9th IEEE-RAS International Conference on Humanoid Robots
December 7-10, 2009 Paris, France

Integrating Grasp Planning and Visual Feedback for Reliable Manipulation

Rosen Diankov* Takeo Kanade* James Kuffner*
*The Robotics Institute, Carnegie Mellon University

Abstract— **We present a vision-centric manipulation framework for reliably performing reach-and-grasp tasks in everyday environments. By combining grasp planning and visual feedback algorithms, and constantly considering sensor visibility, the framework can recover from sensor calibration errors and unexpected changes in the environment. Although many current robot systems include planning and vision components, these components are treated independently, which reduces the capability of the system from making informed decisions. Our proposed framework incorporates information in a data-driven way from both planning and vision modalities during the planning and execution phases of the task. The planning phase generates a plan to move the robot manipulator as close as safely possible to the target object such that the target is easily detectable by the on-board sensors. The execution phase is responsible for continuously choosing and validating a grasp for the target while updating the environment with more accurate information. We stress the importance of performing grasp selection for the target during visual-feedback execution because more precise information about the target's location and its surroundings is available. We evaluate our framework on several robot platforms in simulation.**

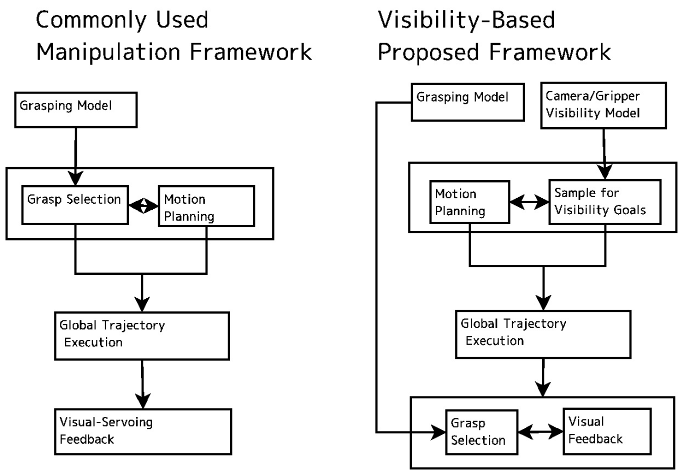

Commonly Used Manipulation Framework
Visibility-Based Proposed Framework
Grasping Model
Grasp Selection
Motion Planning
Global Trajectory Execution
Visual-Servoing Feedback
Grasping Model
Camera/Gripper Visibility Model
Motion Planning
Sample for Visibility Goals
Global Trajectory Execution
Grasp Selection
Visual Feedback

Fig. 1. Comparison of a commonly used grasping framework with the proposed framework. Because the grasp selection phase is moved to the visual feedback step, the proposed framework can take into account a wider variety of errors during execution. The robot platforms used to test this framework (bottom).

I. Introduction

Manipulating objects in everyday environments is a fundamental task for any autonomous robot. The key to successful manipulation is to choose a grasping strategy and move to observe the target of the task so that execution is informed of the surroundings. Reaching and grasping the target object requires multiple pieces of information including: knowledge of the robot kinematics, robot geometry, the gripper capabilities, the available sensor modalities, the grasping strategy of the target, and the environment obstacles that must be avoided. In a successful system, each of these sub-components should build a task-specific model during an offline training process, and then extensively use this model for the online robot execution process. As components start being introduced in a robot system, the interplay between each other becomes very important to the success of the task.

Usually robot systems treat the grasp selection and motion planning stages independently of the sensing capabilities of the robot, leading to the ubiquitous **Sense→Plan→Execute** robot strategy. In the first stage, all the sensors agree on a common snapshot of the environment and pass this information to the planners. The planning process then selects target grasps and computes possible configuration goals for the robot [1, 2]. A global plan for the geometric motion of the robot is then created by enforcing any task specific constraints. An execution monitor [3] takes this global plan and uses sensor feedback loops that follow the plan while quickly correcting for errors due to uncertainty, dynamics, or unexpected occurrences [4, 5, 6]. Because the feedback loop relies on a global plan and needs to run faster than 5Hz, it commonly takes the form of a controller that greedily minimizes an error metric using the partial derivatives of that metric with respect to the control inputs. In this three-stage architecture, the gap between the motion planning stage picking the grasps and the sensor feedback stage attempting to reach those grasps can lead to an early commitment of a global plan that can cause failures in the final execution stage. Although these architectures have been shown to work in many scenarios [7], the performance of such systems still do not match the performance of a person executing the same task; particularly, considering the combination of the visibility capabilities from the planning capabilities can increase robot execution.

In the standard visual-servoing formulation, the robot goal is defined with respect to a *task frame*; as new information comes in, the task frame is updated and the robot attempts to get closer to the goal defined in this frame using gradient descent techniques [8, 9, 10]. Here the entire environment with obstacle avoidance is rarely considered because of the computational complexities. It is also hard for such gradient descent techniques to consider non-linear constraints like properly maintaining sensor visibility and choosing grasps.

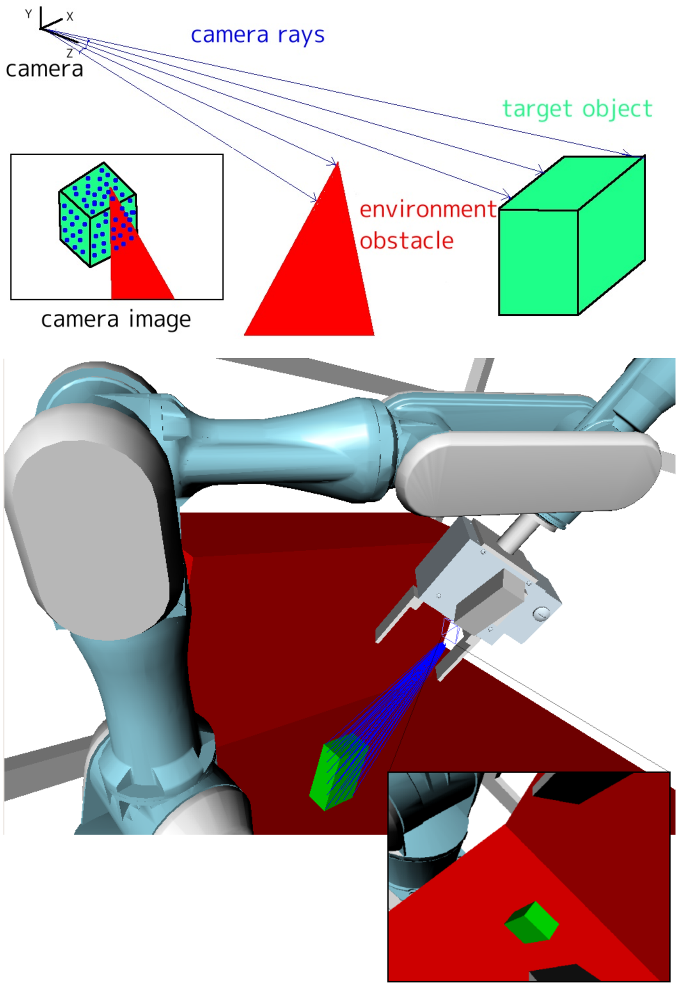


Fig. 2. For every camera location, the object is projected onto the camera image and rays are uniformly sampled from its convex polygon. If a ray hits an obstacle, the camera sample is rejected.

Recently, a framework for combining randomized planners and visual servoing techniques has been proposed [9], which maintains constraints using a planner and locally modifies this trajectory in real-time using visual servoing. Our integrated framework is different from past research in that it re-analyzes the updated environment and grasp selection during the visual-feedback stage.

We propose a framework for reaching tasks that avoids early commitment of a global plan by delaying the decision of the grasp until the execution phase. We particularly focus on the interplay between manipulation planning and the visibility capabilities of a camera attached to the robot gripper. Figure 1 compares the structure of our framework with previous manipulation systems. The fundamental observation driving our framework is that a robot cannot accurately determine the state of the target objects when its camera is far away; therefore, the robot needs to move closer to the object to guarantee accuracy. Systems that combine motion planning with visual servoing usually assume that the grasps remain valid with respect to the task frame, and continue to servo until the robot cannot continue any further. Because the scene can drastically change, the original grasp with respect to the target coordinate system may not be physically possible anymore. For example, it is possible that the new object location can cause parts of the robot to collide with environment obstacles when trying to maintain the previous grasp. There are two parts of the framework we analyze in this paper: the initial motion planning for that takes into account visibility, and the vision-feedback phase for real-time adjustments to the object pose.

II. Planning For Visibility

Before entering the visual feedback loop, the robot should move itself so that its camera has good observability of the target object. This movement should consider the kinematics of the robot, the environment obstacles, and the sensor placement on the robot. It should not, however, consider how to grasp the object. By not considering grasping at this stage, we only need to know a rough location of the object. Furthermore, the required precision of robot execution is reduced because only the environment obstacles need to be considered. The planner can first set a preshape for the gripper that must be maintained throughout the entire time, and only plans for the robot configurations responsible for moving the arm in the environment.

A. Sampling Valid Camera Locations

The first step in planning for the robot approach is to find all camera locations that are within the detection extents of the target object and can observe it with no environment occlusions.

1) Detecting Environment Occlusion: To check if the environment obstructs the target object when viewed from a specific camera location, we shoot all rays from the camera toward the object and check if they hit an obstacle before they hit the object. In the past, this was done by rendering the image and letting the graphics hardware perform the depth computations [11]. For speed and much simpler parallel processing capabilities, we randomly sample the projected object surface and check the ray obstructions using a collision checker (Figure 2).

Consider a pin-hole camera at some transformation T_{camera} with intrinsic matrix K, and let $\mathcal{O}$ represent the object points in the world. The set of all projected points on $\mathcal{O}$ in the camera image is

$$proj(T, \mathcal{O}) = \{proj(K\,T\,p) \mid p \in \mathcal{O}\}. \qquad (1)$$

Each of these points represents a ray coming from the camera. To check if any part of $\mathcal{O}$ is obstructed by the environment, we sample the projected region of the object surface and compute the rays to the object using

$$\mathcal{R}(T) = K^{-1}\ SampleArea(proj(T, \mathcal{H}(\mathcal{O}))), \qquad (2)$$

where $\mathcal{H}$ computes the convex hull and $SampleArea$ uniformly samples the area of the convex hull at a dense enough level not to miss small objects. The object is fully visible in the camera if all rays hit the object and $\mathcal{H}(proj(T, \mathcal{O}))$ is fully inside the camera image (Figure 2). When the projected region of $\mathcal{O}$ is convex, we can greatly speed up sampling the rays in the image. Therefore, we replace $\mathcal{O}$ with its convex hull during environment occlusion checking.

The problem becomes more involved when the robot links are considered since the robot can block the camera view depending on its current configuration. Therefore, environment occlusions have to be checked after a robot configuration candidate is computed. Certains links like the gripper of the robot will be constantly visibile in the camera and will produce a mask in the camera image that the target object should never intersect with. When looking at the scene through the camera image, the gripper should produce a constant mask, which blocks the camera from observing that part of the environment (Figure 3). The camera sampling problem then becomes sampling robot goal configurations such that the attached camera meets the environment occlusion constraints and the target object in the camera image lies outside of the *gripper mask*.

To create a very fast sampling function, we compute the convex polygon of the largest free space in the camera image that does not contain the gripper mask (Figure 3). Computing the largest collision-free convex hull is a hard problem [12, 13], so we use a randomized algorithm that samples supporting points on the boundary of the gripper mask and checks if the resulting convex polygon is free. Depending on the number of supporting points, running this procedure for an hour should yield a good approximation of the largest volume.

A valid camera sample has to fully contain the projection of the object in the free convex region, which is effectively equivalent to checking if the object itself lies in a 3D cone with a convex base. Furthermore, this 3D cone represents the target camera visibility volume and can be treated as a convex hull. The problem is simplified to checking if the object is fully inside this convex hull: $\mathcal{H}(proj(T_{camera}, \mathcal{O})) \subseteq \mathcal{H}(RobotMask)$. However projection is usually slow, so we perform the convex checking in 3D space by converting all 2D lines of the convex polygon in the camera image to 3D planes:

$$\mathcal{H}(\mathcal{O}) \subseteq \mathcal{V}(RobotMask, K, T_{camera}) \qquad (3)$$

In order to simplify sampling robot configurations, we specifically consider cameras attached to the gripper where the manipulator inverse kinematics equations can be directly applied to move the arm to a desired camera location.

2) Vision Algorithm Detectability Extents: Each vision algorithm dictates the camera positions in which the object can be successfully observed in. We can automatically compute these detection extents by waving the camera around the object and recording all extracted object poses (Figure 4). This data is parameterized with respect to the direction of the camera and its distance to the object. Since we assume a pin-hole camera model, the in-plane image offset and the camera roll

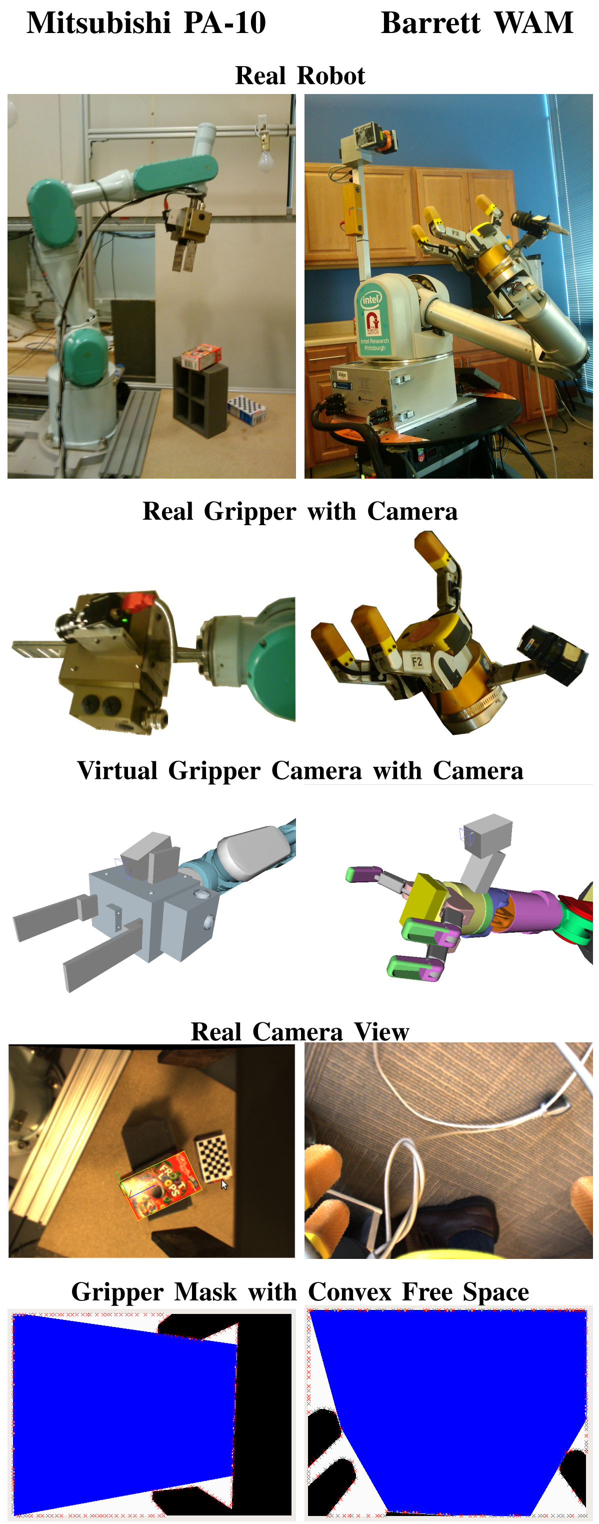
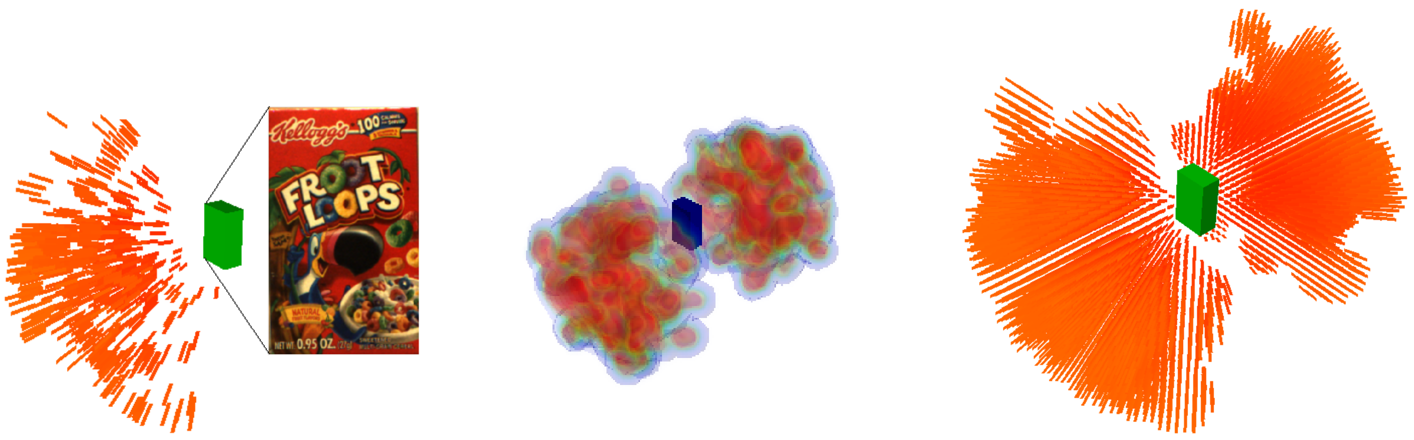


Fig. 3. The real robots with the a close-up simulation of the wrist cameras are show in the top two rows. Given the real camera image, the silhouette of the gripper (bottom) is extracted (black) and sampled (red), then the biggest convex polygon (blue) not intersecting the gripper mask is computed.



(a) (b) (c)

Fig. 4. Camera locations that can successfully extract the object pose are saved (a), this data is then filtered and the isosurface is extracted (b), finally the space is uniformly sampled to get the object detection extents (c).

do not affect detectability as much, so we parameterize the extents with respect to the distance to the object λ, and the direction of the camera view axis v.

Given a real scene, the first task is to sample a camera transformation with respect to the object coordinate system so that object detectability is guaranteed. We first sample the camera distance λ and direction v from the extents model. Then given a random roll angle θ, the camera rotation in the object coordinate system is computed by

$$R(v,\theta) = Rodrigues(\begin{bmatrix}0\\0\\1\end{bmatrix} \times v, \cos^{-1} v_z) \begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix} \tag{4}$$

where $Rodrigues(v,\alpha)$ is the rotation around an axis v by angle α. To compute the camera translation, we constrain the ray pointing towards $\mathcal{H}(RobotMask)$ to point toward the object origin. The camera translation in the object coordinate system then becomes

$$-\lambda\, R(v,\theta)\, K^{-1}\, \mathcal{H}(RobotMask)_{center}. \tag{5}$$

3) Sampling with Robot Kinematics: Algorithm 1 shows how to efficiently sample a camera location such that all the constraints are maintained. The algorithm first samples the transformation of the camera outlined above using the detectability extents model specific to the vision algorithm. Then the object is checked if it lies completely inside the camera visibility volume $\mathcal{V}$ computed by the gripper mask. If this test passes, we sample inverse kinematics solutions of the manipulator until we find a collision free robot configuration. For every solution, we set the robot configuration q and check if any part of the environment or robot is occluding the object using ONOBJECT. Since we need to guarantee that a ray r will hit the object if there is nothing occluding it, ONOBJECT works directly on $\mathcal{H}(\mathcal{O})$ as shown in Figure 2.

To achieve faster sampling times, we perform the following optimizations:

Algorithm 1: $q \leftarrow$ VISIBLECONFIGURATION($\mathcal{O}$)

```
1 while $\{v,\lambda\} \leftarrow$ SAMPLEDETECTIONEXTENTS() do
2     $T_{camera} \leftarrow T_{\mathcal{O}} \begin{bmatrix} R(v,\theta) & -\lambda\, R(v,\theta)\, K^{-1}\, \mathcal{H}(RobotMask)_{center} \\ 0 & 1 \end{bmatrix}$
3     if $\mathcal{H}(\mathcal{O}) \subseteq \mathcal{V}(RobotMask, K, T_{camera})$ then
4         for $q \leftarrow$ INVERSEKINEMATICS($T_{camera}$) do
5             if $\forall_{r \in \mathcal{R}(T_{camera}^{-1})}$ ONOBJECT($r, q, \mathcal{H}(\mathcal{O})$) then
6                 return $q$
7         end
8 end
```

- SAMPLEDETECTIONEXTENTS samples without replacement.
- Once we have the camera transform we check collision with just the gripper before going through the full inverse kinematics computation. Because the gripper preshape is set, all the child link transforms of the gripper can be determined regardless of the arm joints.

B. Motion Planning

In order to plan to the sampled camera regions, we use a slightly modified BiRRT [14]. Instead of fixing the number of robot goal configurations sampled using Algorithm 1, we randomly add these samples with a given probability during the searching process. This allows all possible goals to be considered while not demanding their full computation during planner initialization.

III. VISUAL-FEEDBACK WITH GRASP SELECTION

One necessary component for robust behavior is a real-time visual-feedback phase that can compensate for changes in the environment. The purpose of this phase is to update the virtual world quickly and to decide if the current plan is still valid or needs to change. In order to motivate the necessity for grasp selection in the visual feedback phase, consider the example of a robot attempting to grasp one of two objects in a scene. When the robot first starts the manipulation planning phase, it computes that the objects are 20mm apart and decides that it can safely put one of its fingers between the objects. As the robot switches to its visual-feedback loop and starts getting better measurements of the objects, it sees they are actually 5mm apart and that it cannot put its finger between the two objects. If the visual-feedback phase does not consider the grasp selection process during visual feedback, it cannot change grasps to compensate for this error and will have to restart the entire planning process from the start.

To compensate for uncertainty and large changes in the environment, we propose a visual feedback framework that includes grasp selection and maintains visibility constraints.The grasp selection process takes into account both the collision obstacles and the current robot position. The camera in every new configuration is validated using the methods in the previous section. Although many grasps could be collision-free and reachable from the robot's perspective, the selection process is more successful when they are prioritized depending on the current environment [15]. We use two metrics to prioritize grasps. The first is the difference between rotations of the target grasp and the current robot wrist:

$$\delta(q,T) = \min(|T_g(q)^{quat} - T^{quat}|, |T_g(q)^{quat} + T^{quat}|) \quad (6)$$

where T_g gives the grasp frame with respect to a robot configuration q. Each grasp is checked for the existence of a collision-free inverse kinematics solution. The second metric for grasp prioritization is the distance between the solution and the current robot configuration. Algorithm 2 shows how the visual feedback algorithm orders the grasps using these two metrics before calling the stochastic gradient descent algorithm. γ forces the closest configuration solutions to be considered first before the farthest ones.

Algorithm 2: path $\leftarrow$ VISUALFEEDBACK(q_{start}, G_{raw})

```
1  $G \leftarrow$ SORT(**using** $\delta(q_{start}, G_{raw})$)
2  $\gamma \leftarrow 2.5$
3  **while** $G \neq \emptyset$ **do**
4      **for** $g \in G$ **do**
5          $q_{goal} = \arg\min_{q \in IKSolutions(g)} \delta(q, q_{start})$
6          **if** $\delta(q_{goal}, q_{start}) < \gamma$ **then**
7              $path \leftarrow$ RANDOMDESCENT($q_{start}$,$q_{goal}$)
8              **if** $path \neq \emptyset$ **then**
9                  **return** $path$
10             **else**
11                 $G$.remove($g$)
12             **end**
13     **end**
14     $\gamma \leftarrow 1.5\ \gamma$
15 **end**
16 **return** $\emptyset$
```

We define $\mathcal{C}_{visible}$ as the space of all collision-free robot configurations in $\mathcal{C}_{free}$ that maintain the visibility constraints with the object:

$$\begin{aligned} \mathcal{C}_{visible} = \{\ & q \mid q \in \mathcal{C}_{free}, \\ & \mathcal{H}(\mathcal{O}) \in \mathcal{V}(RobotMask, K, T_{camera}(q)), \\ & \forall_{r \in \mathcal{R}(T_{camera}(q)^{-1})} \text{ONOBJECT}(r, q, \mathcal{H}(\mathcal{O}))\ \} \end{aligned} \quad (7)$$

Algorithm 3 shows the visual feedback algorithm. Given a grasp transform, we first find the closest inverse kinematics solution in configuration space and set that as the goal. Then we greedily move closer to the goal and validate with $\mathcal{C}_{visible}$. After the grasp frame T_g gets within a certain distance τ from the goal grasp, we start validating with $\mathcal{C}_{free}$ instead since it could be impossible for the object to be fully observable at close distances. Because the gripper is already very close to the object and the object is not blocked by any obstacles due to the visibility constraints, such a simple greedy method is sufficient for our scenario. Another advantage of greedily descending is that an incomplete plan can be immediately returned for robot execution when planning takes longer than expected.

Algorithm 3: path $\leftarrow$ RANDOMDESCENT(q_{start}, q_{goal})

```
1  path $\leftarrow q_{start}$
2  **for** $i$ = $1$ to $N$ **do**
3      $q_{best} \leftarrow \infty$
4      **for** $i$ = $1$ to $M$ **do**
5          $q' \leftarrow$ SAMPLENEIGHBORHOOD($q$)
6          **if** $q' \in \mathcal{C}_{free}$ **and** $\delta(q', q) < \delta(q_{best}, q)$ **then**
7              **if** $\delta(T_g(q_{goal}), T_g(q')) > \tau$ **or** $q' \in \mathcal{C}_{visible}$
               **then**
8                  $q_{best} \leftarrow q'$
9      **end**
10     **if** $q_{best} \neq \infty$ **then**
11         path.$add(q_{best})$
12         **if** $\delta(q_{best}, q_{goal}) < \epsilon$ **then**
13             **return** $path$
14 **end**
15 **return** $\emptyset$
```

One advantage of this framework is that he combined planning times for the first planning stage and the second visual feedback stage are actually comparable to the planning times of previous manipulation systems that do not even consider visibility [15, 16, 17]. Because the goal of the first stage is only to get close to the object, it finishes really quickly. Finally the second stage does not have to consider complex planning scenarios since the target is right in front of the camera, this allows it to also finish quickly. By dividing the problem in two steps and conquering each individually, we are able to achieve very fast global planning times.

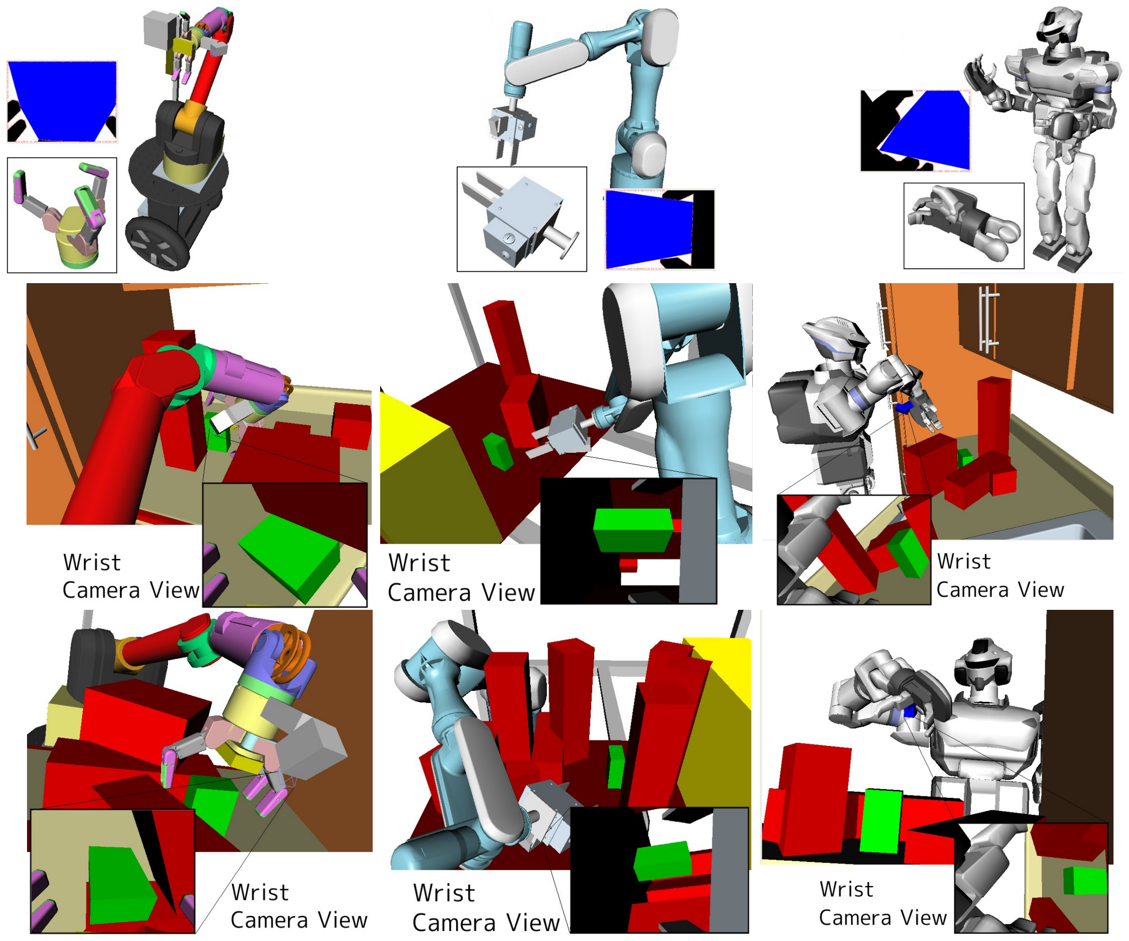


Fig. 5. The scenes tested with proposed visibility framework. The images show the robots at a configuration such that the visibility constraints are satisfied.

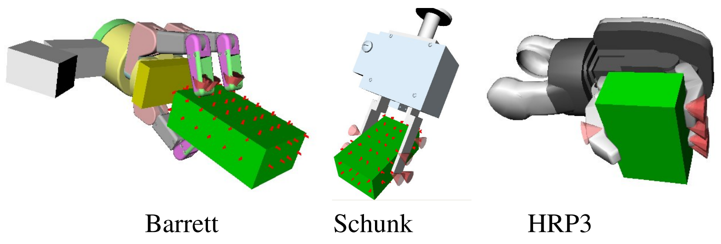


Fig. 6. When building grasp tables, the object surface (green) is sampled (red) and friction cones for contacts are extracted for computing grasp stability.

IV. Experiments

To test the effectiveness of the framework, we have three different robots perform reach-and-grasp tasks in complex environments (Figure 5). All planners and experiments were implemented in the OpenRAVE [18] system using C++, Python, and Octave. First we show results of the planning times and success rates of the individual components. Then we describe how we implemented and tested this framework on the Herb robot system.

In order to successfully get a robot to function in this framework, we perform the following steps within the OpenRAVE framework:

1) Attach camera to wrist and compute its intrinsic and extrinsic calibration.
2) Automatically create analytical inverse kinematics solver using OpenRAVE's **ikfast** generator.
3) Record all poses of target object that can be successfully detected by the camera.
4) Automatically generate a stable grasp set for the target object.
5) Using the robot gripper mask, compute all possible camera locations within the target object coordinate system that can observe the object without any occlusions.

To compute average running times, we create 20 scenes with randomly placed obstacles and record the times for each individual component.

For the first planning stage, we record the sampling and planning times. The average time it takes to sample the first good valid inverse kinematics solution that meets all the visibility constraints is shown in (Table I). Even with all the constraints, it surprisingly takes a short amount of time to sample a goal. Furthermore, we combine the sampling with the BiRRT planner and compute the average time it takes to move the robot from an initial position to a sampled goal. Note that planning times include the time taken for sampling

	PA-10	WAM
Sample First Solution	**0.026s**	**0.009s**
Sample First Solution (many obstacles)	**0.538s**	**0.097s**
Planning Time	**0.188s**	**1.215s**
Planning Time (many obstacles)	**0.905s**	**1.289s**

TABLE I

AVERAGE PROCESSING TIMES FOR THE FIRST VISIBILITY STAGE FOR THOUSANDS OF SIMULATION TRIALS.

	PA-10	WAM
Few obstacles/Visibility Constraints	**0.626s (93%)**	**1.586s (96%)**
Many obstacles/Visibility Constraints	**0.512s (83%)**	**0.773s (67%)**
Few obstacles/No Visibility	**0.117s (94%)**	**0.406s (97%)**
Many obstacles/No Visibility	**0.098s (86%)**	**0.201s (71%)**

TABLE II

AVERAGE PLANNING TIMES (WITH SUCCESS RATES) OF THE VISUAL FEEDBACK STAGE.

the goals.

For the visual feedback stage, we record average time to complete a plan while maintaining visibility constraints (Table II). In order to compute the how much visibility constraints affect the times, we also record statistics for the feedback stage ignoring the camera. Although the times vary greatly depending on the situation, results show that the feedback algorithm can execute at 2-10Hz. Looking at the planning times, scenes with more obstacles usually finish faster than scenes with fewer obstacles. This phenomena is most likely because obstacles constraining the feasible configuration space of the robot and guide it toward the goal faster.

V. CONCLUSION

In conclusion, the advantage of our visibility-based framework is that we don't limit the motion of the arm at the very beginning by choosing a grasp, we instead choose grasps when we have the most accurate measure of our target object and its surroundings. In previous planning systems we have experimented with, performing both grasping and manipulation planning takes 3-5 seconds. In comparison to the proposed system, the total time of moving a robot arm to a feasible grasp while considering visibility constraints takes less than 2 seconds. We showed timing and success rate experiments with several robot platforms demonstrating the flexibility of the system.

VI. ACKNOWLEDGMENTS

We would like to thank Quality of Life Center and Intel Research Pittsburgh for permission to use the Barrett WAM (special thanks to Mike Vande Weghe for help with hardware). We also thank AIST for allowing us to use humanoid HRP3 model for simulation results. This work is based on research supported by the National Science Foundation under Grant No. EEC-0540865 NSF.

REFERENCES

[1] F. Gravot, A. Haneda, K. Okada, and M. Inaba, "Cooking for humanoid robot, a task that needs symbolic and geometric reasonings," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[2] K. Okada, A. Haneda, H. Nakai, M. Inaba, and H. Inoue, "Environment manipulation planner for humanoid robots using task graph that generates action sequence," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[3] C. Baker, D. Ferguson, and J. Dolan, "Robust mission execution for autonomous urban driving," in *10th International Conference on Intelligent Autonomous Systems*, 2008.

[4] M. Prats, P. Martinet, A. del Pobil, and S. Lee, "Robotic execution of everyday tasks by means of external vision/force control," *Journal of Intelligent Service Robotics*, vol. 1, no. 3, pp. 253–266, 2008.

[5] D. Ferguson, T. Howard, and M. Likhachev, "Motion planning in urban environments: Part i," in *Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems*, 2008.

[6] A. Jain and C. C. Kemp, "Behaviors for robust door opening and doorway traversal with a force-sensing mobile manipulator," in *Proceedings of the Manipulation Workshop in Robotics Science And Systems*, 2008.

[7] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. V. Weghe, "Herb: A home exploring robotic butler," *Journal of Autonomous Robots*, 2009.

[8] M. Prats, P. Martinet, A. P. del Pobil, and S. Lee, "Vision/force control in task-oriented grasping and manipulation," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[9] M. Kazemi, K. Gupta, and M. Mehrandezh, "Global path planning for robust visual servoing in complex environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[10] D.-J. Kim, R. Lovelett, and A. Behal, "Eye-in-hand stereo visual servoing of an assistive robot arm in unstructured environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[11] P. Michel, "Integrating perception and planning for humanoid autonomy," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, July 2008.

[12] L. P. Chew and K. Kedem, "A convex polygon among polygonal obstacles: Placement and high-clearance motion," *Computational Geometry: Theory and Applications*, vol. 3, no. 2, pp. 59–89, 1993.

[13] G. Borgefors and R. Strand, "An approximation of the maximal inscribed convex set of a digital object," in *Image Analysis and Processing ICIAP*, 2005.

[14] J. Kuffner and S. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[15] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner, "Grasp planning in complex scenes," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2007.

[16] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. Kuffner, "Bispace planning: Concurrent multi-space exploration," in *Proceedings of Robotics: Science and Systems (RSS)*, 2008.

[17] S. Srinivasa, D. Ferguson, M. V. Weghe, R. Diankov, D. Berenson, C. Helfrich, and H. Strasdat, "The robotic busboy: Steps towards developing a mobile robotic home assistant," in *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, 2008.

[18] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," Robotics Institute, Tech. Rep. CMU-RI-TR-08-34, July 2008.