IACV Homework

Emanuele Ghelfi

Politecnico di Milano

February 1, 2018

Contents

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. In the given image (Figure 1), the bandoneon is placed on a horizontal floor. Four rectangular faces are visible in the image: two coplanar horizontal faces, and two non-coplanar vertical faces. The long side of the horizontal faces is 243 mm, and it is slightly longer than the long side of the vertical face. For each horizontal face, two groups of parallel lines can easily be identified. The two groups of lines are mutually perpendicular. One of the (short) lines on each horizontal face is also common to a vertical face. Part of the horizontal floor is also visible: we can assume to see groups of parallel lines (the groups are also mutually orthogonal): These lines can be used to help in robustly find, e.g., the image of the horizontal vanishing line (i.e. the image of the line at the infinity of the horizontal plane). We can NOT assume square patterns on the floor. In addition, the long lines in the vertical faces are vertical (i.e. orthogonal both to the horizontal faces and to the floor).

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:

Figure 1: Input Image

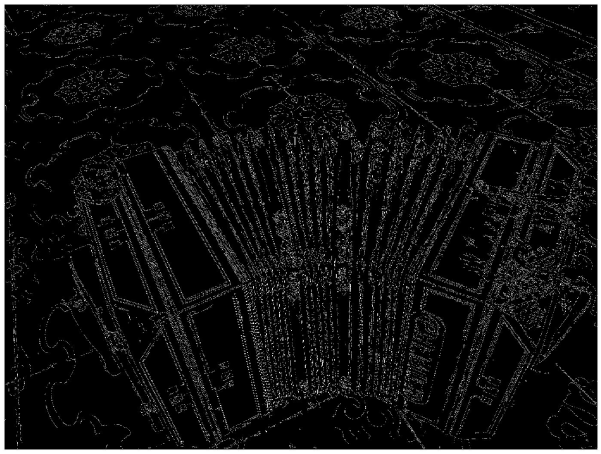


Figure 2: Edge detection through Canny

- Edge detection with Canny
- Line detection using Hough

2.1 Canny

The canny algorithm is based on 3 steps:

- 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:

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

Figure 3: Line detection through Hough

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

Steps:

- Discretize model space. Set the number of votes for each model = 0
- For each datum computes the hough transform $H(x_i)$
- Let x_i vote 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.

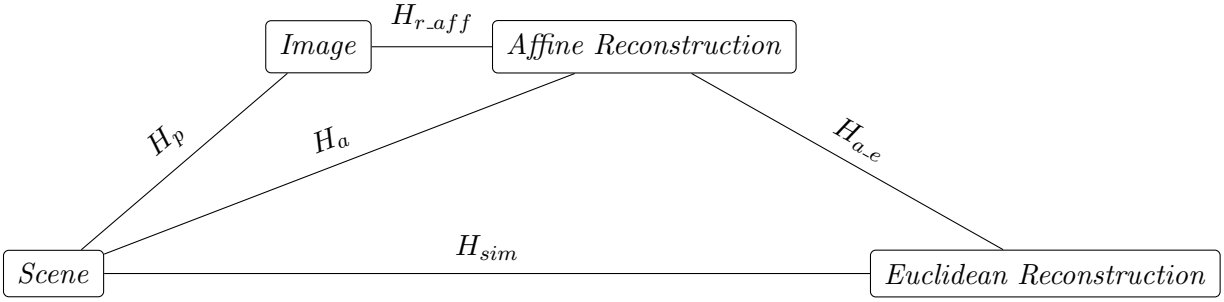
The result of the application of the Hough Transform for line detection is shown in figure 3.

In order to obtain more lines two hough transforms with different parameters have been used and only the best lines have been selected.

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.

A stratified approach to the shape reconstruction problem has been used. 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. A two-step approach increases the robustness of the shape reconstruction avoiding non-linear constraints.

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

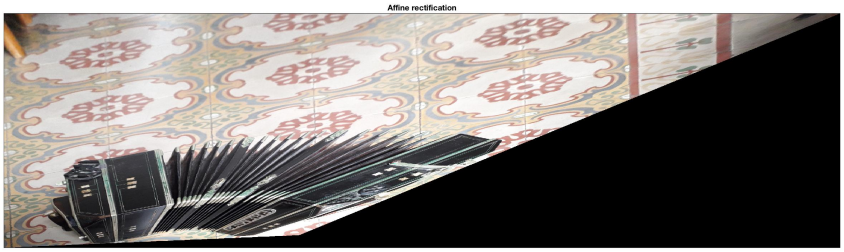


Figure 4: Affine rectification

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].

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:

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

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

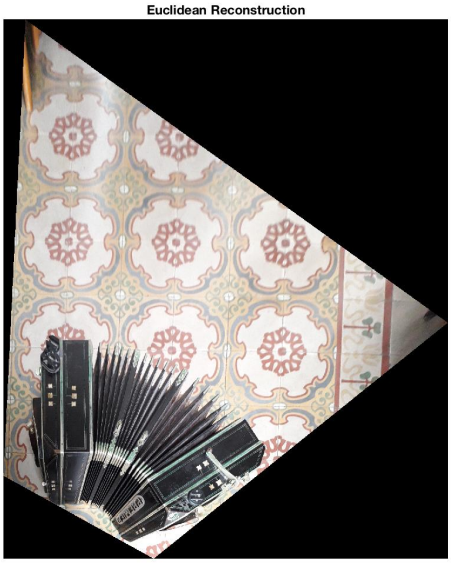


Figure 5: Shape reconstruction

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 (transformed) lines representing corresponding lines in each

face. We need to transform original lines according to the transformation H_r:

$$l' = H_r^{-t} l$$

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:

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

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 world with respect to the camera reference frame) and o is the location of the camera in the world reference frame in cartesian coordinates.

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 $H_r\ [1, \pm i, 0]^t$. Writing $H_r = [h_1, h_2, h_3]$, the imaged circular points are $h_1 \pm i h_2$.
- 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:

$$[l_{inf}]_\times \omega v_p = 0 \tag{12}$$

Where $[l_{inf}]_\times$ is the vector product matrix of l_{inf}. 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. More constraints can be determined from the orthogonality of vanishing points. Given v_{p1} and v_{p2}, vanishing points of orthogonal directions we obtain:

$$v_{p1}^T \omega v_{p2} = 0 \tag{13}$$

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_π is the position of the point in the plane reference frame. The plane reference frame has the x axis identical to the line at the bottom of the face, going right, the y axis with the same direction of the left line and the z axis orthogonal to both axis using the right hand rule. In this way a point on the plane can be written (in homogeneous coordinates) as:

$$X_\pi = \begin{bmatrix} x \\ y \\ 0 \\ w \end{bmatrix}$$

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 $o = [0\ 0\ 0]'$):

$$u = [K|\underline{\mathbf{0}}] \begin{bmatrix} i_\pi & j_\pi & o_\pi \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$u = K[i_\pi|j_\pi|o_\pi]x$$

Where x are the coordinates of the point on the plane. By inspection we can identify $K[i_\pi|j_\pi|o_\pi] = H_omog$ 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:

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

Where H_{omog} is the transformation mapping world points to image point.

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.

Figure 6: Camera Localization from left face

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.

$$R_{right} = \hat{R}R_{l->r}$$

Where $R_{l->r}$ is the rotation matrix that represent the reference frame of the right face in the reference frame of the left face. In figures 6, 7 8 the camera is represented in red while the points on the horizontal plane represent the coordinate of the extreme points of the faces. Red dots are the extreme points of the left face while blue dots are the extreme points of the right face.

Figure 7: Camera Localization from right face

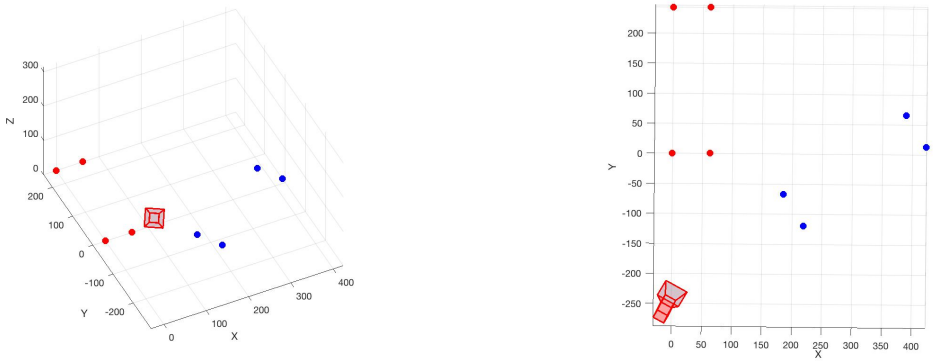


Figure 8: Camera Localization from both face

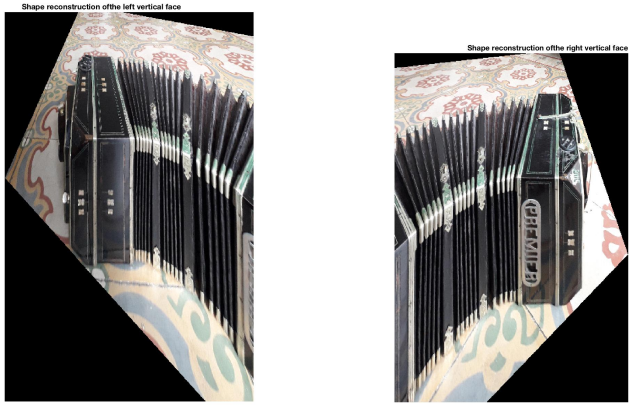


Figure 9: Shape Reconstruction of the vertical faces

7 Shape Reconstruction Vertical Faces

For the shape reconstruction of vertical faces we can use the result of the localization, namely the matrix $\hat{R}$, 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:

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

The reference frame of the vertical face it's the same of the horizontal face, a point on the vertical face (reference frame π') has always $y = 0$: $X_{\pi'} = [x \;\; 0 \;\; z \;\; w]^t$. 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} as $\hat{R}$. The reconstructed vertical faces are shown in figure 9.

8 3D Shape Reconstruction

Once obtained the real shape of the vertical faces it's possible to reconstruct the 3D shape of the object, that is determination of shape and scale.

In order to obtain the position (in the world) of an image point ($\underline{x}_{image}$ on the left vertical face with respect to the reference frame on the left horizontal face is possible to proceed in this way:

$$\begin{bmatrix} x \\ z \\ w \end{bmatrix}_\pi = H_{vert\_sr} \; \underline{x}_{image}$$

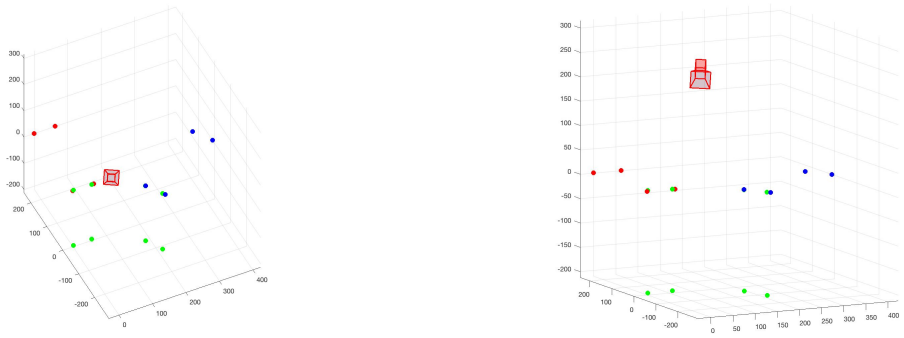


Figure 10: 3D shape reconstruction

The position on a image point on the right vertical face is not as easy, since $H_{vert\_sr\_r}$ maps an image point to its position in the reference frame π', so we need to add the transformation from π' to π:

$$\begin{bmatrix} x \\ z \\ w \end{bmatrix}_{\pi'} = H_{vert\_sr\_r}\ \underline{x}_{image}$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}_{\pi} = relative\_pose\_from\_left\_to\_right + R_{l->r} \begin{bmatrix} x \\ 0 \\ z \\ w \end{bmatrix}_{\pi'}$$

Using these relations the position of the extreme points of the vertical faces can be determined, the result is shown in figure 10.

Green points are points on the vertical faces, red point and blue points are points on the left and right horizontal faces, respectively.

9 Implementation Details

In this section implementation details are explained.

9.1 Fitting Vanishing Point

Vanishing points are fitted 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:

$$f(l, v) = 0$$
$$l'v = 0$$

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 ($E = |Y - XW|^2$) is then found simply using:

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

9.2 Fitting 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.

Given a set of points the line passing through all points can be fitted considering two ways: normal LS or total LS. In normal LS the objective function is:

$$min \sum_i (y_i - mx_i - b)^2$$

Where the line is parametrized as $y = mx + b$ So the objective is to minimize the sum vertical distances from to points to the line. This can be easily solved using the same approach as in section 9.1. The limitations is that this method is not rotation invariant and fails completely with vertical lines. The total LS recover this problem using

another parametrization of the line ($ax + by = d$) and another error measure: the sum of perpendicular distances from the points to the line:

$$min \sum_i (ax_i + by_i - d)^2$$
$$s.t.\ a^2 + b^2 = 1$$

The problem can be rewritten as a quadratic constrained minimization problem:

$$min\ ||A'Ah||$$
$$s.t.\ ||h|| = 1$$

Where the matrix A has in each row: $[x_i - \bar{x}\ ,\ y_i - \bar{y}]$. This can be solved by considering as h the eigenvector corresponding to the smallest eigenvalue of $A'A$, that is the last column of V in the svd decomposition. In the homework these two methods give similar result regarding the extraction of the line at infinity.

9.3 Fitting Conics

The image of the dual conics corresponding to vanishing point in section 3.2 and the image of the absolute conic in section 5 are fitted using a LSA over constraints. Regarding the image of the absolute conic in section 5 different constraints set has been used, generating different conics. For mixing different constraints we need to make the matrix X well scaled, otherwise we can obtain an inaccurate solution. Using a well scaled matrix gives similar result using different constraints set. These are the K matrix obtained along with the constraints set used:

- Using constraints 12 and 11:

```
K =

   1.0e+03 *

   3.367907059719495                   0   2.091030392617472
                   0   3.392768272051243   1.572523700957307
                   0                   0   0.001000000000000


fx = 3.367907059719495e+03
```

```
fy = 3.392768272051243e+03


u0 = 2.091030392617472e+03


v0 = 1.572523700957307e+03


alfa = 0.992672292848130
```

- Using only vanishing points (constraints like equation 13) normalized with the scaling matrix:

$$H_{scaling} = \begin{bmatrix} \frac{1}{MAX\_SIZE} & & \\ & \frac{1}{MAX\_SIZE} & \\ & & 1 \end{bmatrix}$$

```
K =

   1.0e+03 *

   3.322361378031902                   0   2.100901897735311
                   0   3.398639154104814   1.691795005370728
                   0                   0   0.001000000000000


fx = 3.322361378031901e+03


fy = 3.398639154104814e+03


u0 = 2.100901897735311e+03


v0 = 1.691795005370728e+03


alfa = 0.977556376945524
```

- Using only vanishing points not normalized:

```
K =

   1.0e+03 *

   3.322361378031897                   0   2.100901897735313
                   0   3.398639154104813   1.691795005370738
                   0                   0   0.001000000000000


fx = 3.322361378031897e+03


fy = 3.398639154104813e+03


u0 = 2.100901897735313e+03


v0 = 1.691795005370738e+03


alfa = 1.022958904042552
```

- Using all the constraints:

```
K =

   1.0e+03 *

   3.327516509370050                   0   2.100856628676403
                   0   3.399949644133486   1.682536922340478
                   0                   0   0.001000000000000


fx = 3.327516509370050e+03


fy = 3.399949644133486e+03


u0 = 2.100856628676403e+03
```

```
v0 = 1.682536922340478e+03
```

```
alfa = 0.978695821307702
```

The scaling matrix $H_{scaling}$ has on the first two elements on the diagonal the inverse of the maximum size of the image. In this way both the axis corresponding to the maximum size is in the range [0,1] wile the smaller axis is in the range [0,Aspect_ratio]. All the matrices are very similar, in all the matrices the α factor is near 1, so the camera is almost natural. The u_0 and v_0 factors are near the image center, that is acceptable.

10 Results Analysis

A simple test on the pinhole model obtained from the camera localization and calibration phase could be the reconstruction of the horizontal plane using P. Since P maps points on the world to image points it's possible to extract from P the reconstructive transformation from the image to a similarity with respect to the horizontal plane. By using as reference frame the one found before (the left horizontal face), a point on the horizontal face can be written as $\underline{x} = [x\ y\ 0\ w]^t$:

$$\underline{x}_{image} = P\underline{x}$$

$$\underline{x}_{image} = P \begin{bmatrix} x \\ y \\ 0 \\ w \end{bmatrix}$$

$$\underline{x}_{image} = [p_1 \mid p_2 \mid p_4] \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = [p_1 \mid p_2 \mid p_4]^{-1} \underline{x}_{image}$$

If the pinhole model found is good the reconstructed image using $[p_1 \mid p_2 \mid p_4]^{-1}$ should be similar to image 5. The result of the reconstruction is shown in figure 11. The figure on the left is the reconstruction using the pinhole model with the reference frame of the world on the left horizontal face while the reconstruction on the right uses as reference frame of the world the right face.

Measures of segments are a bit different from their real measures, this can be due to numerical errors or errors in measurements on the reconstructed images. Despite small

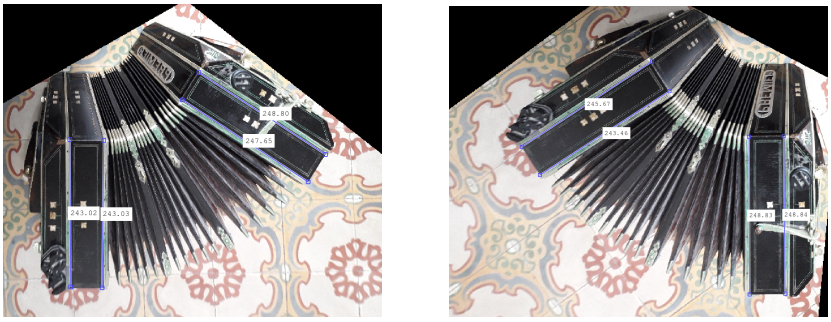


Figure 11: Shape Reconstruction of the horizontal faces using P

errors the reconstructed shape is satisfactory, this means that the pinhole model is quite good.

11 Computed Results

```
Horizontal Vanishing Line

l_inf_prime =

   0.000043143973982
   0.000697947637111
   1.000000000000000

Angle from left to right horizontal face

theta =

  57.087157363489169

Rotation matrix from left to right. This is the rotation of the right
Reference frame in the left reference frame.

R_from_left_to_right =

   0.543362633983245   0.839498092904797                   0
  -0.839498092904797   0.543362633983245                   0
                   0                   0   1.000000000000000
```

```
Position of the right reference frame in the left reference frame

relative_pose_from_left_to_right =

   1.0e+02 *

   1.853883835850114
  -0.667861096818124
                   0

Camera Intrinsic Parameters

K =

   1.0e+03 *

   3.327516509370050                   0   2.100856628676403
                   0   3.399949644133486   1.682536922340478
                   0                   0   0.001000000000000


fx =

     3.327516509370050e+03


fy =

     3.399949644133486e+03


alfa =

   0.978695821307702


u0 =

     2.100856628676403e+03
```

```
v0 =

     1.682536922340478e+03

Rotation of the left face in the camera frame

R =

   0.878572476320691  -0.475699162684000  -0.042669784080417
  -0.355103489978936  -0.590863747207724  -0.724418072414284
   0.319393041966332   0.651605969016083  -0.688039058401559

Position of the left face in the camera frame

T =

   1.0e+02 *

  -1.029171696730963
   0.528455399488923
   3.783697597007948

Rotation of the camera in the frame of the left horizontal face

cameraRotation =

   0.878572476320691  -0.355103489978936   0.319393041966332
  -0.475699162684000  -0.590863747207724   0.651605969016083
  -0.042669784080417  -0.724418072414284  -0.688039058401559

3D Position of the camera in the frame of the left horizontal face

cameraPosition =

   1.0e+02 *

  -0.116628402576252
  -2.642810915981000
   2.942239839695089

Rotation of the right face in the camera frame
```

```
R_from_cam_to_right_plane =

   0.876731994748419   0.479082768330263  -0.042669784080417
   0.303079021295868  -0.619161984629154  -0.724418072414284
  -0.373475723755637   0.622188185261582  -0.688039058401559

Position of the right face in the camera frame

T_from_cam_to_right_plane =

   1.0e+02 *

   0.917300580288374
   0.264749689643228
   3.940632917632010

Rotation of the camera in the frame of the right horizontal face

camera_orientation_wrt_right =

   0.876731994748419   0.303079021295868  -0.373475723755637
   0.479082768330263  -0.619161984629154   0.622188185261582
  -0.042669784080417  -0.724418072414284  -0.688039058401559

Position of the camera in the frame of the left horizontal face

cameraPos_wrt_right =

   1.0e+02 *

   0.587263886602338
  -2.727355201929570
   2.942239839695089

3d coordinates of the points on the vertical faces.
First four points are on the left vertical face.
The other four are on the right vertical face.
Coordinates are in the frame of the left horizontal face.

vert_points =
```

```
1.0e+02 *

0.015414201042978                   0   0.019170226246530
0.016678879198411                   0  -2.082483315996573
0.568770513884049                   0  -2.078203725292395
0.567257454649088                   0   0.003425679758380
1.852564469873516  -0.665822669528639  -0.004663160952759
1.858600813021336  -0.675148851057844  -2.082725200459683
2.150603873877010  -1.126295097375289  -2.119507205189755
2.145882756202694  -1.119000946036775  -0.006334383951559
```

Where $relative\_pose\_from\_left\_to\_right$ is the distance (in mm) from the origin of the left face to the right face. The origin of the left face is the intersection of lines 15 and 1 and the origin of the right face is the intersection of lines 72 and 76. From the camera localization the camera height is about 29 cm above the horizontal faces. Negative numbers on the vert_points' z axis means that they are above the horizontal faces. The height of the vertical faces is around 21 cm, this means that it's slightly smaller than the longside of the horizontal face.

12 References

- Image Analysis and Computer Vision course Polimi 2017/2018
- Multiple View Geometry in Computer Vision (Hartley, Zisserman)
- A Flexible new Technique for Camera Calibration (Zhang)