Report Computer Vision Project 2

Single Michael
08-917-445

1 Binocular Stereo (Due on 28/10/2014)

Some notes about the my code:

- To run a demo showing results from task 1, run the Matlab script *main1.m*. There is a flag called *useZytglogge*. Set it to true in order to run the Zytglogge data set. Otherwise the Cow-data set will be loaded/used.
- To run a demo showing results from task 2, run the matlab scrit *main2.m*. There are two flags in this script: *shouldRunSamples* specifying whether we should use the provided R_l, t_l or R_r,t_r extrinsic parameters. The other flag - called *useLeft* - specifies what extrinsic parameters should be used. If it us true, the script uses R_l, t_l otherwise it will use R_r,t_r.

1.1 Epipolar lines Estimation (50 points)

For given two images - called left and right - both showing a certain object from different viewing positions (i.e. Multiview geometry) we were supposed to estimate the epipolar lines in both images. Epipolar lines are the intersections of epipolar plane with the image plane. Furthermore, when selecting a point from the left image, our program should be able to estimate the corresponding point in the right image.

Initially, a user has to select a certain point in the left image and then mark the corresponding position in the right image by clicking at the appropriate position.

For each click a user performs we store the corresponding image coordinates at the position where the user performed its click action and also remember from which image this click came from. We store these image coordinates as two dimensional homogenous positions. Remember that two-dimensional homogenous coordinates that represents a point in space have three components, where the last is equal one and the first two correspond to the regular cartesian coordinates of the clicked image position. A user has to mark in both images eight positions using our GUI, alternating, first in the left, then in the right image the corresponding position. The collected homogenous coordinates resulting from the selection-process in the left image are both stored in a 3×8 matrix called *left* (each column represents a homogenous point). Similarly, the collected selections in the right image are stored in a 3×8 matrix called *right*.

Using these *left* and *right* positions we compute (estimate) the fundamental matrix F by applying the *normalized eight point algorithm* on them. This algorithm is called *normalized* since all selected positions from the left image and those from the right image are centred towards their mean center which is supposed to denote the approximate image origin $(0, 0)$ (each image has its own origin).

The normalization process for a given set of points works as the following: using the (x,y) from a set of points (in our case a matrix carrying homogenous coordinates), we compute the mean position by summing all positions and dividing by the number of used points (in this exercise this is equal to eight points). Then, we shift all the given point positions towards this center by subtracting each position by their mean position.

In order to increase the numerical stability of the eight-point algorithm, we are supposed to ensure, that the mean squared distance between the origin (i.e. the mean position of the point set) and each point is equal to 2 pixels.

Thus, we also have to rescale each point in the left and in the right position collection by a factor $s = \frac{\sqrt{2}}{\bar{d}}$ where $\bar{d}$ denotes the average distance of all point distances (of the points in a certain point set) to their average position (i.e. their origin). Since we are using homogenous coordinates, we can formulate this *position normalization* as a homogenous transformation, capturing a translation towards the average position, and a rescaling such that the mean squared distance to the origin is equal to two pixels. Formally, such a transformation looks like the following:

$$T = \begin{pmatrix} s & 0 & -c_1 s \\ 0 & s & -c_2 s \\ 0 & 0 & 1 \end{pmatrix}$$

Where the scaling factor s is the distance scaling (such that the mean squared distance of each position to the origin is equal two pixels) and $(-sc_1, -sc_2)$ is the translation to the origin.

For both point sets, *left* and *right*, we have to compute such a homogenous transformation matrix. Then we have to apply these transformations to their points. We denote the normalization transformation for the left point set as T_l and for the right point set as T_r.

Next, let us consider the two points $\mathbf{x} = (u, v, 1)^T$ and $\mathbf{x'} = (u', v', 1)^T$ where $\mathbf{x}$ is a normalized point from the left image (using T_l) and $\mathbf{x'}$ a normalized point from the right image.

Given the following epipolar constraint

$$\mathbf{x'}^T F \mathbf{x} = 0 \tag{1}$$

Where F denotes the (yet) unknown Fundamental matrix we are looking for. By solving the minmizing problem

$$\min_{\mathbf{F}} \sum_{k=1}^{N} \left(\mathbf{x'}_k^T \mathbf{F} \mathbf{x}_k \right)^2 s.t. ||\mathbf{F}||^2 = 1 \tag{2}$$

We we can therefore retrieve F.
In the following I describe some further steps in detail how to compute the fundamental matrix F using above's insights.

We start by explicitly showing all components in the epipolar constraint from equation 1:

$$\begin{pmatrix} u' & v' & 1 \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = 0 \tag{3}$$

which is the same as

$$\begin{pmatrix} u'u & u'v & u' & v'u & v'v & v' & u & v & 1 \end{pmatrix} \cdot \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

Let us define

$$Y = \begin{pmatrix} u'u & u'v & u' & v'u & v'v & v' & u & v & 1 \end{pmatrix} \tag{4}$$

and

$$f = \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} \tag{5}$$

thus, we solving

$$Y \cdot f = 0 \tag{6}$$

will solve the equation 2. Such a system of equations can usually be solved by using a SVD docompositon of the system. We choose f as the left singular vector that corresponds to the smallest singular value of Y. Thus let us apply the singular value decomposition on Y:

$$Y = U\Sigma V^T \tag{7}$$

Where U,V are orthonormal (rotation) matrices (i.e. $UU^T = I = U^T U$ same for V), and Σ is a diagonal matrix containing all singular values of Y in an ascending order. This means, the highest singular values is the first diagonal element, the smalles singular value is the last diagonal element of Σ.

Since we are interested in the left singular vector that corresponds to the the smallest singular values in Σ we have to take the last column of U. Therefore we assign f by the last column of U, i.e.

$$f = \begin{pmatrix} U_{19} \\ U_{29} \\ U_{39} \\ U_{49} \\ U_{59} \\ U_{69} \\ U_{79} \\ U_{89} \\ U_{99} \end{pmatrix} \tag{8}$$

Going back to the definition from equation 3 of the fundamental matrix F we get

$$\begin{pmatrix} U_{19} & U_{29} & U_{39} \\ U_{49} & U_{59} & U_{69} \\ U_{79} & U_{89} & U_{19} \end{pmatrix} \tag{9}$$

Next we have to enforce the rank-2 constraint by setting the smallest singular value of F's SVD to zero. Formally, this corresponds to decompose F into its SVD components, set the last component of its matrix Σ to zero and compute

$$F_{rank=2} = U\Sigma' V^T \tag{10}$$

where Σ denotes the modified 3×3 singular value diagonal matrix Σ with the last singular value set to zero.

the last remaining step is to transform the fundamental matrix $F_{rank=2}$ back to original units. This can be achieved by multiplying the point-set normalization matrices T_l and T_r to $F_{rank=2}$:

$$F = T_r^T F_{rank=2} T_l \tag{11}$$

This final transformation undoes all previous unit transformations (resulted from transforming into left image, right image positions). The matrix F from equation 11 denotes the Fundamental matrix fulfilling the rank two constraint used in order to estimate the epiplar geometry.

Next, we use F and apply the following epipolar constraints to it:

$$Fe = 0$$
$$F^T e' = 0$$

Where e is the epipole in the left image and e' denotes the epipole in the right image. We can solve for e and e' by computing the kernel of the matrix F and F^T respectively. The cross product between the left epipole and any point in the left image gives us the the coefficients of a general line in a two dimensional space. This will us allow us to fit a line through the left image.

Mathematically, this corresponds to:

$$(a, b, c) = e \times p_{left} \tag{12}$$

and then we have to evaluate the line general line quation

$$ax + by + c = 0 \tag{13}$$

We solve this equation for y using $x = [0, lengt(Image_{left})]$.

The corresponding epipole line on the right image can be obtained by applying the fundamental matrix to the point selected in the left image. This gives us a vector with three components. We again use these three components as line coefficients for a line though space.

In the following I demonstrate some sample results produced by my Matlab code:

- The figures figure 1 (Cow) and figure 2 (Zytglogge) visualize the feature points selected by a user. Each feature point selected on the left image should correspond to one point in the right image.
- The figure 1 shows the estimated epipolar line in the Cow image using the user specified image shown earlier and a new user selection in the left image (indicated as a cyan dot on the left red epipolar line). The blue line corresponds to the epipolar line on the right image. Similar plots of the epipolar line using the Zytglogge data-set are shown in the figures figure 4, 5 and 6.
- Last we visualize the numerical plots of the epipoles of both our data-sets. Figure 7 shows the epipoles of the Cow data-set and figure 8 shows the epipoles of the Zytglogge data-set.

1.2 Model reconstruction (50 points)

From given (large) set of matching points of both cow-images, left and right, we want to reconstruct its corresponding 3d geometry by relying on the eight-point algorithm. Our goal is to estimate the depth of the given matching points.

In this task we are only provided by the intrinsic parameter K which is describing the camera calibration.

We first estimate the fundamental matrix F, using all matching points from the provided data-set. Note that there are more than eight given points that we pass to our eight-point algorithm.

This gives us the following matrix for F:

$$F = \begin{pmatrix} -0.0000 & -0.0000 & 0.0075 \\ -0.0000 & 0.0000 & 0.0183 \\ 0.0001 & -0.0137 & -1.9810 \end{pmatrix} \tag{14}$$

For solving this task we were provided by the camera calibration matrix K. Using the estimated Fundamental matrix F we can compute the essential matrix E.

$$F = K'^{-T} E K^{-1} \tag{15}$$

In general, the essential matrix gives us the relative rotation and translation between the cameras of their extrinsic parameter. An illustration of this definition is shown in figure 9.

The provided matching points from both images were taken by the same camera. Thus, for our case K is equal to K'. Thus, we simply can solve for E in equation 15 which is thus equal to

$$E = K^T F K \tag{16}$$

When solving for the essential matrix in this task we get:

$$E = \begin{pmatrix} -0.0005 & -0.1340 & -0.2203 \\ -0.0025 & 0.0188 & -1.5708 \\ 0.0013 & 1.4839 & 0.0002 \end{pmatrix} \tag{17}$$

In the following let x denote a point from the left image and x' a point from the right image. From the class we know that the essential matrix is formed by a translation $[t]_x$ (that is expressed as a skew symmetric matrix and a rotation matrix R.

$$E = [t]_x R \tag{18}$$

Figure 9 again shows the epipolar constraint for the calibrated case and illustrated the relationship of the matrices R and the translation vector t.

$$x' \cdot (t \times Rx) \rightarrow x' E x = 0 \tag{19}$$

Thus, under ideal conditions, we can conclude, that the essential matrix E is singular. Hence, we are going to compute the singular value decomposition (SVD) of the essential matrix:

$$E = U \Sigma V^T \tag{20}$$

In summary, we found that for the essential matrix (under ideal conditions) the following identities hold true:

$$E = [t]_x R = U \Sigma V^T \tag{21}$$

Under ideal conditions, the singular values stored in the 3×3 diagonal matrix Σ are the same. However, in practise, the singular values differ due to various reasons such as numerical issues, not perfect calibrations and many more. Nevertheless, according to the rank-2 constraint on F, the last component singular value is equal to zero.

The diagonal entries of $\mathbf{\Sigma}$ are the singular values of $\mathbf{E}$ which, according to the internal constraints of the essential matrix, must consist of two identical and one zero value. Therefore we define

$$R_{90} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{22}$$

with

$$R_{90}^{-1} = R_{90}^{T} \tag{23}$$

and make the following ansatz:

$$[t]_x = V R_{90} \Sigma V^T \tag{24}$$
$$R = U R_{90}^{-1} V^T \tag{25}$$

Instead of using the provided definition of $[t]_x$ we can alter it to

$$[t]_x = \begin{pmatrix} U_{13} \\ U_{23} \\ U_{33} \end{pmatrix} \tag{26}$$

Where we use the last column of us (which is paired with the smallest singular value, when multiplying the decomposition out) U is the one component of the essential matrix' SVD. Thus, in short t corresponds to the last column of U up to a certain scaling.

Using the ansatz from equation 25 we just get one particular candidate solution. However there are more. Actually there exist four possible rotation transformation matrices and two possible translation matrices, i.e.

$$R = \pm U R_{90}^{\pm 1} V^T \tag{27}$$

$$t = \pm \begin{pmatrix} U_{13} \\ U_{23} \\ U_{33} \end{pmatrix} \tag{28}$$

The cartesian product between R and t gives us the total number of candidate transformation solutions we have to examine. This is equal to eight candidate solutions.

It is possible to reduce the solution space we have to take into account to. We only take rotation transformation matrices that have a determinate equal to one (i.e. they exhibit a positive rotation orientation). Remember that the determinant of a matrix describing a rotation is always either equal one or equal to minus one. Considering the possible sign and transpose combinations we see that there will always two rotation matrices having a determinant equal to one and the other two equal to minus one.

For the remaining solution candidates, we reconstruct their depth z - following the procedure as described below and take the rotation-translation combination exhibiting the most points lying in front of the camera (i.e. having a positive z-value/depth value).

Last we want to reconstruct 3d points (x, y, z) from corresponding image points. More precisely, we want to compute the 3d points given corresponding normalized image coordinates

$$\mathbf{p}^l = (p_x^l, p_y^l) \tag{29}$$

from the left image and image coordinates from the right image

$$\mathbf{p}^r = (p_x^r, p_y^r) \tag{30}$$

Assuming that the essential matrix E is known and the corresponding rotation R and translation t transformations have been determined, we can apply the

algorithm from Longuet-Higgins's Paper.

In the following let r_k denote the k-th row of the rotation transformation R.

Combining the above relations between 3D coordinates in the two coordinate systems and the mapping between 3D and 2D points described earlier gives

$$p_x^r = \frac{x}{z} \tag{31}$$

$$= \frac{\mathbf{r}_1 \cdot (\mathbf{p}^l - \mathbf{t}/z)}{\mathbf{r}_3 \cdot (\mathbf{p}^l - \mathbf{t}/z)} \tag{32}$$

Reformulating equation **??** we can solve for the height component z from the 3d points as the following:

$$z = \frac{(\mathbf{r}_1 - p_x^r \, \mathbf{r}_3) \cdot \mathbf{t}}{(\mathbf{r}_1 - p_x^r \, \mathbf{r}_3) \cdot \mathbf{p}^l} \tag{33}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = z \cdot \mathbf{p}^l \tag{34}$$

In practise we also have to take the focal length f of the camera in order to perform the computations stated in equation 34. Thus in practise we compute the following:

1. $\forall \mathbf{p}^l, \mathbf{p}^r : z = \frac{(\mathbf{r}_1 - p_x^r \, \mathbf{r}_3) \cdot \mathbf{t}}{(\mathbf{r}_1 - p_x^r \, \mathbf{r}_3) \cdot \mathbf{p}^l}$
2. $\forall \mathbf{p}^l : (x, y) = \frac{z \cdot \mathbf{p}^l}{f}$

The resulting, reconstructed 3d points (x, y, z) are then plotted using Matlab's scatter3 function that allows us to illustrate a reconstructed image as a point cloud (without the need of performing a mesh triangulation).

In the following I am going to show some reconstruction images using the method described in this report, when running my code. I also compare my result to those one would get when knowing the exact rotation and translation matrices. These matrices are given as extrinsic parameters in order to compare the results.

The figures figure 10, figure 11, figure 12 and figure 13 show the reconstructed 3d points using our estimated rotation translation matrix R and estimated translation vector t. Figure 14 shows the reconstructed 3d points using the given R_l and t_l data. Last, figure 15 shows the reconstructed 3d points we get when using the given R_r and t_r data.

We observe that when we are using R_r and t_r for the 3d point reconstruction we obtain a cow that has almost no depth, i.e. this model is pretty flat. In contrary, when we are using the R_l and t_l data for the reconstruction process, we get an image which has appropriate depth values. This mean using R_l and t_l for the 3d reconstruction gives us a good looking 3d model having viable, visually convincing depth values. Applying our estimated R and t data gives us a reconstruction that produces depth values that are somewhere in between the both previous results. Using our data, the reconstructed model does not look that flat as when we were using R_r and t_r. However, our reconstructed 3d points also do not exhibit depth details as when we were using R_l and t_l. In my honest opinion, our reconstruction looks closer to the flat results when using R_r and t_r.

Potential causes for this not that well looking results, when using our data:

- Idea 1: The singular were differing too much from each other. The estimation approach we are using in order to reconstruct R assumes, that the singular values of the svd of the fundamental matrix are the same. Thus, when having a different Σ sigma matrix as it was supposed to be there (having the same singular value twice (last singular values is zero - rank 2 constraint)). This also will affect the U and V matrices of the SVD of F, which we are using in order to compute R and V. One possible solution to this problem would be some kind of rescaling or we could overwrite the 2nd singular value by the largest one. I have not tried this out - this is just an idea.
- there is a bug present: Even this is possible I do not think this is the result for this kind of difference. At least I have verified that I have computed matrices that describe a rotation (all their determinant is equal to ± 1).

- Maybe the eight point algorithm is not sufficient enough in order to produce viable results. Even we use normalized points, we maybe should rely on a non-linear approach.
- Maybe there is noise present. However, I exclude this possibility, since then, in my opinion, none of our reconstructions should be able to produce nice depths.
- Maybe there is some kind of normalization issue still present, regarding the image normalization.
- On wikipedia there is also another method described in order to reconstruct the depth $z = \frac{(\mathbf{r}_2 - y_2' \mathbf{r}_3) \cdot \mathbf{t}}{(\mathbf{r}_2 - y_2' \mathbf{r}_3) \cdot \mathbf{y}}$. However, when I tried this out I did not get better looking results.

In general, the eight-point algorithm is very sensitive to the presence of noise. This means if there was noise to the correspondence, the results would look very different from those we obtained. Furthermore, when using de-normalized matching-points, the eight point algorithm would also perform rather bad due to numerical instabilities.

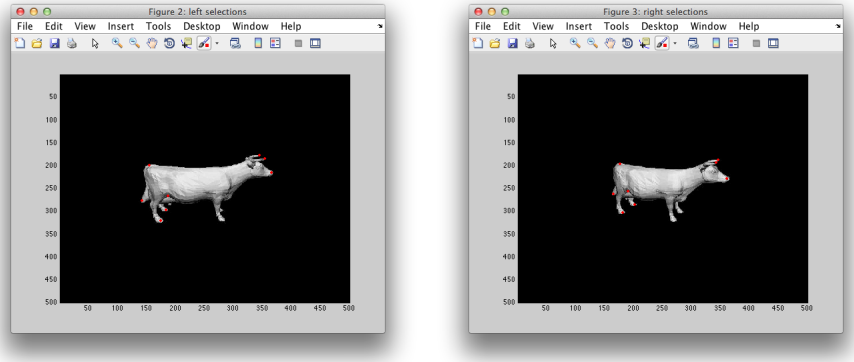

Fig. 1: Showing the eight user specified points in both cow images, the left image (on the left) and the right image (on the right.

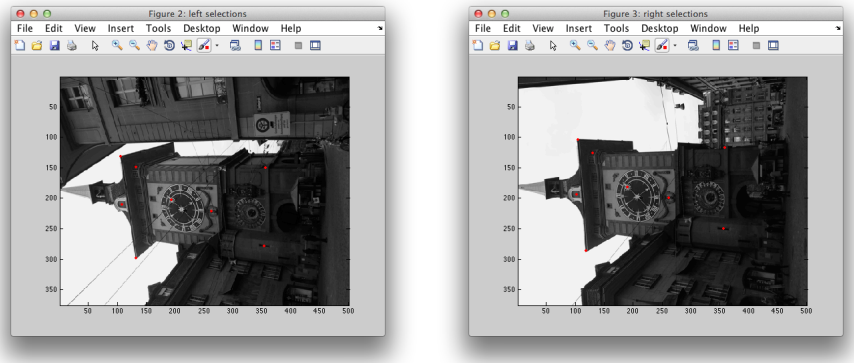

Fig. 2: Showing the eight user specified points in both Zytglogge images, the left image (on the left) and the right image (on the right.

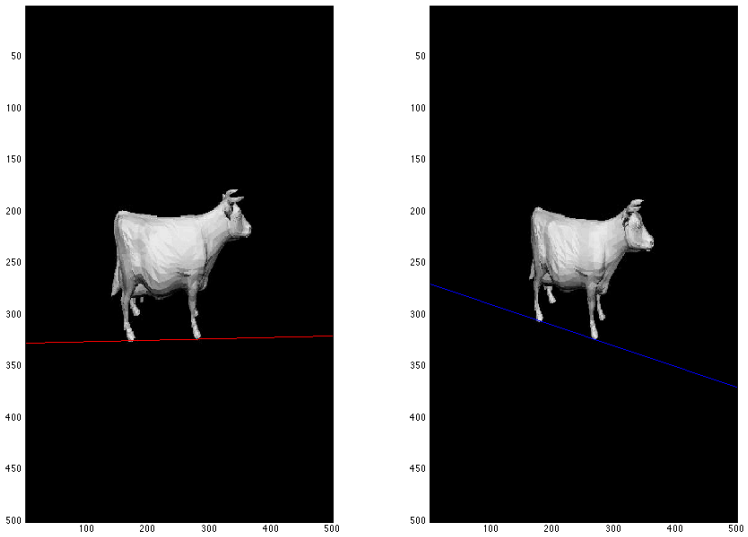

Fig. 3: Visualization of the estimated Epipolar Lines in both Cow images (left,red and right,blue) using a certain user selection which is indicated by a cyan colored little dot (you probably have to zoom-into the left image in order to see this dot) the left image.

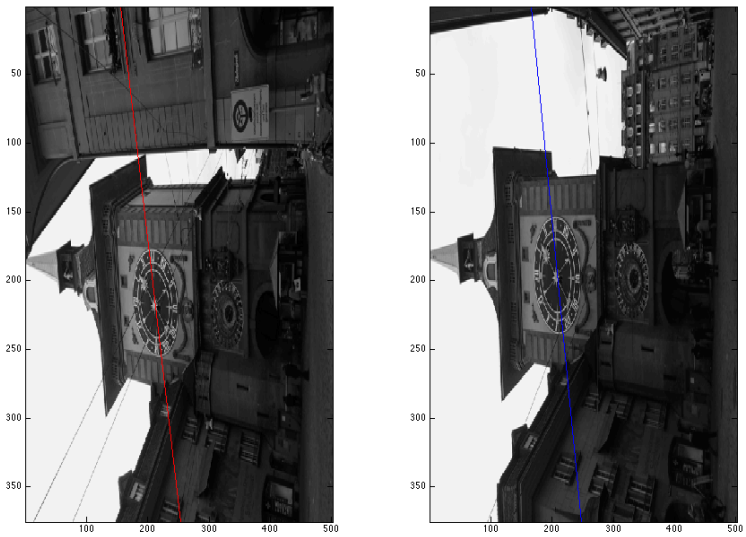

Fig. 4: Visualization of the estimated Epipolar Lines in both Zytglogge images (left,red and right,blue) using a certain user selection which is indicated by a cyan colored little dot (you probably have to zoom-into the left image in order to see this dot) the left image.

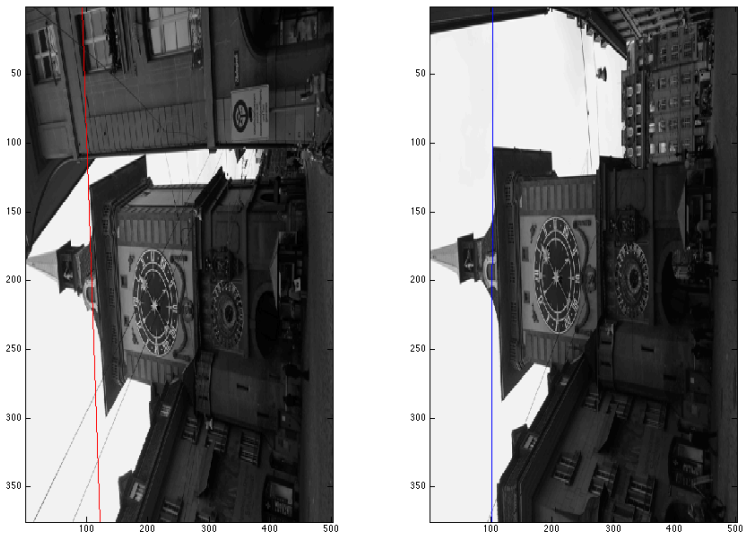

Fig. 5: Visualization of the estimated Epipolar Lines in both Zytglogge images (left,red and right,blue) using a certain user selection which is indicated by a cyan colored little dot (you probably have to zoom-into the left image in order to see this dot) the left image.

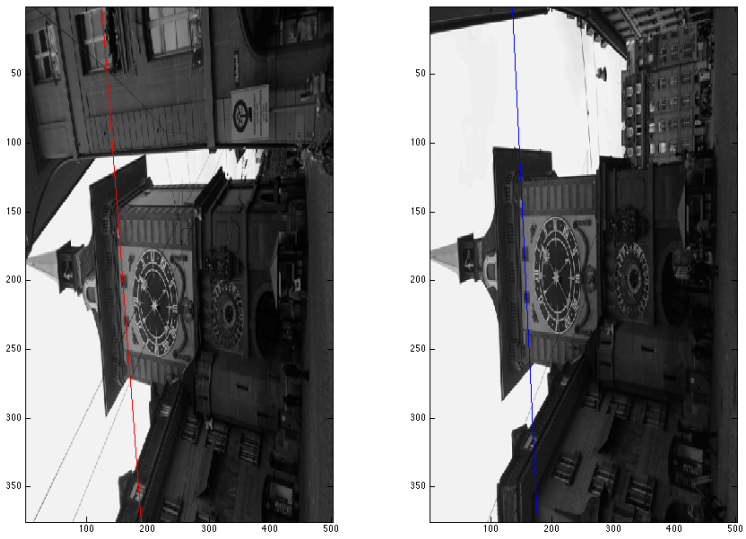

Fig. 6: Visualization of the estimated Epipolar Lines in both Zytglogge images (left,red and right,blue) using a certain user selection which is indicated by a cyan colored little dot (you probably have to zoom-into the left image in order to see this dot) the left image.

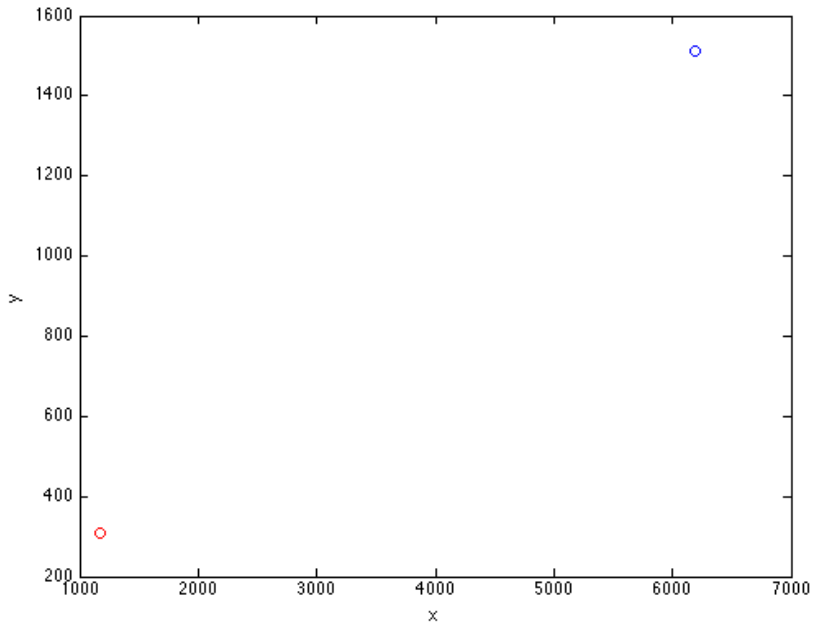

Fig. 7: Visualization of the coordinates of the epipoles of the Cow data-set. The red circle corresponds to the epipole in the left image, whereas the blue circle corresponds to the epipole in the right image.

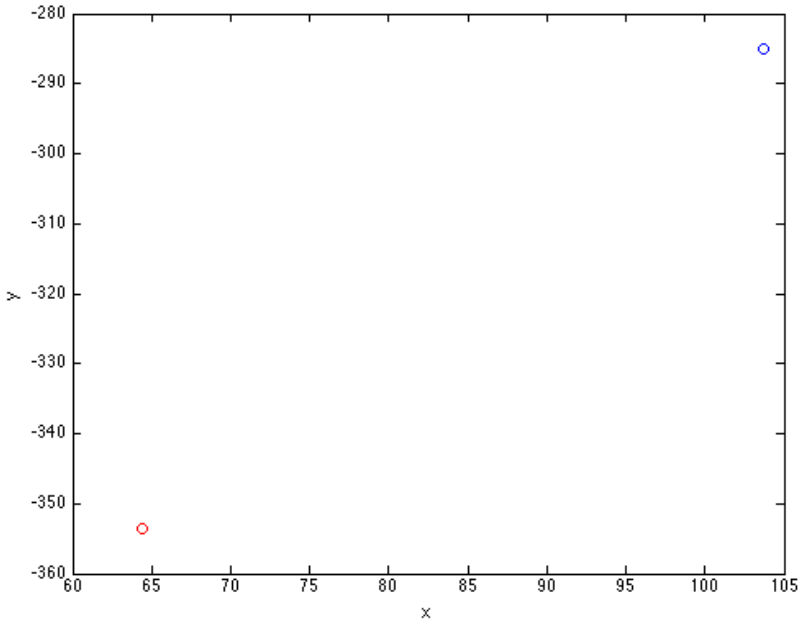

Fig. 8: Visualization of the coordinates of the epipoles of the Zytglogge data-set. The red circle corresponds to the epipole in the left image, whereas the blue circle corresponds to the epipole in the right image.

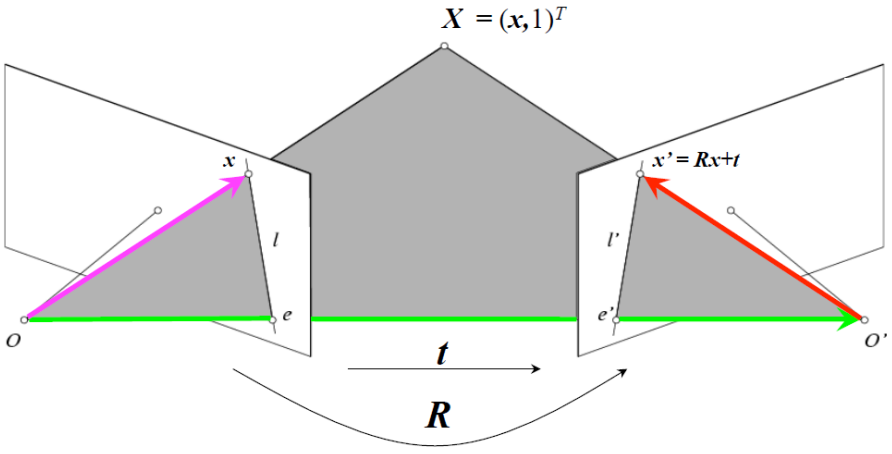

Fig. 9: Illustration of the epipolar constraints for the calibrated case: The vectors Rx, t and x' are coplanar.

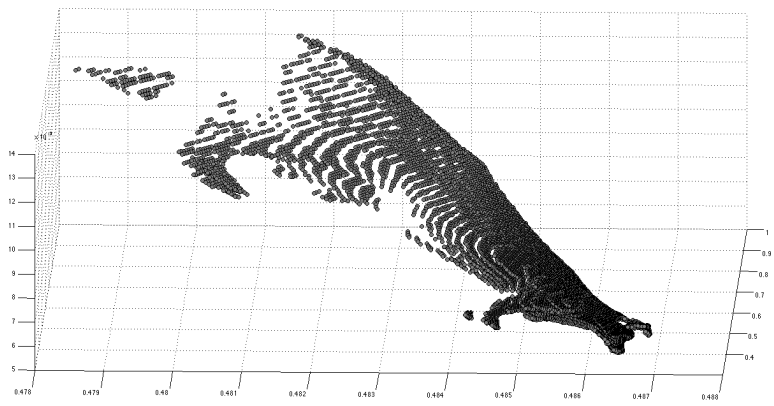

Fig. 10: Visualization of the reconstructed 3d points (point-cloud) of the Cow model using our estimated translation and rotation transformation matrices. Viewing diagonally, from the size

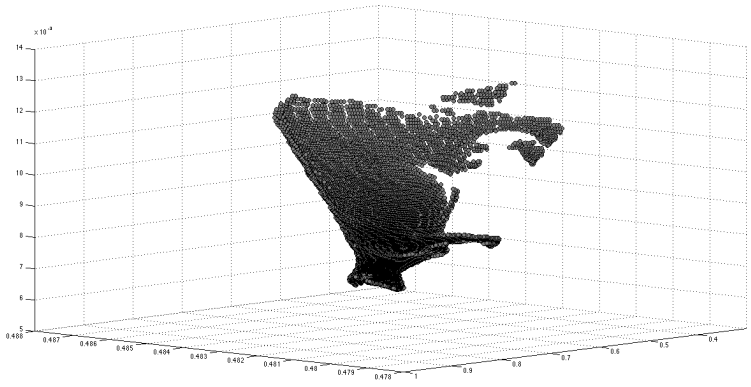

Fig. 11: Visualization of the reconstructed 3d points (point-cloud) of the Cow model using our estimated translation and rotation transformation matrices. Viewing from the side, head downwards.

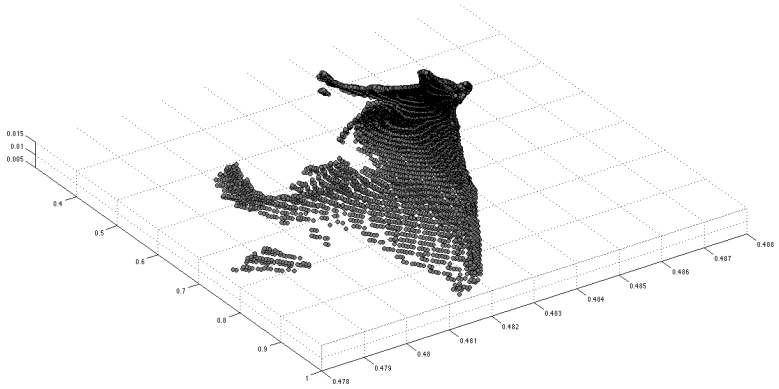

Fig. 12: Visualization of the reconstructed 3d points (point-cloud) of the Cow model using our estimated translation and rotation transformation matrices. Viewing from top the side profile of the cow.

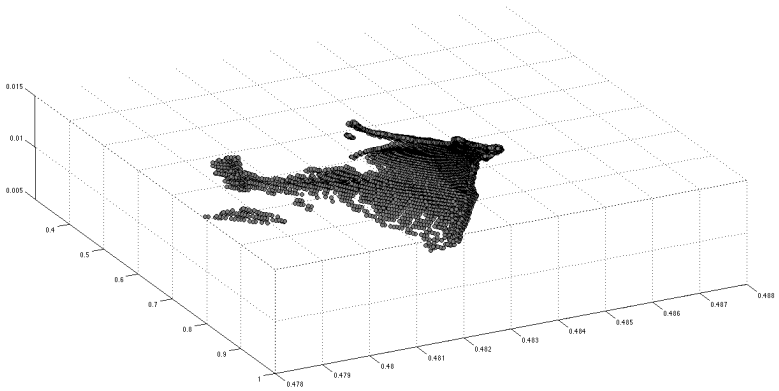

Fig. 13: Visualization of the reconstructed 3d points (point-cloud) of the Cow model using our estimated translation and rotation transformation matrices. Viewing from top the side profile of the cow using a narrow angle in order to better recognize the width of the cow.

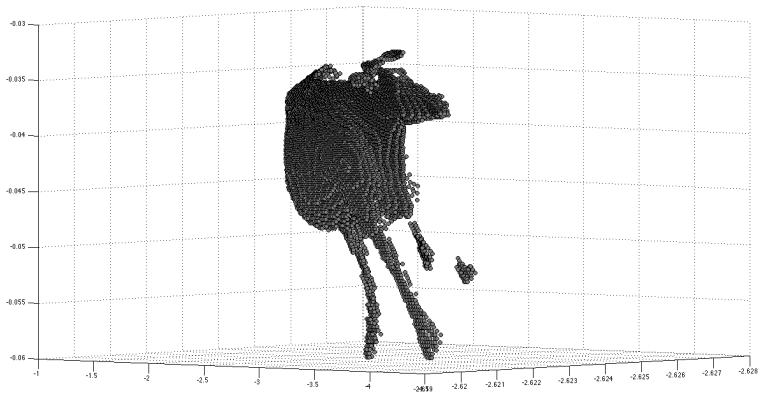

Fig. 14: Visualization of the reconstructed 3d points (point-cloud) of the Cow model using the provided R_l and t_l

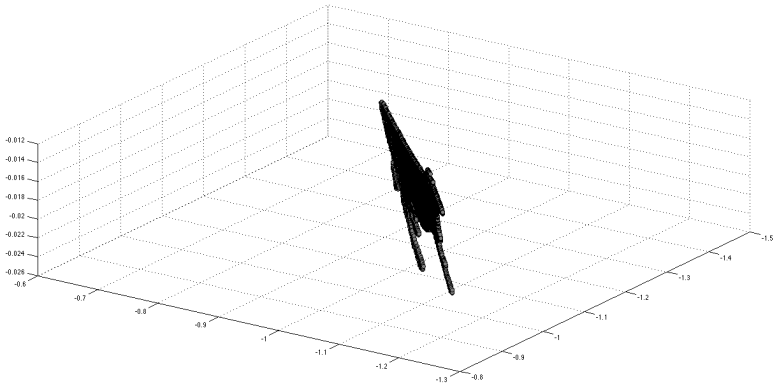

Fig. 15: Visualization of the reconstructed 3d points (point-cloud) of the Cow model using the provided R_r and t_r