Chapter 7

Path planning for informational effects in the real world

From Chap. 6, we have seen that sequential re-planning can achieve better quality grasps that a single attempt in the presence of object pose uncertainty. We have also seen that trajectories designed to maximise tactile information gain achieve successful grasps with fewer iterations.

However, in the previous chapter, several assumptions were made which might be problematic for real world robot deployments. First, a complete model for the object to be grasped was available, in the sense of a dense point cloud or mesh model, as well as a pre-computed grasp for the object's model, therefore only uncertainty in the object pose remained. Second, algorithms all assumed noisy visual sensors to localise the target object, but relied on perfect tactile sensing abilities. Third, a bounding box was constructed around the object to be grasped. This allowed planning of collision-free reach-to-grasp trajectories only for convex objects. For example, a "rim" grasp on a mug would not be possible to plan because it requires placing at least one finger inside

the bounding box. In addition, each re-planning iteration was treated as independent and thus the robot's manipulator was withdrawn to a safe position after a failed attempt to grasp. Chapter 6 demonstrates the validity of the algorithms mainly in simulation and, only as an initial attempt proof of concept, the algorithms were implemented and tested on the Justin robot using an object rigidly fixed to the table (see Fig **??**).

These assumptions all prevent the core methods working on a real robot. In this chapter we discuss how the previous algorithms can be extended and enhanced, to enable relaxation of each of these assumptions, bringing us to a first convincing demonstration of this approach on a real robot. The resulting algorithms can also:

- Compute the target grasp on-the-fly, by incorporating the grasp planning method of [Kopicki et al., 2014].
- Interpret the noisy contact sensors of a real robot hand (using improved Bayes filtering).
- Re-plan trajectories without requiring withdrawal of the manipulator to a safe pose, but where the manipulator can remain in contact with the object.
- Planning dexterous grasping trajectories for non-convex objects.

This work is demonstrated in trials in simulation and on Boris, a half-humanoid robot platform. Empirical results confirm that sequential re-planning achieves a greater success rate than single grasp attempts, and trajectories that maximise information gain require fewer re-planning iterations than conventional planning methods before a grasp is achieved.

7.1 Organisation

This chapter proceeds as follows. In Sec. 7.2, we discuss the limitations of the set of algorithms presented in the previous chapter due to the several assumptions made. We also discuss how to relax these assumptions in order to validate the sequential re-planning approach on a real robot.

Section 7.3 presents the technical contributions of this chapter. We discuss a new method to explicitly address the multi-modal nature of the belief state for the problem of robot grasping. In addition, the section presents a set of engineering or algorithmic solutions which enabled us to implement the sequential re-planning approach for a real robot platform.

Section 7.4 presents empirical results in a real scenario in which a dexterous grasping trajectory has to be planned under non-Gaussian object-pose uncertainty in 6D with shape incompleteness. In addition, this approach is also demonstrated in a virtual scenario. Three strategies are presented.

Section 7.6 summaries the contributions of this chapters.

7.2 Introduction

In Chap. 2, we have seen that the grasp planning problem is composed of four sub-problems: state estimation, grasp synthesis, grasp planning and control. As we have seen in Sec. 2.5, a typical approach is to represent the belief state using prior distributions (usually a Gaussian), select a grasp more robust in the face of uncertainty (pose uncertainty or shape incompleteness) and finally to use tactile feedback to adjust the grasping trajectory, see e.g. [Nikandrova et al., 2013]. The reach-to-grasp trajectory is typically computed using some conventional sampling-based techniques which minimise

Table 7.1: ReGrasp & ReGrasp+IG vs Mycroft & IR3ne at a glance

	ReGrasp & ReGrasp+IG	Mycroft & IR3ne
Target grasp	Pre-computed	Computed on-the-fly
Pose estimation	Mean-shift algorithm	Mean-shift + clustering algorithm
Re-planning	From a safe robot pose	From current pose of the robot
Contact sensing	Assumed to be perfect	Gaussian filter to remove noise

the cost, in Euclidean space, to transfer the robot's end effector to the selected grasp configuration. Comparatively little work has explored the more complex problem of reasoning about uncertainty while planning a dexterous reach-to-grasp trajectory. This is mainly due to the high dimensionality of the configuration space of a dexterous manipulator, and the complexity of determining the effects of an action on the object we wish to grasp. This chapter presents a new set of sequential re-planning algorithms that enable a robot to autonomously operate under object-pose uncertainty in a real scenario.

Nevertheless, the innovative approach I chose in this thesis comes at a cost. First, even in this chapter, it is assumed that a reference model for the object is available. Nevertheless, completeness in this model is not necessary for this set of algorithms to work. Incomplete point clouds can be used as models. Second, it is assumed that a separate sequence of views given by a depth camera give an incomplete point cloud which can be aligned to this model. The shape incompleteness of both the model and the observation of the object results in the object-pose uncertainty being encoded in a belief density for the pose. Given this, the approach presented here selects an active information gathering reach-to-grasp trajectory with respect to the current belief state.

Note that for the new implementation of the sequential re-planning algorithms, no further knowledge on the object we wish to manipulate is needed. Regarding the generation of the target grasp configuration, the system is capable of generating this for a novel object by using the techniques described in [Kopicki et al., 2014]. However, the choice of using a point cloud model of the object has other implications. In the literature, there are no fast

collision detection procedures for non-convex objects represented as point clouds. As a contribution to this thesis I have therefore developed an efficient KD-tree based collision detection algorithm that allows us to efficiently plan dexterous grasping trajectories in such a situation (Sec. 7.3.5). This is useful for real grasping. Point clouds have the appealing property that they can be built on-line and incrementally[1], and that they do not have to be complete. These three properties make point clouds a more desirable representation for manipulation in some ways than surface-based, meshes or volumetric representations such as representation of shapes.

In Chap. 6, the IG planner works with a cost function that allows deviations from the shortest path. These encourage gathering tactile observations that will reduce pose uncertainty in the object location. Section 6.4.2 showed how to reason about informational effects of an action by simply encoding the expected informational value into a cost function. Such a cost function can be minimised using any motion planning technique (here a PRM), such as the ones described in Chap 3. As we have seen at the end of the previous chapter, a drawback of encoding the information value into the cost function is the extra computation during the query phase of the PRM algorithm in order to compute the likelihood of reading a contact. This requires finding the closest surface to the finger tips for each hypothesis every time a node of the PRM is in a neighbourhood of the uncertain region. The new set of algorithms in this chapter also use the procedure described in Sec. 7.3.5 to compute the expected (tactile) observations in the planning process (Sec. 6.4.1), erasing the computational discrepancy between the two sequential re-planning algorithms.

The rest of this chapter presents the new features implemented in the sequential re-planning algorithms in more detail.

[1]Registration procedures allows us to merge several point clouds, collected from different point views, into one.

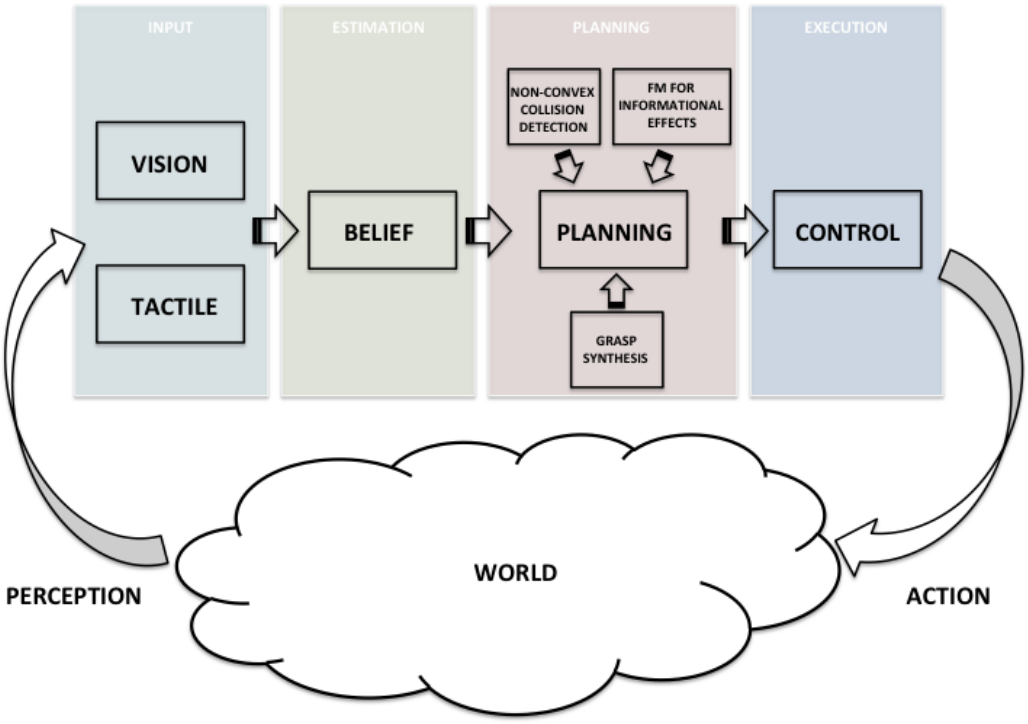


Figure 7.1: The architecture of the system is composed of 4 components: sensory input, pose estimation, motion planning and active control. The arrows show the flow of the system and its interaction with the external world. The system presented in this chapter involve 3 different modules in the motion planning phase: i) collision detection procedure for non-convex objects represented as a point cloud, ii) a forward model (FM) is used to maximise the chance of gathering tactile observations that will reduce pose uncertainty, and iii) a grasp synthesis procedure to compute grasps on point clouds.

7.3 Technical contributions

As explained, the set of algorithms presented in this chapter have a similar formulation to those in Chap. 6. However, there are several key differences to be discussed. This section describes each of them separately.

7.3.1 Mean pose estimate

Section 6.3.2 presented this as the mean of the set of particles in the belief filter. In this chapter a new state estimator is proposed. In chapter 6, the mean was estimated using a mean-shift algorithm on the particle set. This simple estimate of the mean does not work well with multi-modal beliefs. Therefore, in this chapter, I combine the mean-shift algorithm with a hierarchical clustering algorithm to compute the mean of the most promising cluster within a possible multi-modal belief.

This procedure uses the mean-shift algorithm to generate a set of cluster centres from the particle set. Each associated with: i) a score which identifies the "goodness" of the estimate, in terms of how well the estimate represents the entire set of particles, and ii) the number of particles that have contributed to construct the estimate. Let c_i be the centre particle of a cluster, C_i, of the entire set of particles $[p_1, \ldots, p_K]$. The sampling-based model-fitting procedure described in Sec. 6.3.2 computes for each particle an importance sampling weight w_k which expresses the likelihood of best aligning the (dense) model point cloud (assumed to be available in the system) to the (partial) query point cloud. The importance sampling weight can be considered as a measure of how good the pose p_k describes the data, and then the score for the cluster C_i is computed as

$$s_{C_i} = \sum_{k \in [1,\ldots,K]} w_k e^{\|c_i - p_k\|_Q}$$

where $\|a\|_Q$ is defined as $a^T Q a$ and Q is the covariance matrix of the set of particle $[p_1, \ldots, p_K]$. Once a fixed number of clusters C_i is generated, the procedure agglomerates the most similar clusters, if any, and computes the best estimates as the mean of the most promising cluster C_{max}, which is defined as:

$$C_{max} = \arg\max_{C_i} \frac{|C_i|}{\sum_j |C_j|} s_{C_i}$$

where s_{C_i} is the score associated with the cluster C_i, and $|C_i|$ is the cardinality of the cluster C_i in term of number of members.

7.3.2 Grasp synthesis

Section 2.4.2 discussed how to learn grasp types that can be generalised to novel object's shapes and discussed the work of [Kopicki et al., 2014; 2015], which proposes an efficient method to learn dexterous grasp types (e.g. pinch, rim) from a single example that is able to generalise within and across object categories, and with full or partial shape information. As a minor contribution to this thesis, I have integrated this method to enable the system to generate grasps on-the-fly. A user of the system can select the grasp type which will be automatically generated on the novel object's shape.

7.3.3 Re-planning

As we have seen in Chap. 6, the sequential re-planning algorithm planned trajectories assuming only the maximum likelihood observations given the current belief state. Therefore we need to rely on sensory feedback during the execution of the planned trajectory in order to detect whether or not unexpected observations occur. This triggers a belief update, using the observation gathered at execution-time, and consequently a re-planning phase.

As described in Chap. 6, and in [Zito et al., 2012a; 2013b], after a contact the manipulator was moved back to a safe configuration (e.g. outside the uncertain region) prior to each new reach-to-grasp trajectory being planned. This approach was necessary for technical reasons due to the robotic platform in use. This approach was tested on DLR's Rollin Justin robot. The only way to communicate with this platform is via its own controller. This controller accepts trajectories with extra parameters to set, for example, compliance

or activate/disactivate a joint's torque thresholds (or guards). However these parameters cannot be changed on-line during the execution of the trajectory and the controller forbids any movement once an active guard has been triggered. It is possible however to send a trajectory with disabled guards. Therefore the only feasible work around, after making a contact, was to withdraw the robot to a safe configuration with no active guard before the next reach to grasp attempt. This, however, is inefficient and fails to exploit the existing contact in completing the grasp.

To overcome these limitations the modified method, proposed in this chapter, has been implemented on a robotic platform called Boris. Boris' controller enables us to modify settings (compliance and thresholds) on-the-fly at execution time. This allows us to make a contact with an object, stop the robot, and then re-plan from the same configuration, while maintaining contact with the object.

In the experiments presented here, the algorithm uses torque sensors, based on current draw, at each joint of the robot's hand to detect whether or not a link of the hand is in contact with the environment. As a minor contribution to this thesis, a contact detection module has been developed to remove the noise from the sensors, which we will discuss briefly now.

7.3.4 Detecting contacts

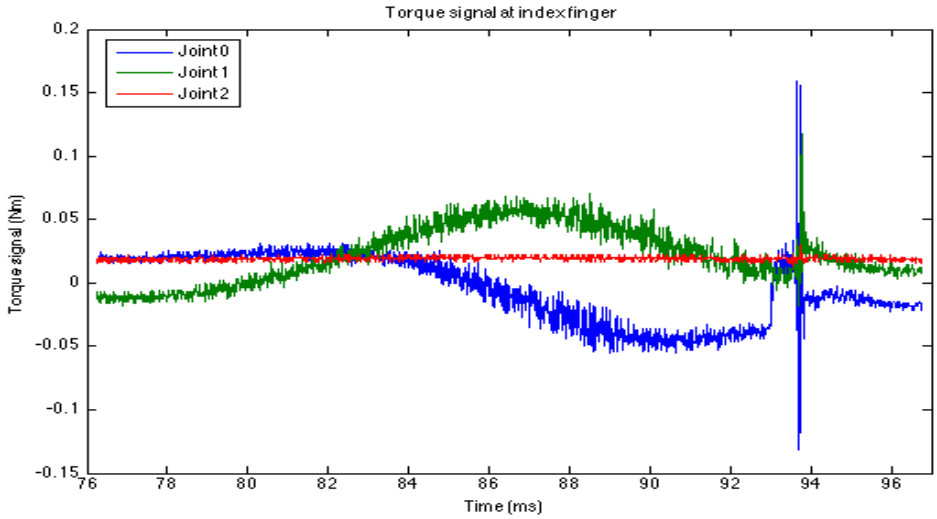
As mentioned, the system developed in this thesis relies on the torque sensors based on the current draw at each joint of the DLR Hit Hand II. Figure 7.2(a) shows the torque signal received from the robot hand during a reach-to-grasp trajectory. Joint 0 (blue) is responsible for the abduction movement, while Joint 1 (green) and 2 (red) are responsible for flexion movements. The fourth joint (Joint 3) of the DLR HIT Hand II is mechanically coupled with Joint 2, so it is not shown in the figure.

Algorithm 5 LOWPASS_FILTER

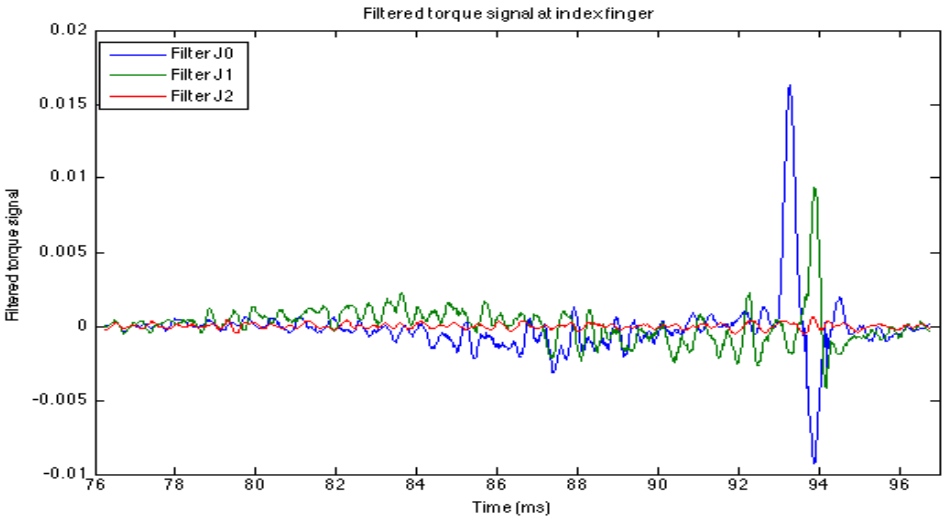
Input: data, mask_size, σ
Output: L
  x ← [-mask_size:.1:mask_size]
  mask ← diff(normpdf(x,0,σ))
  L ← conv(data, mask)

When the robot is commanded to move, the inertias of each finger link, cause significant torque signals to be sensed, even without any contact with external objects. Therefore, an important practical problem is how to remove that portion of the torque signal that is due to the robot's own motion, in order to detect torque changes due to external contacts. To do so, a Gaussian low-pass filter with a fixed window size is used to filter out the noise. Alg. 5 shows the pseudo code for a low-pass filter. The functions diff($\cdot$), normpdf($\cdot$) and conv($\cdot$) are considered as in the MATLAB API, where diff(X) calculates differences between adjacent elements of X along the first array dimension whose size does not equal 1; normpdf(X, μ, σ) computes the pdf at each of the values in X using the normal distribution with mean μ and standard deviation σ; and conv(u, v) returns the convolution of vectors u and v. In this implementation $mask\_size$ = 2 and σ = 1.0, while *data* is the vector of torques for a particular joint. The length of *data* is equal to the selected window size. In this implementation the window size is equal to 40, which means *data* contains the latest 40 readings from the joint's torque.

Figure 7.2(b) shows the corresponding filtered signals along the trajectory. Ideally, we would like to have a zero signal until the finger makes a contact. However, this is not possible to achieve due to the activity of the joints' actuators, the acceleration of the hand along the trajectory and the changes in the gravity vector according to the changes in the orientation of the hand. Nevertheless, the system allows us to define thresholds to further filter the signals. Via empirical experiments on Boris we determined that a threshold of 0.05 gives an appropriate trade-off between false positives and false negatives.



(a)



(b)

Figure 7.2: The images show the real torque signals (a) and the corresponding Gaussian low-pass filter signal (b) for each joint of the index finger of the DLR Hit Hand II during a reach-to-grasp trajectory. Joint 0 (blue) is responsible for the abduction movement, while Joint 1 (green) and 2 (red) are responsible for flexion movement. The fourth joint (Joint 3) of the DLR Hit Hand II is coupled with Joint 2, so it is not shown. The torque signal is plotted over time. At approximately 93ms a contact with the target object is made, as shown by the increasing of the torque signals.

7.3.5 Planning a dexterous grasping trajectory for non-convex objects

The implementation of this planner uses a modified version of Probabilistic Roadmap (PRM) planning, [Kavraki & Svestka, 1996], to plan trajectories. Crucially, sampling-based approaches like PRMs rely on the ability of rejecting points that are in collision. Testing collision between simple geometrical primitives, such as planes, spheres or cubes, is substantially faster than between non-convex polyhedra. On the other hand, if an object to be grasped is wrapped in a simple primitive bounding box, to achieve computational efficiency, many grasps would simply be impossible to achieve. The left image in Fig 7.3 shows the case of a grasp over the rim of a jug. This grasp requires the thumb to penetrate inside the convex hull of the object. Thus, even if this configuration does not produce any true collisions, it would be rejected by such "naive" collision detection.

Here a fast and efficient collision detection module to cope with objects represented by a point cloud is proposed. Whilst the robot's rigid links are represented by an open-chain of convex polyhedra, the object to be grasped is represented by a 2-level structure. The first level of this structure wraps the point cloud in a bounding box. This level can efficiently avoid checking collisions between the robot's links and the object to be grasped when they are far apart. The second level contains the point cloud organised in a KD-Tree. The KD-Tree implementation is based on the FLANN library [Muja & Lowe, 2014]. An example of the 2-level collision detection is shown in Fig. 7.4.

The planning process uses a collision detection in two cases: i) to reject PRM nodes in collision with an object and ii) to compute the information value for each edge of the PRM. In the former case, the planning process only targets the object as if it were in its expected location (the mean pose of the density function), therefore only the mean pose is used for collision detection. In contrast, the latter case requires us to compute the information value for all the sub-sampled hypotheses. In both cases, when at least

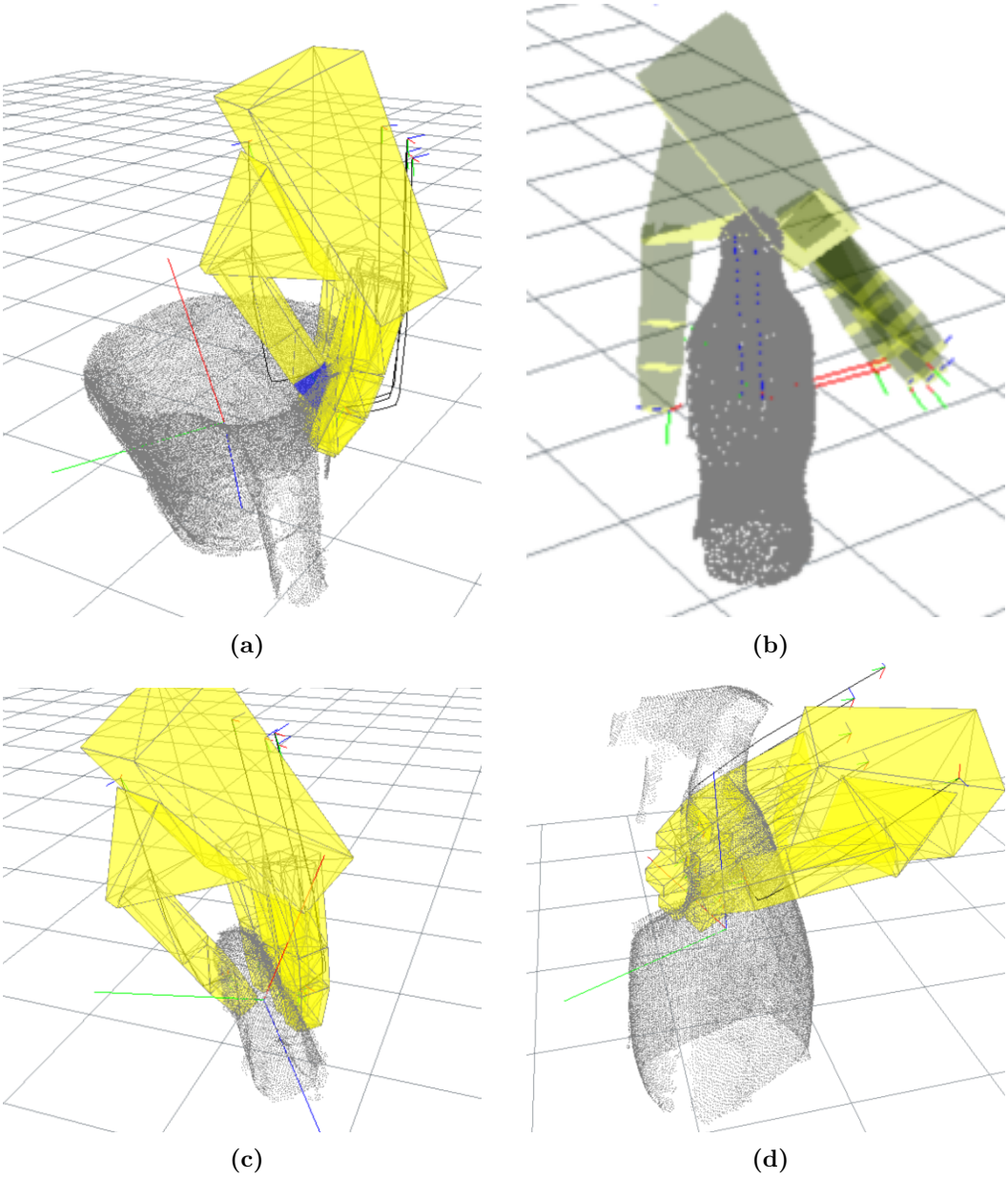


Figure 7.3: Models of the objects and their associated grasps. The image (a) shows a rim grasp on a jug. In this case, the target grasp requires that the thumb be placed on the internal surface of the object, thereby penetrating inside the convex hull. The image **??** shows a top grasp on a bottle of coke. The images (c) and (d) show respectively a rim grasp on a stapler and a Mr Muscle spray bottle. The grasp configurations are computed using the method described in [Kopicki et al., 2014].

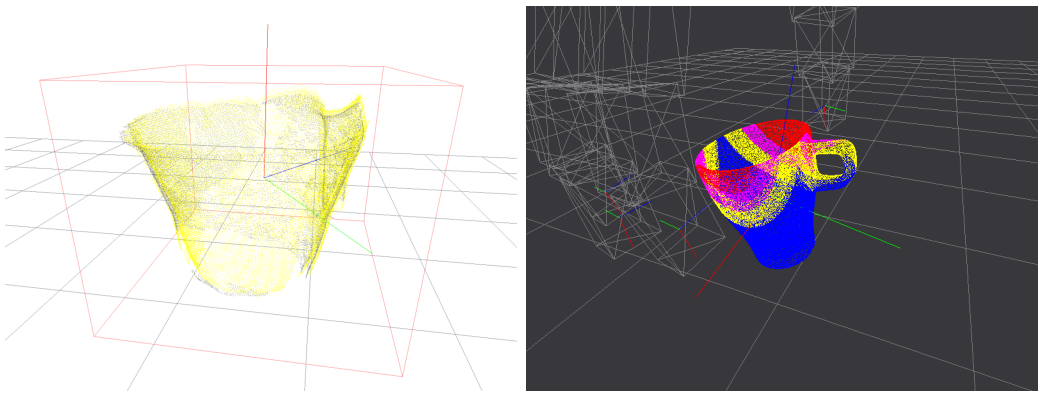


Figure 7.4: The left image shows the target object to be grasped. In this case the object is a jug. The grey point cloud represents the ground truth pose of the object, as it was acquired by a noiseless input source. The yellow point cloud identifies the best pose estimate. The yellow point cloud is organised as a KD-Tree for faster collision detections. The red box represents the bounding box, which prevents unnecessary collision checking between the object and robot's links when they are far apart. The right image shows a point cloud for a mug and the robot hand's configuration. Each point is coloured with respect to the relative distance to the closest finger's link of the robot's hand: red (closest) to blue (furthest).

one bounding box of the robot collides with the bounding box of the object, the second level of the collision detection module is called. By querying the KD-Tree it is possible to retrieve the closest points on the surface of the object to the robot's links, see Fig. 7.4 (right). Each point is then checked to determine whether it collides or not, or to estimate the likelihood of observing a contact.

Algorithm 7 shows the core of the proposed collision detection procedure. This procedure computes a *penetration* value for a particular point p in the point cloud and a set of bounds for the robot hand. The penetration value is an approximation of the distance between p and the closest surface of the robot hand, and it is positive if the point sits inside the bounds, or negative otherwise. Algorithm 6 shows the collision detection procedure used to reject a robot pose in collision. In this case, the procedure rejects a robot pose if the associated penetration value is greater than or equal to zero. It is also possible to treat a point p as a sphere with centre p and radius ρ and this allows us to plan more conservative trajectories.

Let b_i be the boundary of the i–th link in the robot hand, represented as a convex polyhedron. Figure 7.5 shows the boundary b_i as a simple 6 faced polyhedron. Then b_i is composed of a set of triangles $T_i = [t_1, \ldots, t_M]$ and a reference frame $O_i \in SE(3)$. Each triangle $t_j \in T_i$ is composed of three vertices v_{j1}, v_{j2}, v_{j3} and it is possible to compute the normalised normal vector n_j as

$$n = (v_2 - v_1)(v_3 - v_1)$$

$$n_j = \frac{n}{||n||}$$

Note that either n_j or $-n_j$ would be a normal for the triangle t_j. The procedure chooses n_j with the direction that points outside the polyhedron, as shown in Fig. 7.5. Note that since all the normals are computed to point outside the boundary, for a point p to be in collision it must satisfy the following condition:

$$||v_{j1}||_{O_i} - n_j^T p > 0, \qquad \forall j \in [1, \ldots, M]$$

where $||v_{j1}||_{O_i}$ is the distance of the triangle t_j from the reference frame O_i, approximated as the distance to its first vertex v_{j1}, and $n_j^T p$ is the dot product between the normalised normal n_j and the point p. In classical geometry, the distance between a triangle and a point is computed as the length of the projection of the point p onto the triangle or, if the projection does not sit on the triangle's surface, distance to the point's projection onto the closest edge of the triangle. In order to speed up the computation, the implementation I propose computes for each triangle the penetration value as the distance between the point p and the first vertex, v_1, of the closest triangle. If the point sits outside the boundary b_j, the penetration value is the negative of the computed distance. If the triangles t_j are small, this is a good approximation.

Note that the lack of complete information about the object's shape may affect collision

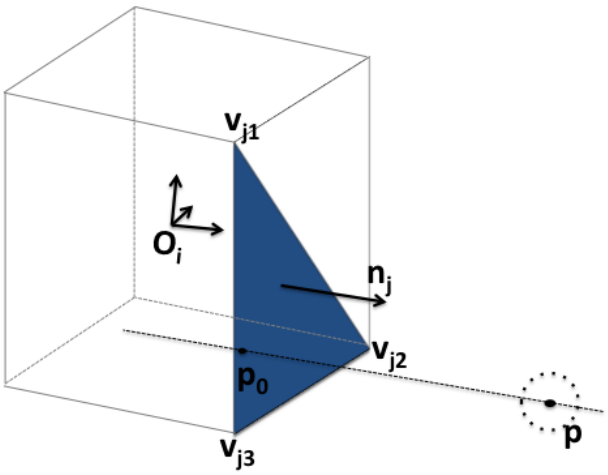


Figure 7.5: The image shows a mesh triangle, t_j, of vertices v_{j1}, v_{j2}, v_{j3} and the normalised normal vector n_j, the reference frame of the mesh O_j and the point p with its projection on the triangle surface. The projection is the point p_0 on the triangle surface which intersects the line passing through the point p with the same direction of n_j. The dotted line around the point p represents the spherical bounding box used for planning more conservative trajectories.

Algorithm 6 CHECK_COLLISION

Input: robot_pose, hand_bounds, kdtree, neighbours, ρ
  closest_points ← KNN_SEARCH(kdtree, robot_pose, neighbours)
  for all $b_i \in$ hand_bounds **do**
    $d \leftarrow$ GET_PENETRATION(b_i, closest_points)
    if $d > -\rho$ **then**
      return true
    end if
  end for
  return false

detection, leading to a reach-to-grasp trajectory which passes through the object. In a real scenario, however the tactile observation that will be generated will cause the robot to stop and the current belief state to update. One limitation of the current

Algorithm 7 GET_PENETRATION

Input: b_i, closest_points
  distance $\leftarrow +\infty$
  for all $p \in$ closest_points **do**
    $d \leftarrow$ GET_PENETRATION_PER_BOUND(b_i, p)
    if distance $> d$ **then**
      distance $\leftarrow d$
    end if
  end for
  return distance

Algorithm 8 GET_PENETRATION_PER_BOUND

Input: b_i, p
  distance $\leftarrow +\infty$
  direction $\leftarrow +1$
  for all $t_j \in b_i$ **do**
    $d_1 \leftarrow ||tj.v_{j1} - p||$
    if distance $> d_1$ **then**
      distance $\leftarrow d_1$
    end if
    if direction > 0 **then**
      $d_2 \leftarrow ||t_j.v_{j1}||_{O_i} - t_j.n_j^T p$
      if $d_2 < 0$ **then**
        direction $\leftarrow -1$
      end if
    end if
  end for
  return direction·distance

implementation, is that this approach does not reason about the relative likelihood that the unexpected observation is given by a mis-estimation of the object pose versus lack of shape information; it simply updates the belief density over the object-pose. As future work, the aim would be to treat this problem as a SLAM problem, which will enable us to compensate for initially incomplete shape information in the object model.

7.3.6 Terminal conditions

The sequential re-planning algorithm terminates its execution when no unexpected contacts occur and the target grasp is achieved. Since we do not have mesh models of the object to be grasped we cannot rely on grasp quality measures (discussed in Sec. 2.4.1) to signal successful termination of the algorithm. Nonetheless, in simulation it is possible to measure the displacement error between the grasp configuration for the final object-pose estimate and the ground truth. A user-defined threshold of tolerance is used to identify whether the grasp has succeeded. On the real robot, the success of the grasp is evaluated by lifting the object. If the robot can hold the object, the grasp is considered successful.

7.4 Results

This section presents the experimental results used to evaluate the new set of algorithms presented in this chapter. As in Chap. 6, three strategies for planning dexterous reach-to-grasp trajectories are evaluated in both the real robot platform Boris and a simulated environment:

- ELEMENTARY: an open-loop trajectory towards the expected object pose without re-planning.
- MYCROFT: a sequential re-planning algorithm without information gathering.
- IR3ne: a sequential re-planning algorithm with information gathering.

In this evaluation the aim is to show that sequential re-planning is capable of achieving higher grasp success rates than single grasp attempts in presence of non-Gaussian object-pose uncertainty in 6 dimensions. It also shows, as in the previous chapter, that planning trajectories that maximise information gain result in fewer re-planning iterations to

achieve a grasp.

First, Sec. 7.4.1 presents empirical results collected on 20 trials for a single object (a jug) using a real robot, in which the ability of these three different strategies to achieve a grasp configuration is tested as well as the benefits of using a mean-shift algorithm with hierarchical clustering to compute the estimate of the object's pose.

In addition, Sec. 7.4.2 presents the results collected on 120 trials in simulation for four objects: a jug, a bottle of coke, a stapler and a Mr Muscle spray bottle. In these experiments, a mean-shift algorithm is still used to compute the pose estimate, and the aim is to evaluate the ability of the three strategies to achieve a grasp configuration.

7.4.1 Experiments on the robot Boris

These experiments are composed of 2 runs of 20 trials each (10 trials using MYCROFT and 10 using IR3ne). The results for the ELEMENTARY strategy are extrapolated from the results collected for MYCROFT, in the sense that the two approaches construct a reach-to-grasp trajectory minimising the same cost function, therefore if MYCROFT achieves (or fails to achieve) a grasp at the first iteration, the ELEMENTARY also would have succeeded (failed) as well.

For both set of trials only one object has been used: the jug shown in Fig. 7.4. A dense point cloud, representing the object model, is assumed to be available at the beginning of each run. This object model is acquired by scanning the operational workspace (Sec. 3.2.5) with a depth camera from 6 different views. These views are merged to generate a single point cloud of the object model. This pre-processing aligns the single view point clouds, registering and eliminating outlier points.

I argue that in order to evaluate the ability of planning more robust reach-to-grasp trajectories in the face of pose uncertainty, a critical parameter is not the shape of the

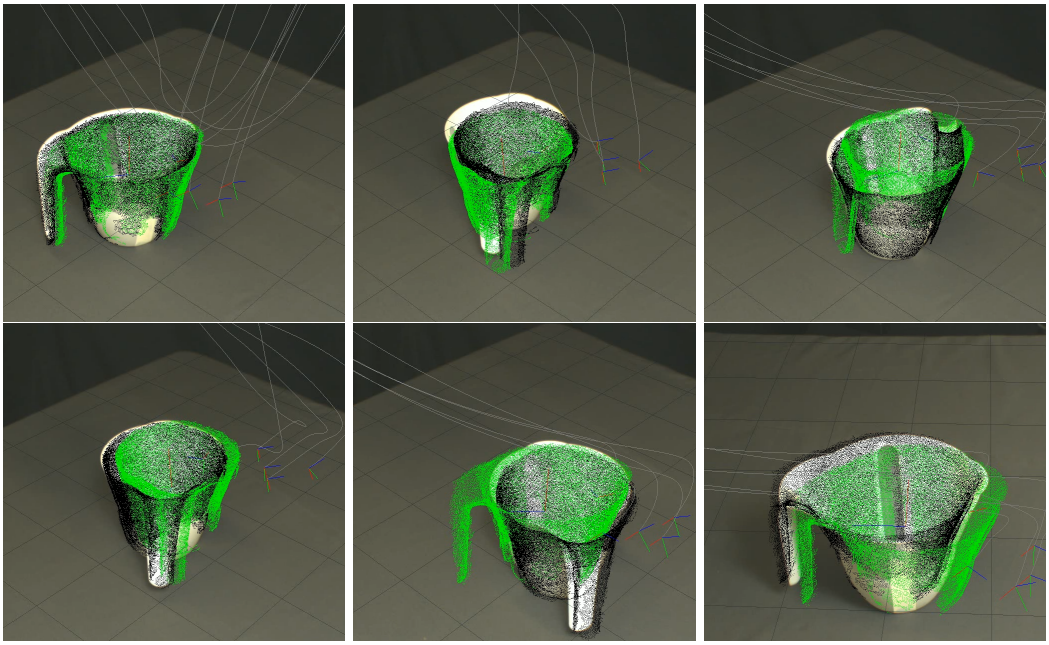


Figure 7.6: Initial pose uncertainty. The sequence of images shows some examples of initial pose estimation during the experiments on Boris. The simulated point cloud overlay the real plastic jug to show the misalignment between the real pose of the object, the ground truth (black point cloud) used to evaluate the simulated results and the initial pose estimate (green point cloud), to which the algorithm attempt the grasp. The lines show the planned trajectory for each fingers.

object but the initial estimation error w.r.t. the ground truth. Thus, at each trial, the target object is placed on a table in front of the robot, within the robot's operational workspace, in a different configuration. In this case, the query point cloud is acquired by scanning the operational workspace from only 3 different views, such so to produce a mismatch between the estimated pose and the ground truth. Once these views are aligned and registered, Boris estimates the belief state with a set of particles. The first run used a mean-shift algorithm combined with the hierarchical clustering algorithm, as presented in Sec. 7.3.1, to estimate the object's mean pose. The second run used only the single mean-shift algorithm as described in Sec. 6.3.2. Figure 7.6 some of the initial estimations from the first run of the experimental trials. For clarity, the black point cloud represents the ground truth computed with the same model-fitting algorithm used

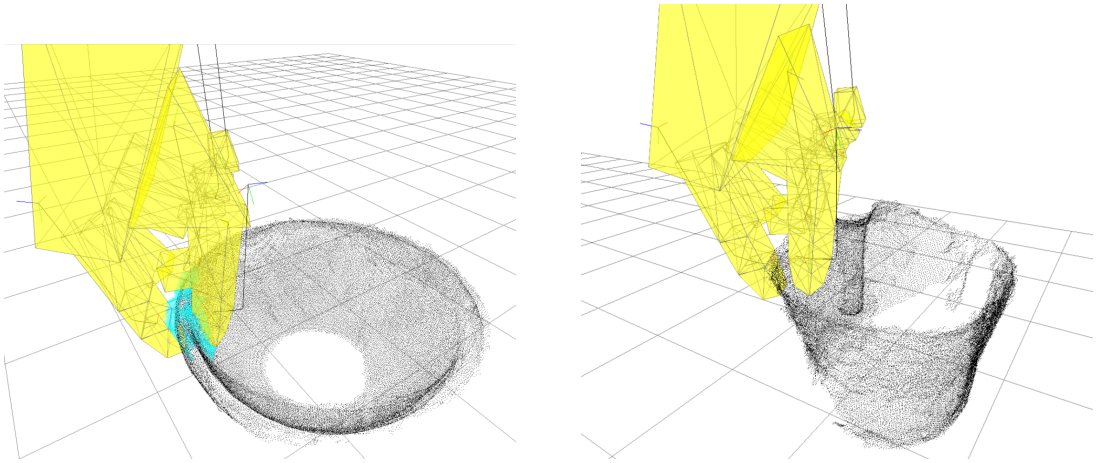


Figure 7.7: Pinch with support grasp learned on a bowl and transferred to a jug, using the method described in [Kopicki et al., 2014]. The image on the left shows the learned contact model (blue points) for all the fingers involved in the grasp. The right image shows a possible grasp adaptation on a jug.

in Sec. 6.3.2, but sampling 500 times more features. The ground truth is also used in the experiments in a virtual scenario as explained in the next section .

In both runs, a target grasp for the jug is computed by adapting a pinch with support grasp learned on a bowl. The training and test grasps are shown in Fig. 7.7. Then a reach-to-grasp trajectory is computed and performed. A trial is considered successful if Boris can converge to the target grasp configuration and lift the object from the table surface.

Figure 7.8 summarises the empirical results collected. In order to test the ability of a sequential re-planning algorithm to converge to the "true" pose of the object, a ground truth pose of the object is calculated at the beginning of each trial by using the same model-fitting algorithm as used in Sec. 6.3.2, but sampling 500 times more features. The algorithms have no knowledge of the ground truth pose.

The results are organised in Fig. 7.8 as follows:

- The number of average planning iterations across all the trials for each run, for

both the re-planning strategies: MYCROFT and IR3ne (Fig. 7.8 top left chart).

- The success rate across all the trials for each run, in the sense of their ability to converge to the planned grasp for all the three strategies (Fig. 7.8 top right chart).
- The averaged reduction in the positional error between the estimated location and the ground truth across all the trials for each run. The error is computed as the Euclidean distance in a 3D space for both strategies: MYCROFT (middle left chart) and IR3ne (Fig. 7.8 middle right chart).
- The averaged reduction in the rotational error between the estimated rotation and the ground truth across all the trials for each run. The error is computed as the distance in the quaternion space for both strategies: MYCROFT (bottom left chart) and IR3ne (Fig. 7.8 bottom right chart). The rotational error is computed for each orientation on the unit hypersphere in 4D quaternion space, and then measure displacement as the length of the arc of the geodesic which connects these two points.

Section 7.5.1 discusses the presented results.

7.4.2 Experiments in a virtual environment

The experiments are composed of 120 trials per object in a virtual environment: a jug, a coke bottle, a stapler and a Mr Muscle spray bottle. Each trial has a different initial probability density over the object pose. We tested the ability of different strategies to achieve a grasp configuration. The algorithm has a model of the object to be grasped, in the form of a dense point cloud, computed by scanning the object with a depth camera from 7 different views. As before, these views are pre-processed to generate a single point cloud of the object model. The pre-processing aligns the single view point clouds, registering and eliminating outlier points. Figure 7.3 shows the pre-processed

point clouds of each simulated object, and the associated target grasp configurations. For these experiments, the models are composed of 7 single-view point clouds, apart from the bottle of coke which is composed of only 5 views.

The algorithms have been tested under the hypothesis that, at each trial, the object is displaced to a different position - but still in the dexterous workspace of the robot - and the robot has to attempt a grasp even if the new point cloud is not as dense as the model point cloud. In simulation, I achieve this by applying a rigid body transformation to the single-view point cloud to move it to a different location within the dexterous workspace of the manipulator. Four different conditions have been tested. In each condition we randomly selected either 1, 3, 5 or 7 view point clouds from the 7 single-views collected, in order to simulate real situations in which the robot's depth camera has been able to observe smaller or larger parts of the object. Once the subset of views has been selected and moved to the simulated object location, the state estimation procedure described in 6.3.2 is performed.

Figures 7.9, 7.10, 7.11, and 7.12 summarise the data collected in our experiments. For each condition mention above, a model coverage is computed, in terms of which percentage of the model surface was covered by the merged point clouds from the individual views. All the results are compared with respect to the model coverage in percentage terms. Again, a "ground truth" pose for the object is calculated at the beginning of each trial by using the same model-fitting algorithm used in Sec. 6.3.2, but sampling 500 times more features. The ground truth is also shown in Fig. 7.6 as a black point cloud. The algorithms have no knowledge of the ground truth pose, however in the virtual environment the ground truth is used to model the real object location, and is used to trigger simulated contacts with the robot hand. These simulated contacts cause the sequential re-planning algorithm to stop and update the belief state.

The results in Figures 7.9, 7.10, 7.11, and 7.12 are organised similarly to the experiments

on the real robot:

- The number of average planning iterations across all the trials for each condition (1, 3, 5, and 7 single-view query), for both the re-planning strategies: MYCROFT and IR3ne (top left chart).
- The success rate across all the trials for each condition, in the sense of their ability to converge to a grasp for all the three strategies (top right chart).
- The average reduction in the positional error between the estimated location and the ground truth across all the trials for each condition. The error is computed as the Euclidean distance in a 3D space for both strategies: MYCROFT (middle left chart) and IR3ne (middle right chart).
- The average reduction in the rotational error between the estimated rotation and the ground truth across all the trials for each condition. The error is computed as the distance in the quaternion space for both strategies: MYCROFT (bottom left chart) and IR3ne (bottom right chart). The rotational error is computed for each orientation on the unit hypersphere in 4D quaternion space, and then measure displacement as the length of the arc of the geodesic which connects these two points.

Section 7.5.2 discusses the presented results and illustrates the behaviour of the sequential re-planning approach (IR3ne) generated by one simulated trial.

7.5 Discussion

In this section we discuss the results presented in the previous section.

7.5.1 Sequential re-planning in a real scenario

Figure 7.8 presents the collected results on the Boris robot platform, as described in Sec. 7.4.1. The results show the benefits of using the hierarchical clustering algorithm to address explicitly the multi-modal nature of the belief space. In particular, the bottom row shows that, in the case of the jug, this approach leads to a lower initial rotational uncertainty with respect to a single mean-shift approach. This is due to the fact that, if the handle of the jug is not visible from the query point cloud, the object looks almost symmetric, which results in a multi-modal rotational uncertainty. Figure 7.13 shows this case from one of the trials from the collected results. Nevertheless, IR3ne is capable of achieving a grasp in a single iteration, thanks to a more robust approaching trajectory with respect to the estimated uncertainty.

Figure 7.14 shows an ideal case where the initial pose estimation does not cover the ground truth, in the sense that the belief state has no hypotheses that can explain the real pose of the object. Hence a first contact with the object (shown in Fig 7.14(b)) is not sufficient to produce an accurate estimation after the belief update. This leads to a second failed attempt, but also to a contact which allows Boris to locate the object and grasp it at the third, and final, attempt. Notice that this example is not part of the experimental results, but it has been recorded during a demonstration.

These two sets of trials have shown the validity of the sequential re-planning approach on a real scenario, however there are several issues that have to be addressed. First, the need for a predictive model for predicting how the target object moves after a contact occurs. Although the contact detection module described in Sec. 7.3.4 reliably stops the

robot after a contact is made, light objects, such as the plastic jug used in these trials, may be perturbed by the contact. For example, often we observe that the jug is tilted by a finger. The belief update by itself cannot deal with these cases and the resulting mean estimations are usually incorrect. However, the choice of point cloud models negatively affects the ability of developing predictive models. Second, the lack of tactile sensors does not allow us to have a good estimate of which link of the finger experienced the contact. This results in an additional uncertainty in the belief update.

Future work aims to address such issues by extending the sequential re-planning approaches to the use of tactile sensors and simple heuristics for an object-independent predictive model. The latter should be based on simple geometric properties of non-penetration between objects and robot's fingers to constrain the belief update and the mean pose estimate.

7.5.2 Sequential re-planning in a virtual scenario

The results collected in the virtual scenario confirms the ability of the sequential re-planning approach to achieve a grasp more robustly than a single grasp attempt (ELEMENTARY strategy) for all the objects (Fig 7.3) and all the conditions (see Sec 7.4.2). In addition, Sec 7.4.2 has also shown that IR3ne is capable of achieving successful grasps with fewer iterations than MYCROFT.

To illustrate the behaviour of the re-planning system Fig. 7.15 shows a typical sequence generated by one simulation trial attempting to perform a rim grasp on the jug. In this case the initial belief state over the object pose is quite narrow (Fig. 7.15(b)), however an error of a few millimetres in the pose estimate leads to a potentially dangerous contact with the object (Fig. 7.15(c)). The contact is used to update the belief and plan a new trajectory from the current configuration of the robot(Fig. 7.15(d)). The second trajectory successfully transfers the hand to the target grasp configuration

(Fig. 7.15(e)).

7.6 Conclusion

This chapter has shown the validity, via empirical results, of the sequential re-planning algorithms on a real scenario. In addition, this approach has been evaluated in a simulated environment on four different objects. Several innovations have been developed to extend the theoretical techniques of Chap. 6, to overcome difficult practical problems encountered when trying to implement simultaneous perception and grasping with a real robot manipulator with 21 DoF and non-Gaussian object pose uncertainty in 6D, as well as incomplete knowledge of the object shape.

The main contribution of this chapter is to describe, demonstrate and evaluate a novel approach for the problem of robot grasping, where a robot is capable of reasoning about object-pose uncertainty while planning a dexterous reach-to-grasp trajectory. This system shown in Fig 7.1 is able to plan grasp operations autonomously in unstructured environments, with no knowledge of the object we wish to manipulate, apart from a model point cloud.

As in Chap 6, this chapter has shown how sequential re-planning can achieve better quality grasps than single attempts to directly grasp an object at its expected pose, and that re-planning with trajectories designed to maximise tactile information gain, achieves successful grasps with fewer iterations than sequential attempts to move directly towards the object's expected pose.

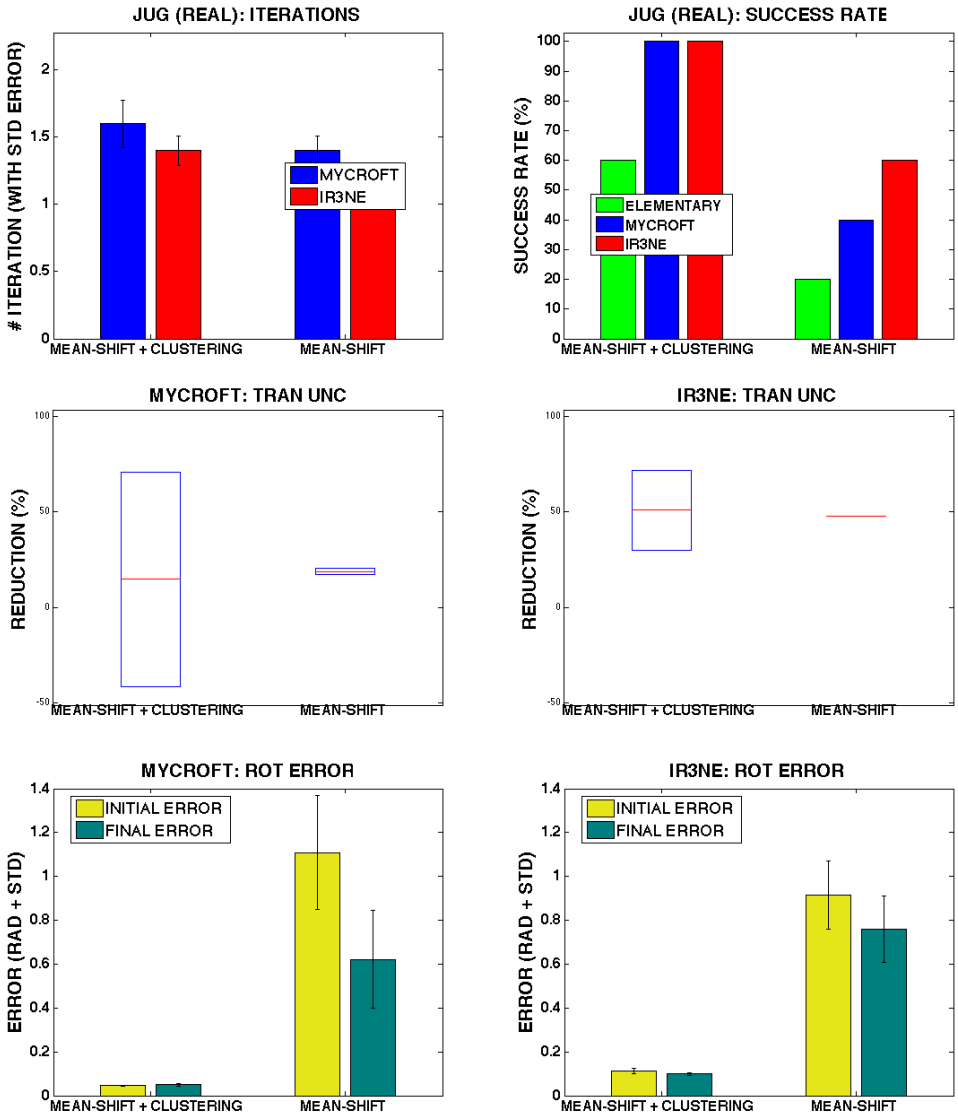


Figure 7.8: Empirical results on the Boris robot platform. The results refer to 2 sets of 10 trials on a jug. Three strategies were used: ELEMENTARY (green), MYCROFT (blue), IR3ne (red). Two conditions are presented. The first uses a mean-shift algorithm combined with a hierarchical clustering algorithm to estimate the object's pose. The second uses only a mean-shift algorithm. The top left chart presents the averaged number of iterations across the trials to reach a grasp. The top right chart shows the success rates across the trials. The middle row shows how the initial linear uncertainty (in percentile) is reduced by sequential re-planning until the algorithms converge to a grasp. The bottom row presents the reduction in the rotational uncertainty. Yellow bars represent the initial rotational error, while green bars are the final error. The rotational error is computed in quaternions where similar rotations yield to error value close to zero and rotations with 180 degrees difference yield an error value of 1.

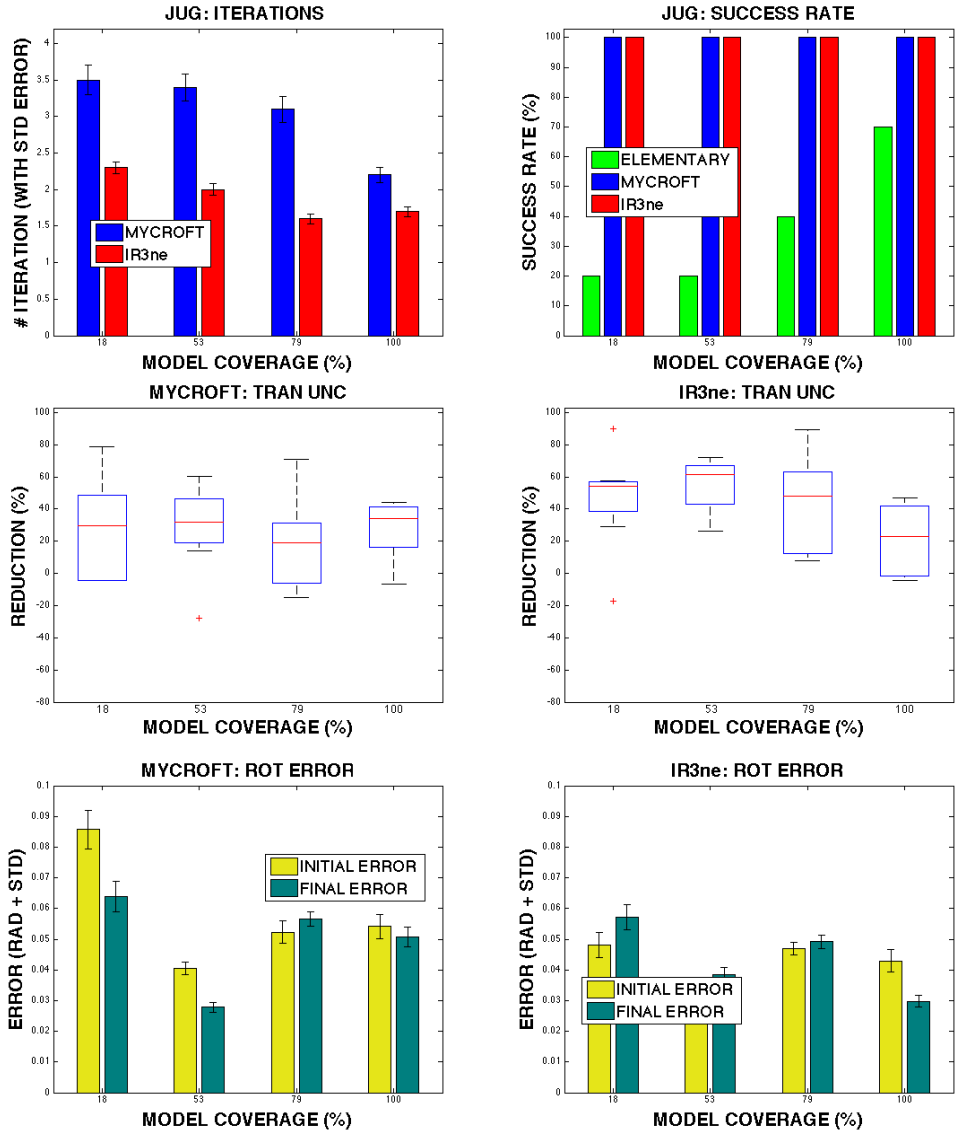


Figure 7.9: Simulated results for 120 trials on a jug. Three strategies were tested: ELEMENTARY (green), MYCROFT (blue), IR3ne (red). All the results are plotted against the initial percentage of model coverage for a total of four conditions. The top left chart presents the averaged number of iterations across the trials to reach a grasp. The top right chart shows the success rates across the trials. The middle row shows how the initial linear uncertainty (in percentile) is reduced by sequential re-planning until the algorithms converge to a grasp. The bottom row presents the reduction in the rotational uncertainty. Yellow bars represent the initial rotational error, while green bars are the final error. The rotational error is computed in quaternions.

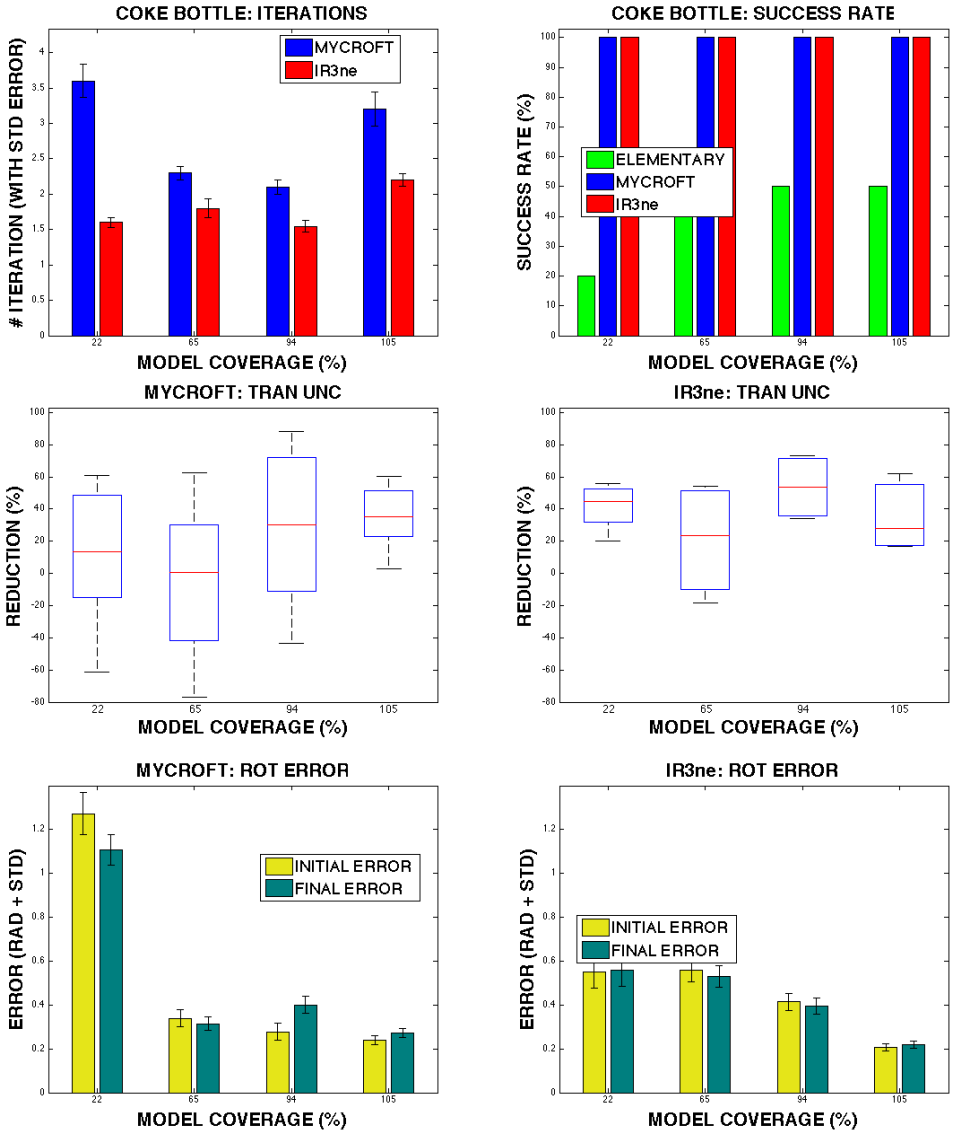


Figure 7.10: Simulated results for 120 trials on a coke bottle. Three strategies were tested: ELEMENTARY (green), MYCROFT (blue), IR3ne (red). All the results are plotted against the initial percentage of model coverage for a total of four conditions. The top left chart presents the averaged number of iterations across the trials to reach a grasp. The top right chart shows the success rates across the trials. The middle row shows the linear uncertainty reduction (in percentile), while the bottom row presents the reduction in the rotational uncertainty. Yellow bars represent the initial rotational error, while green bars are the final error. The rotational error is computed in quaternions.

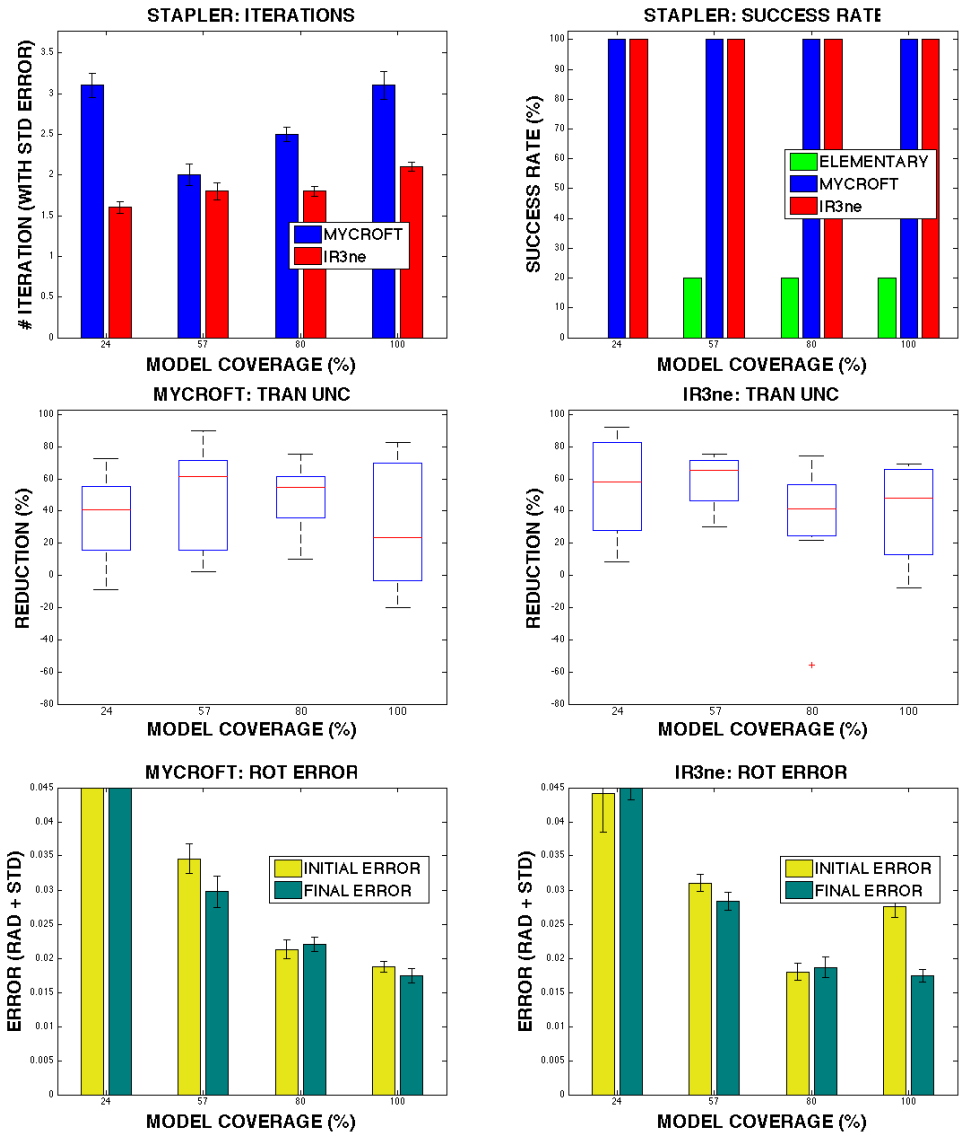


Figure 7.11: Simulated results for 120 trials on a stapler. Three strategies were used: ELEMENTARY (green), MYCROFT (blue), IR3ne (red). All the results are plotted against the initial percentage of model coverage for a total of four conditions. The top left chart presents the averaged number of iterations across the trials to reach a grasp. The top right chart shows the success rates across the trials. The middle row shows the linear uncertainty reduction (in percentile), while the bottom row presents the reduction in the rotational uncertainty. Yellow bars represent the initial rotational error, while green bars are the final error. The rotational error is computed in quaternions.

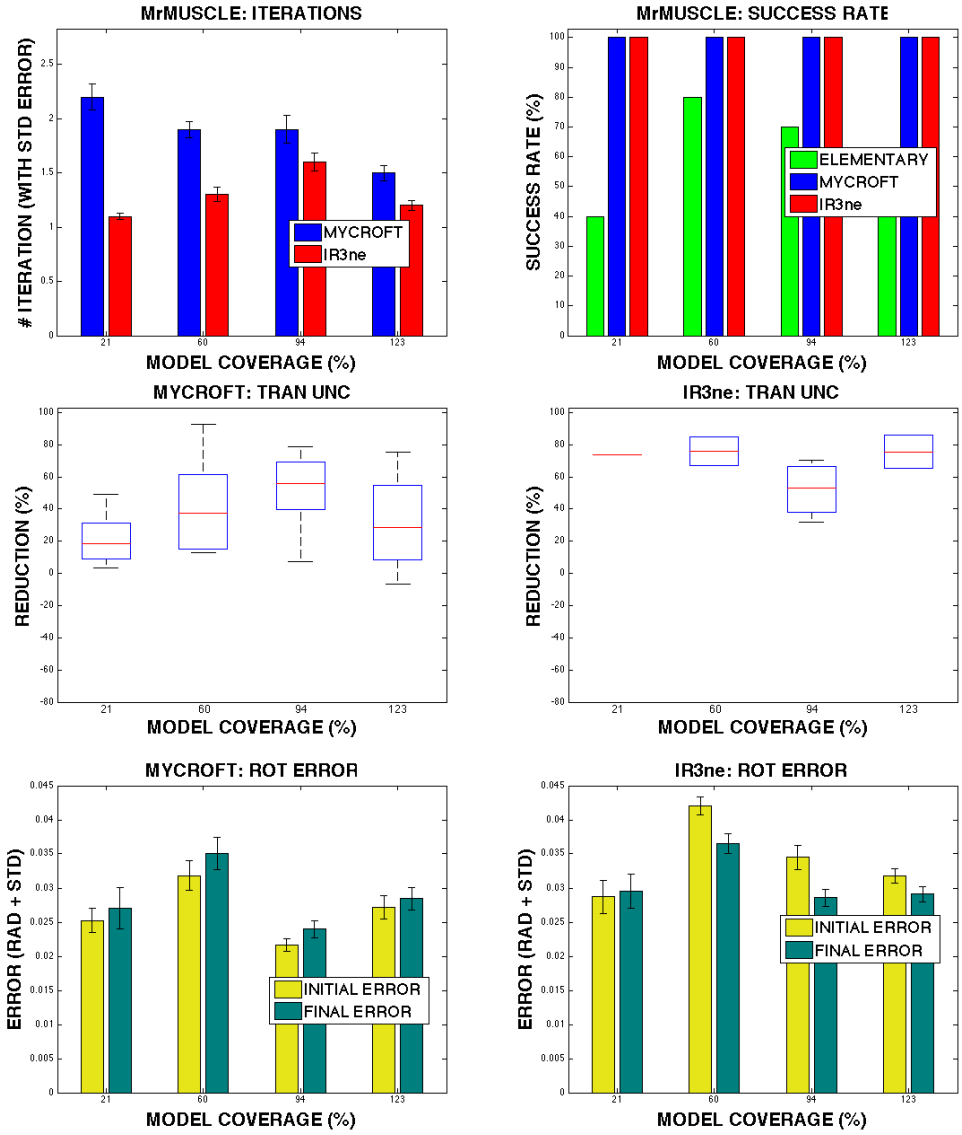


Figure 7.12: Simulated results for 120 trials on a mr muscle spray. Three strategies were used: ELEMENTARY (green), MYCROFT (blue), IR3ne (red). All the results are plotted against the initial percentage of model coverage for a total of four conditions. The top left chart presents the averaged number of iterations across the trials to reach a grasp. The top right chart shows the success rates across the trials. The middle row shows the linear uncertainty reduction (in percentile), while the bottom row presents the reduction in the rotational uncertainty. Yellow bars represent the initial rotational error, while green bars are the final error. The rotational error is computed in quaternions.

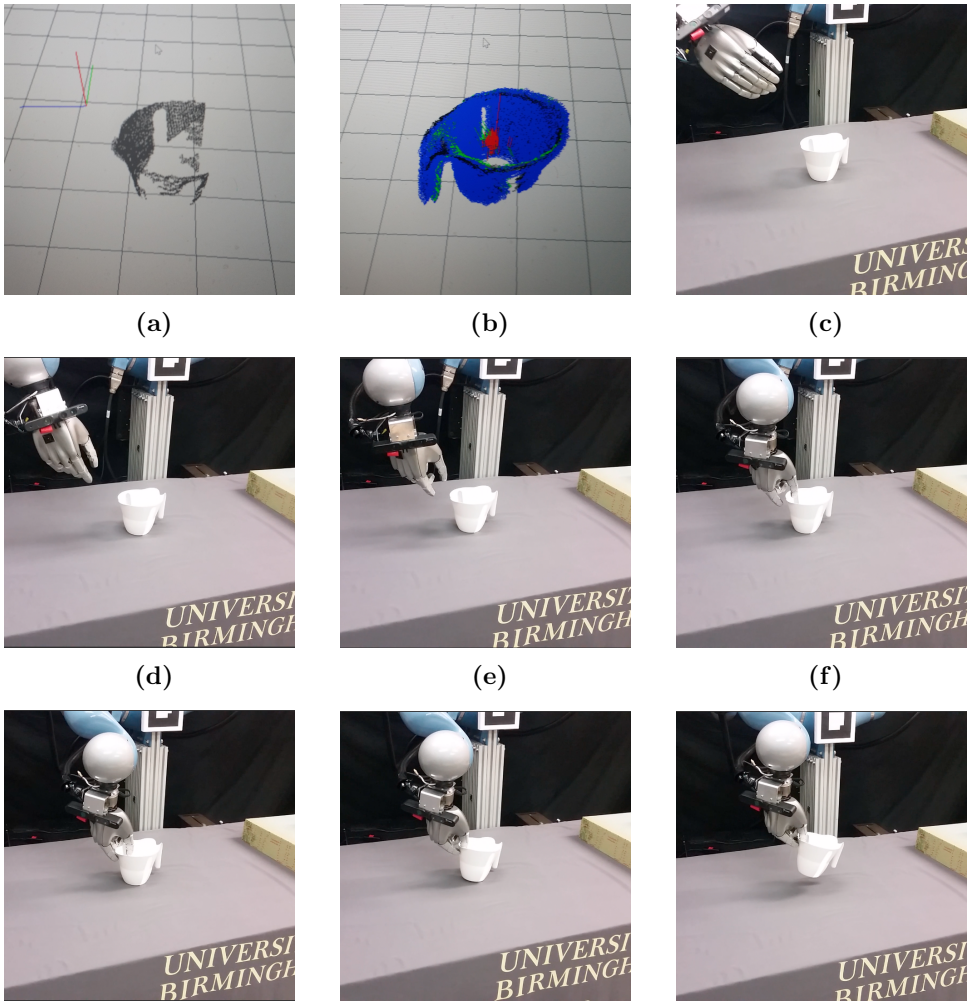


Figure 7.13: Example of plan execution for IR3ne. The top row shows: (a) the partial query point cloud (3 merged views), (b) the belief state (as sub-sampled hypotheses (blue), mean pose (green) and ground truth (black)), (c) the real pose of the object. This example shows the worst case in which the the query point cloud covers only a 22.5% of the model and the handle is not visible. Note that the ground truth (b) (black point cloud) is also estimated with the wrong orientation. Nevertheless, IR3ne executes the planned trajectory (middle row) and achieve e grasp (g) and (h). In (i), Boris lifts successfully the jug.

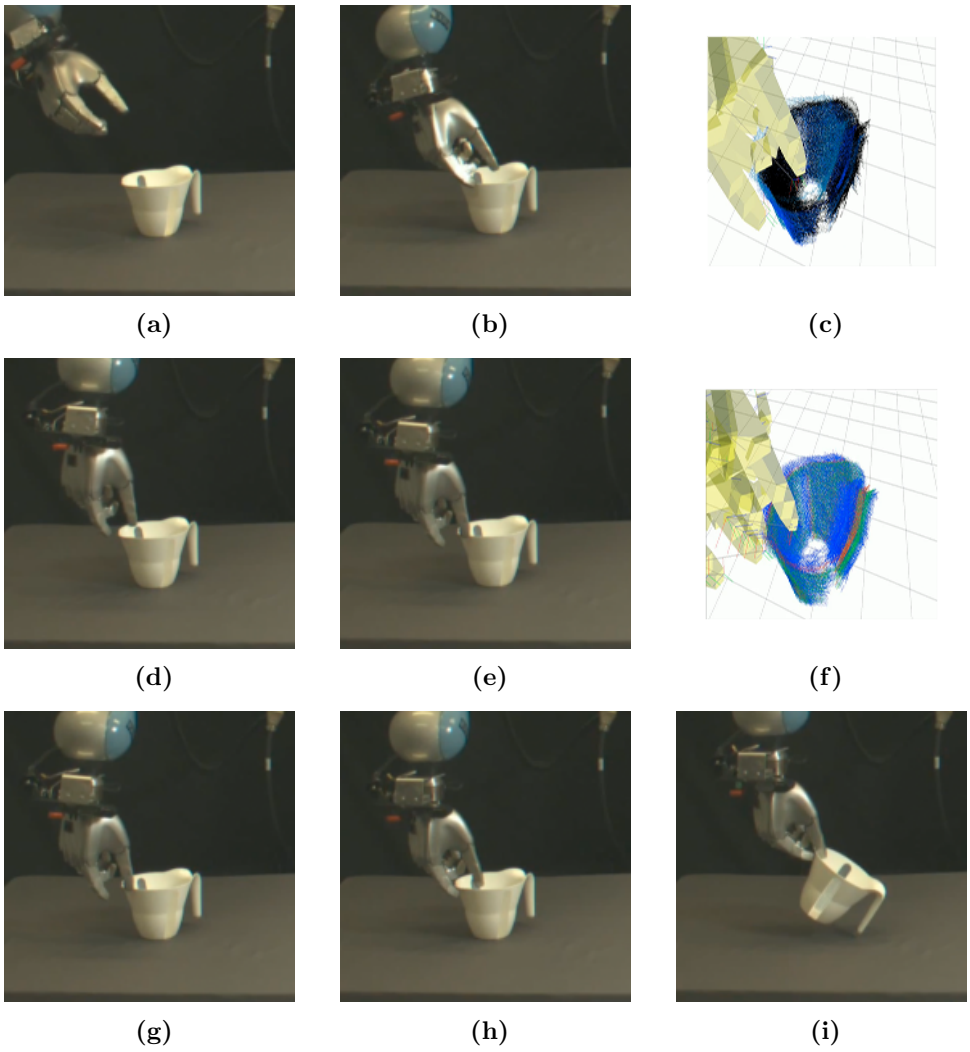


Figure 7.14: The sequence of images presents an interesting case in which the initial belief state does not cover the ground truth, however IR3ne is capable to converge to a grasp in 3 attempts. Each row shows an attempt of real robot and either the belief update after each attempt or the final grasp. The simulated images show the belief update as PF. The colour of each hypothesis is associated with its likelihood, from red (high likelihood) to black (zero likelihood).Top row: due to a erroneous localisation the first grasp attempt collides with the plastic jun on the rim (b). The contact is used to refine the belief state but, since none of the hypotheses match the contact, the hypotheses have all low probability associated (c). Middle row: a second attempt fails but this time the contact allows Boris to localise the object and, finally, to grasp it (third row).

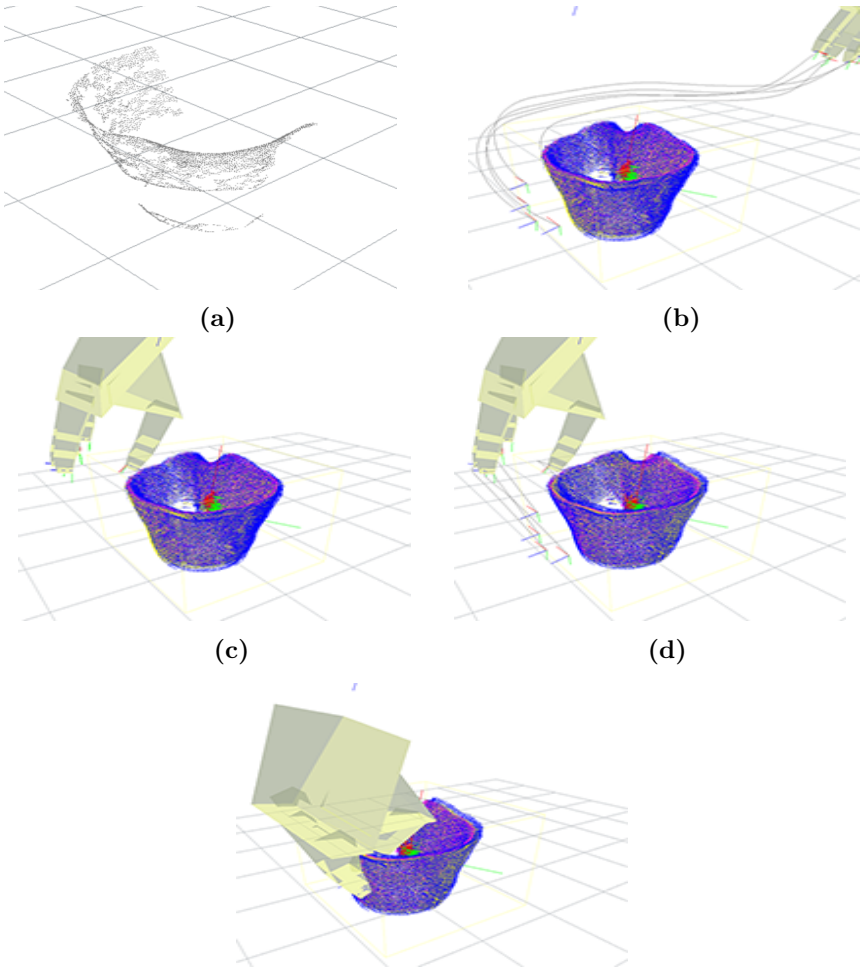


Figure 7.15: Example of plan execution for IR3ne. Top row: a partial point cloud is acquired (a) and a density distribution represented by a set of particles is computed and a reach-to-grasp trajectory is planned (b). The red point cloud identifies the ground truth which exactly matches the partial point cloud in (a). The blue point clouds represent the low dimensional belief states sub-sampled from the corresponding high dimensional belief states. The yellow point cloud represents the estimated pose for the object. Second row: despite the tiny misplacement between the estimate pose and the ground truth, a contact occurs before the grasp configuration with respect to the mean pose (yellow point cloud) could be reached (c). The belief state is therefore updated from the contact (d) and a new trajectory is planned with respect to the new estimate. Bottom row: the hand reaches the target grasp pose (d).