Technical Report: Active Segmentation

Gregor Gebhardt, Philipp Becker

October 24, 2017

Abstract

Autonomous manipulation of objects in unknown, unstructured environments requires robotic exploration of their geometric and physical properties. These properties are necessary to plan motions and reason about the effects of an action. Active segmentation approaches this problem by iteratively perturbing the environment and inferring models from the effects of the perturbation. In this report, we describe the current state of our implementation of active segmentation.

1 Introduction

To give robots the ability to interact with unknown objects in unstructured environments requires to enable them to explore the objects autonomously. From this exploration, the robots should infer the geometrical and physical properties of the objects. These properties could then be later used to reason about the state of the environment or to predict effects of actions on the objects. Active segmentation tries to solve this problem by combining segmentation algorithms with probabilistic reasoning on the observed effects of robotic perturbation of the environment. Initially the robot observes the environment and performs a basic segmentation based solely on visual clues. Afterwards, the robot iteratively performs an action which perturbs the scene and incorporates the effects of the perturbation into its probabilistic model of the segmentation.

In this report we present an implementation of active segmentation based on [2] in which the robot observes the scene through an RGB-D camera and performs perturbations by pushing the objects with a stick mounted on the end-effector. While the depth information from the RGB-D camera was used to separate objects from the table on which they are positioned and to initialize the segmentation, the effects of the perturbations are only detected using visual features. In our implementation, we extend this work by using features of the point-cloud data instead.

Our implementation is based on ROS (Indigo) [1] and developed on Ubuntu 14.04 LTS (Trusty Tahr). We use the point-cloud library in version 1.8 which is not available through the Ubuntu package manager and needs to be installed manually. In this document we describe the differences of our implementation to [2] and present the details of our implementation.

2 Preliminaries

TBA

3 Software Architecture

Our implementation of active segmentation is structured as a pipeline using ROS nodes and nodelets as components. The data is passed from one component to the next using the topic-based publisher-subscriber system of ROS. The processing of the data is triggered by ROS-service calls. In this section we describe the purpose and implementation details of each component.

Depth Camera

Needs to be able to provide colored point cloud data (or alternatively colorless point cloud data and RGB images) via ROS (PointCloud2 messages). Also the current pose of the camera needs to be available. In simulation this can easily be obtained, for real systems it needs to be inferred from the robots position.

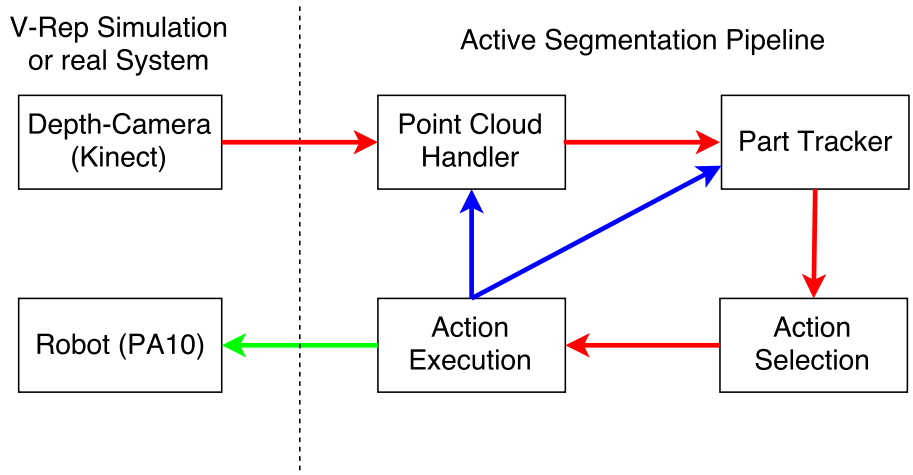

Figure 1: Overview of the modules of the Active Segmentation Pipeline. Red arrows indicate communication via publish/subscribe, blue ones via service and the green one via action server

Robot

Needs to be controllable in task space via the IAS Ros Robcom bridge. Needs to support inverse kinematic constrained on position only.

Action Execution

The Action Executor translates a pushing action into 3 individual actions and sends them to the robot via the IAS Ros Robcom bridge. The 3 actions are:

1. Go to point of contact with object (with arbitrary end effector orientation)
2. Push (along object normal)
3. Go back to save position

Furthermore, once in the beginning and after each push it records a new point cloud and initializes/updates the Part Tracker, together with the part tracker. The viewpoint for the recording of the point cloud are currently hard coded.

Interfaces

- Subscribes to Pushing Actions, published by Action Selection
- Requires the cloud handler service offered by the Point Cloud Handler
- Requires the initialize part tracker and update part tracker services advertised by the Part Tracker
- Advertises a service to reset the whole pipeline (clear Point Cloud Handler and Part Tracker, go back to safe position)

Point Cloud Handler

Receives a stream of point clouds with color information. Upon request snapshots of the current point cloud and camera pose can be taken and are stored in a list. Upon another request all snapshots currently in the list are merged, resampled as a equidistant grid with fewer points and cropped. The cropping removes the ground as well as all points beyond a certain horizon. Merging automatically clears the list, which can also be done manually by sending a clear request.

With this a compact point cloud is obtained for the part tracker to work with. The usage of multiple viewpoints allows for a more detailed view of the world and circumvents occlusions.

Interfaces

- Subscribes to the stream of RGB Point Cloud Data published by the Depth Camera using the PointCloud2 message type.

- Alternatively it can subscribe to color less point cloud data (also PointCloud2 messages) and color images, for more details see Cloud Color Combiner.
- Subscribes to the current pose of the camera.
- Advertises a point cloud obtained by merging all snapshots.
- Offers the cloud handler service, which offers commands for taking snapshots, merging and clearing the list.

Cloud Color Combiner

Some depth camera interfaces, like for example the V-Rep implementation of the Kinect (as of 08.17), do not offer direct publishing of point cloud data with color information. In that cases the additional Cloud Color Combiner Nodelet can be launched. It subscribes to a Point Cloud channel without color information as a corresponding RGB image channel, combines the two and publishes a colored point cloud.

Part Tracker

Initialized with a over segmentation of the scene. Upon request correspondence between parts over time is established. Furthermore semantic labels describing the observed changes are added. Those labels are specified by the Observation message This provides a compact representation of the parts and their relation and movement for the action selection to work with.

Interfaces

- Offers initialization service. Initializes with over segmentation, can also be used to reset.
- Offers update service. Triggers update of the parts, establishing correspondence and identifying motion and changes in the scene.
- Subscribes to current merged point cloud view published by Point Cloud Handler
- Publishes inferred observations as Part Array (see message types section)

Action Selection

Selects which part to push next in order to obtain new information and computes a push vector. Internally the selection is realized by maximizing a information gain criterion.

The start point for the push ("contact point") is the point closest to the average of all points that are not in a plane parallel to the ground. The push is directed along the surface normal at the contact point in x and y direction and parallel to the ground. The push length is currently fixed.

Interfaces

- Subscribes to the observations published by the Part Tracker
- Publishes pushing actions

Visualization

Multiple modules publish visualization information to use with rviz. Those information include:

- Current part centers (Marker Array published by Part Tracker)
- Inferred part movements (Marker Array published by Part Tracker)
- Last pushing action (Marker published by Action Selection)

4 Setup

Currently the setup consist of a single robot (PA10) with a special end effector for pushing, see Figure 2. In the future more robots using grippers for grasping should be integrated.

5 Message Types

Several non standard message types are defined for the pipeline and used by it

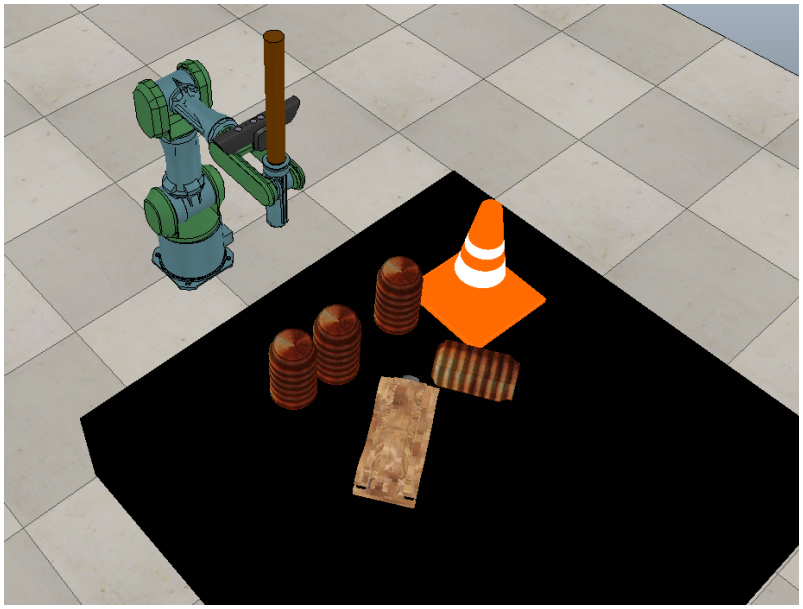

Figure 2: Exemplary setup in simulation. The robots end effector is a stick with a kinect mounted on it. This can be used to take pictures from different viewpoints and pushing

Observation

Enumeration over possible observations made for the parts. Each part may be assigned with one of the following labels:

appeared, no motion, motion, disappeared, reappeared no motion, reappeared motion.

Additionally each observations has an id.

Part

For each part the last observation made, a translation vector and the corresponding point cloud is published together with an id.

Pushing Action

Each pushing action consist of the origin of the push ("contact point") and the direction of the push

Cloud Handle Control

A cloud handle control request may contain either of the following commands:

(take) snapshot, merge (list, publish and clear), clear (list).

The response is a flag indicating success or failure of the operation.

IAS Ros Robcom GoTo Action

For further Information see the IAS Ros Robcom Bridge documentation

6 VRep Models

PA10 V-Rep Model

Todo

PA10 V-Rep Action Server

The PA10 V-Rep Action Server implements an IAS Ros Robcom Server for the interaction with the V-Rep simulator. Its name might be a bit misleading since it is fairly general and will most likely be reusable for robots other than the PA10.

End Effector and Kinect in V-Rep

The stick of the end effector is modeled as a simple cylinder with uniformly distributed mass.

There exists a model for the Kinect in V-Rep which is straight foreword to use. Also publishing the point cloud, color image and pose is straight foreword using V-Reps ROS-Plugin.

References

[1] ROS - The Robot Operating System.

[2] Herke Van Hoof, Oliver Kroemer, and Jan Peters. Probabilistic segmentation and targeted exploration of objects in cluttered environments. *IEEE Transactions on Robotics*, 30(5):1198–1209, 2014.