Pick and Place

Group 8
email@student.sdu.dk
University of Southern Denmark

May 28, 2015

Abstract

Text

Contents

Chapter 1

Vision

1.1 Known object

For this project we have two objects, which are created using the 3D design software called Autodesk Inventor 3D CAD. The result of this gave us an accurate 3D model which we were able to recreate using a 3D printer. The end result was an object the robot is able to pick up, with known dimensions.

1.1.1 3D to 2D

Since the world observed by a camera is 2D, some work has to be done to be able to perceive the 3D object. To do this, the objects were rendered from a multitude of different angles. The simple nature of these objects made it so only a few viewing angles were needed to capture the essence of the 3-dimensional shape. The first object used is a sphere, this shape was chosen as this is the least complicated shape to test with. No matter from which angle the object is observed, the shape will always be the same. See Figure 1.1. The second object, a cube, is a bit more complicated shape. When a cube is observed from a different angle, the shape will look different. For this object it was necessary to render images from different angles. A small selection of the resulting images can be seen in Figure 1.2, Figure 1.3 and Figure 1.4.

1.1.2 Contour detection

There are different ways to extract the features of an object. As for this project we used a contour detection to extract the main features from the object. This way we received enough information from the object to compare with, but we didn't overcomplicated the problem. As is described in[1] , there a more complicated but robust ways to better extract features from an object. Though with

[1]Stefan Lanser, Olaf Munkelt, and Christoph Zierl. "Robust Video-Based Object Recognition Using CAD Models". In: *Intelligent Autonomous Systems IAS-4*. IOS Press, 1995, pp. 529–536.

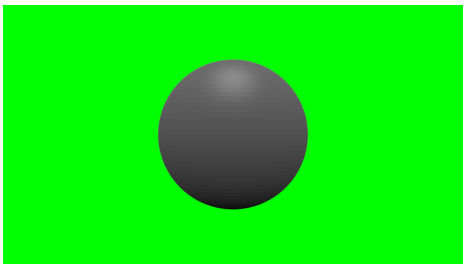

Figure 1.1: 2D image derived from 3D sphere

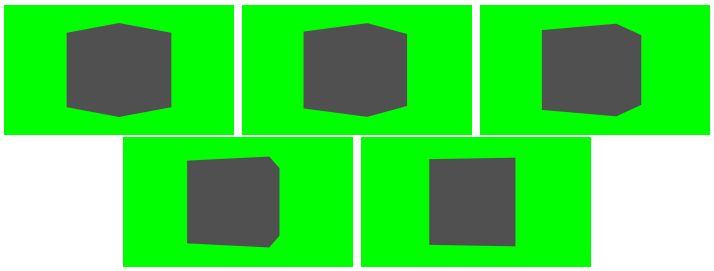

Figure 1.2: 2D image derived from 3D cube

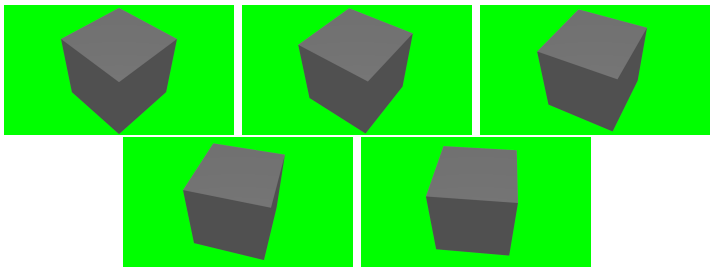

Figure 1.3: 2D image derived from 3D cube

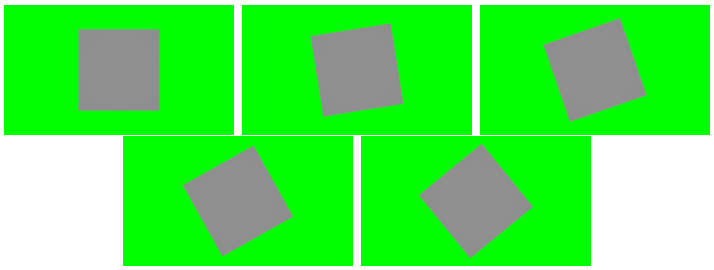

Figure 1.4: 2D image derived from 3D cube

the current time limit, we didn't pursue this method. As can be seen further in this report, the current method used was, how crude it might be, sufficient for the problem at hand. For the contour method we used OpenCV. This library provided us with tested methods to use and enough flexibility to be tailored for our problem. First the received image is converted to a gray scaled image. As can be seen in[2] , using a RGB image over Gray scaled image will not increase the accuracy by much. "Almost 90% of edge information in a colour image can be found it the corresponding gray scaled image." So in compliance with the K.I.S.S. [3] principle, we used a gray scaled image which is also a prerequisite to be able to use the standard contour detection build in OpenCV. Figure 1.5 shows the end result of the contour detection on a 2D converted image from the sphere.

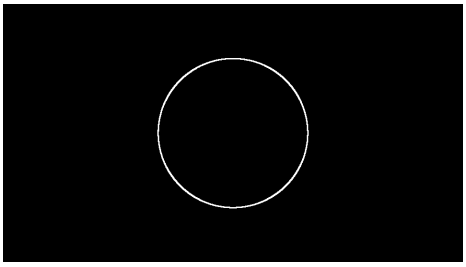

Figure 1.5: Contour detection from sphere

1.2 Camera

For this project we use a BubleBee2 and an Asus XTion Pro (also known as a "Kinect"). Each of these cameras have their own features which are usable in different parts of the project. Figure 1.6 shows the result of using the Kinect camera. The cameras shows us the workplace of the robot. The vision part here is to detect the object derived from the video feed. Detecting the object can be done in several ways. For this project we already have a image of the contours of the object we're searching for. This gives us one way of searching for it.

1.2.1 Colour segmentation

As the colour of the object is already known, to perform a colour segmentation on the video feed is a logical step to take. This will exclude any other objects that has nothing to do with the object we're searching for. If the object was build using multiple colours, using a colour segmentation method would still be

[2]S. Dutta and B.B. Chaudhuri. "A Color Edge Detection Algorithm in RGB Color Space". In: *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09. International Conference on.* 2009, pp. 337–340. DOI: 10.1109/ARTCom.2009.72.

[3]Stands for "Keep it simple, stupid" or "Keep it short and simple"

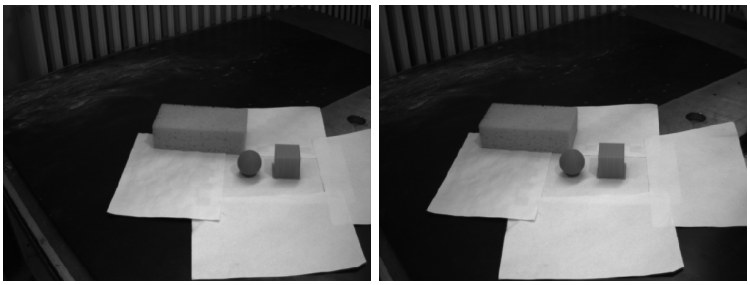

Figure 1.6: Kinect Camera

valuable as long as there are not too much interference of overlapping colours. This object doesn't have any colour patterns, which you could have also used otherwise, but it consist of one colour. This makes this step a bit easier, though detecting an object with multiple colour would have been a nice expansion. For this object, the cube has the same colour, we returned a thresholded image which only contains the colours in a range between H=0-179, S=130-255 and V=80-255. As shadows and different light conditions have an effect on the result, detecting the colour within a certain range is necessary. This result will then be (morphological) opened and closed in order to remove any small irregularities. Figure 1.7 shows the end result.

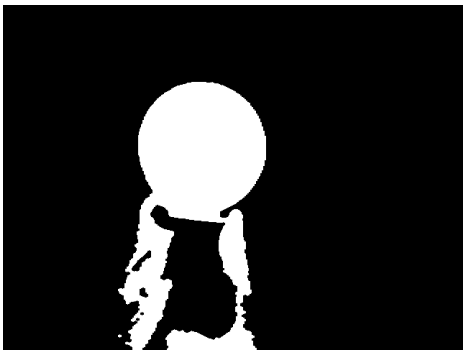

Figure 1.7: Colour detection

1.2.2 Contour detection

The contour detection used on the video feed is the same one used for the 2D image of the object. Only minor changes where necessary in order to change the video feed to the correct colour space. See Figure 1.8 for the result of the contour detection after a colour segmentation.

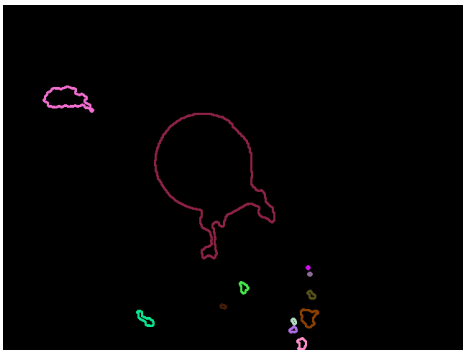

Figure 1.8: Colour contour detection

1.3 Feature detection

The feature detection uses several different methods in order to compare features from the 2D image to the converted video feed. The feature detection mainly consist of a SURF algorithm with a FLANN matcher.

1.3.1 SURF

To detect features from an image, a SURF algorithm is performed on both the object as the scene (i.e. video feed). SURF stands for Speeded Up Robust Features detection method. SURF was inspired from the SIFT algorithm but is several times faster. As for this part we're using a live feed, the speed of the algorithm is crucial. OpenCV has a SURF implementation which we used for this project. We used SURF with an hessian threshold of 400. This resulted in enough keypoints to use for the comparison.

1.3.2 FLANN matcher

FLANN stands for Fast Approximate Nearest Neighbor Search Library. FLANN is a library for performing fast approximate nearest neighbor searches. The algorithm used to match points from the object and scene is based on this. The used FlannBasedMatcher function trains the descriptor collection and find the best matches[4] . After finding the best matches, a quick calculation is done between the keypoint distances. In the end only matches who aren't too far away from each other are considered "good" matches and are used. The end result is then drawn and can be seen in Figure 1.9.

[4]OpenCV. *OpenCV FlannBasedMatcher*. URL: `http://docs.opencv.org/modules/features2d/doc/common_interfaces_of_descriptor_matchers.html?highlight=flannbasedmatcher#flannbasedmatcher` (visited on 05/29/2015).

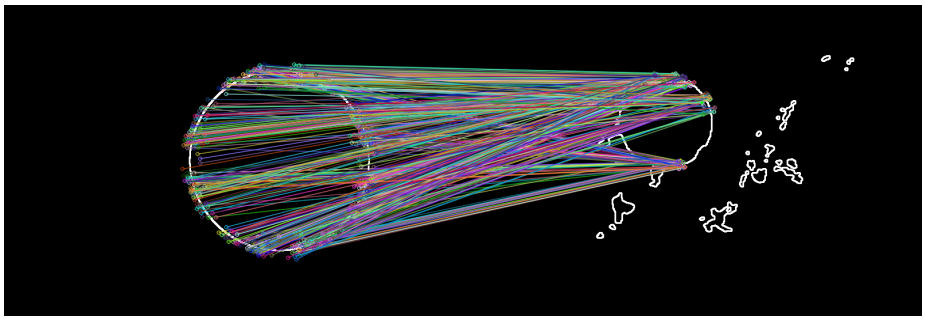

Figure 1.9: Colour contour detection

1.4 ROS

The connection with ROS comes down to receiving the video feed from different cameras and publishing information.

Chapter 2

Wrapping Up

2.1 Conclusion

In the end the filter works. There were some issues with one of the ADCs, which contributed to a colour channel not working properly. The A.I. implemented is quite rudimentary, but works reasonably well up until a certain speed.

2.2 Acknowledgements

I would like to express my thanks to Xander Bos, as my partner for this project. Also special thanks to Patrick Stolc for discussing various ideas.

Bibliography

Dutta, S. and B.B. Chaudhuri. "A Color Edge Detection Algorithm in RGB Color Space". In: *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09. International Conference on.* 2009, pp. 337–340. DOI: 10.1109/ARTCom.2009.72.

Lanser, Stefan, Olaf Munkelt, and Christoph Zierl. "Robust Video-Based Object Recognition Using CAD Models". In: *Intelligent Autonomous Systems IAS-4*. IOS Press, 1995, pp. 529–536.

OpenCV. *OpenCV FlannBasedMatcher*. URL: http://docs.opencv.org/modules/features2d/doc/common_interfaces_of_descriptor_matchers.html?highlight=flannbasedmatcher#flannbasedmatcher (visited on 05/29/2015).