Heuristically initialized motion planning in a low cost consumer robot

Nandan Banerjee, Erik Amaral, Ben Axelrod, Steven Shamlian, Mark Moseley

***Abstract*— In this paper we address the problem of designing a consumer robot capable of manipulating objects typically present in a home. One reason for lack of consumer adoption of manipulator robots is that planning for grasps while negotiating obstacles is costly in terms of time, power, and computational resources. Also, robot arms are generally expensive, thus confining their usage to research labs and the industry. The contribution of this paper is twofold. First we present the hardware design of robot arms resulting in an order of magnitude reduction in cost over the state of the art. Second, we propose an efficient motion planning algorithm which is able to generate motion plans for grasping consistently within 1s everytime using heuristic initialization. We evaluate the algorithm on a challenging task of grasping objects in a cluttered home environment, using a proprietary physical system using two low-cost 7 DoF arms, 3 fingered underactuated hands, and a 1 DoF torso and neck on a holonomic drive base.**

I. INTRODUCTION

Manipulation for consumer applications still has a long way to go. There are certain challenges that need to be addressed first before robot arms can safely make their way into consumer homes. Typically, robot arms are large and dangerous to be around. While this problem is partially a result of their industrial pedigree, it is also a simple matter of physics. To have a required lift capacity at the end-effector of the arm, electric motors need to have a high gear ratio. This makes the arm very stiff with a high amount of inertia.

Recently, some arms such as the Rethink Sawyer, Universal Robotic, and Kinova have been developed to use expensive precision harmonic drive transmissions [1][2][3]. These arms can work closely with people, but are far outside the price range of traditional consumer products. Other arms use series elastic actuators to add compliance to make them safer while still using traditional gearing like Rethink Baxter.

We recognize that there are engineering trade-offs that we must exploit in order to achieve a low-cost home manipulation robot. With no analog available for Moore's Law in the hardware world, we have worked to move solutions from the hardware domain into software in order to take advantage of dropping processor costs.

In Section III, we briefly describe the design and construction of low cost arms and torso with an order of magnitude price improvement over state of the art which is a significant starting step towards building manipulation robots for households. Since having legged robots in the home poses significant control and cost challenges, we have chosen to go with a proprietary mobile holonomic base (not low cost) to support the upper body of the robot. Our objective was to create a strong and capable upper body (arms and torso) comparable to manipulators that are available in the market - but much less expensive. We have not addressed the design of a low cost home navigation platform in this paper.

The authors are with the iRobot Corporation, Bedford, MA, USA {nbanerjee, eamaral, baxelrod, sshamlian, mmoseley}@irobot.com

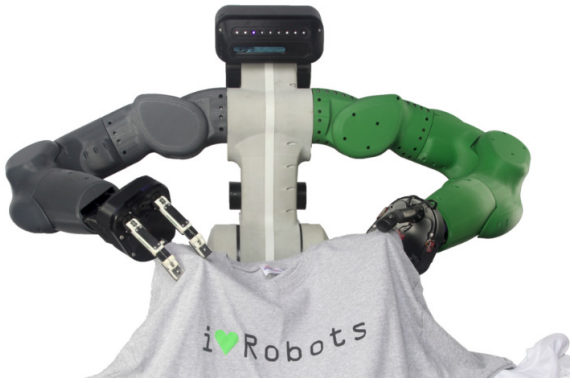

Fig. 1: Arms with torso.

Apart from having a low cost humanoid like platform, fast motion planning is also essential if consumers are to feel comfortable around robots and be able to effectively use them to do household chores. The generated motion plans should also be intuitive to a consumer, i.e. they should "look natural". In Section IV, we introduce a motion planning framework using a grasp heuristic initialized optimization based planner for generating fast and natural-looking plans.

In Section V, we compare the heuristic initialized optimization based planner with BiRRT and show that our method finds solutions in about 1s or less whereas BiRRT solve times range from 0.2s to 4.6s depending on the initial configuration of the robot. We present our Conclusions in Section VI.

II. RELATED WORK

Robotic arms that are inexpensive and also robust enough for consumer applications are not very common in the literature. On one hand, there are hobbyist robotic arms which while generally inexpensive lack in precision, safety, speed, or lift capacity. On the other hand, there are commercial arms manufactured by Rethink, Kinova, Franka Emika, and others, which have all of the specifications but are an order of magnitude more expensive [4][5][6].

The state of the art grasp planning work has been mostly based on the idea of trying to find ideal grasp solutions by simulating grasps on object shapes. Miller et al [7] produced very promising results by using shape primitives to do grasp planning. Research on using a Bayesian framework for planning for grasps taking robot kinematics and object shape and

pose uncertainty into account was done at Willow Garage [8]. These methods usually require previous knowledge about the object to be grasped and also take some time in simulating valid grasps.

There has been a lot of work done in the area of motion planning for articulated robots, human robot interaction, and active obstacle avoidance. In motion planning, two different approaches are usually taken - a sampling based approach, and an optimization based approach. The first approach of using sampling based planners like RRT and BiRRT [9] yields a feasible path in the search space. These methods are probabilistically complete. However, searching in high DoF configuration spaces with a mesh representation of the robot environment, and finally performing post processing on the generated trajectories is computationally intensive and time consuming. Bialkowski et al [10] proposed a method to reduce the number of collision checks in sampling based algorithms but those only work well with multi query methods like RRT* and not RRT. King et al [11] worked on the rearrangement planning problem on a planar surface by incorporating a physics model in the planner to generate trajectories with full arm manipulation and object interaction.

On the other hand, optimization based planners like CHOMP [12], STOMP [13], and TrajOpt [14] can generate collision free trajectories from an initial trajectory which might be in collision. The result of a trajectory optimization problem can be obtained in a short amount of time and scales well with accurate mesh representations of the robot environment. But these methods also have their drawbacks, the most important being sensitivity to the choice of the initial trajectory guess and getting stuck in local minima.

To tackle some of these problems of optimization based planners, Li et al [15] came up with a method for generating initial trajectories to improve the success rate of finding a solution. They used BiRRT to produce an initial trajectory and then used that as a seed to an optimization based planner. Although it gave better success rates, it took a long time to find that particular solution.

III. HARDWARE

The robot is intended to be used for everyday tasks around the home. Since the average home is designed around human function, it was logical to design a manipulator that closely matched human morphology [16][17]. The capacity of the arm was determined by measuring human joint speeds and lifting capacity. The layout of the arm is: Roll – Pitch – Roll – Pitch – Roll – Pitch – Yaw. In the interest of removing cost and complexity, the hand was designed with three underactuated fingers (see Fig. 2 and Fig. 3 b).

In order to make the manipulator feasible for consumer use, cost was taken into serious consideration. Injection molded parts and repeated parts were used whenever possible. The same actuator was used for the top five joints, with the first two joints having a larger motor to handle the increased torque. The actuator consisted of a three stage planetary gearbox with a brushless DC (BLDC) motor. The lower two joints consisted of two custom linear actuators forming the wrist gimbal driven by the same BLDC motors. Each actuator was almost entirely made of injection molded parts bringing the price down to less than $80 per degree of freedom. A detailed cost breakdown can be seen in Table I (1000 unit per year volumes).

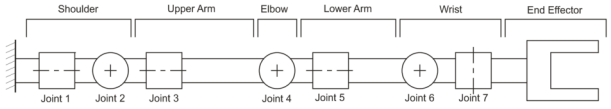

Fig. 2: Joint layout of the manipulator.

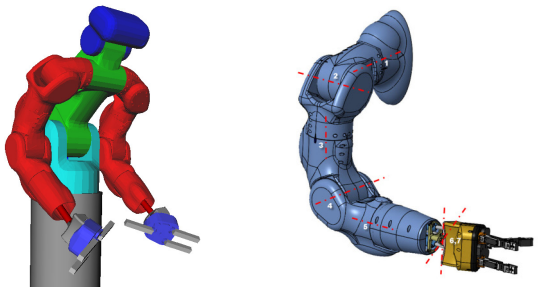

Fig. 3: (a) Bend in the torso leading to an increased workspace and more natural looking to users. (b) Rendition of the arm with the joint axes.

TABLE I: Cost breakdown of the arm

Description		Cost
Joint	Gearing, housing, motor, misc. hardware (per joint)	∼ $45
	Motor driver and other electronics	∼ $35
	Total	∼ $80
Arm	7 x Joints (without end effector)	∼ $560

Extensive analysis was performed to pick the joint layout. We looked at various metrics such as manipulability measure [18][19] and distance from joint limits for the workspace in front of the robot as well as simple motions such as opening doors, cabinets, drawers, and reaching objects. A Roll – Pitch – Roll type shoulder was found to have the best trade-off between usable workspace and ease of mechanical construction. Keeping in mind that the arm's workspace is primarily in front of the robot, we were able to safely limit the joint ranges allowing us to do away with costly slip-rings or wire service loops.

A robot in the home should be able to reach objects on the floor. We decided to add this ability with a 1 DoF waist instead of an extra arm link or linear actuator in the torso. Not only is this more natural looking to users, but it significantly increases the arm's workspace when working at a table (see Fig. 3a). This bending waist also necessitated a bending neck since we want to keep our cameras fixed on the target as we bend the waist and reach forward. To keep the cost down, we did not include panning joints in the waist or neck since a mobile base is able to provide this degree of freedom. We also offset the robot's hand by 30 degrees. This faces the robot's palms inward and allows for more human-like grasps.

We chose inexpensive brushless outrunner motors; their mass to torque ratio far exceed more conventional inexpensive motor technologies [20]. Three different sizes of

motors were used in the system. Smooth control of this arm requires a very high dynamic range on speed – between 0.05 and 20 kRPM on the motor shaft. Usually, a controller with this kind of capability for a mass market sensorless BLDC motor costs hundreds of dollars. So, we moved away from the hardware domain by designing our own brushless driver and communications stack for smooth control of these brushless outrunner motors at a lower cost. We implemented torque control (at 40 kHz) and velocity control (at 300 Hz) using an 8-bit, 32 MHz microprocessor which costs less than $2. The whole controller costs approximately $35 in low volumes (1000 units per year). The robot computer is a single Gigabyte BRIX Pro GB-BXi7G3-760, residing in the robot torso. An Intel RealSense R200 RGB-D sensor was mounted on a tilting neck as the primary visual input. The RealSense R200 was chosen due to its compact size, minimum range, relative low-cost, and availability compared to many of the other common RGB-D sensors being used.

The torso is mounted on a proprietary holonomic mobile platform with indoor navigation capabilities. This mobile base reads a map of the environment generated apriori and uses SLAM to navigate to a given target.

This robot provides comparable range of motion to other robots mentioned in the Introduction but offers a higher payload capacity at a cheaper price point. The robot can provide a 3.63 kg lift at full extension of one arm, and a 6.8 kg lift when the robot is curled at the elbow. Joints 1-2 have a max torque capacity of $\sim$46.32 N m, joints 3-4 have a max torque capacity of $\sim$33.33 N m and joint 5 max of $\sim$9.04 N m at the output. The gearing has integrated slip clutches to protect the joints from overload. Unloaded joint speeds are 120 degrees/sec and full load joint speeds are 30 degrees/sec. Range of motion - joint 1 ($0^\circ - 330^\circ$), joint 2 ($0^\circ - 190^\circ$), joint 3 ($0^\circ - 330^\circ$), joint 4 ($0^\circ - 170^\circ$), joint 5 ($0^\circ - 330^\circ$), joint 6 and 7 (end-effector moves in a 60° cone).

IV. METHOD

First, the robot is moved to a place of interest, i.e. in front of a table or a counter-top. The proprietary robot base which is given high level instructions like "Go to the kitchen table" handles this part. After that, the robot detects the target object and manipulates it as per the instructions received by the robot. This section briefly discusses some of the perception algorithms which are responsible for the detection and localization of the target object and then the heuristic initialized optimization based planner is presented. The perception algorithms are presented for the sake of completeness of the pipeline. A high level outline of the perception-manipulation pipeline is as follows:

- Navigate to location of interest ("go to kitchen table").
- Based on task specification, choose perception pipeline.
- Use 3D stereo to scan and identify objects of interest.
- Generate a ranked set of candidate grasp poses based on a heuristic.
- Generate IK (inverse kinematics) solutions using IKFast and pick the best solution.
- Plan using an optimization based planner.
- Execute the plan on the robot.
- Use a low-level servo controller using ranging sensors on the robot hand for performing a final grasp.

A. Perception

In the course of this work two vision pipelines were developed, both designed to serve the goal of providing a real-time point cloud processing framework for manipulation in cluttered 3D environments. The goal of the first pipeline (graspable shape pipeline) was to find graspable primitive shapes. By limiting the representation to primitive shapes instead of a dense representation such as a mesh, it is possible to drastically reduce the complexity of both obstacles for collision avoidance as well as grasp planning. This approach works well for a wide range of objects geared towards pick-and-place style tasking. The first pipeline is used for tasks such as cleaning a table by picking up objects represented as generic primitive models and placing them in a container.

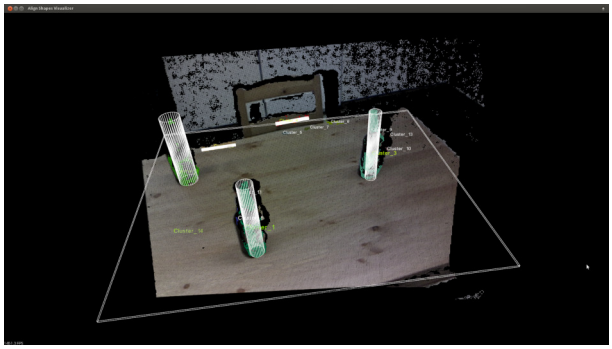

Fig. 4: Example output of the graspable shape vision pipeline

Overall, this pipeline generated a simple representation of the world and operated at approximately 7 fps on our Intel i7 computation platform. It worked well for many objects and enabled very fast grasp planning and collision avoidance as shown in Fig. 4. Some disadvantages - Limiting object representations to only primitive shapes results in poor grasps for some objects and the shape fitting can be quite noisy with translucent, specular, and complex self-occluding objects.

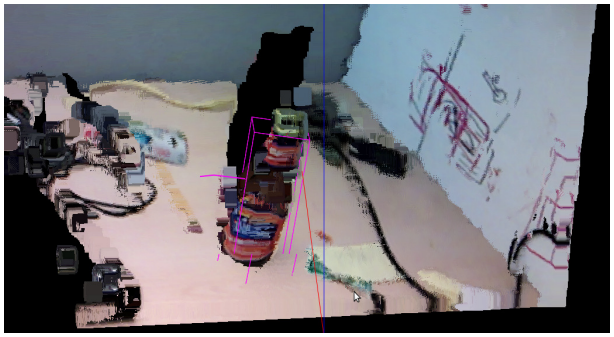

Fig. 5: Example of a 3D bounding box of a detected object tracked over time. The track history is the pink line off the side of the box

The second pipeline (object detection pipeline) leverages RGB trained objects detections to find graspable objects based on deformable part histogram of oriented gradients, but newer detectors could easily be leveraged as the final output need only be 2D bounding boxes. Furthermore, the entire

point cloud within graspable range is meshed for a richer representation of both objects and obstacles. The second pipeline is used for tasks where the robot is asked to pick up a particular type of object, like a beer bottle from a cluttered table full of other objects (Fig. 5).

B. Grasp Heuristic

There are various methods for generating valid grasps, but most of them usually simulate grasping solutions on a model of the object to be grasped. These methods are usually very computationally intensive and need an accurate model of the object. In the case of household manipulation, when one thinks of tasks like cleaning a table by removing objects from it, or picking up a bottle of water to hand over to someone, accurate representations of the object are not necessary and a crude object model is enough to generate grasp points. This fact has been exploited here along with generic grasp heuristics that compute potential grasp poses.

Algorithm 1 Grasp heuristic for cylindrical objects on tables

Require: Target pose T, Center position of the upper-arm in the ready config. $joint3$, Table plane normal N

1: Approach vector, $A \leftarrow \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}$
2: $\theta_{best} \leftarrow \arctan2(T.y - joint3.y, T.x - joint3.x) +$ wrist offset
3: **for** $d\theta$ in range$(0, \pi/2)$ **do**
4: **for** θ in $(\theta_{best} + d\theta,\ \theta_{best} - d\theta)$ **do**
5: rot_N $\leftarrow AngleAxis(\theta, N)$
6: grasp_pose $\leftarrow T \times$ rot_N $\times A$
7: grasp_poses.add(grasp_pose)
8: **return** grasp_poses

One of the heuristics used to grasp cylindrical objects like bottles and cans from a table or a counter-top is described here. The heuristic creates natural looking grasp poses. This particular heuristic aligns the robot's forearm along the vector between its shoulder joint and the object center in the XY plane. This has the effect of minimizing extreme joint angles and keeping the arm in a natural looking configuration.

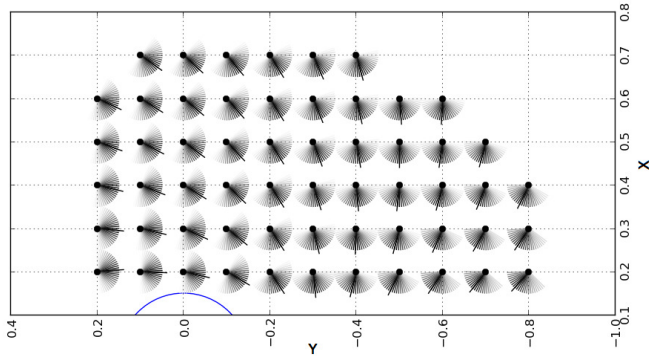

Fig. 6: Grasp heuristic for natural looking grasps for right arm for different positions on the table.

Fig. 6 shows the grasp heuristic for an array of positions on a table (black dots). The robot is centered at the blue circle at (0,0) of the image. The longer, darker arrows point to the approach angle of the hand. Darker angles are tried before lighter angles.

Algorithm 1 describes the algorithm in detail. Basically, θ_{best} is the angle from the target point to the center of the upper-arm in the ready configuration, offset by the 30 degree wrist wedge. Then lesser quality grasp angles are on either side.

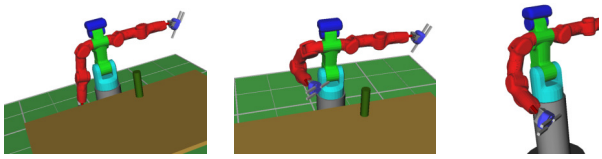

Fig. 7: (a) The right arm in the hang position. (b) The right arm in the wing position. (c) The right arm in the natural looking configuration used for weighting the joint positions.

Another simple heuristic that can dramatically speed up planning time is to simply plan from an easier arm configuration. If the start configuration is close to the goal configuration and is relatively free of obstacles, the planner can be very fast. Therefore, before doing some table manipulation, we put the robot's arm in a "ready" configuration where the robot's hand is above the table with the elbow bent (Fig. 7b).

C. Heuristic initialized optimization based planner

In our approach, a sequential convex optimization problem is formed by creating constraints from the kinematic properties of the robot system and the interactions between the environment and the robot. The trajectory is then represented as a set of T waypoints. The following objective function is used -

$$f(\mathbf{q}_{1:T}) = \sum_{t=1}^{T} ((\mathbf{q}_{t+1} - \mathbf{q}_t)^T Q_1 (\mathbf{q}_{t+1} - \mathbf{q}_t) + (\mathbf{q}_t - \mathbf{q}_{nom})^T Q_2 (\mathbf{q}_t - \mathbf{q}_{nom}) + \mathbf{d}_\Delta^T Q_3 \mathbf{d}_\Delta) \quad (1)$$

where $\mathbf{q}_t \in \Re^K$ describes the K DoF joint configuration at the t-th timestep. The entire trajectory, $\mathbf{q}_{1:T}$ is represented as a sequence of joint configurations, Q_i are weights, $\mathbf{q}_{nom}$ denotes a nominal configuration and $\mathbf{d}_\Delta$ is the Cartesian deviation between the robot state $\mathbf{q}_t$ and the desired configuration. These quadratic terms represents penalizations on the weighted sum on the joint deviations between the waypoints, joint deviations from the nominal configuration, and the Cartesian deviation of a link frame from a desired frame. The first term smoothens the trajectory, the second term pushes the robot configuration towards the nominal configuration, and the third term pushes various link frames to desired poses.

Many different constraints on the robot motion can be added. Some of the various robot constraints that are used in this planner are -

- Joint limits constraint can be specified as $\mathbf{q}_t - \mathbf{q}_{min} > 0$ and $\mathbf{q}_{max} - \mathbf{q}_t > 0$.

- Joint configuration constraint can be written as $\mathbf{q}_t - \mathbf{q}_d = 0$, which will basically lock the joint configuration to $\mathbf{q}_d$ at time t.
- Collision avoidance constraint - The robot is modeled as a group of convex collision elements and the constraint limits all the collision pairs to be greater than a min tolerance threshold.

After potential grasp poses are computed using the grasp heuristic described above, IK solutions are generated using IKFast [21]. A weighted distance based on the robot's link geometry is calculated in the configuration space of all the solutions from the "natural" robot configuration (Fig. 7c). The solutions are then sorted based on the weighted distance from the current configuration. The planning space is constrained by putting barrier walls behind the robot in the planning environment task space so that the motions that are generated by the planner look human-like and do not have giant arm swings.

Algorithm 2 Heuristic initialized optimization based planner

Require: Target pose T, Environment mesh $\mathbf{M}$, Seed trajectories $\mathbf{t}_{seed}$

```
Potential grasp poses, G ← graspHeuristic(T)
for grasp_pose in G do
    Q ← ikFast(grasp_pose)
    q ← Natural IK configuration
    distance ← Σ_{i=1}^{n} (w_i × (Q_i − q_i))^2
    ik_sols.add(Q, distance)
sort(ik_sols)
for id in t_seed do
    p ← problem(t_seed[id], M, costs, constraints)
    NEW_THREAD(id, solve(p), result[id])
traj = result[successful_thread_id]
return traj
```

The planning environment now has a decimated mesh of the environment, virtual barrier walls, and a bunch of potential grasp joint configurations for planning a trajectory. The planning problem is then constructed as an optimization problem with a seed trajectory, the robot environment, the costs and the constraints. The costs and constraints added are a joint velocity cost on every joint based on the joint weights calculated earlier from the link geometry, a collision cost that penalizes when the robot gets closer to within $0.1m$ of an obstacle, the target configuration constraint, and a Cartesian velocity of the end effector constraint. The maximum number of allowed solver iterations is set at 20. Since convex optimization solvers are prone to getting stuck in local minima, different seed trajectories are used to construct several problems. These problems are then solved in separate threads on the robot computer. An open source convex optimization solver (BPMPD [22]) is used. Once a solution is found with a particular seed trajectory, all the other threads are killed and the collision free planned trajectory is returned.

Depending on the computational capabilities of the on board computer on the robot, the number of threads that can be spawned will vary. In the tabletop manipulation scenario, two threads with a no waypoint seed trajectory, and a wing waypoint seed trajectory worked very well. Based on the application, the planner can be tuned to seed itself with waypoints that are spaced to avoid obstacles in the manipulation workspace by statistically estimating the probability of major obstacle positions in the environment.

Finally, for the final approach, range sensors in the palm of the hand and on the bottom of the hand are used to servo the robot hand close to the object for grasping. This is required for two reasons - one, planning exactly to the grasp pose is difficult as the object to be manipulated is modeled as an obstacle in the planner, and two, the lower precision of the arm due to a low cost construction.

V. Experiments and Results

Obstacle avoidance while planning for a trajectory to do a task is vital for manipulating in a household environment. We did a few experiments to figure out a good motion planner that works well when obstacle meshes are taken into account. We pitted our optimization based planner against a sampling based planner to see how good each one performs by looking at their planning times w.r.t the number of meshes in the planning environment. The amount of planning time required for a BiRRT planner increases with the number of mesh triangles but it is not the case for our optimization based planner (Fig. 11).

We compared the performance of the trajectory optimization planner with the grasp heuristic and without the heuristic. The target pose was added as a constraint to the planner and costs were added to simulate the effect of the grasp heuristic. The planner failed to converge to a solution in some of the cases and took a much longer time to plan a path.

Experiments to gauge the performance differences of the heuristic initialized trajectory optimization planner and the BiRRT planner were also done. The results are shown in Table II. The plan times for both the planners were considered by giving the robot a target pose on a table with the starting positions being under (hang) and over (wing) (Fig. 7a, Fig. 7b) the table. The average plan time for TrajOpt in the hang position is better than the average BiRRT plan

TABLE II: Performance of BiRRT with our optimization based planner with a grasp heuristic

Planner type	Initial positions	Planning time					
		0-1 s	1-2 s	2-3 s	3 -5 s	5-10 s	Average
BiRRT	Hang (198 poses)	0%	0%	2.53%	60.1%	37.37%	4.6 seconds
	Wing (198 poses)	93.43%	5.05%	1.52%	0%	0%	0.21 seconds
Heuristic initialized trajectory optimization	Hang (198 poses)	87.37%	11.61%	0%	1.02%	0%	1.12 seconds
	Wing (198 poses)	98.98%	1.02%	0%	0%	0%	0.59 seconds

Fig. 8: Executing a plan generated by the grasp heuristic initialized optimization based planner to a beer bottle in a cluttered environment.

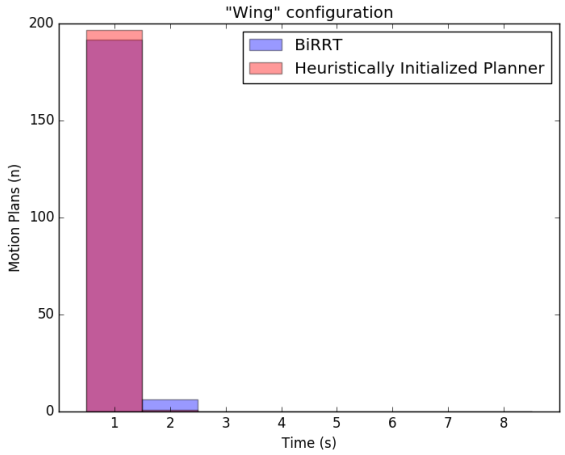

Fig. 9: BiRRT and heuristically initialzed planner with a seed wing configuration trajectory and a natural IK goal.

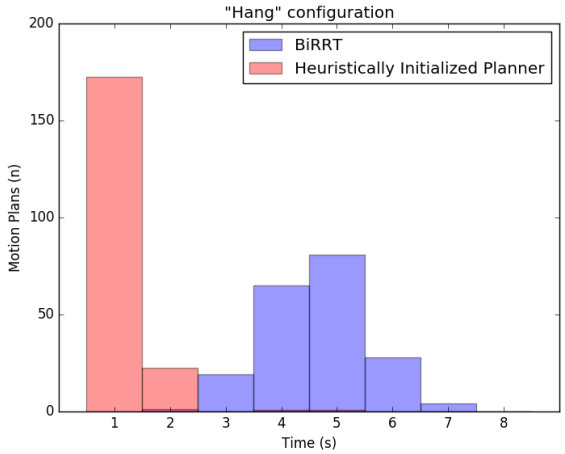

Fig. 10: BiRRT and heuristically initialzed planner with a seed hang configuration trajectory and a natural IK goal.

time by more than 3 seconds and the plan time in the wing position is about 0.4 seconds slower in TrajOpt than BiRRT.

Fig. 9 and Fig. 10 shows the histogram plots of the number of plans where the X axis shows the seconds taken to plan and the Y axis shows the number of plans generated. The BiRRT and optimization planner results for the wing position look very similar but the results for the hang position are very different showing the optimization planner to be more reliable in terms of planning time.

We also performed experiments by using a random IK solution instead of the "natural IK" solution as the goal configuration. Both the BiRRT and the optimization based planner successfully planned a path in most cases but the plans looked very non-intuitive and in some cases, it took more than 20 seconds to plan a path.

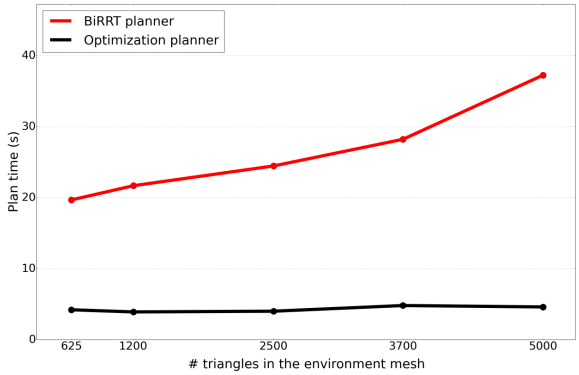

Fig. 11: Planning time using BiRRT and our optimization planner vs. number of mesh triangles in the environment.

VI. Conclusions

We have presented our low cost approach to designing and manufacturing the arms and torso of a wheeled humanoid robot for performing tasks in the household. With commodity motors and commodity parts, building custom gearboxes and electronics, it is possible to build consumer-capable manipulation without sacrificing speed, controllability, or lift capacity required to operate in a home environment. While we believe there is still an order of magnitude improvements left before the price point could be palatable to a consumer, the design of our arm provides guidance for future development.

Based on our results, it was shown that optimization based planners are better suited for doing manipulation tasks in a household. We can add new costs/constraints like end effector velocity in the task space, secondary goals of holding something upright while manipulating, etc. in the planning problem easily. We introduced a grasp heuristic initialized trajectory optimization planner to plan for trajectory solutions reliably and quickly which uses a decimated mesh representation of the environment to avoid obstacles in the environment (Fig. 8). The planner reliably plans within a couple of seconds every time.

VII. Acknowledgements

Thanks to our previous team members: Annan Mozeika, Mark Claffee, Jamie Milliken, Erik Steltz, and Tim Ohm.

REFERENCES

[1] "Rethink robotics sawyer - harmonic drive motors," https://www.allied-automation.com/meet-sawyer-baxters-little-brother-robot/, accessed: 2017-01-26.

[2] "Universal robots - harmonic drive motors," http://harmonicdrive.de/en/applications/universal-robots, accessed: 2017-01-26.

[3] "Kinova robotics - harmonic drive motors," http://www.maxonmotorusa.com/medias/sys_master/root/8817214586910/Kinova-Robotics.pdf?attachment=true, accessed: 2017-01-26.

[4] "Rethink robotics - robot prices," http://www.rethinkrobotics.com/build-a-bot/baxter/, accessed: 2017-01-26.

[5] "Kinova robotics - robot prices," http://www.kinovarobotics.com/service-robotics/build-prices/, accessed: 2017-01-26.

[6] "Franka emika - robot datasheet," https://www.franka.de/, accessed: 2017-10-07.

[7] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003, September 14-19, 2003, Taipei, Taiwan*. IEEE, 2003, pp. 1824–1829. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.2003.1241860

[8] *Bayesian Grasp Planning*, 2011.

[9] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 2, 2000, pp. 995–1001 vol.2.

[10] J. Bialkowski, M. Otte, S. Karaman, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning via safety certificates," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 767–796, 2016. [Online]. Available: http://dx.doi.org/10.1177/0278364915625345

[11] J. E. King, J. A. Haustein, S. S. Srinivasa, and T. Asfour, "Nonprehensile whole arm rearrangement planning on physics manifolds," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2508–2515.

[12] N. Ratliff, M. Zucker, J. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 489–494.

[13] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 4569–4574.

[14] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, 2014.

[15] L. Li, X. Long, and M. A. Gennert, "Birrtopt: A combined sampling and optimizing motion planner for humanoid robots," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov 2016, pp. 469–476.

[16] T. W. Dorn, A. G. Schache, and M. G. Pandy, "Muscular strategy shift in human running: dependence of running speed on hip and ankle muscle performance," *Journal of Experimental Biology*, vol. 215, no. 11, pp. 1944–1956, 2012. [Online]. Available: http://jeb.biologists.org/content/215/11/1944

[17] M. Mihelj, "Human arm kinematics for robot based rehabilitation," *Robotica*, vol. 24, no. 3, pp. 377–383, May 2006. [Online]. Available: http://dx.doi.org/10.1017/S0263574705002304

[18] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[19] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[20] J. W. Sensinger, S. D. Clark, and J. F. Schorsch, "Exterior vs. interior rotors in robotic brushless motors," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2764–2770.

[21] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010. [Online]. Available: http://www.programmingvision.com/rosen_diankov_thesis.pdf

[22] C. Mszros, "The bpmpd interior point solver for convex quadratic problems," *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 431–449, 1999. [Online]. Available: http://dx.doi.org/10.1080/10556789908805758