April 2015

Exploring Human-Robot interaction in Collaborative Tasks

Rafi H. Hayne
Worcester Polytechnic Institute

Exploring Human-Robot Interaction in Collaborative Tasks

Rafi Hayne
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
rhhayne@wpi.edu

***Abstract*—My Major Qualifying Project takes the form of focused independent research on the subject of Human-Robot Collaboration. In performing this research I divided the project into two separate subproblems: automatic motion segmentation and motion planning in the presence of a human. For the first subproblem, I frame motion segmentation as a supervised learning problem. Demonstrations of proper segmentation, labeled by hand, are fed into three different classifiers. I find that two of the three classifiers are able to properly segment motions with varying tradeoffs, while the third seems unfit for the problem. For the second subproblem, I propose a novel cost function that aims to minimize robotic interference to a human collaborator's workspace.**

I. Introduction

For my Major Qualifying Project I performed independent research in the field of Human-Robot Collaboration. While automation in a factory setting is a growing field, many manufacturing tasks prove difficult to fully automate because they must be performed in close proximity to a human, or because they dictate a level of dynamic decision making that is not currently in the scope of automation technology. As a result of this difficulty, allowing a robot and human to perform manipulation tasks in a collaborative setting is a necessary step towards fully autonomous robotic assembly. I approach Human-Robot Collaboration through the lens of two disciplines: Artificial Intelligence, and Motion Planning. In the field of Artificial Intelligence I present a machine learning approach towards automatic segmentation of manipulation recordings. In the field of Motion Planning I present a novel cost function that aims to minimize robotic interference to a human's used workspace in a collaborative setting. The following sections of this report will follow this outline.

II. Motion Segmentation

A. Introduction

Proper segmentation of human motion is becoming more attractive as more algorithms and planners seek to make inferences from human demonstration. In order to properly analyze human demonstrations, individual motions must be separated from longer recordings. For example, [1] uses a hand collected library of human demonstrations to create classes of grasping motions that are used for online prediction of human workspace occupancy. This library of motions was hand segmented, limiting the learning phase to be entirely offline. With automatic segmentation of motions, on-line classification could be used to simultaneously predict workspace occupancy and to reinforce or expand the library of motion classes. The segmentation required for this specific approach could potentially be done naively by taking advantage of the fact that all motions belonging to the library start at the same resting position and end at a specified goal location.

However, as motions become more fluidly connected, as seen in a collaborative setting for example, proper segmentation becomes more involved. Firstly because natural human motion does not have a defined start and end zone. Additionally, the inclusion of another human can cause conflicts in grasping motions. In this instance it may be necessary to segment a grasping motion into more fine grained parts: the initial intended trajectory up to a conflict as well as the replanned portion.

Originally, my goal for this project was to be able to properly segment human motions at replanning points. Because of this focus, I believed simply segmenting motion at a task's start and goal to be an afterthought. With no result nearing the deadline of the project, I reformulated the project's goals to be only segmenting human motion at task goal regions. Without the requirement

of segmenting motion at replanning points, I changed my approach to a naive supervised learning problem, in which classifiers are trained to label goal regions from a supervised dataset. I chose to compare the ability of Support Vector Machines, Artificial Neural Networks, and the K-Nearest Neighbor classifier to properly label these regions because they were the main classification methods presented to us in class.

After this restructuring of the project, I found that all three classifiers could be used to properly segment at goal regions with varying success. Further, I found that Support Vector Machines can in some settings, segment at replanning points in addition to task goal regions.

The remainder of this section is structured as follows: In the next section I give a description of related work and how I planned it could be applied to finding replanning points. In Section II-C I describe the approach used to train each of the selected classifiers to segment human motions. In Section II-D I present the results of each classifier's ability to properly segment test data.

B. Related Work

Prior to the reformatting of my project I investigated traditional approaches for automatically segmenting human motions. In [2], Dana Kulic uses a stochastic approach to segment continuous human motion. More specifically, the Kohlmorgen and Lemm algorithm [3] is used without any *a priori* knowledge of human motions under the assumption that data belonging to the same segment will have the same underlying probability distribution. The author notes that this method works well to find the boundary of two different motions, but does poorly at finding fine grained transitions such as the replanning points described earlier. Kulic thus extends [3] by including motion primitives in the segmentation loop. As motions are segmented they are clustered in a similar manner to [1] and used to more accurately guide future segmentation. This approach appears to work well, but does not encode any other outside information into segmentation making it less desirable for collaborative motions. Additionally, the results of this research still showed many false positives and false negatives. This is likely because the algorithm aimed to be a general solution to segmentation of fully body motion, instead of simply attempting to segment manipulation tasks.

The CHAMP algorithm [4], an extension of [5] uses a similar stochastic approach, but allows for the inclusion of a candidate likelihood model. I believed that this candidate model would allow me to take advantage of features in collaborative manipulation tasks to make identifying replanning points a simpler task. Unfortunately, I was unable to properly formulate candidate models for the algorithm in the time frame of the project.

(a) Experiment Setup (b) Experiment Being Performed

Fig. 1. Example of experiment setup and experiment being performed.

I believe that both of the methods listed above are a better, yet more complicated solution to the problem of human motion segmentation. This is mainly because both of these approaches make segmentation decisions based on windows of sequential data. Because my approach attempts to classify an individual data point, it is prone to over segment data.

C. Approach

My approach to segmenting human motion consists of four distinct phases. In the first phase, I create a library of continuous reaching motions in which two human's are recorded performing a simple manipulation task. Next, this library is then labeled and segmented by hand to create a training and testing dataset. As stated earlier in the paper, I consider motion segmentation as a supervised learning problem. As a result, the following two phases consist of a learning phase, followed by a classification phase. In the learning phase, labeled motions from the training set are used to train the SVM, KNN and ANN classifiers. Finally, in the classification phase, motions from the testing set are segmented and compared to their original ground truth labels to compare the efficacy of each classifier for this problem.

1) Data Collection: The experiment designed to gather training and test data consisted of two participants standing shoulder to shoulder parallel to a table; each working on an individual task within a shared workspace. In order to execute their task, the participants must place a correnspondingly colored ball on each of their pegs in a specified order, displayed as an ordered set of pegs. The participants look at the color of the first empty

peg in their plan, pick up a ball from the corresponding color zone, and place the ball on top of the peg until all pegs in the plan are filled with balls. An example of the experiment setup and two participants performing their task can be seen in Figure 1.

As a means to record these interactions, I used a Vicon motion capture system consisting of eight Bonita cameras running at 100 Hz. Subjects wore a suite consisting of three rigid plates and nine markers consisting of a waist-belt and headband attached to rigid objects, a marker on the back of the hand, two on each side of the wrist, an elbow pad, two markers on either side of the shoulder, and two markers straddling both the sternum and the xyphoid process. In total, twenty participants were recorded performing the experiment six times each to create a dataset of ten blocks which contain six runs of the experiment. Because markers can occationally become occluded in collaborative manipulation tasks, two runs were selected from the block with the most accurate data to generate the initial training and test data. These will be refered to as the training run and test run in the future.

2) Data Labeling: The ultimate goal of this system is to recover a single point to segment continuous data into individual grasping motions. Within the constraints of the experiment, a segmentation point can be intuitively defined as picking or placing a ping pong ball. Unfortunately the boundary between picking and placing is not entirely black and white; it is difficult to definitively choose a single segmentation point. For example, consider the participant wearing a black shirt in Figure 1(b) who is reaching to pick up a green ball. It is trivial to define a lower and upper bound for when the subject picks up the ball, but difficult to determine an exact point of contact.

Following this intuition, each frame of the training and test runs are labeled in a boolean fashion : True for frames in which the participant can be considered picking or placing a ball and false for all other frames. As a result, the labeled data manifests itself as alternating regions of True and False frames. This regionally labeled data is convenient because it prevents the classifiers from training to human error, and from overfitting an overly sparse dataset.

3) Training: In order to prevent this segmentation method from being overly task specific, I opted to use features that should generalize to a multitude of manipulation tasks. Following the picking and placing intuition described in Section II-C2 and this requirement, x,y,z velocities of the palm marker, and wrist and elbow joints were selected for a 9 dimensional feature space. Prior to training each classifier, the data was normalized and reduced to two dimensions using Principal Component Analysis. Justification for these assumptions can be seen in Figure 2. Positively labeled regions are very clearly clustered around the origin.

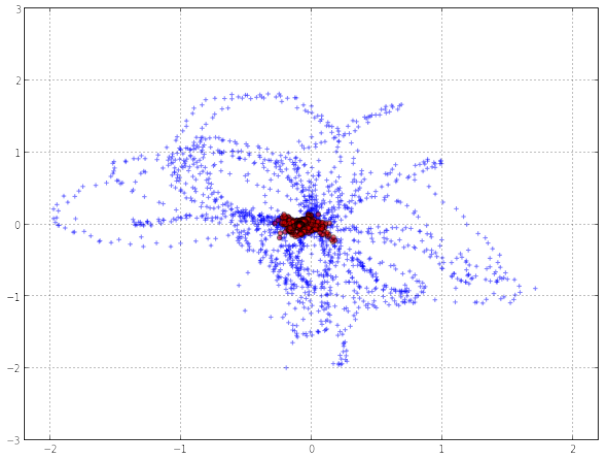


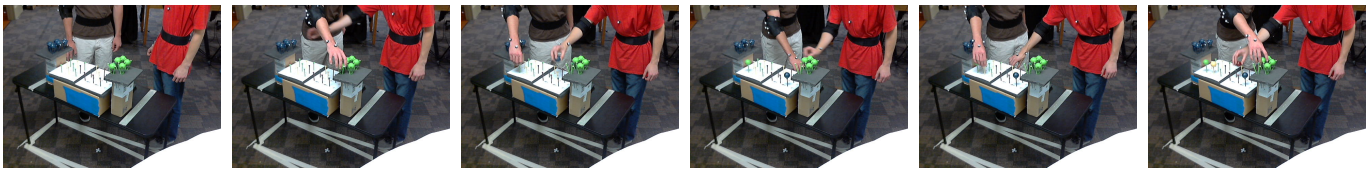
Fig. 2. Plot of labeled training run after principal component analysis. True frames in red, False in blue.

4) Classification: To recover segmentation regions, each frame of the test run has its label predicted individually by each of the classifiers. In order to prevent over segmentation of the data, a minimum region size of 20 frames is imposed. This minimum is smaller than every hand defined region in the ground truth data and accounts for only 200 milliseconds. The final predicted segments are compared to the ground truth segments both qualitatively and quantitatively. First, the false negative and false positive classification rates are calculated. Second, a single segmentation point is naively chosen at the mid point of each positively labeled region and visually sanity checked.

D. Results

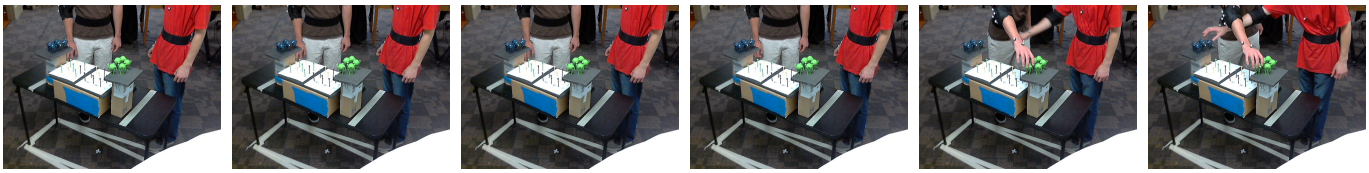
In this section I will present results illustrating each classifiers ability to properly segment human motion. I will first detail the configuration used for each classifier. Next, I will present the classifier's ability to properly recover segments by comparing their error rate's when compared to true labels and by comparing visual examples of the recovered motions.

1) Classifier Setup and Parameters: The Support Vector Machine implementation used was the scikit-learn[6]



(a) 133 / 127 (b) 401 / 402 (c) 541 / 540 (d) 676 / 676 (e) 801 / 803 (f) 900 / 903

Fig. 3. The first six segmentation points recovered by the SVM and ANN classifiers. Recovered frame numbers are listed respectively.



(a) 29 (b) 80 (c) 142 (d) 226 (e) 389 (f) 429

Fig. 4. The first six segmentation points recovered by the KNN classifier. Recovered frame numbers listed below each image.

python library, which in this case, acts as a front end to LIBSVM. Linear and polynomial kernels were tested, but did not recover labels as well as the radial basis function kernel with default paramers of C = 1 and gamma = 0.5.

The Artificial Neural Network implementation used was the PyBrain[7] python library, a popular library for neural networks. A three layer network with 2 input nodes, 10 hidden nodes and 1 output node was used. The network was trained with backpropagation until convergence using 35% of training data for validation. Because labels in the training data are sparsely distributed, frames of the training set were reordered to be evenly distributed to obtain a proper validation set.

Finally, the K-Nearest Neighbor implementation used was also from the scikit-learn[6] package. All values of k ranging from 2 to 50 were tested using both uniform and distance based weights. A k of 4 with uniform weights was found to have the highest classification rate.

2) Raw Classification Results: Figure 5 shows the confusion matrix for each classifier when attempting to predict labels for the test run. Each table shows the number of properly recovered labels, and improperly recovered labels. The main diagonal of each table shows the true number of false and true labels.

The first thing to note, is that each classifier has a much different ratio of true to false labels than the ground truth set (0.27). The SVM has significantly more true than false labels with a ratio of 1.87, the ANN has slightly more true than false with a ratio of 1.08 and the KNN has slightly less with a ratio of 0.79. While none of these ratios are close to the ground truth, on first impression the KNN labeling seems to most closely match the ground truth labels.

However, it is important to additionally consider false positives (true 0 labeled as 1) and false negatives (true 1 labeled as 0). Intuitively, a false positive will lead to over segmentation by splitting a single motion into multiple parts, while false negatives will lead to under segmentation by considering two motions as one.

The SVM classifier has the lowest false negative rate out of the group, but also has the highest number of false positive. Alternatively, the KNN classifier has highest false negative rate and the lowest false positive rate. The ANN falls between the other two classifiers. As a result, I would expect the SVM to properly recover the true segments, with many additional segments overly splitting each motion. Similarly, the KNN may provide fewer segments than in the ground truth. Finally, the ANN should perform closest to ground truth. These assumptions are tested visually on the data in the next section.

3) Visual Results: In order to test the recovered labels visually, segments were recovered from labeled regions as described in Section II-C4. These recovered segments were used as pause points in a video playback of the experiment. In addition to visualizing the segments recovered on the test run, I also had each classifier recover segments from an unlabeled run in a different block. Using a different test block shows the methods's ability to generalize to different participants.

Figures 3 and 4 show screen captures of the first six segmentation points for each classifier on the unlabeled

		Predicted		
		0	1	Total
True	0	1212	1553	2765
	1	17	753	770
	Total	1229	2306	3535

(a) SVM

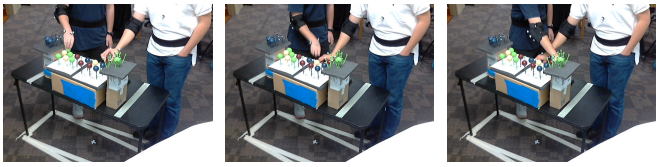
		Predicted		
		0	1	Total
True	0	1525	1240	2765
	1	178	592	770
	Total	1703	1832	3535

(b) ANN

		Predicted		
		0	1	Total
True	0	1733	1032	2765
	1	243	527	770
	Total	1976	1559	3535

(c) KNN

Fig. 5. Confusion matrices for each of the classifiers.



(a) 2926 (b) 3021 (c) 3076

Fig. 6. SVM method segmenting test run at a replanning point

block. The SVM and ANN results share images because they are within a few frames for each segment. All three classifiers are able to properly segment each motion in the unlabeled run and the test run. However, both the SVM and KNN add additional segments.

In both the unlabeled run and the test run the KNN over segments before the experiment has started (Figures 4(a) through 4(d)) and occationally segments upon entering and exiting a goal region (4(e) and 4(f)).

In the unlabeled run, the SVM performs exactly as the ANN and only recovers the true segmentation points. In the test run, the SVM adds two additional segments that may have been missed when originally labeling the data where one participant has to avoid the other participant's arm in the workspace. Figure 6 shows the SVMs three segments that split one single reaching motion into two parts at the replanning point. These additional two segmentation points could explain it's high false positive rate found in the previous section.

E. Discussion & Future Work

While the numbers from the confusion tables in Section II-D2 were useful in initially validating a classifiers ability to recover true labels, they were unhelpful in actually demonstrating the ability to successfully segment individual motions. From the visual results, we can see that the ANN learning algorithm did best at learning to reproduce the model described by the ground truth labels; in both visual tests the ANN provided the correct number of segments. The SVM provides a more general model that can potentially be used for more fine grained segmentation of motions as described in Section I. Finally, the KNN can properly segment individual motions, but may require significant post processing to remove extra segments.

In future work, I would try to use a higher value of k in the KNN algorithm. This may cause the recovered labels to fit the overall dataset worse, but will hopefully correct double segmenting goal regions. Additionally, I would further investigate the SVMs ability to properly segment at replanning points by using additional features, or entirely new features in conjunction with the ANN model that properly segments at goal regions. Finally, all of these classifiers can be queried in real time, potentially allowing online segmentation of motions.

F. Conclusion

I have shown that all three of the selected classifiers can be used to segment human motions with varying effects. Visual analysis of the three methods shows that both the SVM and neural network perform better than the k-nearest neighbor approach. The ANN can reliably segment motions at the boundary of a task while the SVM could potentially be used in continued work to segment motions at human replanning points in addition to task based replanning points.

III. Motion Planning in Human Proximity

A. Introduction & Background

The next subproblem of Human-Robot Collaboration I tackled was that of manipulation planning for a robot in close proximity to a human collaborator. I present a cost to be used in a motion planner or trajectory optimizer that attempts to aid human comfort in collaborative manipulation tasks. More specifically, this new cost function attempts to embed an understanding of which areas of a shared workspace a human is utilizing.

As mentioned in Section I, allowing a robot and human to perform manipulation tasks in a shared workspace

is a necessary step towards fully autonomous robotic assembly. Because of this need for collaboration in advanced automation, planning for a robot while maintaining human safety is an active research topic. Most safety efforts in this area appear to fall into two categories based on whether the robot is assumed to be functioning in an active or passive role when interacting with a human.

In the case where a robot is assumed to be taking an active role, the human is generally considered to take the role of a more passive observer. Planned motions are thusly made to be more understandable to the human. For example, [8] aims to generate more human-like motions by first generating a reachability map that is used to choose an ergonomic and understandable goal configuration. However, such a goal configuration in a factory setting may further occlude a human's workspace, which is contrary to the goal of this work. Similarly, [9] takes into account a human's visibility, comfort and reachability in the robot's configuration space. While this approach is useful when performing hand over tasks to a human, it does not appear to be of much use when performing separate pick-and-place tasks in a shared workspace. Finally, [10] aims to convey manipulation intent by purposefully bending a trajectory to better communicate which goal is being reached for. Legible motions are certainly important when collaborating with a human and could be used in tandem with this work, but the primary goal of this proposal is to minimize the need for the human to understand a robotic collaborator's intent. Ultimately, work of this nature seems better suited for a robot assistant than a collaborator in ones workspace.

In pick-and-place manufacturing tasks, a human should feel safe enough to work freely in a workspace; giving minimal consideration to its robot counterpart. In this scenario, to maintain safety the robot must take an observational role of the "active" human. As a result, research in this case generally aims to predict a human's future motion. In [1] Mainprice and Berenson use a precomputed library of human motions to create GMMs that are queried on-line to predict a human's future workspace occupancy while multiple trajectories are planned simultaneously in parallel. The trajectory with the least penetration of the predicted workspace occupancy is chosen for execution. This work is closest in nature to my proposal, in that it aims to avoid the collaborating human's workspace. However, it differs in that the predicted occupancy is used largely for task selection, not in motion generation. Finally, in [11] Mainprice et al. use Inverse Optimal Control to learn an assumed optimal cost function from human demonstration. This cost function is used in iterative replanning to predict human reaching motion. Continued work in this topic used the recovered cost function to plan human-like motion for a robot working collaboratively with a human. Unfortunately these human-like robotic trajectories caused the human subject to become visibly uncomfortable.

B. Methods

In a perfect scenario, generating human-like motions for a robot would most likely lead to the best possible human-robot collaboration. Human collaborative motions in a shared workspace are generally fluid, rarely colliding or stalling with minimal verbal communication. Regrettably, as mentioned above, human-like robotic motions do not maintain this grace in practice for a number of reasons. Firstly, human motion generation does not appear to globally resolve conflicts. When working together two humans do not appear to minimize conflicts by watching for legibility or even distance to the other human. Motions move in a generally straight path towards a goal where conflicts are quickly resolved locally by replanning or stalling. Conflicts are minimized in a global sense through task planning; humans choose tasks that will cause the least need for local conflict resolutions with their partner. Additionally, humans are not comfortable performing this type of local conflict resolution with a robotic collaborator, for fear of safety or simply distrust due to minimal robotic exposure in their lifetime.

Because of these observations, I worked on extending existing motion planners that allow a human to work naturally alongside a robot not through more human or legible motions, but rather through avoidance of the human's intended workspace. The underlying idea of my method is that humans will use the same area of a workspace to perform a repetitive task if there is minimal interference from a collaborator in a workspace. This is a fair assumption to make as humans and robots both attempt to take a straight line path to their task goal if possible. I call these regions of the workspace that a human traverses when performing a task "workspace lanes". The goal of my proposed cost function is to model the growth of these "lanes" over time such that a robot collaborator can minimize its lane penetration.

1) Generation of Costmap: To generate a workspace lane cost, I discretize the shared workspace into a voxel

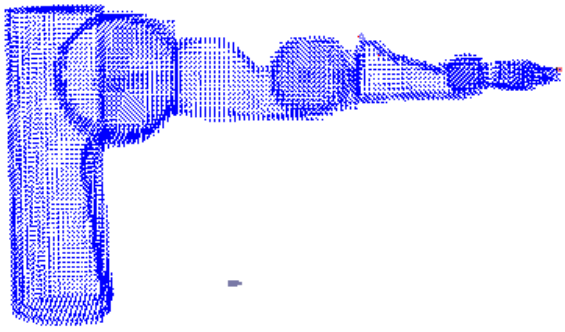


Fig. 7. Sampled Points of a PR2s Right Arm

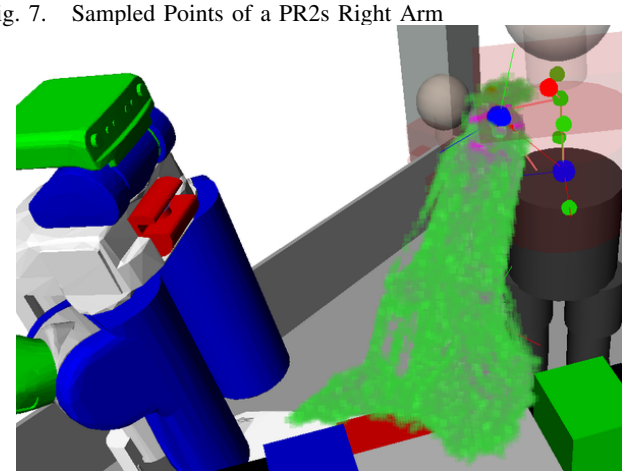
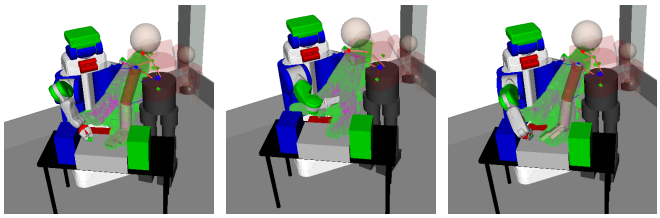


Fig. 8. Generated Lane Costmap

grid with 1cm cells. Next, I take uniform samples of both the human and robot's arm. An example of these samples can be seen in Figure 7. At each frame update from Motion Capture data, I apply the transform of the appropriate agent's arm to these samples to align them properly in the voxel grid. The transformed human samples can then be used to increment the cost of each corresponding voxel in the voxel grid. As the human performs its given task in the workspace voxel costs are incremented to generate a cost map (Figure 8) that describes how often a given voxel in the workspace is used by the human. In a similar fashion, the transformed samples from the robots arm can be given to evaluate the cost of a given configuration.

2) Planning with Lane Cost: In order to plan trajectories while minimizing this workspace "lane" cost, I selected the TrajOpt [12] trajectory optimizer due to its ability quickly generate feasible trajectories often in sub-second planning time, and because the authors provide a set of convenient python bindings that minimized implementation time. Planning to minimize lane cost with TrajOpt is accomplished in two stages. In the first stage, a stochastic approach is used to select a goal configuration of minimal cost. In traditional motion planning, a single goal configuration is used as the goal for a planned trajectory. However, finding a feasible goal configuration with minimal cost is non trivial. To find these optimal goal configurations I randomly initialize the PR2s arm and construct a request to optimize lane penetration cost with an xyz pose constraint. This optimal goal configuration is then used in the second phase which attempts to plan an optimal trajectory. A second optimization request is now made with a straight line trajectory from the start configuration to the newly found goal configuration. Because TrajOpt is a sequential convex optimizer it can often converge to a local minimum that is an infeasible trajectory. In this case subsequent optimization requests are made with the addition of a random workspace waypoint included in the mid-point of the trajectory. Finally, if a feasible trajectory can not be found after exhausting all waypoints, a path is planned using a Bidirectional RRT and is subsequently optimized with TrajOpt.



(a) (b) (c)

Fig. 9. First three planned goal configurations

3) Results: Unfortunately, I have not completed the implementation described in Section III-B2 prior to the deadline for this report. Because of this, I have no numerical results that compare the cost of planned paths with my method to a path planned not considering workspace cost. However, by viewing screenshots of recovered goal configurations, one can see that the recovered paths are indeed avoiding this cost (Figure 9). Each grasp positions the PR2s shoulder joint very high in the workspace to avoid interfering with the human's intended workspace lanes. Figure 9(b) is the hardest goal region in the entire task plan that the robot must complete. This is because the goal region exists directly below the human's workspace. The planner correctly produces a grasp that chooses the midpoint of a fork in the human's lanes. Unfortunately the PR2's shoulder joint is incuring higher cost than it should be (It should be raised higher to avoid cost near the blue goal region). I believe this is because

trajopt converged to a suboptimal minimum because I am computing a numerical gradient as opposed to the proper analytical gradient.

4) Future Work: As briefly mentioned in the previous section, I would like to implement an analytical gradient for workspace lanes to improve TrajOpt's convergence. I hope to compare my planned paths that attempt to minimize cost to a naively planned path to verify the method is indeed working. Finally, I would like to perform this experiment in the real world as opposed to in simulation. While the underlying intuition in avoiding workspace lanes appears solid, it may not make task completion for the human any easier. I will have to define a way to numerically analyze a humans comfort levels when performing a task.

5) Conclusion: For my Major Qualifying Project I have performed research in various aspects of Human Robot Collaboration. I have produced an efficient supervised learning technique for automatically segmenting human grasping motions from larger recordings. I have completed a basic implementation of a human-workspace aware motion planner for a robotic collaborator. Finally, I have outlined where I hope to continue this motion planning research in the near future.

REFERENCES

[1] J. Mainprice and D. Berenson, "Human-robot collaborative manipulation planning using early prediction of human motion." in *IROS*, 2013, pp. 299–306.

[2] D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *The International Journal of Robotics Research*, pp. 330–345, 2011.

[3] J. Kohlmorgen and S. Lemm, "A dynamic hmm for online segmentation of sequential data," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., 2001, pp. 793–800.

[4] S. Niekum, S. Osentoski, C. Atkeson, and A. G. Barto, "Champ: Changepoint detection using approximate model parameters," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-14-10, June 2014.

[5] P. Fearnhead and Z. Liu, "On-line inference for multiple changepoint problems," *Journal of the Royal Statistical Society*, vol. 69, no. 4, pp. 589–605, 2007.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber, "PyBrain," *Journal of Machine Learning Research*, vol. 11, pp. 743–746, 2010.

[8] F. Zacharias, C. Schlette, F. Schmidt, C. Borst, J. Rossmann, and G. Hirzinger, "Making planned paths look more human-like in humanoid robot manipulation planning," in *ICRA*, May 2011, pp. 1192–1198.

[9] J. Mainprice, E. Sisbot, L. Jaillet, J. Cortes, R. Alami, and T. Simeon, "Planning human-aware motions using a sampling-based costmap planner," in *ICRA*, May 2011, pp. 5012–5017.

[10] A. Dragan and S. Srinivasa, "Generating legible motion," in *Robotics: Science and Systems*, June 2013.

[11] J. Mainprice, R. Hayne, and D. Berenson, "Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning," in *ICRA*, May 2015.

[12] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems*, 2013.