Computer Vision Project
Winter Term 2017/2018
Franz Köferl, André Aichert, Tobias Geimer,
Vincent Christlein, Franziska Schirrmacher

Sheet 2, starting from November 6th, 2017, due December 12th, 2017

Exercise 2.1: Algebraic Estimation

a) The Direct Linear Transformation

The Direct Linear Transformation (DLT) provides an algebraic estimate for projective transformation $\mathbf{x}' \cong \mathbf{H}\mathbf{x} \in \mathbb{P}^2$. The DLT is based on the observation that two linearly dependent vectors fulfill $\mathbf{x}' \times \mathbf{x} = \mathbf{0}$. Same can be written

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & x'_3 & -x'_2 \\ -x'_3 & 0 & x'_1 \\ x'_2 & -x'_1 & 0 \end{pmatrix} \mathbf{H}\mathbf{x} \text{ with } \mathbf{H}\mathbf{x} = \begin{pmatrix} \mathbf{x}^\top \mathbf{h}^1 \\ \mathbf{x}^\top \mathbf{h}^2 \\ \mathbf{x}^\top \mathbf{h}^3 \end{pmatrix}.$$

We have

$$\mathbf{0} = \begin{pmatrix} & +x'_3\mathbf{x}^\top & -x'_2\mathbf{x}^\top \\ -x'_3\mathbf{x}^\top & & +x'_1\mathbf{x}^\top \\ x'_2\mathbf{x}^\top & -x'_1\mathbf{x}^\top & \end{pmatrix} \cdot \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} \in \mathbb{R}^{3\times 9},$$

where any row is a linear combination of the other two. For a set of matched points $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$, $i \in \{1, 2, \ldots, n\}$, we obtain a matrix $\mathbf{M} \in \mathbb{R}^{2n\times 9}$ with $(\mathbf{h}^{1\top}, \mathbf{h}^{2\top}, \mathbf{h}^{3\top})$ in its null-space. For at least four non-collinear points we can solve directly using QR or singular value decomposition ("last column of $\mathbf{V}$")[1]. Implement the function `H = dlt_homography(x, x_prime)`, such that `x_prime=H*x` . Test your function using the script `dlt_test.m` from supplementary material.

b) Planar Rectification

The points on a plane as seen from two pinhole cameras are related by a homography. The goal of this exercise is to estimate a frontal view of a building. Load the images `schloss/facade.jpg` and `schloss/photo.jpg` from supplementary meterial. Write a script to select a number n of points on an image and thus ask the user to establish at least four point correspondences. Use your implementation of DLT to estimate a homography relating the two views and map the facade in the photograph to the historic plan. (hint: in Octave use `imperspectivewarp`, MATLAB use `maketform('projective',...)` and `imtransform`).

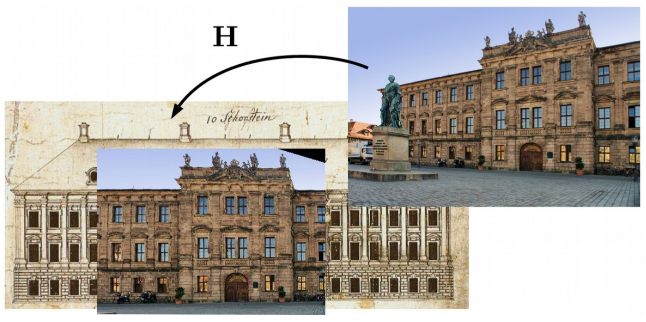

Figure 1: Mapping the facade of a historic building to an original plan for a virtual frontal view.

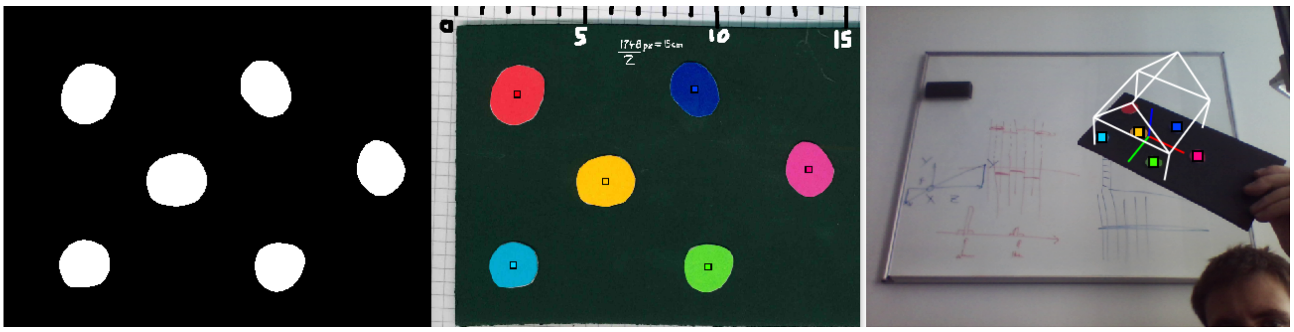
Figure 2: Mask (left) of the marker pattern (center) and an augmentation of a video frame with a wire frame house (right). Note how marker locations and average hue are visualized.

Exercise 2.2: Augmented Reality

a) Marker Detection

In this exercise we will learn how to add a virtual object to a video of a real scene ("Augmented Reality"). As a simple example, we chose a colored marker pattern, on top of which a wire-frame representation of a house shall be drawn. In the first step, the marker pattern needs to be calibrated, i.e. measured accurately. To do so, we provide a 2D scan of the pattern in the file `TargetA.png`, in which $\sim$ 5.8 pixels correspond to one millimeter. The first task is to segment the colored markers from the image by detecting bright, saturate regions.

1. An RGB image is given as a $w \times h \times 3$ tensor, where w and h are width and height of the image and the third dimension contains the RGB color. The value ("brightness") of an RGB color is defined as the maximum color channel. Compute the color value of the whole image.

2. The chromacity of an RGB color is defined as the difference between the maximum and minimum color channels. Saturation is the ratio between chromacity and value. Compute the saturation.

3. Saturation can be high for pixels with low value. However, our marker pattern has bright saturate color. Set pixels of a color value less than 25% to zero. Create a mask image by application of an appropriate threshold to the result, so that the colored markers are clearly visible. You can improve your results if you apply image filters to suppress noise, for example a Gaussian convolution or a median filter.

b) Connected Components

In this exercise we will extract the locations of colored markers. This can be done by analyzing the connected components in the mask image.

1. Apply morphological operators to the mask image to remove small components. (hint: Octave/-MATLAB `imopen`)

2. Compute the connected components (hint: Octave/MATLAB `bwlabel`). Iterate over all components and compute their center and mean RGB color. Compute the color hue of the mean RGB color as defined by the HSV color model. In Octave/MATLAB this can be done with `rgb2hsv` function.

3. Establish a coordinate system in millimeters on our marker pattern relative to its approximate center. Write a function to visualize your results on the image (for example by drawing colored dots on the center of the markers, compare Figure 2).

c) Establishing Correspondences

We provide several webcam images extracted from a video sequence showing the pattern from various positions. Load any of them (e.g. `test/cam1001.png`) and apply the same marker detection and localization as to the calibration image. In order to apply the DLT algorithm to the detected marker positions, we have to establish correspondences first.

1. The color hue should be easy to match. Iterate over the detected markers and assign the best match in color hue to each marker in the calibration pattern. If the match is ambiguous (due to spurious or incorrect matches) it is sufficient to ignore this match altogether. Only four of the six markers, no three of which may be collinear, have to be detected for a successful augmentation.
2. Apply the DLT algorithm to establish a projective mapping $\mathbf{H}$ between the maker pattern in the webcam image and calibration image.
3. Visualize the $x - y$ coordinate system of the marker in the webcam image.

d) Estimating a projection matrix

In the previous exercise we learned how to artificially add flat objects to existing photographic images in a perspectively correct manner. However, to add three-dimensional objects we need some additional information. The intrinsic parameters of the webcam are $\alpha_x = \alpha_y \approx 625$ with a principal point in the center of the image at $(320,\, 240)^\top$. We will now use our estimate of the homography of the plane to build a full projection matrix, which maps 3D points in millimeter coordinates of the marker pattern to the webcam image.

1. Assuming the marker pattern is on the $z = 0$ plane, the marker positions fulfill

   $$\mathbf{x}' \cong \mathbf{PX} = \mathbf{K}[\mathbf{Rt}](x, y, 0, 1)^\top = \mathbf{H}(x, y, 1)^\top = \mathbf{Hx},$$

   with a rotation matrix $\mathbf{R} = (r_1, r_2, r_3)$. We have

   $$\mathbf{K}(\mathbf{r}_1,\, \mathbf{r}_2,\, \mathbf{t}) = \mathbf{H}.$$

   Due to measurement errors, we obtain estimates $\mathbf{r}_1',\, \mathbf{r}_2',\, \mathbf{t}'$ via $(\mathbf{r}_1',\, \mathbf{r}_2',\, \mathbf{t}') = \mathbf{K}^{-1}\mathbf{H}$. Compute the translation vector as

   $$\mathbf{t} = \frac{2 \cdot \mathbf{t}'}{\|\mathbf{r}_1'\| + \|\mathbf{r}_2'\|}$$

   and the approximate third rotation vector $\mathbf{r}_3' = \mathbf{r}_2' \times \mathbf{r}_1'$.
2. Note that the above equations are not homogeneous. Specifically, multiplying by negative scalars flips the direction of $\mathbf{r}_3'$. To ensure that orientation is consistent in all views, make sure that $\det(\mathbf{H}) > 0$. In addition, $\mathbf{r}_1',\, \mathbf{r}_2',\, \mathbf{r}_3'$ do not necessarily form an orthogonal basis due to measurement errors. Compute the closest rotation matrix by singular value decomposition

   $$\mathbf{U\Sigma V}^\top = (\mathbf{r}_1',\, \mathbf{r}_2',\, \mathbf{r}_3') \text{ and } \mathbf{R} = \mathbf{UV}^\top,$$

   to make sure the 3D coordinate system is not distorted. Finally the projection matrix is given by multiplying $\mathbf{P} = \mathbf{K}[\mathbf{Rt}]$.
3. Verify your result by drawing the axis vectors of the 3D pattern coordinate system with 100mm length in red, green, blue for x, y, z. Try all test images. For the ones that fail, propose an improvement to the algorithm which might be able to work with the failed images.

References

[1] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2004.