the data could give an a priori prototype probability. The spread of the cluster could give a spread to the prototype's prediction.

VII. Conclusions

Our results do support our hypothesis that the edit distance used for clustering unclassified data will produce inferior prediction prototypes compared to first classifying with a SVM then clustering. We also saw that adding a term to the edit distance that measures the actual metric distance between curves along their length leads to better prototypes. These were our main two goals in this paper.

VIII. Acknowledgments

The work presented in this papers has been funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No 600623 ("STRANDS").

References

[1] M. Hanheide, A. Peters, and N. Bellotto, "Analysis of human-robot spatial behaviour applying a qualitative trajectory calculus," in *IEEE RO-MAN: Symposium on Robot and Human Interactive Communication*, 2012, pp. 689–695.

[2] G. Cielniak, M. Bennewitz, and W. Burgard, "Where is . . . ? learning and utilizing motion patterns of persons with mobile robots," *IJCAI*, vol. 25, pp. 909–914, 2003.

[3] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1726–1743, Dec. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2013.05.007

[4] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "Understanding human interaction for probabilistic autonomous navigation using risk-rrt approach," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 2014–2019.

[5] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "A mobile vision system for robust multi-person tracking," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[6] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, New York, 1990.

[7] T. Kanda, D. Glas, M. Shiomi, and N. Hagita, "Abstracting peoples trajectories for social robots to proactively approach customers," *IEEE Trans. on Robotics*, vol. 25, pp. 1382–1396, 2009.

[8] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," *The International Journal of Robotics Research*, vol. 24, no. 1, pp. 31–48, 2005.

[9] Z. Wang, R. Ambrus, P. Jensfelt, and J. Folkesson, "Modeling motion patterns of dynamic objects by IOHMM," in *to appear: Intelligent Robots and Systems (IROS14)*, Chicago, 2014.

[10] A. Bruce and G. Gordon, "Better motion prediction for people-tracking," in *Proc. of the Int. Conf. on Robotics & Automation (ICRA), Barcelona, Spain*, 2004.

[11] M. Luber, L. Spinello, J. Silva, and K. Arras, "Socially acceptable robot navigation: A learning approach," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012.

[12] S.-Y. Chung and H.-P. Huang, "A mobile robot that understands pedestrian spatial behaviors," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 5861–5866.