Active Perception for Object Manipulation

Ady-Daniel Mezei
Technical Univeristy of Cluj-Napoca
Cluj-Napoca, Romania
Email: adydani_mezei@yahoo.com

Levente Tamas
Technical Univeristy of Cluj-Napoca
Cluj-Napoca, Romania
Email: Levente.Tamas@aut.utcluj.ro

***Abstract*—The active perception makes an important step towards the integration of the sensing in the planning phase of the object manipulation. It ensures the existence of uncertainty in the perception phase, while with active planning helps to reduce this in order to achieve a better results for scene parsing and object manipulation applications. In this paper we show the active perception concept using an industrial like scene with different pipes which are sensed with a depth camera. The pose information from the camera is used for a 7 degree of freedom robotic arm mounted on a mobile base in order to perform the planning for object manipulation purposes. For the whole active perception pipeline we have implemented and successfully tested the scene parsing using object parts decoupling and the planning with the robot arm.**

I. Introduction

Object pick and place applications in an industrial environment are still an important research topic especially in case of the applications without preset positions. The idea of using a 3D camera with a stationary robotic arm for object selecting is reported in several works including [1]. However the idea of using a mobile manipulator for this task is not a solved problem in general [2], even though the number of potential applications in the industrial area is rather high. Mobile object pick and place became feasible by integrating advanced concepts from robotics like object detection, planning and navigation. Within this context, in this paper we are focusing on the first part of this pipeline including the scene parsing and object handling in an active perception framework, i.e. planning in order to enrich the information for the scene parsing.

The proposed setup includes a 7 degree of freedom Cyton light-weight robotic arm mounted on a mobile base with a 3D projected light depth sensor on it. The perceived scene includes a box with cylindrical shaped pipes from which the robot is performing the bin picking like operations. The experiments were carried out both in a simulated environment as well as on real life robot. The robot model that we developed for the highly dexterous arm was already proved to be appropriate in other planning application [3]. Compared to our previous feature based recognition pipeline [4], in this paper we adopted a object decomposition in geometric primitives technique which parts are later on stored in a graph containing the topology of the analysed object [5].

The main advantage of this approach is that even in case of partially visible objects, one can reason about the position of the hypothetically detected object and can give a next best viewpoint. This next best view is useful in order to obtain the information about the missing part of the object, thus making the detection more reliable [6].

The current paper is validating the active perception idea in an industrial scene with cylindrical objects. The structure of the paper build around the main processing pipeline of the experiments: first details about the scene camera-arm calibration, data acquisition and scene registration is presented. Next, the object decomposition in geometric primitives is introduced with the object detection and manipulation part as well. Finally the experimental results using synthetic and real data are concluding the paper.

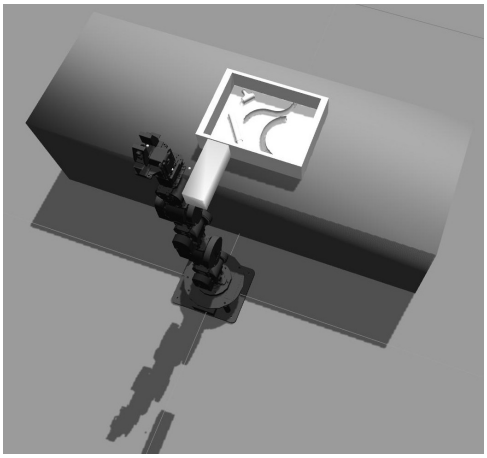

Fig. 1: Simulated test environment with the projective light depth camera mounted on a Cyton robot arm

II. Active perception pipeline

In this section the components of the active perception pipeline are presented. The pipeline consists of several interconnected modules, which are running in parallel in order to perform the desired object pick and place. The pipeline is similar to the one presented in the work of [7], but with the a hand in eye camera configuration.

A. Hand-eye camera calibration for the arm

The hand eye camera calibration is a special case of the external camera calibration performed with respect to a mobile arm. The main aim is to get a common reference for the camera and the end effector. This is a necessary step to be

performed in order to have the be able to pick a certain object seen by the camera[8].

Usually with the inverse kinematic chain a spatial transformation from the robot base to the gripper is available, thus this can be also used for calibration purposes, however in case of a slight error induced by the physical misalignment in the arm, this can cause differences between the camera and the gripper reference frame. In order to avoid this, we proposed a marker based calibration method using an external camera network system. The Optitrack camera system is able to deliver position and orientation information about the markers at high frequency and high spatial resolution. The later was essential in our case, as the aim was retrieving a precise enough calibration between the arm and the camera.

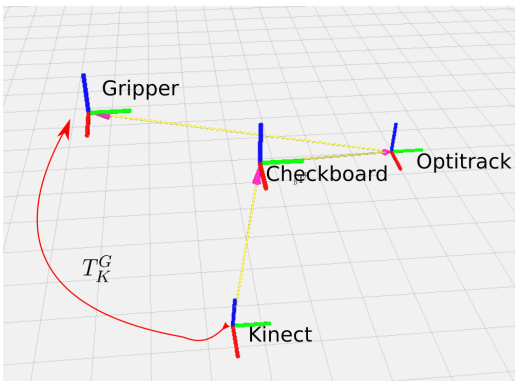

Fig. 2: Frames used for the calibration of the depth camera and the end effector using and external camera network system for pose estimation

With the configuration shown in Figure 2 the calibration of the camera and end effector from the arm can be done in an indirect manner: both the depth camera and the external Optirack camera system observes a landmark (in our case a checkboard), and based on the spatial transformation chain between these observations, one can determine the external calibration parameters of the depth camera with respect to the gripper, denoted in the figure with T_K^G. For the closed transformation chain additional reflective markers were placed on the end effector in order to localise it in the 3D space with the external camera network.

We achieved with this calibration a precision will up to $0{:}5cm$ in translation and less than two degrees in rotation. These values are fine for typical object manipulation scenarios.

B. Scene data acquisition and registration

Scene data acquisition is an important step of the active perception process. For a robot in order to be able to manipulate objects from an environment the perception of the environment has to be carried out. In our case, the scene analysis is performed on data acquired using a projected light type sensor from different views of the scene, more exactly views around the box containing the objects that are to be manipulated.

The data has been acquired both from a simulated environment as well as from a real sensor. In both cases, the acquired data from different view points needs to be registered in order to have a consistent analysis model of the environment. The pointclouds are registered using the iterative closest point (ICP) algorithm [9]. The ICP algorithm, first tries to find corresponding points in the input pointclouds, then computes rigid transformations needed to align the input pointclouds with respect to a global coordinate framework. Once the registration process is performed a consistent model of the environment from the input pointclouds is obtained.

The initialisation of the registration is done using the rough sensor position returned from the inverse kinematic model of the arm. As in our case the robot joints and physical structure is not a perfect one, the position is prune to errors. Also another error source in this case is the one due the hand-eye calibration. With these errors a typical scanning from two different viewpoints will be registered as shown in Figure 3.

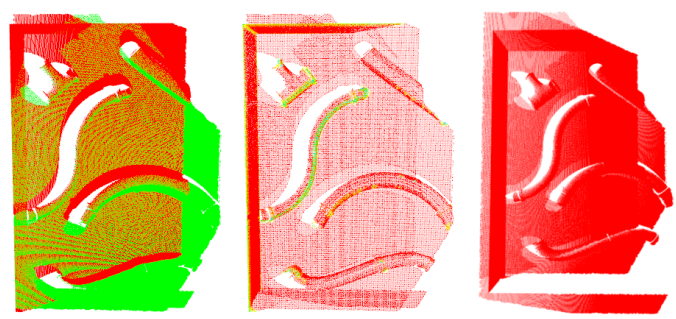

Fig. 3: The pointcloud alignment from different views(from left to right): input clouds (red-green), intermediate step during the registration and final registered variant

The dense pointcloud obtained from the ICP registration is filtered with a voxel filter in order to obtain a uniformly distributed cloud needed for the segmentation part of the pipeline.

C. Object decomposition in primitives

Once the registered input data is available, the next step of the processing chain involves the planar surface segmentation. This step typically removes the wall of the box containing the objects of interest. In our approach we used the Euclidean cluster extraction segmentation [10] which groups the regions of a pointcloud in consistent sets according the the metric distance among the points. For this algorithm two important tuning parameters have to chosen: the maximum distance threshold between the connected points as well as the difference of normal angles at the edges of the objects. A typical output in our case for these steps of the processing chain is shown in Figure 4.

Further on each object further decomposed into shape primitives and inserted into a graph model. Due to the fact that pipes have cylindrical shapes, in our experiments the objects are decompose into cylindrical shapes. The cylinder fitting is

Fig. 4: Segmenting out the major planes from the data and dividing into objects(from left to right): input data, planes segmented out, segmented object of interest

done in using a sample consensus approach from the Point Cloud Library (PCL).

From the detected cylindrical primitive shapes a graph is built keeping the topological structure of the pointcloud. The centroid of the each primitive is computed and is inserted into the graph and connected to the closest candidate that is already present in the graph. Thus for each object from the pointcloud a topological graph representation is obtained after this step.

A typical decomposition is shown in Figure 5, containing the input object, the partially decomposed variant and the graph representation of the extracted pointcloud sets.

In the next part the robustness of the proposed pipeline is analysed using synthetic and real data too.

D. Object detection

The next step of the processing chain is the object model recognition for the segmented objects. For this step the graph based model of the object is used in order the recognise the same object in the environment. Once each object of interest is decomposed into primitives and has a *template graph* based representation this can be searched later on in the *observations*.

In order to match an *template* graph model against *observation* one, first a random node from the template graph is selected and is compared with all the nodes from the observation graph.

The comparison is done taking into account the radius and the size of the underlying primitive components. If this step is successful, the next step is to compare the distances to the connected nodes, looking for similar distances. Theses two tasks are repeated until similarities between the two graphs are found. Pipe objects consists of small numbers of basic shape primitives, thus this graph matching process is a proper approach for this particular case. The summary of the proposed detection algorithm is described in Algorithm 1.

Algorithm 1 The proposed algorithm for component matching

Require: Scene model graph and query graph
Ensure: Detected object

1: Select random node from query graph
2: Find most similar node from model graph
3: Find similar edges of the two matched nodes
4: Repeat steps 2,3 until all nodes from the query graph are visited or no similar nodes are found

III. Results

This section contains the results of the test cases for the pipeline presented till now. The data has been acquired both from a simulated environment and a real one too. The Blensor simulation toolbox [11] has been used for the generating synthetic data. Blensor is developed on top of the 3d graphics tool, Blender having the capability of simulating a multitude of sensing devices such as:

- Velodyne
- Kinect
- Time of flight cameras
- Generic LIDAR
- Ibeo LUX

For our experiments the simulated Kinect sensor from Blensor was considered, as a similar sensor was available for the real data test cases too. For the data capturing the real environment, the pointclouds have been acquired using an ASUS Xtion PRO Live 3D sensor. In this case the same CAD T shaped model was printed out on a 3D printer in order to test the detection on the same template form. A typical test case is presented both for the synthetic and real cases in Figure 6.

The proposed pipeline has been tested in different test cases, where both the template data and the observation data has a certain level of noise corruption and comes both from a simulated as well as a real environment.

A. Noise free template–Noise free observations

In this case, both the training and query data comes from a simulated environment and are not affected by noise. The T shaped pipe is the object of interest for detection and for this object are presented the plots in Figure 7.

B. Noise free template–Noisy observations

In this test case, the same model built from a noise free data is matched against models constructed from data having different noise levels. The generated data has been affected by Gaussian noise of different scales means such as `0:1`, `0:16`, `0:19`.

The results of this test case is shown in Figure 8. As it can be seen from this figure, the method is detecting even in case of high level noise the template object in the observation list.