Technische Universität München
Fakultät für Informatik
Martin Groher, Daniel Pustka,
Marco Feuerstein

WS 2005
Solution for Exercise Sheet 3
Dec 08 2005

Exercises in 3D Computer Vision

Exercise 1 Mosaicing (Exemplary Solution)

a) One way - the easiest - of acquiring point correspondences is the manual approach: Load the images into a graphic program, e.g. `gimp` and determine the pixel coordinates of the image points in both images. The point correspondences can also be determined in MatLab directly using the function `ginput`.
However, sometimes one gets better results when many point correspondences are used, but determining, say, 150 point correspondences manually would require quite some work. Thus, point correspondences can also be determined automatically by using the intensity properties of an image. For that the corners in one image are determined. These are points with neighbouring pixels representing two strong edges, i.e. two edges with a high intensity strength. For each corner in image 1 a match in image 2 is determined by examining the pixels in the neighbourhood. This examination is performed by taking the 8 surrounding pixels and measure the similarity of any 8-pixel square in the second image. The square with the highest similarity is the neighbourhood of the corner in the second image. The similairty can, for instance, be measured with the "Sum of Squared Differences" (SSD) of the intensity values of the pixels. The smaller the SSD the more similar the neighbourhood. Keep in mind that such automatic methods for acquiring point correspondences can lead to point mismatches (outliers) that have to be either eliminated in advance or considered in the homography algorithm. Thus, robust estimation algorithms such as RANSAC are recommended when doing automatic point acquaintance.

b) Changing of coordinate systems or scaling can be encapsulated in a similarity transformation T. Applying them to the image also applies them to the points of the point correspondences $\{x'_i \leftrightarrow x_i\}$. Let $\tilde{x}'_i = T'x_i$ be the transformed points of image 1 and $\tilde{x}_i = Tx_i$ the transformed points of image 2. Let H be the 2D homography before the transformation, i.e. $x'_i = Hx_i$, and let $\tilde{H}$ be the homography after the transformation, i.e. $\tilde{x}'_i = \tilde{H}\tilde{x}_i$, defined by $\tilde{H} = T'HT^{-1}$. Now, consider the basic equation in the DLT algorithm and apply the transformations to it:

$$\begin{aligned} \tilde{x}'_i \times \tilde{H}\tilde{x}_i &= T'x'_i \times (T'HT^{-1})Tx_i \\ &= T'x'_i \times T'Hx_i \\ &= T'^{*}(x'_i \times Hx_i) \end{aligned}$$

T'^{*} is the cofactor matrix of T' and defined as $T'^{*} = det(T)T^{-T}$ if T is invertible. Any 3×3 cofactor matrix M has the property that for any given 3-vectors x, y

$$Mx \times My = M(x \times y)$$

If T' is a similarity transformation, then $T' = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$. Now, $T'^{-1} = \begin{bmatrix} \frac{1}{s}R^T & -\frac{1}{s}R^T t \\ 0^T & 1 \end{bmatrix}$ up to scale. The determinant of T' is $s^2 det(R) = s^2$ since R is orthogonal and a rotation

matrix, so $det(R) = 1$. Hence, $T'^* = s \begin{bmatrix} R & 0 \\ -t^T R & s \end{bmatrix}$. We consider now the matrix-vector notation of the DLT equation ($x_i' = Hx_i$), $A_i h = 0$. Note that A_i just consists of two rows, since the third one is linearly dependent to the first two. If we apply T'^* to this notation, we see that

$$\tilde{A}_i \tilde{h} = T'^*_{|3,3} A_i h = sRA_i h,$$

where $T'^*_{|3,3}$ is T'^* with only the first two rows and columns, i.e. $T'^*_{|3,3} = sR$. The rotation R does not affect vector norms and thus $||\tilde{A}\tilde{h}|| = s||Ah||$. Hence, the norm remains the same up to scale and minimizing the algebraic error as required in the DLT algorithm is the same before and after transformations T, T'.
However, the DLT algorithm has to minimized this algebraic error obeying a constraint, namely $||H|| = 1$. This constraint is **not** related in a simple manner before and after transformations T, T'! To put it in a nutshell,

$$\begin{aligned} & \text{minimize} \quad \sum_i ||A_i h||^2 \quad \text{subject to} \quad ||H|| = 1 \\ \Leftrightarrow \quad & \text{minimize} \quad \sum_i ||\tilde{A}_i \tilde{h}||^2 \quad \text{subject to} \quad ||H|| = 1 \\ \not\Leftrightarrow \quad & \text{minimize} \quad \sum_i ||\tilde{A}_i \tilde{h}||^2 \quad \text{subject to} \quad ||\tilde{H}|| = 1 \end{aligned}$$

To improve the outcome of the DLT algorithm one needs to undo the changes made by different coordinate systems or scaling. This can be achieved by transforming all images such that they have equal scaling and coordinate origin. This is done as follows:

(i) Compute a similarity transformation T (translation and scaling) that takes points x_i to a new set of points $\tilde{\vec{x}}_i$ such that the centroid of points $\tilde{\vec{x}}_i$ is the coordinate origin $(0,0)^T$, and their average distance from the origin is $\sqrt{2}$.

(ii) Compute a similar transformation T' for the point set $\vec{x}_i'$ to $\tilde{\vec{x}}_i'$.

(iii) Apply the DLT algorithm to point correspondences $\{\tilde{\vec{x}}_i' \leftrightarrow \tilde{\vec{x}}_i\}$ to obtain homography $\tilde{H}$.

(iv) Undo the normalization to obtain $H = T'^{-1} \tilde{H} T$.

Note that here, the transformations T, T' are performed before the calculation of a homography such that the coordinate systems of both images indeed do have the same origin and same scaling and the non-invariance to these problems is ommitted.

c) Given is a transformation H and an Image I. We want to apply the transformation to I. Intuitively, one would just apply the transformation to each pixel $p(i,j) \in \mathbb{N}^{1\times 1}$ to get the new image I', $p'(i,j) = Hp(i,j)$. This is called a **forward warping** and has the disadvantage that wholes could arise in the warped image I'. For instance, the scaling-transformation

$$H = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

on a 640×480-pixel image I would warp to an image I' of size 1280×960 pixel. Thus, new pixels would be added, which do not have any color value. Hence, the gaps in the image.[1]
In order to avoid those gaps, the **backward warping** approach is used. Here, a forward warping H is applied only to the corners of the image to determine the bounding box of the new image I'. Then, the inverse homography is applied to each pixel in this bounding box,

[1] Those gaps could be deleted if all pixels would be "pumped up", but this would create artifacts. Moreover, the gaps could be filled via interpolation.

i.e. to the image I', $I = H^{-1}I'$. Thus, for every pixel in the target image I' the appropriate pixel in the source image I is found.
Since the pixel coordinates are discrete and thus the color values are just at discrete locations, and the inverse transform will generally map pixel coordinates to real points one has to perform an interpolation on the color values (red, green, and blue separately) of the pixels p in order to get the values at pixels p'.

d) The Matlab code for the mosaicing can be viewed in `/u/halle/groher/home_sun/3DComputerVision/MatLab/exercise05/solution/`. The function dlt.m does a direct linear transformation on 2D/2D point correspondences, normalizedlt.m implements the normalized DLT algorithm. Homography.m is a test script that executes the algorithm and warps the images.

Exercise 2 Normalized DLT for Image Mosaicing (Exemplary Solution)

see `/u/halle/groher/home_sun/3DComputerVision/MatLab/exercise05/`.

Exercise 3 Non-linear estimation methods (Exemplary Solution)

a) Refer to `JacobianReprojection.pdf` on the 3D Computer Vision website.

b) The Matlab code for the mosaicing can be viewed in `/u/halle/groher/home_sun/3DComputerVision/MatLab/exercise05/solution/`. The file GoldStandard.m contains the ML estimate and the error function (reprojection error). Again, Homography.m will test the code when adjusted slightly.

Exercise 4 Error Propagation (Exemplary Solution)

a) Mean of $||\vec{x}||$:

$$\overline{||\vec{x}||} = ||\vec{\bar{x}}|| = \sqrt{3^2+4^2} = 5$$

Standard Deviation:

$$grad(||\vec{x}||) = \begin{pmatrix} \frac{x}{\sqrt{x^2+y^2}} \\ \frac{y}{\sqrt{x^2+y^2}} \end{pmatrix}$$

The variance (forward propagated) of $||\vec{x}||$ is

$$C_N = grad^T_{||\vec{x}||}(\vec{\bar{x}}) * C * grad_{||\vec{x}||}(\vec{\bar{x}}) = \frac{1}{25}(3,4)\begin{pmatrix} 2 & 0 \\ 0 & 9 \end{pmatrix}\begin{pmatrix} 3 \\ 4 \end{pmatrix} = 162/25$$

Thus, the standard deviation is $\frac{1}{5}\sqrt{162}$.

b) This exercise is analogous but with a vector-valued function. The only difference is that the forward propagation of the covariance is calculated via the Jacobian (first derivative of a vector-valued function), and not the gradient.