Real-time CAD Model Matching for Mobile Manipulation and Grasping

Ulrich Klank, Dejan Pangercic, Radu Bogdan Rusu, Michael Beetz
Intelligent Autonomous Systems, Technische Universität München,
Boltzmannstr. 3, Garching bei München, 85748, Germany

Abstract—Humanoid robotic assistants need capable and comprehensive perception systems that enable them to perform complex manipulation and grasping tasks. This requires the identification and recognition of supporting planes and objects in the world, together with their precise 6D poses. In this paper, we propose a 3D perception system architecture that can robustly fit CAD models in cluttered table setting scenes for the purpose of grasping with a mobile manipulator. Our approach uses a powerful combination of two different camera technologies, Time-Of-Flight (TOF) and RGB, to robustly segment the scene and extract object clusters. Using an a-priori database of object models we then perform a CAD matching in 2D camera images. We validate the proposed system in a number of experiments, and compare the system's performance and reliability with similar initiatives.

I. INTRODUCTION

As personal mobile manipulation platforms come to age, they require perception systems that can recognize and track objects with real-time performance. These perception systems should also include inference mechanisms for determining the most probable grasping points such that the objects of interest can be manipulated using stable grasps. However, the current state of the art in 3D perception for manipulation architectures is represented by implementations which require substantial computational efforts to recognize and reconstruct object models from sensed data, and generate good strategies for grasping them.

In this paper, we propose a 3D system architecture for perception that addresses the problem of precise object modeling and tracking for grasping applications. Our approach makes use of databases of a-priori acquired object models, and identifies objects of interest and their 6D pose in the sensed data. The hardware setup consists of a combination of stereo and Time-Of-Flight (TOF) cameras, as shown in Figure 1. The CAD object models are fit in 2D images with real-time performance, using constraints automatically determined from the 3D point cloud data (PCD) obtained using the TOF camera.

Our 6D pose object recognition algorithms can handle arbitrarily complex shapes, without any assumptions about occlusions, distance to object, or object rotation with respect to the camera. The object models are correctly identified and matched in real-world scenes, independent of the type of surface the objects are made of, and most importantly, without making use of any texture information. The latter is of extreme importance for objects present in household environments which are most often textureless. This constitutes an important advantage over other similar recently proposed research initiatives [1], [2], which fail to work in the absence of good texture-based features.

The system presented in this paper is based on our recently proposed work on matching approximated CAD models [7], [19], using extensions of the work presented in [16], [18]. Here we continue our work by incorporating 3D range data into the existing camera based algorithms to improve the results both in terms of speed and robustness.

Our application scenario can be described as follows. Given a surface representation of an object, either acquired in an a priori step, or extracted from a database of shape models, identify the object and its position in real-time in a scene, and provide a suitable grasping pose for a mobile manipulator. The grasping points for a model are estimated offline using OpenRAVE [4]. An example object models identified in a cluttered table scene can be seen in Figure 2.

The object models are used to identify the most probable edge occurrences in 2D images. While this can be performed directly in the entire image space, the search can lead to increased computational demands which might not be suitable for application with real time constraints. Another

Fig. 1. The mobile manipulation platform used for the purpose of the experiments presented herein. The hardware setup consists of a B21 mobile base with two 6-DOF PowerCube arms, stereo cameras and a TOF camera mounted near the end effector. The right part of the figure presents interpretation results given by the perception system, from top to bottom: i) a textureless occluded white mug identified and localized; ii) 3D segmentation of object clusters on the table; iii) four distinct objects localized in a cluttered table scene.
disadvantage of the brute-force search is that extreme clutter could potentially interfere with the solutions found by the matching algorithm, and result in matches with a poor geometric correspondence (i.e., objects flying in thin air). Our proposed extensions include the inference of supporting planes (e.g., tables) from 3D depth data provided by a TOF camera, in order to generate consistent places where objects could be located, and region growing and clustering mechanisms for the segmentation of individual objects into potential Regions of Interest in the image space. The novelty of our approach is represented by the realization of a real-time integrated object identification system based on CAD model matching for the purpose of grasp identification in mobile manipulation applications.

The software components necessary to perform the tasks discussed in this work are modular so they can be reused individually. Because of their modularity, they can be used for robots other than the one presented in this paper. In particular, we make use of the ROS open source initiative\(^1\) to build our distributed system.

II. RELATED WORK

The problem that we are trying to solve falls in the context of perception for manipulation. We therefore split the related work discussions into three distinct parts that correspond to the respective subjects covered in this paper, namely: i) similar mobile manipulation platforms with detection-localization-grasping capabilities, ii) object classification and localization in indoor environments based on depth data and iii) 2D camera images respectively.

The STAIR platform [10] uses a peripheral visual object detection system working with a steerable pan-tilt-zoom camera to obtain high-resolution images. The authors demonstrate grasping in a comprehensive “fetch a stapler” application, based on a large training set of labeled natural and synthetic household objects [13].

\(^1\)ROS (Robot Operating System - http://ros.sourceforge.net) is an Open Source project to build a meta-operating system for mobile manipulation systems.

The humanoids robot ARMAR-III is presented in [17], together with a perception system that can localize small and large-sized household objects. The system makes use of a recognition step using 2D-2D feature point correspondences from Hough transforms and an iterative estimation of an affine transformation, while the 6D pose estimation method makes use of a stereo triangulation algorithm.

A perception system for the PR2 (Personal Robot 2) manipulation platform is presented in [11]. The authors make use of a 2D laser tilted continuously on a pan-tilt unit, which provides a 3D point cloud map representing the surrounding world in front of the robot. The perception system is capable of constructing dynamic collision maps used by a motion planner to avoid obstacles while moving the robot arms, and can recognize tables and object clusters supported by them.

A humanoid two-arm system built for studying dexterous manipulation is presented in [9]. Its real-time perception system is based on a 3D-Modeller head that comprises a laser-range scanner, a laser-stripe profiler, and a stereo camera sensor. Object pose hypotheses are obtained by applying the invariant Haar measure of the Euclidean group \(SE(3)\) on the data provided by the composite sensor, and then fed to the manipulation system. The work of [12] discusses the extraction of planes in range data. A system for object recognition and segmentation in range data is presented in [5].

The last step of our problem is to match many 3D objects in 2D images. This problem was solved before and there are many different methods, such as [6] for example, which is based on generating many views of a 3D object. We are using a state of the art 3D shape model matching technique, which is described in [16], [18]. This method matches 3D-CAD-models in an image by simulating the 2D appearance in a shape model generation phase. A range of position relative to the camera must be specified, which can only be done if information about the scene is available. A similar approach which makes use of line and circle segments was presented in [8]. Another approach that can recognize shapes via geometric features was presented in [15].

III. SYSTEM ARCHITECTURE

The overall architecture of the proposed system is depicted in Figure 3. Our mobile manipulation platform (Figure 1) has a SwissRanger 4000 TOF camera mounted on one of its end-effectors (6-DOF arm) and two RGB color cameras attached to a Pan-Tilt-Unit (PTU) on top of the robot body. The platform is controlled via several subsystems including Player [3], and ROS.

The task executive is in charge of controlling the overall system operation and creates/starts separate threads based on the current state of the system. In particular, the executive controls:

- the way the detection and localization of objects is performed;
- the arm motion planning and grasping.

The modules responsible for the detection and localization of object models in camera images are outlined in the
In the previous version of our system we used priors like the size and orientation of supporting planes in a room. This required a precise localization of the robot, especially with respect to the orientation of the cameras towards the supporting plane. Even if this plane is often given by the hardware setup, it might be disturbed by obstacles and in general, an inaccurate localization.

In the work presented herein, we make use of a TOF camera (see Figure 1) to estimate the best supporting planes in real-time, and thus improve the segmentation of object candidates (see section IV-A). Having such object candidates and their respective RoI (Regions of Interest) in the camera image, they can be transformed into appropriate search spaces (section IV-B) for CAD matching (section IV-B.2). An example of how such search spaces look like can be seen in Figure 4.

The transformation between the 3D estimates obtained from the point cloud data in the TOF camera frame, to the RGB camera coordinate system makes sense only after a careful calibration of the two different cameras. The calibration procedure uses 2D-2D correspondences between the TOF camera (four channels, x,y,z coordinates and intensity) and the RGB camera using a standard calibration plate as a reference object. By interpolation (upsampling) of the neighborhood pixels in the TOF camera image, we obtain a good approximation of the depth image (and thus the measured 3D points), which leads to an accurate localization of the corresponding pixels in the RGB camera image. A standard pose estimation [14] is then used to obtain the pose of the TOF camera ($P_T$) in RGB camera ($P_C$) coordinates while minimizing the re-projection error of those 3D points in the calibrated RGB camera, as follows:

$$P_C = C H_S \cdot P_S$$

Fig. 3. The architecture of our system using the ROS paradigms. The planar estimation and cluster segmentation steps are performed by the 3D PCD processing node (right) whose output is fed into an image processing node (middle) which performs the detection and localization of objects. Our motion planning and grasping is spawned as a separate system thread (left).
consider points with their normals \( \mathbf{n}_i \) approximately parallel with the Z axis;
- a clustering in normal space is performed in \( \mathcal{P}^d \), to simplify and reduce the overall errors that might affect the planar segmentation step;
- in each cluster, the best planar support candidate is estimated using a sample-consensus approach. Each plan is weighted with respect to the camera viewpoint, and the model with the largest weight is selected as a candidate. The weight is a simple measure of the number of inliers in the dataset, together with the estimated 2D area of the model;
- once the table equation is estimated, a search for all the point clusters supported by it is performed, by looking whether the projection of the points \( p_i \in \mathcal{P} \) falls inside the bounds of the 2D table polygon.

Algorithm 1 outlines those steps in pseudo-code while a visualization is provided by Figure 5.

**Algorithm 1** Find tables and objects

\[
\mathcal{P} = \{p_1, \ldots, p_k\}  \quad \text{# set of 3D points}
\]

for all \( p_i \in \mathcal{P} \) do

- estimate \((n_i, \text{from } \mathcal{P}^k)\) // estimate surface normal from nearest neighbors
  - if \((\alpha = (n_i, Z) \approx 0)\) // check if the normal is parallel to the Z axis
    - then \( P_2 \rightarrow p_i \) // add \( p_i \) to the \( P_2 \) set

- estimate \((C = \{p_1, \ldots, p_k\}, \mathcal{P}_2 \subseteq \mathcal{P}_1\) // break \( \mathcal{P}_2 \) into clusters

for all \( c_i \in \mathcal{C} \) do

- // find the best plane fit using sample consensus

- estimate \((a, b, c, d), a \cdot p_i^t = b \cdot p_i^t + c \cdot p_i^t + d = 0, p_i \in c_i\)

- estimate \((\alpha_{\text{min}}, \alpha_{\text{max}})\) // find the min/max bounds of the planar area

- \( T M \rightarrow \mathcal{P}(c_i) \) // add the table parameters to the table map

for all \( p_i \in \mathcal{P} \) do

- if \((\alpha_{\text{min}} \leq \alpha_i \leq \alpha_{\text{max}}, a_{\text{min}} \leq a \leq a_{\text{max}})\) // within bounds?

  - then \( p_i \rightarrow P_3 \) // add \( p_i \) to the \( P_3 \) set of cluster candidates

- estimate \((O = \{p_1, \ldots, p_k\}, \mathcal{P}_2 \subseteq \mathcal{P}_1)\) // break \( \mathcal{P}_2 \) into clusters

**B. Search Space Calculation**

Before propagating the point cloud data to the visual system we want to represent the search space in a compact form. Therefore, we describe the clusters \( \mathcal{O} \) using their center points and their maximal extensions along the \( xyz \) coordinate axes. The extents are then converted to a \( 6 \times 6 \) diagonal matrix, representing the estimated uncertainty of the cluster poses.

The matrix representation can be efficiently propagated through the different coordinate systems, to the final extents in the camera based spherical and image coordinates which we use to set up the search space for the CAD matching step. In our experiments we did not encounter any drawbacks from this compression scheme, since we have to overestimate the visual search space anyway because of the relatively large baseline between the RGB and the TOF cameras.

1) **3D Points to Search Space:** Given that we have a point with its orientation in a 3D space, we interpret this point as an object to camera relation. To include such a point in the RGB camera search space we have to add its projection into a Region of Interest (RoI) image and transform it into the spherical coordinates. The latter enables modelling of the RGB camera image on a sphere with a radius \( r \) at a pose described by 3 angles: the longitude \( \alpha \), latitude \( \beta \) and camera roll \( \gamma \). Given an already defined search space with \( \langle \alpha_{\text{min}}, \alpha_{\text{max}}, \beta_{\text{min}}, \beta_{\text{max}}, \gamma_{\text{min}}, \gamma_{\text{max}}, r_{\text{min}} \text{ and } r_{\text{max}} \rangle \), we have to adapt it as soon as the point falls outside the RoI. For each point found in the adjacency of an already existing RoI, we extend the respective region to include the point by modifying its outer dimensions. Figure 6 shows an example of the 3D clusters estimated using the methods presented in Section IV-A, here back-projected into the RGB camera image.

![Fig. 6. Backprojection examples of the 3D clusters estimated using the methods in Section IV-A into the RGB camera image.](image)

2) **CAD Matching:** The CAD models of the objects that we want to localize are usually acquired as an a-priori step using precise laser sensors followed by a number of geometric processing steps (sometimes involving manual labor to clean the models), or can be alternatively downloaded from the Internet [7]. Most household objects available in the internet databases can be found lying on supporting xy-planes in the 3D model coordinates. This is of course an assumption we do not rely completely upon, but we assume that the majority of the models are aligned in such a way. Since we additionally also align all selected inlier models to each other, we obtain the major “upright” direction. Following this we can assume that giving a supporting plane we only have to account for one of the three rotational degrees of freedom.

Shape matching approaches require significant preprocessing of the given 3D model, whose complexity is polynomially increasing with the number of faces in the model and linearly with the number of views that have to be generated to get a complete search. The number of views in our application depends mainly on the given search space in the spherical coordinates system. Thus the constrained regions of interest reduce the search phase significantly.

To obtain fast results, we build a tree containing the projection of the model in different resolutions (image pyramids), with all leaves similar to their parents but at a higher resolution. The similarity of a match is measured in the 2D appearance of the projection. The information resulted from the segmentation step in the point cloud data gives a significant reduction in \( r \) (the distance to the camera) while \( \alpha, \beta \) are only slightly affected by our search space restrictions. This is due to the search for the same classes of possible orientation on the table. If it was possible to reduce the RoI and the search range of \( \gamma \) this would only affect the calculation time in the search phase. A very rough approximation of the estimated model calculation time is given as follows:

\[
\text{step} \sim \left( \max(x_i^t, y_i^t) \right) / r_{\text{min}} 
\]

\[
I_{\text{model}} \sim N \gamma_{\text{step}} (\alpha_{\text{max}} - \alpha_{\text{min}})(\beta_{\text{max}} - \beta_{\text{min}})(r_{\text{max}} - r_{\text{min}}) 
\]
where $x_i^r$ and $y_i^r$ represent the resolution of the image and $N_f^v$ is the number of visible faces. Similarly, we can approximate the expected calculation time for the search phase by:

$$t_{\text{search}} \sim \text{step} \times (\gamma_{\text{max}} - \gamma_{\text{min}}) \times N_{\text{cand}}^1$$

(4)

where $N_{\text{cand}}^1$ is the number of candidate matches on the highest pyramid level, which corresponds directly to the size of the regions of interest used.

V. DISCUSSIONS AND EXPERIMENTAL RESULTS

For the sake of completeness, we briefly review some of the previous results we obtained in [7] in order to introduce the new approach presented in this paper. In our previous work we showed how to obtain CAD models from the Google 3D Warehouse and successfully match them against simple objects in table setting scenes. Using a set of preprocessed grasping points, our mobile manipulation platform was capable of picking up the object using a stable grasp approach, as seen in Figure 12. However, the major problems that we faced concern the missing robustness against clutter and the high calculation time of the model generation phase. The following sections demonstrate significant improvements in terms of computation speed and detection robustness in cluttered scenes, using in this work proposed system architecture.

A. COMPUTATIONAL EFFICIENCY

To demonstrate the computational advantages obtained by incorporating the 3D clustering information into the image search space, we performed several tests for different objects located in various cluttered scenes. In each experiment, we recorded the calculation time necessary for the model generation phase and the one for the search phase. Since the four objects present a large variety in the number of faces that their CAD models contain, the overall calculation times can vary significantly.

The experiments include a comparison between the results obtained using the entire search space of the full table versus the one automatically generated by the point cloud data processing node. Figure 8 shows the resultant calculation times for the model generation phase. As presented, the highest relative calculation time gain is obtained when a number of faces in the object is high as well. Additionally, the relative size of the cluster with respect to the initial search space influences the calculation time. The smaller the
segmented clusters are, the higher is the gain. This can be seen by comparing the calculation times for large and small clusters. An example of a large cluster is an ice tea box, while a small cluster can be caused by an appearance of the mug, as seen for example in Figure 10.

The time needed for the clustering added up to $\approx 50\text{ms}$ per frame and is omitted in Figures 8 and 9. The model generation can be skipped for new search spaces if there was another model generated before, that contains the current search space completely. Figure 9 shows the calculation times for the search phase. Again, the graph compares the calculation time with and without the segmentation. This experiment included the assumption of the object standing upright on the table having an arbitrary orientation. All measurements were done on an AMD Athlon(tm) 64 X2 Dual Core Processor 3800+, which constitutes one of the two stock computers of our mobile manipulation platform.

**B. Robustness through Search Space Segmentation**

To show that this approach increases the robustness of the system we compared the search of several objects in more complex scenes.

Table I shows the final results we obtained. For the simple scenes with one or two objects we got comparable robustness for both approaches, but when looking at the densely clustered scene the added value of the segmentation in 3D is significant and boosts up the robustness of the matching. The most interesting fact is the improvement we achieved on cluttered scenes. The ratio of well detected objects to all present objects was increased by more than 20%.

<table>
<thead>
<tr>
<th></th>
<th>Ice Tea</th>
<th>Ice Tea</th>
<th>Mug</th>
<th>Plate</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Object</td>
<td>0.5</td>
<td>1.0</td>
<td>1.0</td>
<td>0.75</td>
<td>0.88</td>
</tr>
<tr>
<td>–with segmentation</td>
<td>0.75</td>
<td>1.0</td>
<td>1.0</td>
<td>0.75</td>
<td>0.88</td>
</tr>
<tr>
<td>Partial Occlusions</td>
<td>1.0</td>
<td>0.75</td>
<td>1.0</td>
<td>1.0</td>
<td>0.94</td>
</tr>
<tr>
<td>–with segmentation</td>
<td>1.0</td>
<td>0.75</td>
<td>1.0</td>
<td>1.0</td>
<td>0.94</td>
</tr>
<tr>
<td>Cluttered Scene</td>
<td>0.2</td>
<td>0.75</td>
<td>0.6</td>
<td>0.2</td>
<td>0.44</td>
</tr>
<tr>
<td>–with segmentation</td>
<td>0.6</td>
<td>0.6</td>
<td>0.75</td>
<td>0.75</td>
<td>0.68</td>
</tr>
<tr>
<td>Overall</td>
<td>0.54</td>
<td>0.66</td>
<td>0.76</td>
<td>0.75</td>
<td>0.72</td>
</tr>
<tr>
<td>–with segmentation</td>
<td>0.76</td>
<td>0.84</td>
<td>0.84</td>
<td>0.83</td>
<td>0.82</td>
</tr>
</tbody>
</table>

**VI. CONCLUSION**

In this paper we presented a comprehensive perception system that integrates 3D depth data from a TOF camera with 2D imagery for the purpose of object model identification and localization in cluttered scenes. Our approach makes
no assumptions about the possible locations of objects or their appearance, which makes it extremely suitable for the identification of stable grasps for textureless objects.

The model search parameters are automatically identified by performing a 3D segmentation in point cloud data to identify the planes supporting objects, and by extracting the object clusters lying on them to provide regions of interest in the camera image. We presented results showing an improved recognition rate and calculation speed compared to our previous, single camera based approach.

Acknowledgments: This work was supported the CoTeSys (Cognition for Technical Systems) cluster of excellence at TUM and by MVTec Software GmbH.

REFERENCES

Fig. 12. Snapshots taken during the grasping experiment of a mug using our mobile manipulation platform.