Cardboard Detection Report

Josef Kahoun

April 2025

Contents

1 Introduction

This short document serves as a report on the Cardboard Detection and Pose Estimation problem, a project that was addressed at the Czech Institute of Informatics, Robotics and Cybernetics (CIIRC).

The structure of this document is as follows. In section 2, the general problem and the hardware setup are described. Section 3 introduces the proposed solution and shows the results.

2 Task setup

The goal of the algorithm is to obtain an SE(3) pose estimation of a cardboard cutout located on a worktable with the use of an RGBD camera. The camera can be located either on a robotic arm or fixedly positioned above the worktable. The algorithm should be able to locate the object even if it is only partially in the camera frame.

The cardboard cutouts can be of various shapes and sizes. Additionally, the 2D CAD files of the cutouts will be provided. Examples of the cutouts can be found in Figure 1.

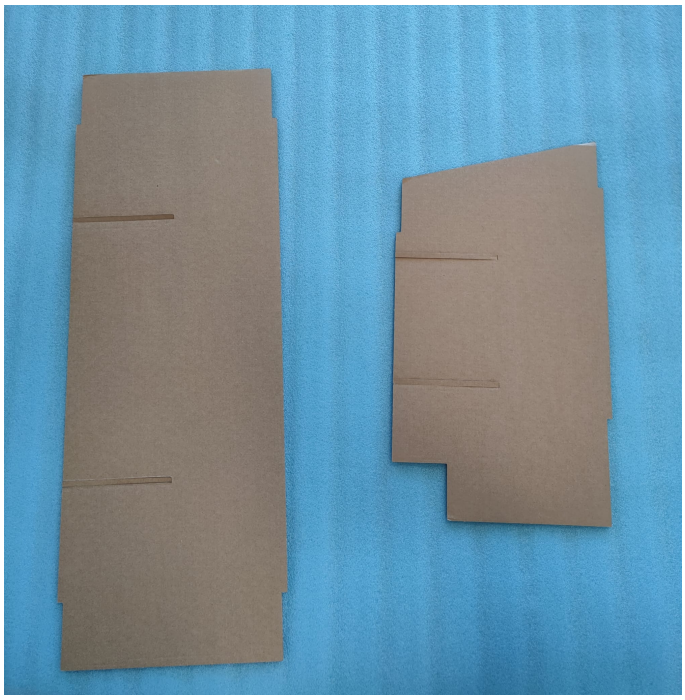


Figure 1: Examples of the cardboard cutouts.

3 Proposed solution

Multiple solutions were tested for this problem. SIFT/ORB[1] keypoint matching has been utilized for obtaining correspondences to be later used in PnP, but the uniform color of the cardboard prevented the algorithms from obtaining enough distinct keypoints. Generalized Hough Transform (GHT)[2] has been able to obtain the rough position of the object in some cases, but proved to be computationally expensive and not robust enough. The best result was obtained by minimizing the Chamfer Distance between the projected points from the outline of the CAD

[1] SIFT, ORB
[2] GHT

template and the edge map points from the frame. This approach was, however, too prone to the influence of outliers.

Based on the observations made during the testing, the following pipeline was devised:

1. **Detecting the object in the frame** in order to get rid of other objects and noise far from the object.
2. **Getting the rough position estimate** from the edge map of the image.
3. **Improving the estimated position through optimization**.

The whole algorithm is built upon the minimization of the Chamfer distance. Chamfer distance is a metric used to evaluate the similarity of 2 different sets of points. Given two point sets A and B, the Chamfer distance is calculated as the sum of distances of each point in A to its nearest neighbor in B, plus the sum of distances of each point in B to its nearest neighbor in A.

We start with an initial guess of the translation and rotation vectors of the cardboard's position, and use those to project points from the CAD template outline onto the image with the use of the camera matrix and distortion coefficients. Iterating over the translation and rotation vectors, we can find the global minimum of the Chamfer distance, thus receiving the best possible match. We therefore formulate an optimization problem

$$\begin{aligned} \min_{\mathbf{t},\mathbf{r}} \quad & \text{chamfer\_distance}(\mathbf{P}(\mathbf{x}_{CAD}, \mathbf{K}, \mathbf{d}), \mathbf{x}_{IMG}) \\ \text{s.t.} \quad & \text{initial parameters: } \mathbf{t}_0, \mathbf{r}_0, \\ & \mathbf{t}_{min} \leq \mathbf{t} \leq \mathbf{t}_{max}, \\ & \mathbf{r}_{min} \leq \mathbf{r} \leq \mathbf{r}_{max}. \end{aligned} \tag{1}$$

$\mathbf{P}$ is the projection function, $\mathbf{x}_{CAD}$ are sampled points from the CAD outline, $\mathbf{K}$, $\mathbf{d}$ are the camera matrix and distortion coefficients respectively, and $\mathbf{x}_{IMG}$ are sampled points from the found edges of the object. The rotation and translation vectors are bound with the parameters $\mathbf{t}_{min}, \mathbf{t}_{max}, \mathbf{r}_{min}, \mathbf{r}_{max}$.

However, the optimization itself shows two main problems:

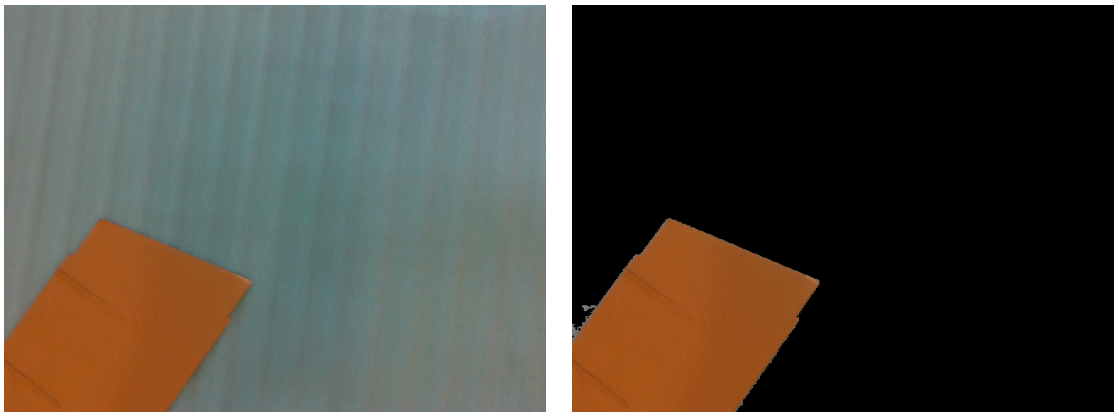
1. The Chamfer distance can be highly influenced by outliers (=noise in the edge map or edges of different objects), and
2. The optimization can converge to a local minima and get stuck there. The minima differ based on the initial parameters.

To solve both problems, we introduced the additional steps in the algorithm, one which firstly detects the object to get rid of the outliers, and one which finds the best starting initial parameters to make sure the optimization converges to a global minimum.

3.1 Detecting the object

The detection of the object can be done in multiple ways. Specifically, using Neural Networks (NNs), either for detection (such as YOLO) or segmentation (SAM2), can be quite a robust approach. However, this requires creating a dataset for every cutout part and training the NN.

We utilized the different background color and performed a simple color segmentation to obtain a mask. This was done using the *cv2* library, with predefined color ranges in the HSV color space. Combining the mask with the image results in an image containing just the cutout object. The results can be seen on Figure 2.



(a) Test image. (b) The segmented image.

Figure 2: The comparison of the provided test image (a) and the result of the segmentation (b).

3.2 Edge detection

To detect the outline, we need to obtain the edges in the picture. For now, we utilize the standard Canny[3] detection algorithm. However, more robust algorithms can be used, f.e. the HED[4].

3.3 Obtaining the rough position estimate

We settled on 2 approaches for the rough position estimate:

1. RANSAC using the optimization method,
2. Template matching.

3.3.1 Template matching

Template matching consists of sliding the template image alongside a given test image, and choosing the highest match. Multiple libraries already implement template matching. However, they require the whole template to be found, and cannot handle only partial visibility of the template. Therefore, own implementation was written, using cross-correlation as the scoring method.

The template matching in our case is done on the detected edges of the image, and the templates were obtained by rendering the supplied CAD template at predefined distances and rotations. Upon matching every template, we choose the one with the best score, obtaining the position of the template on the image.

The rough rotation is obtained from the rotation of the matched template. For the position, we first extract the center pixel from the matched template. The general transformation between real world points and image pixels is given as

$$\lambda \mathbf{u} = \mathbf{K}\mathbf{x} \tag{2}$$

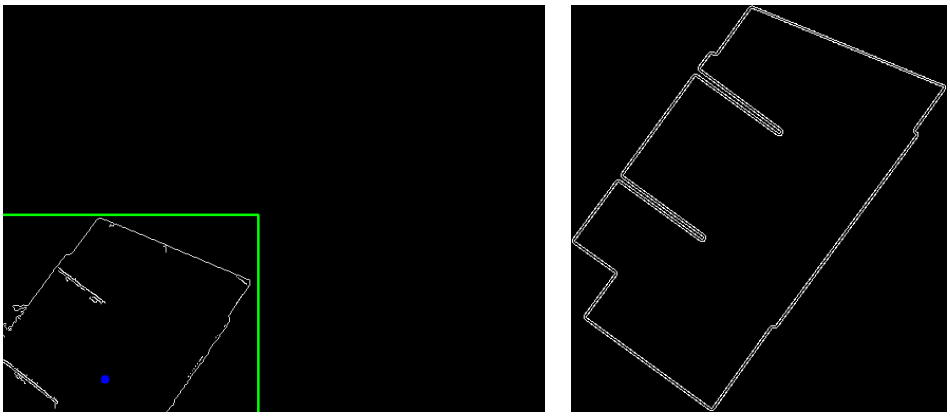
where $\mathbf{K}$ is the camera matrix, λ is the scale, $\mathbf{x}$ are the world points in camera coordinates, and $\mathbf{u}$ are the homogeneous pixel coordinates. Rewriting, we can obtain the formula for the world coordinates as

$$\mathbf{x} = \lambda \mathbf{K}^{-1}\mathbf{u}, \tag{3}$$

where the scale/distance λ is known from the used template.

[3] https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html
[4] https://arxiv.org/abs/1504.06375

The results of the rough position estimation through template matching are shown in Figure 3.



(a) The found position of the template. The blue dot represents the center.

(b) The rendered template image with the best score.

Figure 3: The results of the position estimation through template matching.

3.3.2 RANSAC

We can also utilize the RANSAC to find the rough position estimate. We fit a model through the chamfer optimization; however, on far fewer points than the final one. This bypasses the problem of outliers and incorrect initialization. Moreover, the final optimization could do more harm than good, since it does not utilize RANSAC but takes all the points.

The exact RANSAC implementation samples a predefined number of points m from the CAD outline, and n points from the edges found in the picture. These points are sampled at random. After sampling the points, the Chamfer distance optimization is performed. We value the result of the optimization based on the following value function: Each sampled CAD point is projected onto the image using the found SE(3) transformation, and the distance to the closest edge point is calculated. If this distance is smaller than a given threshold, we classify this point as an inlier. The final value of the RANSAC iteration is given by the number of inliers divided by the number of projected points, the CAD points that were projected onto the image, not outside it[5].

The RANSAC is usually more robust than the template matching, but setting the correct number of points to optimize is crucial. The results of the RANSAC approach are seen in Figure 4.

3.4 Refining the position

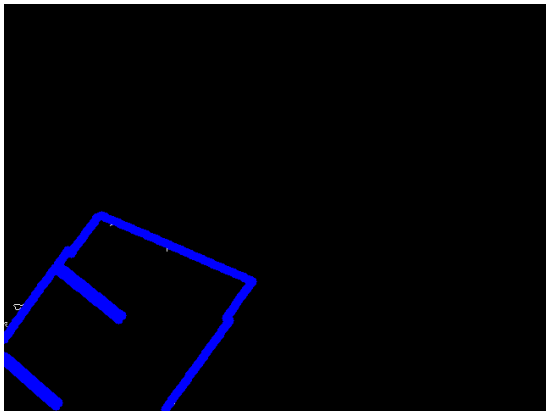
The final refinement is done using the Chamfer distance optimization in (1). As initial parameters, the rough position and rotation obtained using the steps described in the previous section are used. If the initial estimate is close enough, the refinement correctly converges to the global minimum, thus finding the desired pose estimation. The optimization was implemented using the *scipy* library. Projecting the CAD points using the resulting transformations, we can visualize the result in Figure 5.

4 Experiments

In the following section, the brief experimental results are provided, as seen in Figure 6. In the left column, the initial captured images are seen, while in the right, we can we the projected

[5]This is needed to work accurately when the object is only partially visible and at the edge of the picture.

(a) The initial image.



(b) Projected CAD template using the obtained translation and rotation vector from the RANSAC optimization.

Figure 4: The results of RANSAC estimation.

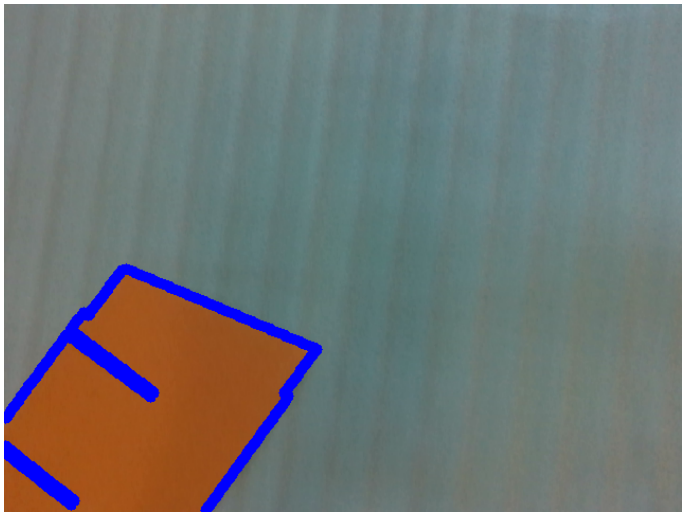


Figure 5: The final optimization results

CAD points in blue using the transformation obtained from the pipeline.

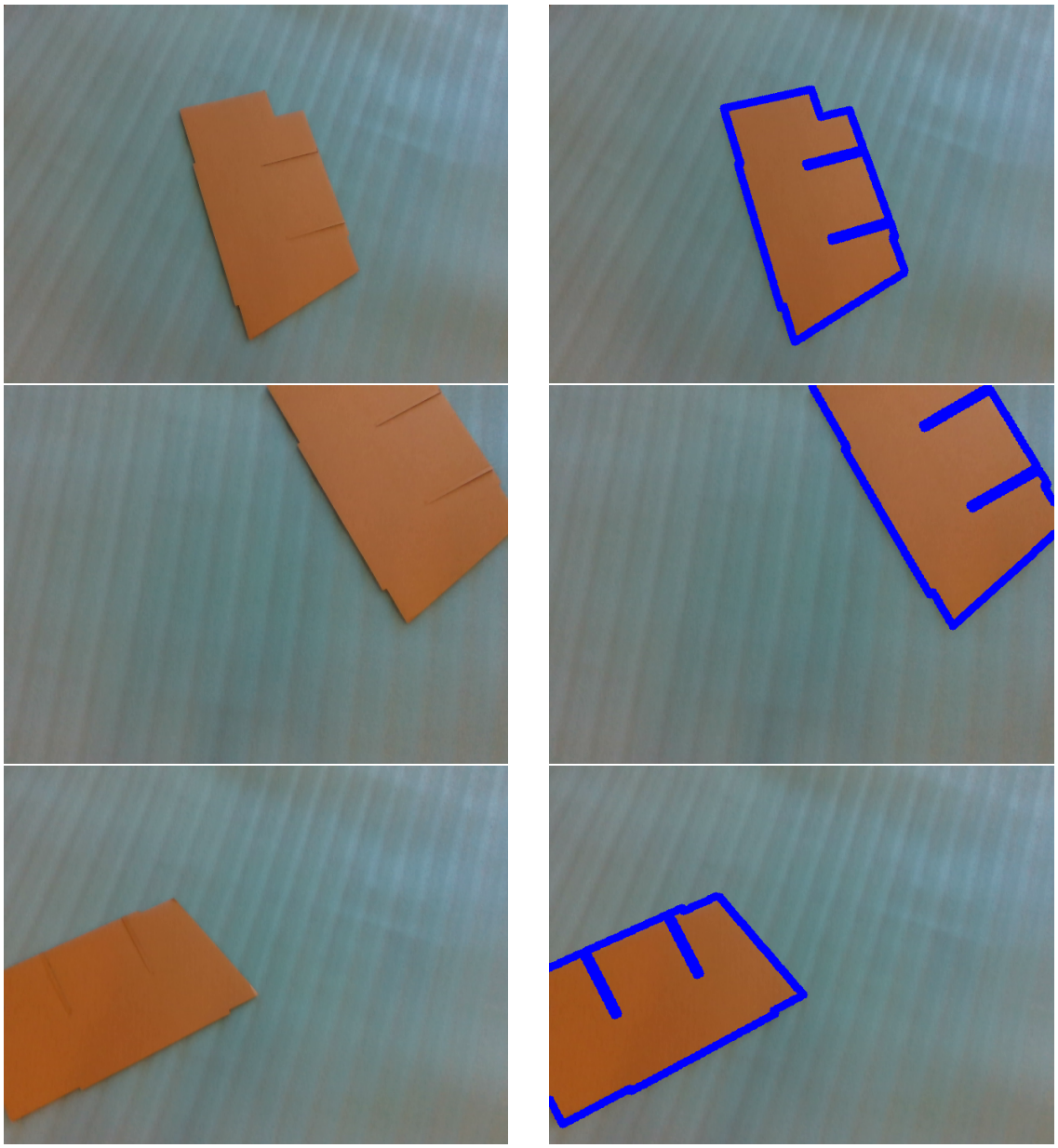


Figure 6: Experimental results.

5 Conclusion

In this document, the pipeline for estimating the SE(3) pose of a cardboard cutout has been described briefly. For more information, please visit the provided Github repository.

5.1 Future work

The work presented could certainly be improved, either by using a better edge detector, incorporating a Neural Network for primary detection, etc. However, at the time of writing, the

specifications of the setup were not provided, so testing the specific situations was impossible.