Augmented Reality - VU WS 2022/23

Alexander Plopski, Ana Stanescu

Task 1 - Camera Pose Estimation (40 points)

Assignment changes

The assignment has been modified to account for issues the DLT algorithm has when using a planar target. All changes throughout the document are highlighted in **bold**. Tips on how to determine the 3D coordinate of extracted locations of interest from the template of the tracked target are provided below the assignment information.

Task Description

Implement a camera pose estimator based on natural feature tracking, using one of the two approaches described in the lecture. Implement the RANSAC algorithm as part of your estimation. Use the minimum number of necessary points for the initial estimation in the RANSAC loop.

For this purpose, we use the *OpenCV* library, version 4.6.0. You need to use a marker with an image of your choice **that is not used in the examples below**. **You will need to construct a 3D target from this image if you use DLT.** With the estimated pose, you have to render a virtual object, e.g., a rectangular cuboid wireframe, on top of the marker.

We recommend using C++ for this assignment, but Python is also permitted.

Aruco marker tracking and OpenCV built-in RANSAC is not allowed in this assignment. You can use the OpenCV camera calibration toolkit.

This task is an *individual* assignment.

Submission

Your submission needs to be uploaded to the TeachCenter, and to contain the following components:

Code: This includes your source code without the OpenCV library.

Report: Write a short report including the following information:

- a description of your application. Include your motivation for selecting the algorithm, interest point detectors, feature descriptors, matching approach, etc.

- an image of the chosen target and a motivation of the choice of target: Why did you chose the particular target? Which kinds of targets would perform differently (better/worse), and why?
- report on the FPS that you achieve

You can include images in your report to support your findings. The report needs to be uploaded in a .pdf format.

Video: Include a video to demonstrate the live behaviour of your tracker.

Copy of source code (from your colleagues or from other sources) is forbidden. The assignments are tested for plagiarism.

The **deadline** of this submission is **December 4, 2022 (23:59)**.

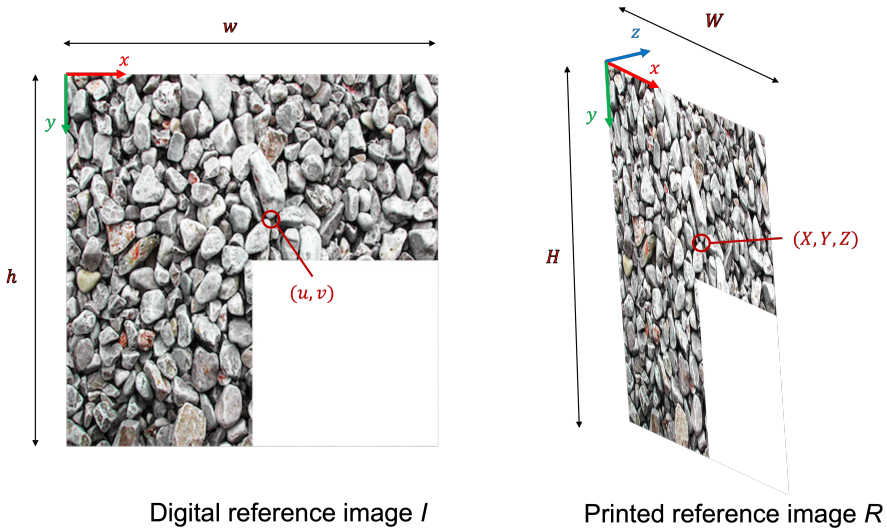

Figure 1: When digital reference image I of size (w,h) is printed as an image of (W,H) meter. The coordinate origin of I and R is in the top left corner and the x and y axes point along the width and the height of the image, respectively.

Creating 3D Locations from Points of Interest on a 2D Target

Consider the 2D reference image I used as a reference. An example is shown in Fig. 1. This image is w pixel wide and h pixel high. OpenCV places its origin into the top-left corner, with the x-axis along its width and y-axis along its height. Now, consider its printed counterpart R with a width of Wm and a height of Hm, as shown in Fig. 1(right). Its origin is also in the top-left corner, with the x-axis along its width and the y-axis along its height, the same as for I. Following the right-hand rule, the z-axis is pointing towards the back. Given that the origin of I and R, and the direction of the x- and y-axis coincide for both, we can convert any pixel (u,v) into a spatial location $\mathbf{P}$ on the printed reference image, where

$$\mathbf{P} = (u * \frac{W}{w}, v * \frac{H}{h})^{\mathrm{T}}. \tag{1}$$

For a planar reference image we can assume that all points lie in the X-Y plane, meaning all points have the z-value 0. Thus we receive the point

$$\mathbf{P}_3 = (u * \frac{W}{w}, v * \frac{H}{h}, 0)^{\mathrm{T}}. \tag{2}$$

Applying Eq. 2 to all points of interest detected in I creates a database of 3D locations with corresponding descriptors.

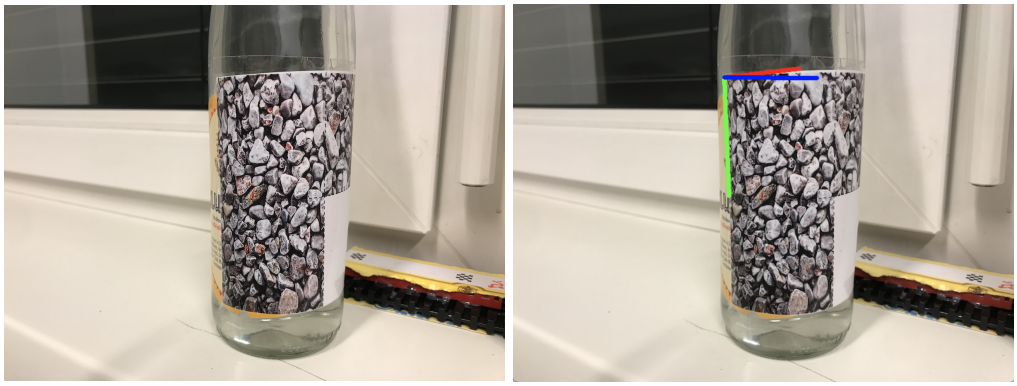

Figure 2: The printed reference image has been attached to a cylindrical object (bottle). (right) The axes of the reference are rendered onto the image.

Creating 3D Locations from Points of Interest on a Cylindrical Target

Once we know how to calculate the location of a point of interest on a planar image, we can compute its location once the target is deformed.

The easiest way to create a non-planar tracked target that works well for the DLT algorithm is by pasting the printed target R onto a cylindrical surface, e.g., a softdrink bottle as shown in Fig. 2. Assume that the coordinate systems of the bottle and the printed reference image align, i.e. the axis of the cylinder aligns with the y-axis of the target.

As shown in Fig. 2 and w.l.o.g. assume that the origin of our target is again in the top-left corner of the printed image, its y-axis points downwards (along the image's height), the x-axis is tangential to the cylinder surface and the z-axis is pointing towards the center of the cylinder. If the cylinder has a diameter of d(meter) and a radius r(meter), its center is located at (0,0,r).

To determine the 3D point for a pixel (u,v), first we compute its location on image plane P=(X, Y) as described in Eq. 1. As Y is not distorted by the cylinder curvature, we can ignore it in the following considerations.

Consider the scenario shown in Fig. 3, where the center of a circle is at the origin of the coordinate system at $\boldsymbol{O}$=(0,0). As point $\boldsymbol{p}$=(-r,0) moves on the perimeter of the circle, it transcribes an arc. Given moving a a, the new location of p will be $\boldsymbol{p}$'. We can determine the angle θ between the vectors $\overrightarrow{\boldsymbol{Op}}$ and $\overrightarrow{\boldsymbol{Op}}$' from the arc length as

$$\theta = \frac{a}{r}. \tag{3}$$

From this, we can easily determine the location of $\boldsymbol{p}$' as

$$\mathbf{p}' = (cos(-\theta) \cdot (-r), sin(-\theta) \cdot (-r)). \tag{4}$$

Note, that in Fig. 3 $\boldsymbol{p}$ is in the X-Z plane with the same axes directions as in our scenario. We

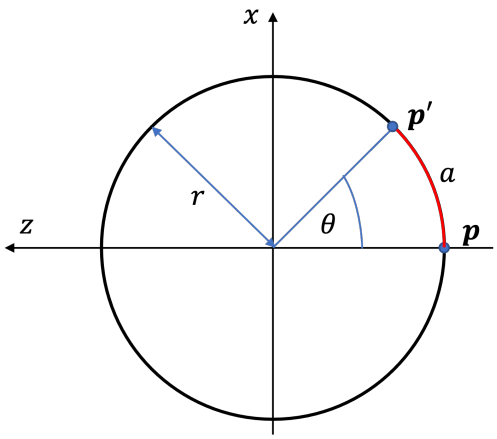

Figure 3: Diagram of the cylinder design when viewed from above.

can thus easily determine the x and z coordinates of our 3D point by applying this 2D case and the y coordinate we have calculated at the beginning.

For (u,v) the corresponding location is thus given as

$$\mathbf{P}_3 = (sin(-\theta) \cdot (-r), Y, cos(-\theta) \cdot (-r) + r)^{\mathrm{T}}, \text{ where } \theta = \frac{X}{r}. \tag{5}$$

Figure 2 shows the results of DLT applied to this scenario where the coordinate system is rendered at the origin of the template used in the example above.

Creating 3D Locations from Points of Interest on a Bent Target

It is possible to create a 3D target from R by bending it at given points. For example, you can bend the target in the center, so it wraps around a corner as shown in Fig. 4. However, we would recommend using the cylinder for several reasons:

- it is very likely that you will select points that lie on one of the bent sides (one of the plane) during RANSAC, something we are trying to avoid
- depending on the bending angle it may be difficult to reliably detect features on all sides, e.g., the 90 degree angle in the example requires very specific viewing angles to detect a sufficient number of features on more than 1 side

To compute the 3D location of a point (u,v) for a bent target, we start once again from the corresponding location $\mathbf{P}=(X,Y)$ on the printed image. W.l.o.g. with the origin in the top-left corner of the printed image and the x-axis along the width and y-axis along the height of the image, assume that we bend the target along the x-axis B as shown in Fig. **??**. If $X \leq B$ we do not need to update the position, and P_3 is defined as in Eq. 2.

If $X > B$ we need to account for the bend. As the bend does not affect Y consider the 2D case in the X-Z plane shown in Fig. 5. For a point $x=(X,0)$ the bend of an angle θ corresponds to a

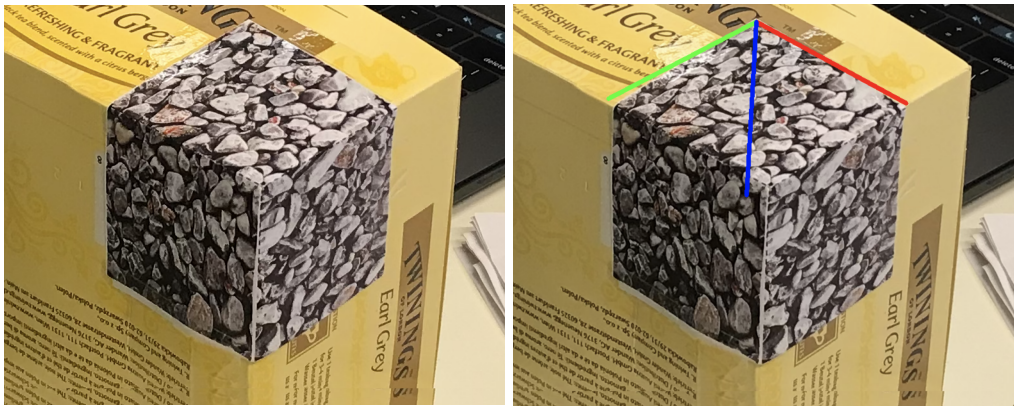

Figure 4: Printed target image bent around the x-axis and the y-axis to create a 3D structure. (right) The axes of the reference are rendered onto the image.

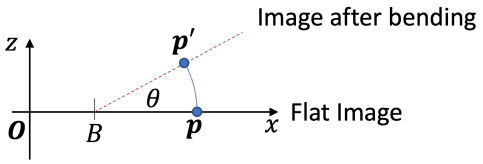

Figure 5: Diagram of the bending design when viewed from above. The printed image is bent at B by θ degrees in the positive z direction (backwards).

2D rotation around the point (B,0). Thus, its new location x' can be computed as:

$$x' = (cos(\theta)(X - B) + B, sin(\theta)(X - B) \tag{6}$$

We now have everything necessary to determine the point $\mathbf{P}_3$:

$$\mathbf{P}_3 = \begin{cases} (X, Y, 0)^{\mathrm{T}} & \text{if } X \leq B \\ (cos(\theta)(X - B) + B, Y, sin(\theta)(X - B))^{\mathrm{T}} & \text{if } X > B \end{cases} \tag{7}$$

Please note, that it is important to ensure that the points selected for the initial guess of the projection matrix during RANSAC, as well as all inliers used to compute the final projection matrix, do not lie on a single plane, otherwise the algorithm will not perform correctly.