IACV Homework

Emanuele Ghelfi

January 19, 2018

Contents

Figure 1: Input Image

1 Introduction

A bandoneon is a musical instrument, consisting in two rigid wooden parts connected by a deformable bellow. The assignment is to reconstruct the bandoneon shape from a single image of it, using additional information.

2 Line Extraction

Use the learned techniques to find edges and lines in the image. Then manually select those lines which are useful for the subsequent steps.
The first step is image feature extraction and selection. Lines are extracted using the following steps:

- Edge detection with Canny
- Line detection using Hough

2.1 Canny

The canny algorithm is based on 3 steps:

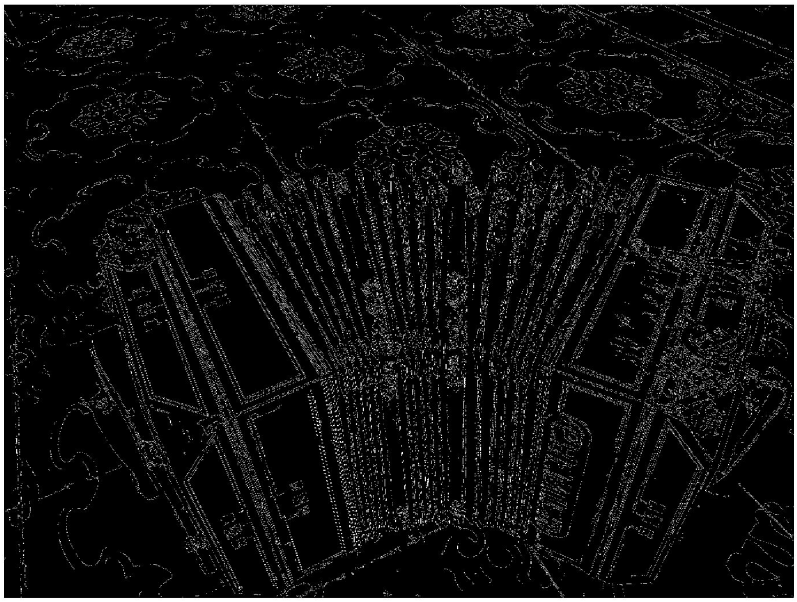


Figure 2: Edge detection through Canny

- Convolution with derivative of Gaussian before computing image derivatives
- Non-maximum Suppression
- Hysteresis Thresholding

The threshold parameters found for Canny are the result of many experiments. The threshold is important since too many edges cause the Hough transform to fail considering useless lines.

The processed image with extracted edges is shown in figure 2.

2.2 Hough Transform

The Hough transform is designed to detect lines, using the parametric representation of a line:

$$rho = x * cos(\theta) + y * sin(\theta)$$

Each individual datum (edge) votes for all the model compatible with him ($f(m, xi) = 0$).

Steps:

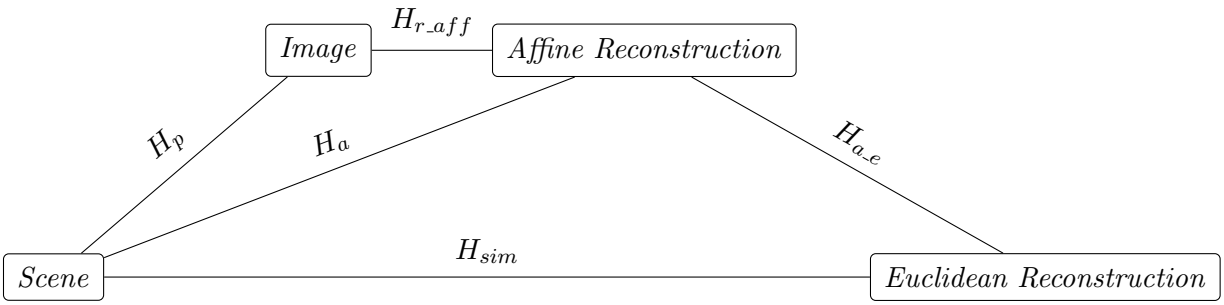
Figure 3: Line detection through Hough

- Discretize model space. Set the number of votes for each model = 0
- For each datum computes the hough transform H(xi)
- Let xi votes for each cell of the hough space crossed by H.
- Selects the local maxima in the hough space
- Apply a threshold to the number of votes.

The result of the application of the Hough Transform for line detection is shown in figure 3.

3 Shape Reconstruction

Using constraints on the horizontal lines, and their images, reconstruct the shape of the horizontal faces, and determine their relative position and orientation.
I used a stratified approach to the shape reconstruction problem. The idea is to pass through two transformations, then the reconstructive transformation is the composition of the two transformations.



The first step is to compute H_{r_aff} that maps the image to an affine reconstruction with respect to the real scene. Then, from the affine reconstruction compute the mapping H_{a_e} that maps the affinity to a euclidean reconstruction with respect to the real scene.

3.1 Affine Rectification

An affine transformation is a non-singular linear transformation followed by a translation. Its matrix representation is the following:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1}$$

Because an affine transformation includes non-isotropic scaling, the similarity invariants of length ratios and angles between lines are not preserved under an affinity. Invariants of an affinity are:

- Parallel lines. An affine transformation maps points at infinity to points at infinity. Consequently, the parallel lines are mapped to lines which still intersect at infinity, and so are parallel after the transformation.
- Ratio of lengths of parallel segments
- Ratio of ares

In order to perform affine rectification we require that the line at infinite in the image is mapped back to itself ($l_\infty = (0, 0, 1)^T$). So we first perform the identification of the imaged line at infinite through LSA using n couples of imaged parallel lines. Once found the image of the line at infinite the reconstruction matrix that rectifies the image it's simply:

$$H_{r\_aff} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{bmatrix} \tag{2}$$

So the last row is the imaged line at infinite [pag 49 Multiple View Geometry in computer vision].

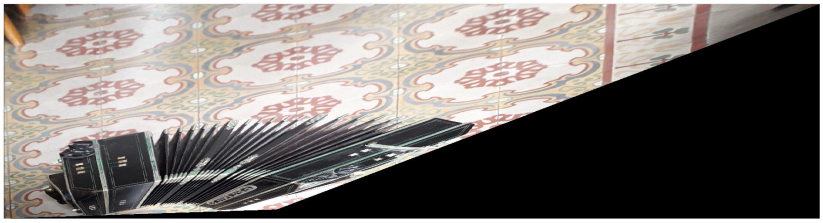


Figure 4: Affine rectification

The result of the affine rectification on the input image is in figure 4.

It's possible to see that parallel lines are now parallel, even if angles do not show their real value.

3.2 Euclidean Rectification

A similarity transformation (or more simply a similarity) is an isometry composed with an isotropic scaling. In the case of a Euclidean transformation composed with a scaling (i.e. no reflection) the similarity has matrix representation:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3}$$

where the scalar s represents the isotropic scaling. A similarity transformation preserves the shape of objects. Invariants:

- Angles between lines (and parallel lines)
- Ratio of lengths
- Ratio of areas

Once the image has been affinely rectified we have obtained an image such that the transformation from the original scene is an affine transformation. The image of the dual

conic corresponding to circular points can be obtained as:

$$C_\infty^{*\prime} = H_a C_\infty^* H_a^t \tag{4a}$$

$$C_\infty^{*\prime} = \begin{bmatrix} a_{11}^2 & a_{12} * a_{21} & 0 \\ a_{12} * a_{21} & a_{22}^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4b}$$

$$C_\infty^* = H_a^{-1} C_\infty^{*\prime} H_a^{-t} \tag{4c}$$

Notice that the upper left part of $C_\infty^{*\prime}$ is a symmetric matrix and homogeneous, so it has only 2 DOF. We can use two pair of orthogonal lines, l and m, to determine its parameters using equation 5.

$$\cos(\theta) = \frac{l_1 m_1 + l_2 m_2}{\sqrt{(l_1^2 + l_2^2)(m_1^2 + l_2^2)}} \tag{5a}$$

$$\cos(\theta) = \frac{l C_\infty^* m}{\sqrt{(l^t C_\infty^* l)(m^t C_\infty^* m)}} \tag{5b}$$

$$\cos(\theta) = \frac{l' C_\infty^{*\prime} m'}{\sqrt{(l'^t C_\infty^{*\prime} l')(m'^t C_\infty^{*\prime} m')}} \tag{5c}$$

That, in the case of orthogonal lines, becomes a linear constraint on $C_\infty^{*\prime}$.

The matrix H_a in equation 4 is the transformation matrix from the real scene to the image, so the matrix H_a^{-1} is the matrix that maps the image to a similarity with respect to the real scene since the matrix C_∞^* is mapped back to its value.

Once found $C_\infty^{*\prime}$ we can use standard cholesky (or SVD) to determine H_a:

$$svd(C_\infty^{*\prime}) = USV^t = H_a C_\infty^* H_a^t$$

Where $H_a = U$. Since SVD does not return the matrix S (C_∞^*) as

$$C_\infty^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

that is required by the algorithm, it's possible to factorize the matrix S returned by SVD through the following decomposition:

$$S = S_{fact} C_\infty^* S_{fact}$$

$$S = \begin{bmatrix} \sqrt{S_{11}} & 0 & 0 \\ 0 & \sqrt{S_{22}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} \sqrt{S_{11}} & 0 & 0 \\ 0 & \sqrt{S_{22}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Doing in this way the factorization of $C_\infty^{*\prime}$ becomes:

$$C_\infty^{*\prime} = USV^t = US_{fact}C_\infty^* S_{fact}V^t = H_a C_\infty^* H_a^t$$

In this way the matrix H_a becomes:

$$H_a = US_{fact}$$

Renaming the matrix H_a^{-1} as H the overall transformation (H_r, reconstructive matrix) that maps the image to a similarity is the composition of the two transformation:

$$H_r = HH_{r\_aff}$$

The final result of the shape reconstruction phase for the two upper faces is showed in figure 5.

4 Measure of Metric Properties

Once we have reconstructed the shape of the object, metric properties can be determined, like angles, since they are invariants of a similarity transformation.

The relative orientation between the horizontal upper faces can be determined using the cosine between the two lines representing corresponding lines in each face.

Given two corresponding lines in each faces the cosine between them can be determined using equations 5.

The relative position can be determined simply by computing the difference between the origin of the two reference frames and multiplying by the scaling factor. The scaling factor can be determined by doing the ratio between the length of the longside of the horizontal face and the length in the image of the corresponding side. The relative position estimated as before gives us the relative position in the reference frame of the image. If we want the position of the right face with respect to the left face in the reference frame of the left face we need to multiply the rotation matrix of the reference frame of the left face and the vector of the relative positions:

$$relative\_pose\_from\_left\_to\_right = R_{from\_img\_to\_left} * relative\_coordinates$$

5 Camera Calibration

Using also the images of vertical lines, calibrate the camera (i.e., determine the calibration matrix K) assuming it is zero-skew (but not assuming it is natural).

Camera calibration is determining the matrix K of the intrinsic parameters of the camera:

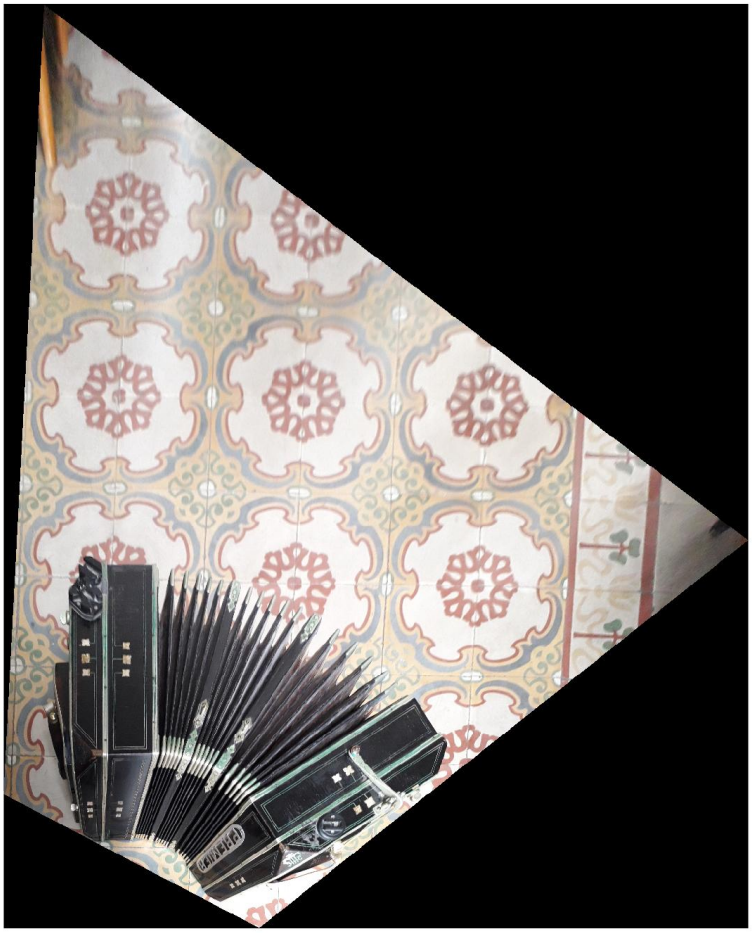


Figure 5: Shape reconstruction

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

P is the projection matrix that maps 3d points (X) of the world to points in the image (x) through the relation:

$$x = PX \tag{8}$$

Where P is the matrix:

$$P = [KR| - KRo] \tag{9}$$

In equation 9, R is the rotation between the camera and the world (represents the rotation of the camera with respect to the world reference frame) and o is the location of the camera in the world reference frame.

K is related to ω through the following equation:

$$\omega = (KK^t)^{-1}$$

In order to determine K we need to specify some constraints on ω (the image of the absolute conic).

For a zero skew camera the image of the absolute conic is given by

$$\omega = \begin{bmatrix} \alpha^2 & 0 & -u_0\alpha^2 \\ 0 & 1 & -v_0 \\ -u_0\alpha^2 & -v_0 & f_y^2 + \alpha^2 u_0^2 + v_0^2 \end{bmatrix} \tag{10}$$

That is a symmetric matrix with 4 DOF, so we need 4 constraints on ω. Here we can use the homography method (p 211 Multiple View Geometry in Computer Vision) adapted with the reconstructive transformation (that we have found in the previous point) on the horizontal faces.

- For each horizontal face we can compute the transformation that maps its corner points to their imaged points (H_r^{-1} since H_r maps the image point to their real shape).
- We can compute the imaged circular points for the plane of that face as $H_r(1, \pm i, 0)'$. Writing H_r = [h1, h2, h3], the imaged circular points are $h1 \pm ih2$.
- This gives us two constraints on the image of the absolute conic since the circular points lie on ω:

$$h_1^T \omega h_2 = 0$$
$$h_1^T \omega h_1 = h_2^T \omega h_2$$

Which are linear equations in ω.

Other constraints that can be used are the constraints deriving from the fact that the line at infinity on the horizontal plane is orthogonal with respect to the vanishing point of the vertical direction on the vertical faces:

$$l_{inf}^T \omega v_p = 0$$

In order to determine the vanishing point of the vertical direction we can use a least square approximation using all vertical lines on the vertical face. Once having determined ω it's possible to obtain the calibration matrix K using the parametrization in equation 10.

6 Localization

Localize the camera with respect to (both) horizontal faces. From the image of the (short) horizontal segments common to a horizontal face and its neighboring vertical face, reconstruct the shape of the vertical faces.
In this point we have to find the relative position of the camera with respect to the reference frame placed on the horizontal faces.

This is possible knowing the shape of the horizontal faces, knowing the size, knowing the image and knowing K.

If we identify the plane of the left horizontal face with π we can write the position of a point in the world reference frame as $X_w = [R_\pi | o_\pi] X_{pi}$ where X_{pi} is the position of the point in the plane reference frame. Using equations 8 and 9 we obtain:

$$u = [KR| - KRo][R_\pi | o_\pi] X_{pi}$$

By putting the world reference frame on the camera ($R = I$ and $o = [0\ 0\ 0\ 1]'$):

$$u = [K|0][i_\pi | j_\pi | o_\pi] \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$
$$u = K[i_\pi | j_\pi | o_\pi] x$$

Where x are the coordinates of the point on the plane. By inspection we can identify $K[i_\pi | j_\pi | o_\pi] = H_o mog$ as the matrix that maps the real points to the image. H is known since the shape of horizontal face is known and also its size.

So we obtain:

$$[i_\pi | j_\pi | o_\pi] = K^{-1} H_o mog$$

Where $H_o mog$ is the transformation mapping world points to image point.

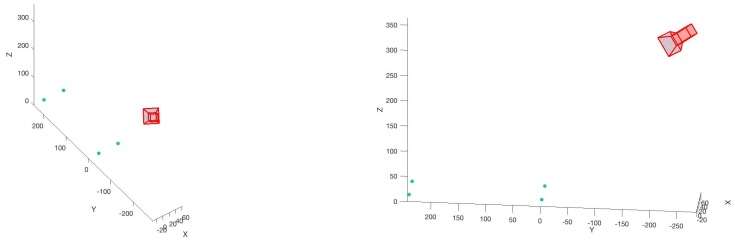


Figure 6: Camera Localization from left face

H_{omog} it's easy to find since knowing the shape it's the transformation that maps the shape of the horizontal face to the image.

A few more steps are needed once found the matrix $H_{omog} = [h_1 \mid h_2 \mid h_3]$.

The matrix $R = [i_\pi \mid j_\pi \mid k_\pi]$ (rotation of the plane with respect to the camera) can be found using these equations:

$$\begin{aligned} \lambda &= \frac{1}{|K^{-1}h_1|} \\ i_\pi &= K^{-1}h_1\lambda \\ j_\pi &= K^{-1}h_2\lambda \\ k_\pi &= i_\pi \times j_\pi \\ o_\pi &= K^{-1}h_3\lambda \end{aligned}$$

Due to noise in the data R may be not a true rotation matrix, it's possible to approximate it through SVD, obtaining an orthogonal matrix:

$$\begin{aligned} [U, \_, V] &= svd(R) \\ \hat{R} &= UV \end{aligned}$$

The result of the camera localization phase is shown in figure 6.

The rotation of the right plane can be found simply by applying the rotation found in section 4.

$$R_{right} = \hat{R} * R\_from\_left\_to\_right$$

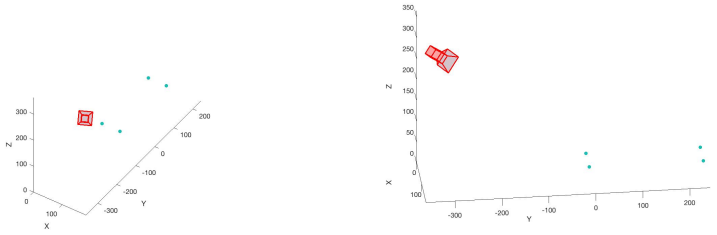


Figure 7: Camera Localization from right face

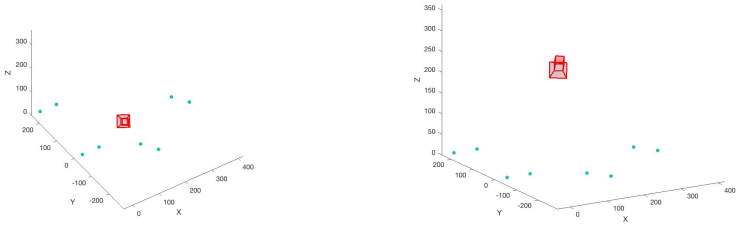


Figure 8: Camera Localization from both face

7 Shape Reconstruction Vertical Faces

For the shape reconstruction of vertical faces we can use the result of the localization, namely the matrix R, the rotation of the plane with respect to the camera. If we put the world reference frame on the horizontal face we obtain the following matrix P:

$$P = [K\hat{R}/\lambda \mid o_\pi/\lambda]$$

The elementwise division by lambda is needed to reconstruct the original ratios. The reference frame of the vertical face it's the same of the horizontal face, a point on the vertical face has always $y = 0$ and the z has the inverse sign. The matrix P maps the point on the vertical face to image points so we can directly use the matrix $H_{vert\_sr} = [p_1 \mid p_3 \mid p_4]^{-1}$ to reconstruct the shape of the vertical left face. The reconstruction of the vertical right face can be performed in the same way using the matrix R_{right}. The reconstructed vertical faces are shown in figure 9.

8 Implementation Details

In this section implementation details are explained.

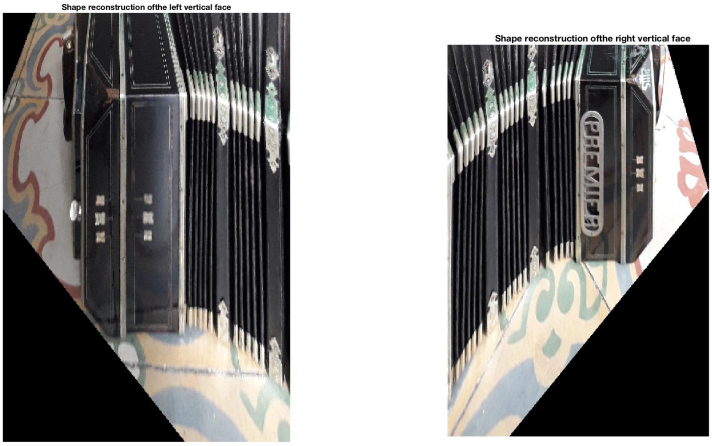


Figure 9: Shape Reconstruction of the vertical faces

8.1 Vanishing Points

Vanishing points are extracted using a Least Square Approximation: A vanishing point is a point common to all line having the same direction. Considering the line l as data and the vanishing point v as model:

$$\begin{aligned} f(l, v) &= 0 \\ l'v &= 0 \end{aligned}$$

Considering the vanishing point with 2DOF ($v_3 = 1$), the equation becomes:

$$l_1 v_1 + l_2 v_2 = -l_3$$

It's possible to rephrase this in a model fitting framework using these matrices:

$$X = \begin{bmatrix} l_{11} & l_{21} \\ \vdots & \vdots \\ l_{1n} & l_{2n} \end{bmatrix}$$

$$W = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$Y = \begin{bmatrix} -l_{31} \\ \vdots \\ -l_{3n} \end{bmatrix}$$

The solution (W) minimizing the error ($|Y - XW|^2$) is then found simply using:

$$W = (X'X)^{-1}(X'Y)$$

8.2 Line at Infinity

The line at infinity on the horizontal face is extracted using these steps:

- Vanishing point extraction using lines on the horizontal faces and on the ground.
- Line fitting using Least Square Approximation.