Navigation

Dibris
- My home
  - Dibris
  - My profile
  - Current course
    - 65863-1314
      - Participants
      - Badges
      - General
      - Lab n. 1
      - Lab n. 2
      - Lab n. 3
      - Motion
      - Projective transformations
        - SLIDES - projective transformations
        - **... Removing the perspective distortion from an image**
      - Single view geometry and camera calibration
      - Stereopsis
  - My courses
  - Courses

Administration

Course administration

My profile settings

Lab 5 - Removing the perspective distortion from an image

Introduction

We have seen that to compute a homography we need *n>=4* points correspondences $(\mathbf{x}_i, \mathbf{x}'_i)_{i=1}^{n}$

Today we consider exactly 4 pairs of finite points lying on two (image) planes and see how we can estimate the transformation between the two planes. Our hypothesis is that points are images of 3D points lying on a plane in the 3D world (planar scene).

The goal of the lab is to implement the DLT algorithm starting from the equation $\mathbf{x}'_i \times H\mathbf{x}_i = \mathbf{0}$

$$\begin{bmatrix} \mathbf{0}^\top & -w'_i\mathbf{x}_i^\top & y'_i\mathbf{x}_i^\top \\ w'_i\mathbf{x}_i^\top & \mathbf{0}^\top & -x'_i\mathbf{x}_i^\top \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

then stacking equations from 4 correspondences we obtain a homogeneous system

$$A\mathbf{h} = \mathbf{0}$$

Matrix A is 8x9 (rank 8 if the 4 points are not in degenerate configurations - eg collinear). A non trivial solution can be found in the 1-dim null space of the matrix.

Part 1 - Let's play with toy data

- Choose 4 points on the plane, for instance the vertices of the square (-2 -2), (-2 2), (2 -2), (2 2).
- Write them in homogeneous coordinates.
- Stack the points in a 4x3 matrix X1
- Choose a transformation that you know in advance of the 4 points on the plane (a simple one, roto-translation, should be enough) and obtain a matrix X2
- Write matrix A (see equations above)
- Compute the null space A, see Matlab functions `null` or `svd`:
  - `h=null(A); H=reshape(h,3,3);`
  - `[U,D,V]=svd(A,0); H=reshape(V(:,9),3,3);`
  - (notice that the above choices are alternative in this case since the rank of the matrix is certainly lower than the number of unknowns otherwise our only choice would be to minimize the algebraic error (svd...)
- have a look at the estimated points H*X1(i,:); how different they are from X2?

It would be good to write your code so that you produce a function:

```
function H = my_homography(X1,X2)
```

Part 2 - Removing the distortion

write a `function I2 = frontoparallel(I1)`

- Provide to the function an image I1
- Choose 4 points representing edges of a distorted shape (with the Image Processing toolbox you may use the command getpts, otherwise you can use ginput). Form a matrix X1 of the points in the first (real) plane
- Form a "virtual" matrix X2 representing where you want to put your 4 points: the new points must lie on a rectangle

- Compute H from those points. To choose their position you may use the getpts again, clicking on 4 virtual positions
- Use the function my_homography to estimate the tranformation H
- Check how good H is by analysing how close (w.r.t the geometric distance are the points in X2 to the HX1
- If you are happy with what you obtain you may apply H to each point of the image I1 producing a frontoparallel view I2 (note: set the size of I2 equal to I1 and don't be afraid to "loose" points;). Thus, for each point (row,col) of image I1:

```
coords = H * [col row 1]';
coords = floor(coords ./ coords(3));
new_row = coords(2);
new_col=coords(1);
if (new_row >=1 && new_row <= size(I1,1) &&  new_col >= 1 && new_col <=
size(I1,2) )
         I2(new_row,new_col,:)=I(row,col,:);
```

- otherwise you may want to try this normalization (it shouldn't be needed in this simple example) before applying the homography

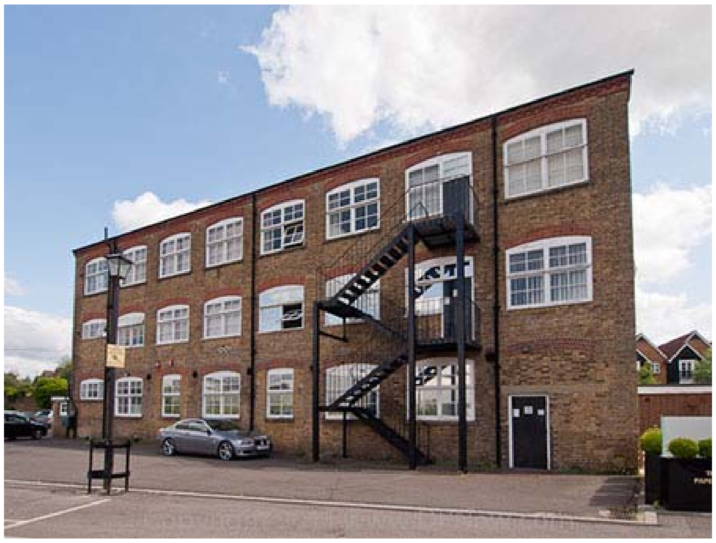
Appendix: inverse mapping

It may be better to compute I2 via an inverse mapping. For each point of I2 you apply the inverse of H and reach a point of I where you may get the gray value. Since in general you will not reach a real pixel (but you will have real an not integer coordinates) some interpolation may be used (eg bilinear interpolation)

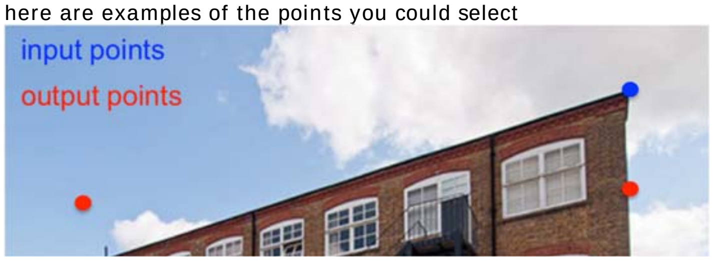
More details can be found here

Stuff

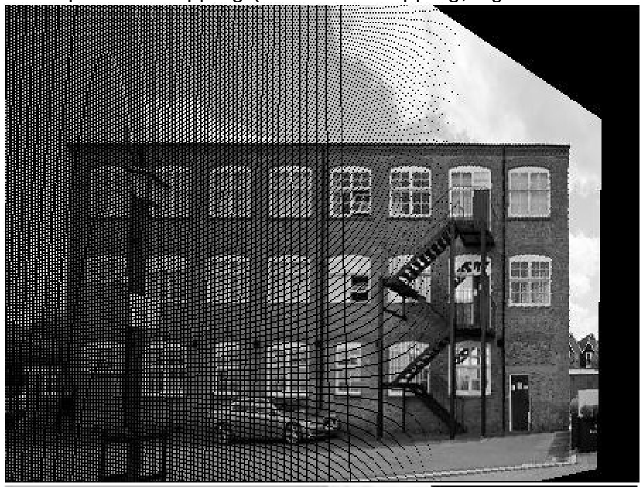
an image to try on (right click the image and save it on your machine):

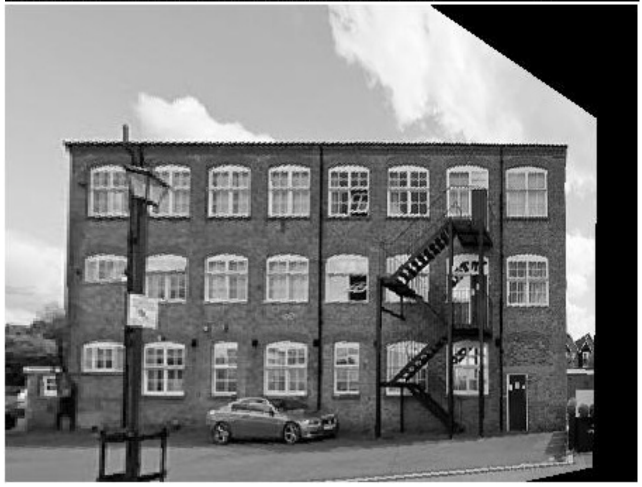


here are examples of the points you could select



frontoparallel mapping (left direct mapping, right inverse mapping)





Other images on google (search on image.google.com: perspective, perspective deformation, ...). For instance, I tried this one:

Submission status

Submission status	No attempt
Grading status	Not graded
Due date	Wednesday, 27 November 2013, 11:05 PM
Time remaining	9 days 21 hours

Add submission

Make changes to your submission