Grasping Characterization by using Symmetry Knowledge

Perception and Manipulation

Tutor: Pedro J. Sanz

Zsolt Pásztori
zspasztori@gmail.com

UNIVERSITAT JAUME I

1. Introduction

“**Robot**, any automatically operated machine that replaces human effort, though it may not resemble human beings in appearance or perform functions in a humanlike manner.”- **Encyclopædia Britannica**

According to the above definition robots need to be able to perform functions in a human-like manner. To be able to achieve this goal robots have to be able to autonomously manipulate objects. This manipulation task may seem easy, since for humans tool usage seems so simple, we do not consider it to be a measure of intelligence. It is a skill which most of us have innately. To be able to see how hard this task is we need to realize that several components of tool usage is born with us others are acquired automatically at a young age. When we are born we have the ability to move our limbs, and the simple movement primitives are coded into our genes. During the first years of our life we lift, push and drop hundreds of items playfully, meanwhile learning about dynamics and manipulation. The same time we are tuning our senses to the world surrounding us. We learn what tools and items are, this way shaping our vision. Meanwhile every time we pick up an item we learn about their weight, surface and dynamical properties.

For robots the manipulation task is hard for several reasons. They are limited by their mechanical design and intelligence. Currently we are not able to come close to the precision, strength and robustness of the human hand. Human hand is an extremely powerful actuator with its 24 degrees of freedom, built-in pressure sensors and ability to heal and get accustomed to the loads. The most problematic part for robots is their lack of pressure sensing. Manipulators with built in tactile sensing are not widely accessible as of yet. Robots are equipped with sophisticated 2D and 3D cameras. Their ability to capture images can be compared to that of humans. The problem comes from their visual intelligence. For a robot it is hard to locate and separate items from the video feed. Luckily computer vision went trough a revolution recently. At some limited tasks computer vision application are able to produce results on par with human performance. Still computer vision needs has a long road ahead.

In my current work I focus on the topic of robotic grasping. Grasping is the starting step of robotic manipulation. During this task the robot needs to be able to grasp a given object, lift it and move it to a given position. This capability seems limited compared to general tool use, such as playing the violin, but it has several clear economically viable possibilities. Robots with grasping capabilities can automate the packing in factories and warehouses. It

is especially important for companies in the field of logistics, such as amazon.com or aliexpress.com, since these companies need to handle millions of items on daily basis. Robots have clear advantage over manual labor since they can work non-stop, and are not burdened the monotonous work and bad working conditions.

During my current work I have implemented a system for automatic robotic grasping with Rethink Baxter. Carried out an extensive literature review of available grasping methods. I chose to use the symmetry information available in 2D images to base my grasping algorithm. I carried out a review of segmentation algorithms. Chose Masrk R-CNN for instance segmentation, for locating objects on images. Using the location of objects calculate the principal axis of inertia. Based on the axis of inertia and the object outline find grasping points. Grasping point locations are made more precise by 3D point-cloud data from a Kinect sensor. Using the built-in inverse kinematic solver of Baxter move the robotic arm and perform the grasping action.

2. Bibliography Research

To be able to choose a good grasping algorithm I have reviewed the state-of-the-art from “Data-Driven Grasp Synthesis – A Survey”. Unfortunately robotic grasping is still in a research phase. There are no general easy to apply methods available today. The available methods can be classified according to several aspects. On the figure below the most important of these aspects can be seen.

Illustration 1: Dimensions of comparing grasping methods

The paper categorizes the methods into two main categories: analytic and data-driven methods. Analytic methods are based on mechanical calculations. They try to calculate grasp points based on previously established heuristics. Analytic methods can be used for any item, they do not need to be trained in advance. Unfortunately for the calculations to be able to carry out, several limitations are needed. These limitations include: linear models, no surface frictions, constant density. Because of these constraints calculations can be carried out, but the grasp point precision is not good. Usually these methods do not perform well in application.

Data-driven methods choose their grasp points according to a machine learning model. There are 3 main types of methods based on how much knowledge they have about the items before execution.

The first category is based on grasping known objects. All objects to be grasped are known beforehand, grasp points are readily available. During execution only the item orientation must be specified. The grasp points are trained either by offline generating grasp candidates and then rating them with the help of a

simulator, such as GraspIT! . They can be provided with teaching by demonstration, when a person moves the robot hand to the grasp points. They can be obtained by reinforcement learning directly by the robot. These approaches work best in practice, but they need a costly training phase for each new object.

The second category is grasping familiar objects. Here the objects are not precisely known beforehand, but the model is trained on object primitives, which can be used as basis. Discriminative approaches learn good and bad grasp point candidates from the training data. Comparison based approaches have built-in grasp candidates for several items. During execution they try to find the most similar object from their database and base the grasp point synthesis on that item. Generative approaches try to infer full grasp configuration from items. They infer not just grasp points but also end-effector position and orientation too. Finally category based approaches learn grasp points for categories. These categories might not look similar, but because of their usage we handle them similarly, such as door knobs and shower heads.

The third category is for grasping unknown objects. These methods need the least training data, but perform the heaviest calculations during execution time. There are some methods which try to approximate the shape of an object based on some limited knowledge. There are some which try to estimate grasp points from low level features such as surface, or handles. Finally some use global features for grasp synthesis.

After going trough these possibilities I chose not to base the grasp synthesis on any previous work. Even tough there are many approaches none of them found fitting to my purpose, rather I tried to use the higher level concepts from these algorithms. In my algorithm tried on not to base the synthesis on previous knowledge, so to provide the most general solution of the problem.

3. Instance Segmentation

The field of computer vision went trough a revolution since the 2012 ImageNet Challange. Before the challenge computer vision applications consisted of several steps. These traditional methods were built as pipelines of edge detection, feature detection and classification. Unfortunately feature based methods turned out to be a dead end, since they were sensitive to noise and had small model capacities. Since the mentioned ImageNet challenge computer vision algorithms are based on deep neural networks and the extensive usage of convolutional filters.

The first task they were used was image classification. In this task each image has one object in focus, and the vision algorithm needs to classify this object. In the classification task computers have surpassed humans in accuracy around 2015. The generalization of image classification is object detection, where the input image contains several objects. These objects need to be classified, and their location needs to be specified by the usage of bounding boxes. Finally, instance segmentation can be considered a generalization of object detection. Here the objects need to be found on a picture, but for their location a bounding box is not enough. For their location each pixel of the picture containing the object must be labeled.

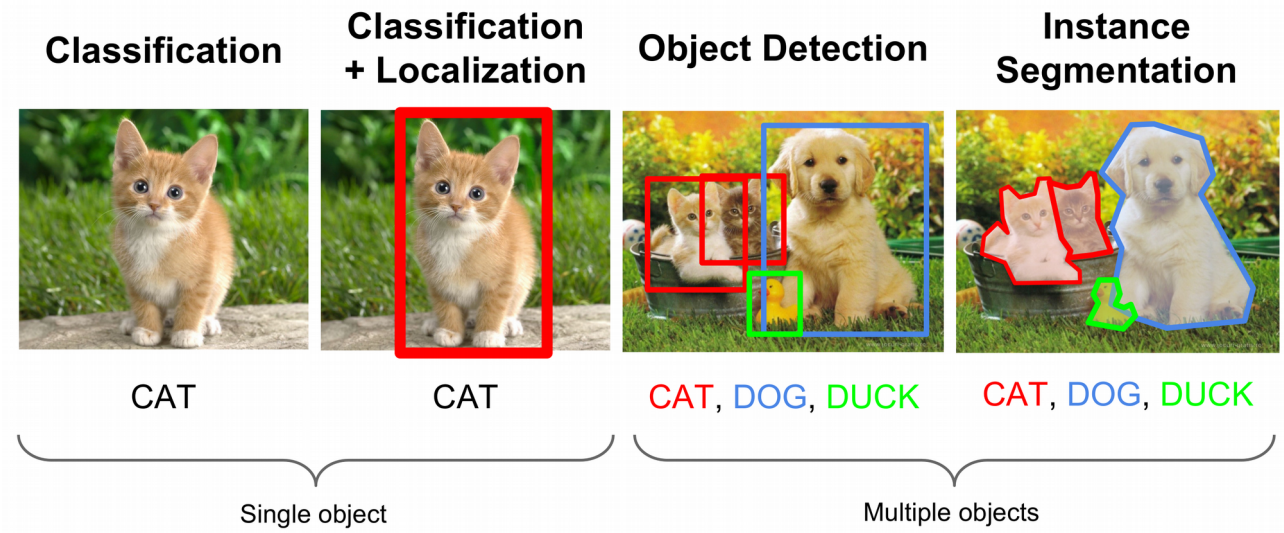

Illustration 2: Comparison of different computer vision tasks

For the grasping task the robot has to be able to locate the objects to the pixel level. Since the grasping points are usually on the edges of the objects. Instance segmentation is the task which the computer vision algorithm has to perform on a two dimensional image. It is important to remark tough that humans have capabilities connected to vision far beyond edge detection and classification. We are able to judge the size, weight, surface properties and dynamics of an object simply by looking at it. There is no application as of yet capable of predicting these informations.

Instance segmentation is a high complexity task. There are some initial trials for it, but these are more on the level of research, and can usually not be directly applied in application. For a computer algorithm to be used comfortably has to have several properties. Algorithm must have high accuracy and confidence, this is the most basic requirement. For application tough it has to be robust, need to work with limited computational resources, and be able to extended with ease to accommodate new items.

For choosing the best method available I have read a detailed survey paper on semantic segmentation. Instance segmentation is such a novel task, that there is no survey available for it. Instance Segmentation is a subtask of Semantic Segmentation. The difference between them is, that during Instance Segmantation each instance of a class have to get separate labels on the image. This means if there are 3 cats on the image, there will be 3 completely different pixel masks each corresponding to one of the cats. On the other hand semantic segmentation simplifies the problem, assigning one pixelmask to on object category. Semantic segmentation is a more mature research topic, and its easier to solve it because of its constraints. It can be used in environments where the number of objects is not necessary to be known, for example automatic driving. On the other hand during the robotic picking tasks it is absolutely necessary to distinguish each item on the pixel level.

Illustration 3: Comparison of semantic and instance segmentation

As of today there are three working architecture solutions for instance segmentation. All of them were published by the Facebook AI Research team (FAIR). These models are respectively: DeepMask, SharpMask and Mask R-CNN. I have decided to use the latest model Mask R-CNN, since it is an improved version of the previous papers.

Mask R-CNN is based on three different type of models. It contains an image classifier model, an object detection models part responsible for region proposition, and a segmentation network. An input image goes trough first the region proposal network, this was taken from faster R-CNN. Faster R-CNN is a 3rd generation object detection model, which is state-of-the-art both in accuracy, while having a relatively fast speed. Then the network patches created by the region proposition are run trough an image classifier. Image classifier can be switched based on accuracy and speed needs. In the original paper they used the ResNet-101 model trained on the ImageNet dataset. The main difference between faster R-CNN and mask R-CNN is in the third phase. Faster R-CNN only outputs the bounding box locations and labels. Mask R-CNN outputs the same information, but it parallel outputs the masks too belonging to each instance. This refinement was possible because of the realization that the object classifier contains the location information too in its layers. Most

classifiers simply discard this information, because they are not needed, in favor of easier regularization. Regularization methods, such as maxpooling and dropout, lose the spatial information contained in the image. By supplementing these techniques with usage of batch-normalization, leaky relu activation, and RoiAlign method the pixel wise masks can be easily obtained. RoiAlign was one of the main new contributions of mask R-CNN. In each image processing deep learning method the down sampling of the information inside the network is necessary. Most down sampling pooling methods down sample according some filter sizes, this way initiating a “rounding down” error to the system. The RoiAlign method gets rid of rounding down, and uses bilinear interpolation instead.

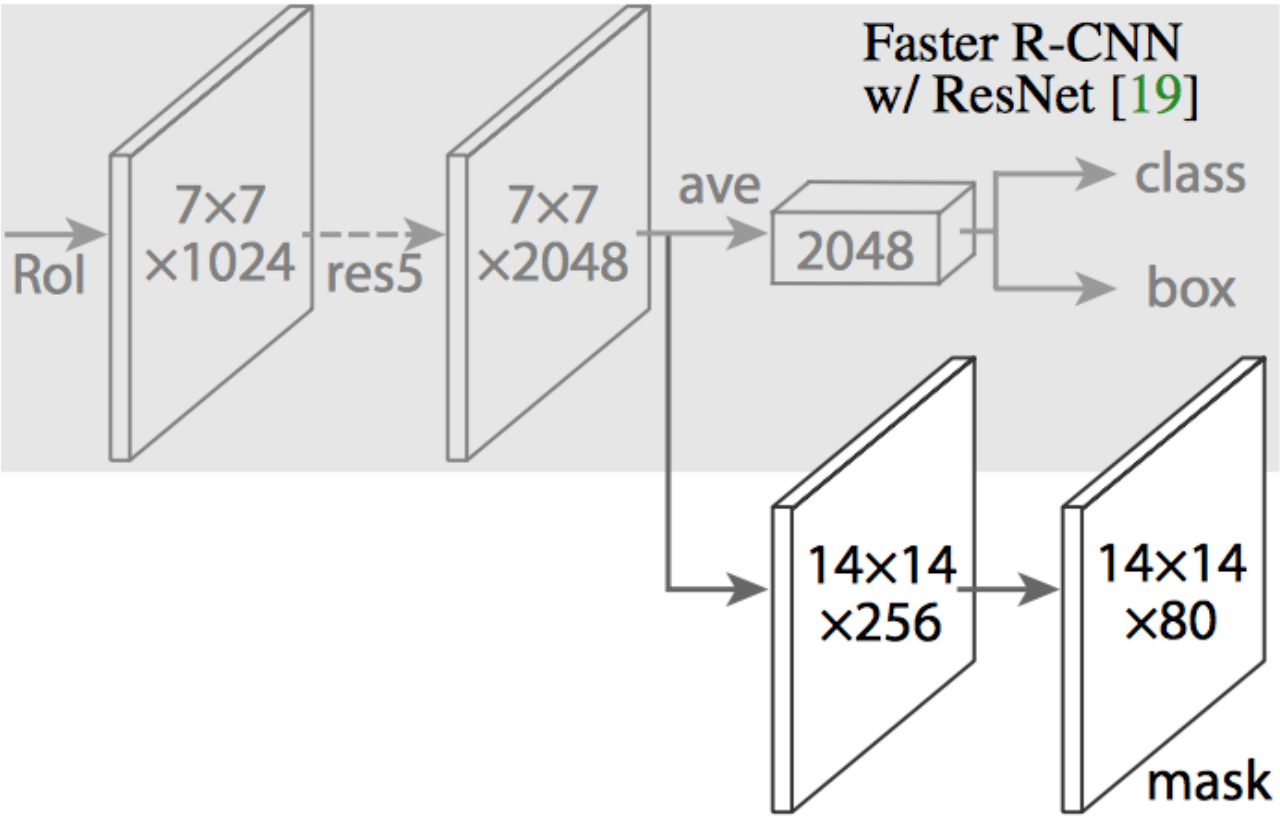

Illustration 4: Output layers of Mask R-CNN

The Mask R-CNN paper was published early 2017. Altough the method was far superior for any previous instance segmentation algorithms for long time it had no open-source implementation. It must be noted that simply implementing the code of the paper was not sufficient. A model also needed to be trained on the COCO dataset. Because of the complexity of training the model, since it does not converge automatically, and the length of the training task deep learning servers are needed for training. Luckily the MatterPort company has implemented, trained and open sourced their model during November. My

system is based on their implementation. Because of time and computational constraints I could not train/retrain their model. It must be noted that their model can be easily customized with a custom dataset. For new items, simply a medium size dataset is needed. Starting from their model it can robustly and fast trained with transfer learning.

The original model was trained on Microsoft Coco dataset. The dataset contains 120000 labeled images, with 81 different classes. These classes include vehicles, road signs, animals, food items and household items. For the picking task I had to choose items which were included among th data classes, such as a teddy bear.

For running the model several libraries are needed: Python3, CUDA, CUDNN and Tensorflow-GPU. The last 3 libraries are needed for the faster inference. Unfortunately, I did not have the privileges to modify these libraries, so the model could be run only on CPU. On the CPU the run time was around 20 second, but on the GPU probably a 20-50x speed up would be possible. Since the model was written in Python3, but ROS Kinetic is based on Python2 I had to write a wrapper between them.

Illustration 5: Mask R-CNN instance segmentation resoults

4. Grasp Synthesis:

Grasp synthesis is the task of specifying grasp points. Grasp points are point that the robotic manipulator has to create contact to be able to manipulate the object. Grasp point synthesis must take into account the sensory inputs and the properties of the available robotic arms and manipulator. In my case grasp points have to be in three dimensional space, although the images from where they are extracted are in two dimensional space. I have used a simple parallel gripper for the grasping. For this reason I needed an algorithm which specifies two points on the object.

During my work for extracting points I have not used any previous knowledge. The algorithm will work with even unknown objects. It is based on the symmetry knowledge of objects. Most man-made objects are mirror symmetric. This is due to the fact that we are mirror symmetric. This makes it possible to use most items with left and right hand too.

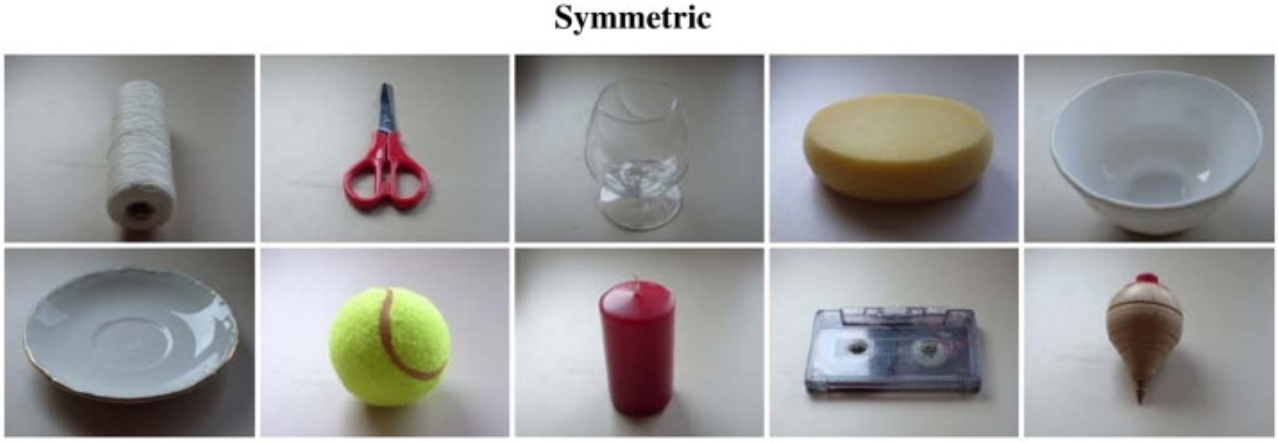

Illustration 6: Common household objects with symmetri

For high amount of objects the mirror axis corresponds to the axis of highest moment of inertia. The moment of inertia is a geometrical property of the object. Intuitively if we want to grasp an object with only two points the line on which the contact points lie must be perpendicular to this axis. To minimize the moments created by gravity on the object, the line which the grasp points lie must go trough the center of mass of the object. This means that the grasp points have to be on the second axis of moment of inertia. Additionally the grasp points have to be the furthest possible from the center of mass to be able to counterbalance the highest amount of torque. These assumptions hold only if the object has constant density and are non-deformable. People when

grasping an unknown object with chopsticks will try to grasp it the same way according to my experience.

The points of the object are the points extracted during the segmentation. Since the object is probably not a geometric primitive and its orientation is random, calculating the principal axis of inertia can not be done trough a closed form equation. Luckily Principal Component Analysis (PCA) can be used for this task.

PCA is a statistical method used mainly for dimensionality reduction. It is analogue for the principal axis theorem in mechanics. Since points in images can be considered a point distribution this method can be applied to them. It can be considered as fitting an n-dimensional ellipsoid to the data. The axis of the ellipsoid are the principal components. The longer the axis the bigger variance it covers.

To calculate the principal axis first the data has to be centered. This is done by calculating the mean of the variables and subtracting it from the values. Then the covariance matrix has to be calculated respect to all variables. From the covariance matrix we can get the axis by eigenvalue decomposition. Each axis has a value which shows how much variance it explains. In two dimensional case there will be two axis. The bigger axis will be hopefully the mirror axis and the smaller one will be the axis of grasp.

By using PCA we obtain the axis, and pick the points on the second axis, which lie on the edge of the item. These points are currently in the two dimension. For grasping we have to find them in three dimensional space. This is done by the help of the Kinect 3D sensor. I have calibrated the sensor, so it returns a pointcloud in the Baxter main coordinate frame. After the calibration simply extract the 3D points corresponding to the previously chosen 2D grasp points. Using these 3D points we can calculate the end effector coordinate, with the necessary orientation.

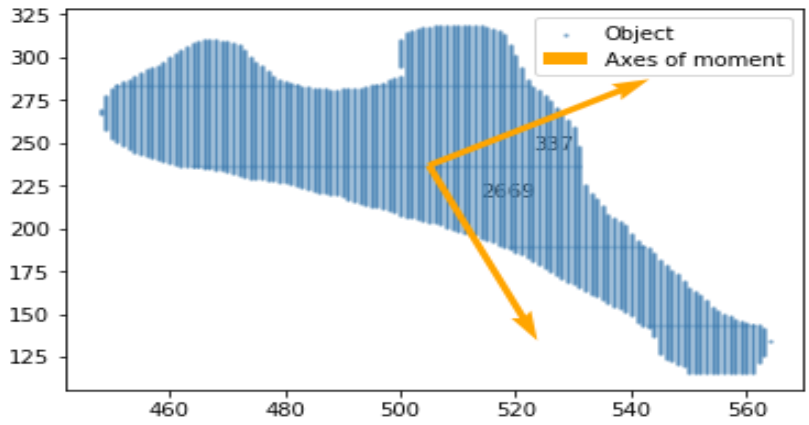

Illustration 7: Two principal axis of an item

5. Movement Execution:

Baxter has a 7 DoF robotic arm. Altough its arms are longer then human arms, because of its parallel grippers and arm configuration its reachable space is small. The parallel gripper has two degrees of freedom respect to the end effector. It can open-close, and it can rotate.

To simplify the endeffector position I made a few assumption. The objects are located on a planar surface of a table. The centerline of the endeffector will be diagonal to the surface of the table. The first assumption does not take away much from the generality of the application, since even in our households and storage facilities items are stored on shelves and tables. The second assumptions is necessary from a computational standpoint. If the orientation of the end effector could be free, we would need several 3D points to calculate the right trajectory. In my current project I wanted to focus on the two dimensional problem, hence the simplification.

Given the previous assumptions the end effector must be positioned above the center of mass of the object. To reach this position the robot must use inverse kinematics. There were two possible solutions for inverse kinematics solvers: MoveIt! and the inbuilt IKSolver. MoveIT! is a full fledged inverse kinematics software. It has useful features, such us being able to give approximate solutions and showing the simulation of the results on the Gazebo simulator. Unfortunately, it requires a hard set-up process, given the time constraints I was not able to utilize it. Baxter robot has an inverse kinematic engine built-in. It can be called from a ros-node, returns the joint solutions if a solution is possible. Unfortunately, it has several limitations in its calculations. It calculates the joint coordinates compared to the actual position, and several times it does not return with a solution. This problem can be improved by moving a bit the end-effector, and trying to calculate the movement from the new position.

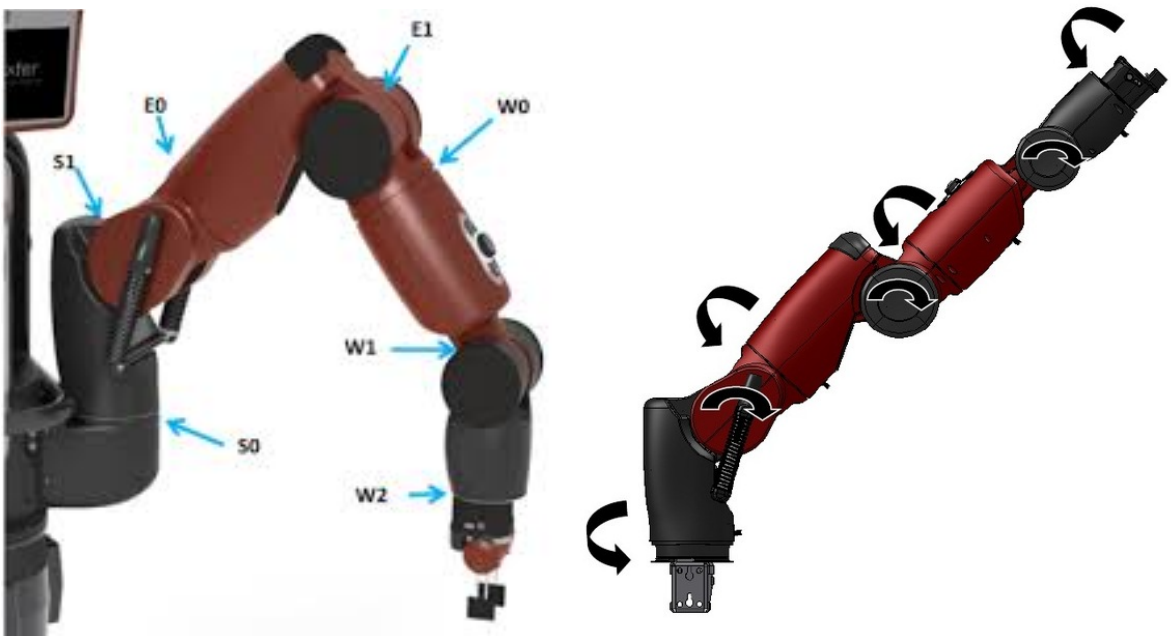

Illustration 8: Robotic arm of Baxter

The parallel gripper does not have pressure sensors at the grasping points. Without pressure sensors it is not possible to know whether the grasp has succeeded. For this reason the grasping is run in open loop. It is only possible to evaluate the performance trough inspection, and error handling is not possible during execution.

6. Experimental Results

To be able to evaluate the execution of the application I have carried out experiments. I have experimented with 3 items: a teddy bear, a ball and a package of a heavy object. The choice of objects were limited by the instance segmentation method. I had to choose an item out of the 81 possible classes.

I have carried out 5-5 experiments with each item. The teddy bear grasping succeeded 2 out of 5 times. The orange and box succeeded 3 out of 5 times. The accuracy of the program leaves something to be desired, but it also highlighted the weak points in the grasping application.

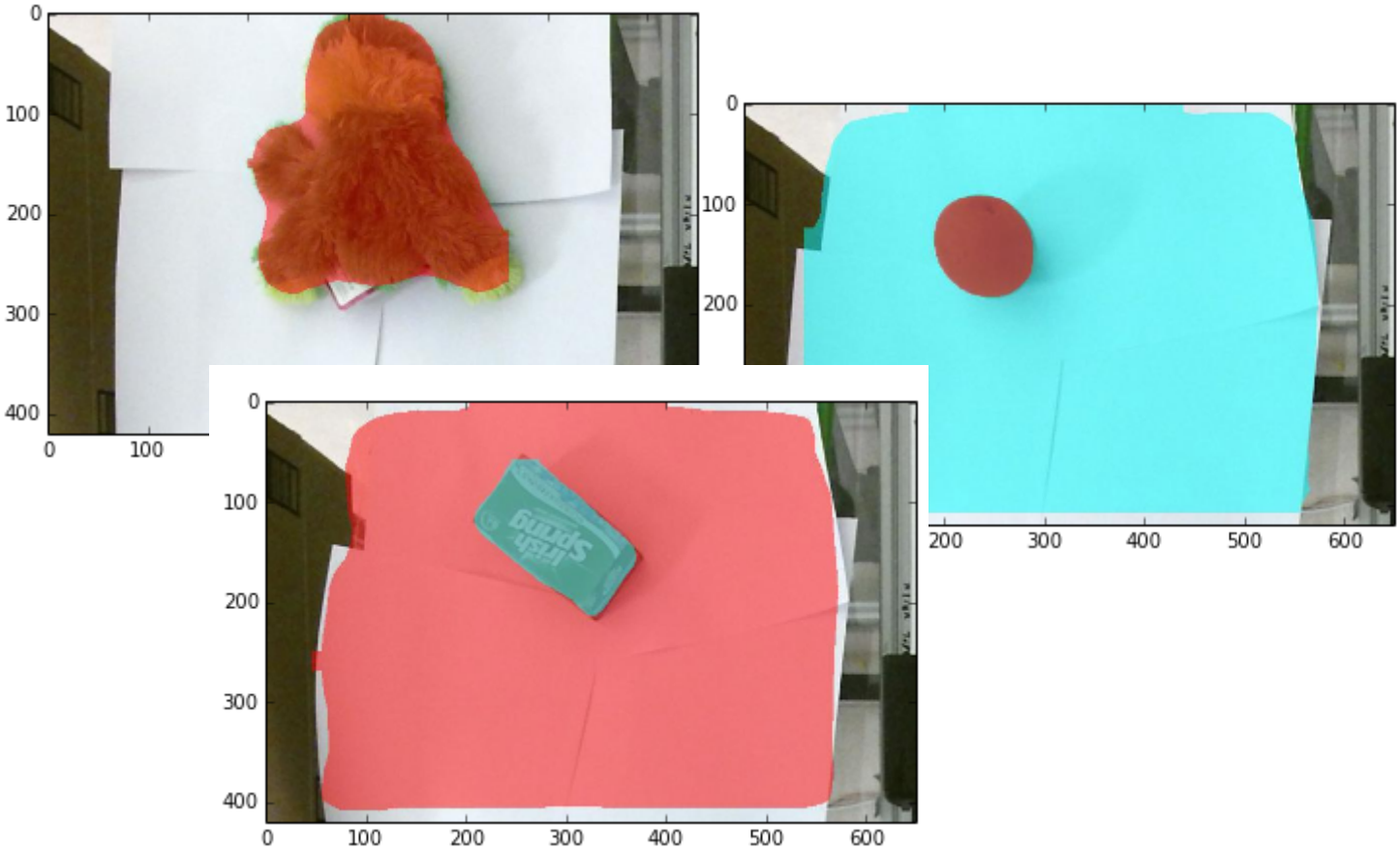

Illustration 9: The experimental items: tedday bear, orange, package of mint

The weakest link during the execution of the grasp was the parallel gripper. For each object the gripper had to be picked out of a selection of possibilities, the gripper had to be mounted with the right span, and it had to be modified at some cases. In my opinion a two point gripper should be enough for lot of the items, but instead of a rigid parallel gripper a claw-like actuator would function much better.

Another problem is the lack of force feedback. Baxter actuator can sense the force between the parallel gripper in theory. This sensor is not precise enough tough to be able to used in wider application. Several times the issue with the grasp was not the grasp points problem, rather that there was not enough force exerted on the item. This could have been easily fixed by retrying the execution when no closure was detected.

7. Discussion

During the timespan of 4 months I have created an application for carrying out grasping with the Baxter robot. The application is capable of automatically grasping objects from a table, even tough the accuracy could be improved.

The application has 3 main parts: instance segmentation, grasp synthesis and trajectory planning/execution. The method for instance segmentation is in my opinion very accurate and reliable. Unfortunately, this can not be said about the trajectory planning. A lot of times it does not return with a meaningful solution, which can make execution unreliable. I did find a solution for this, by assigning intermediate points to reach along the path. In the future implementing or using a better inverse kinematic service would be needed.

The project's main goal was to find good grasping points. In my opinion grasping is a very knowledge heavy problem. The heuristics I have used for the solution are good. Probably the brain uses a similar algorithm when it encounters unknown objects. The problem is that most items we encounter during our everyday lives are previously known. We have experience with their dynamical properties and do not calculate grasp points simply use the previously found optimal ones. I think using the symmetry knowledge of objects is a good way to go, but it should serve only as a basis for automatic learning.

8. Bibliography

(1) SPETH, Johannes; MORALES, Antonio; SANZ, Pedro J. Vision-based grasp planning of 3D objects by extending 2D contour based algorithms.

(2) Data-Driven Grasp Synthesis—A Survey.

(3) A Review on Deep Learning Techniques Applied to Semantic Segmentation

(4) Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick: Mask R-CNN