Computational Photography

Matthias Zwicker, Siavash Bigdeli
University of Bern

Fall 2014

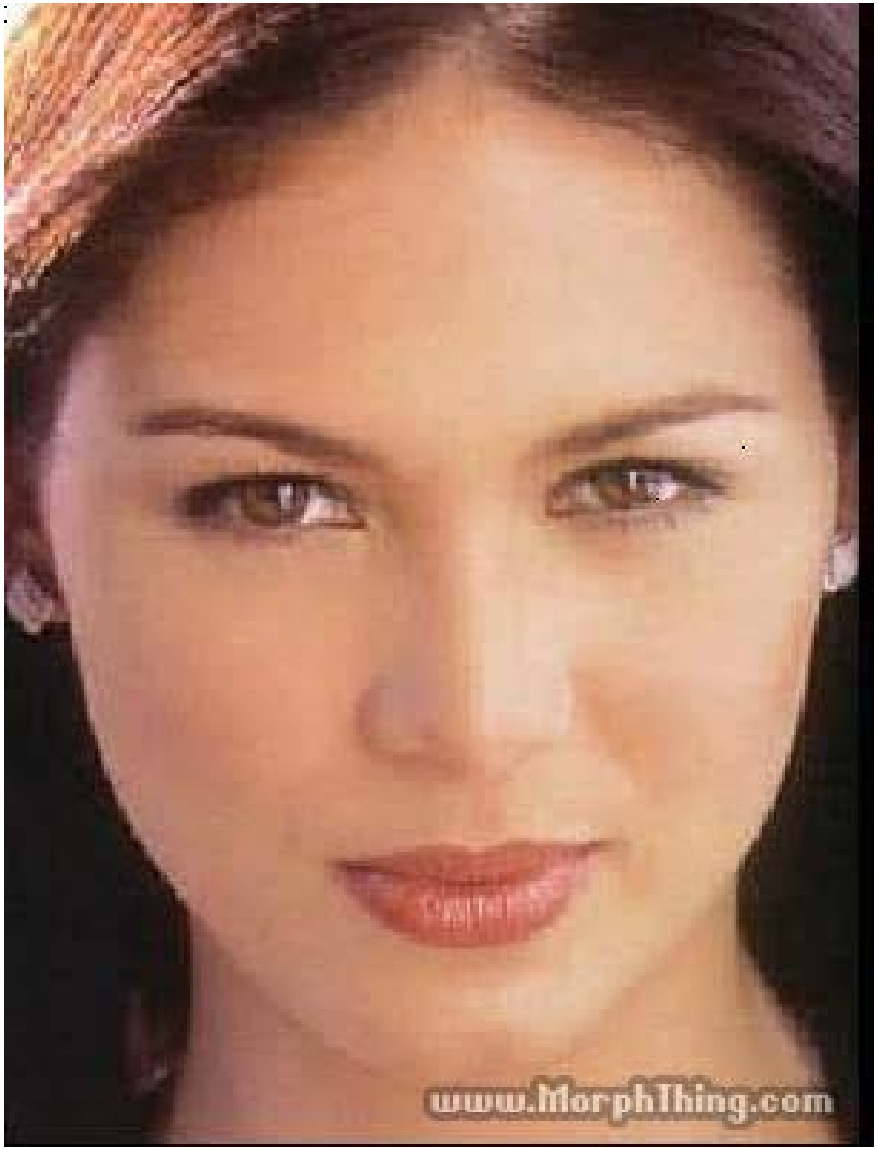
Project 5

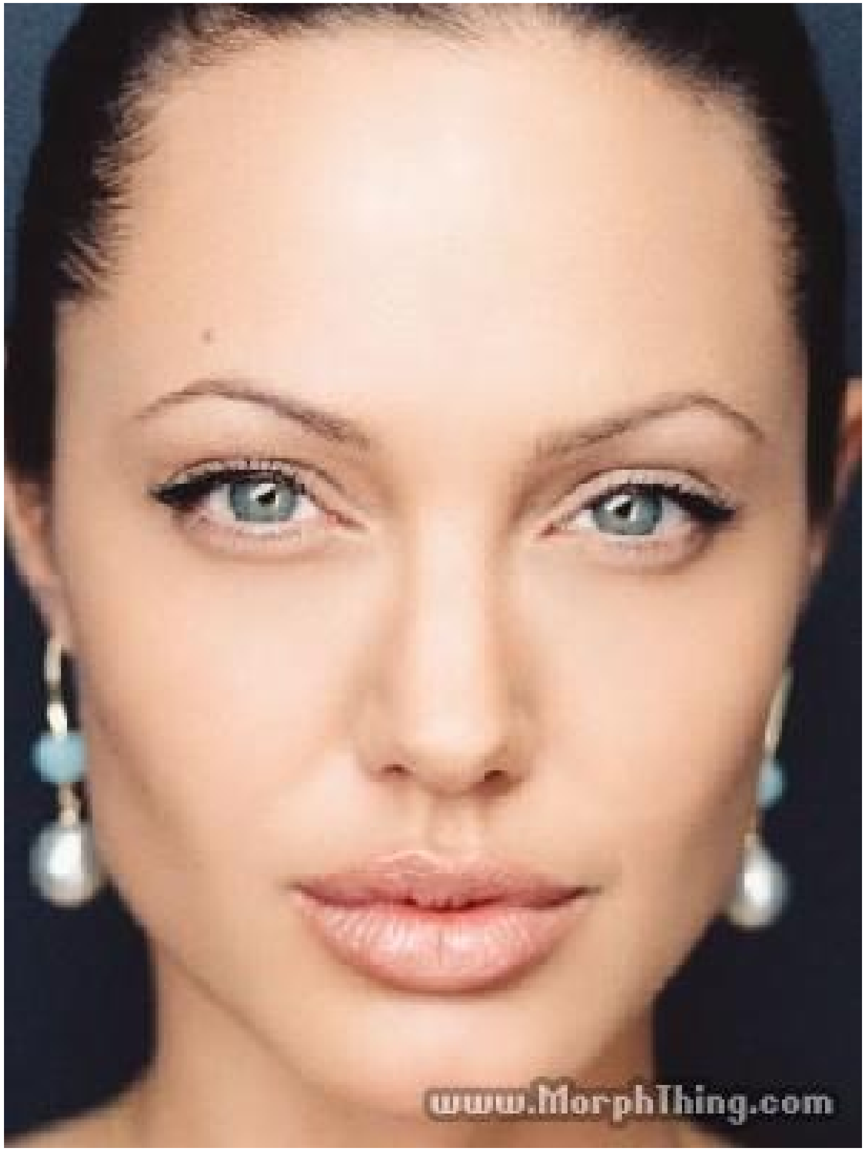
1. Morph Tool
2. Image Rectification
3. Panorama Stitching

Demo

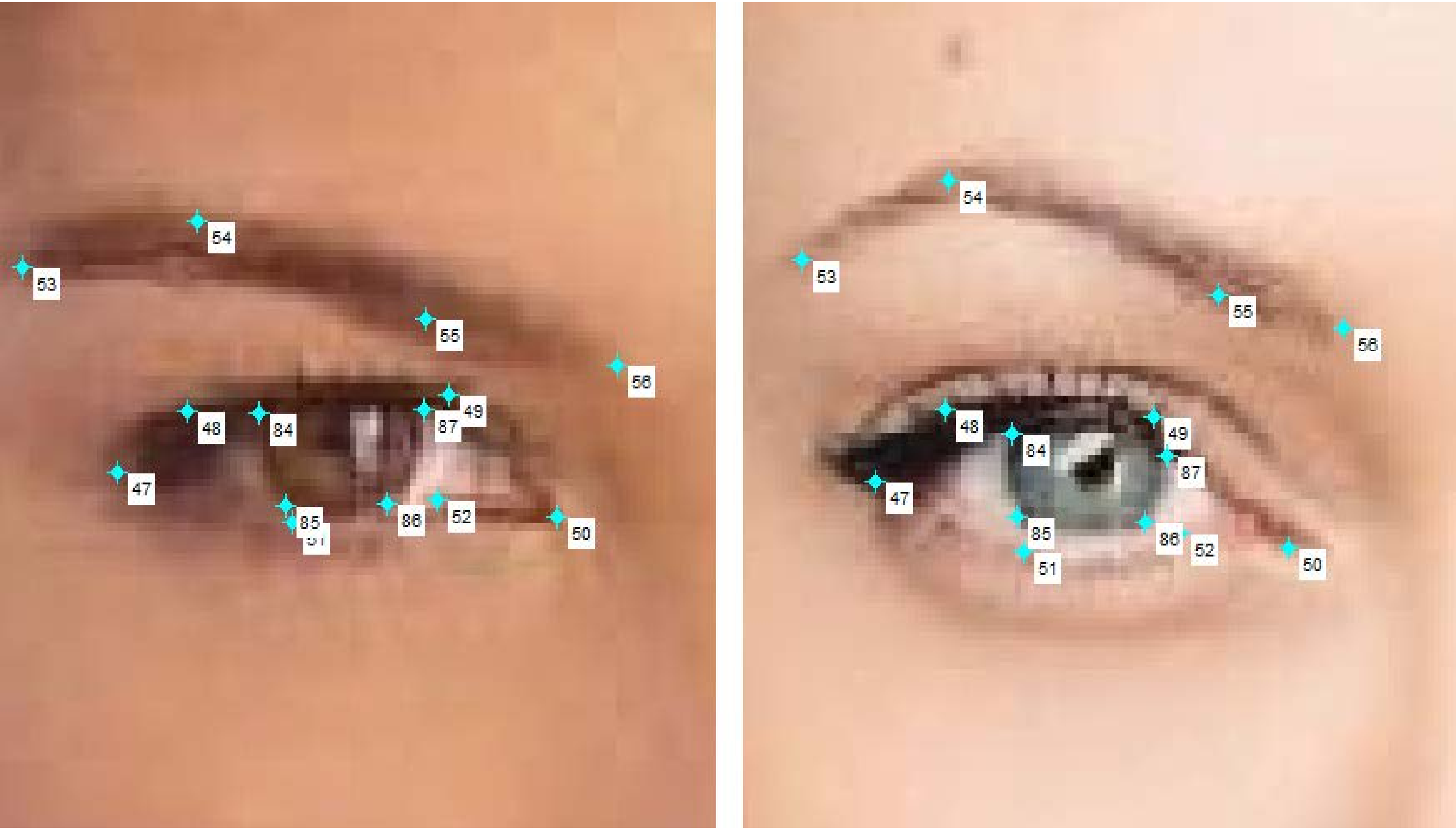
Kristine Hermosa - Angelina Jolie

Select Source & Target Images



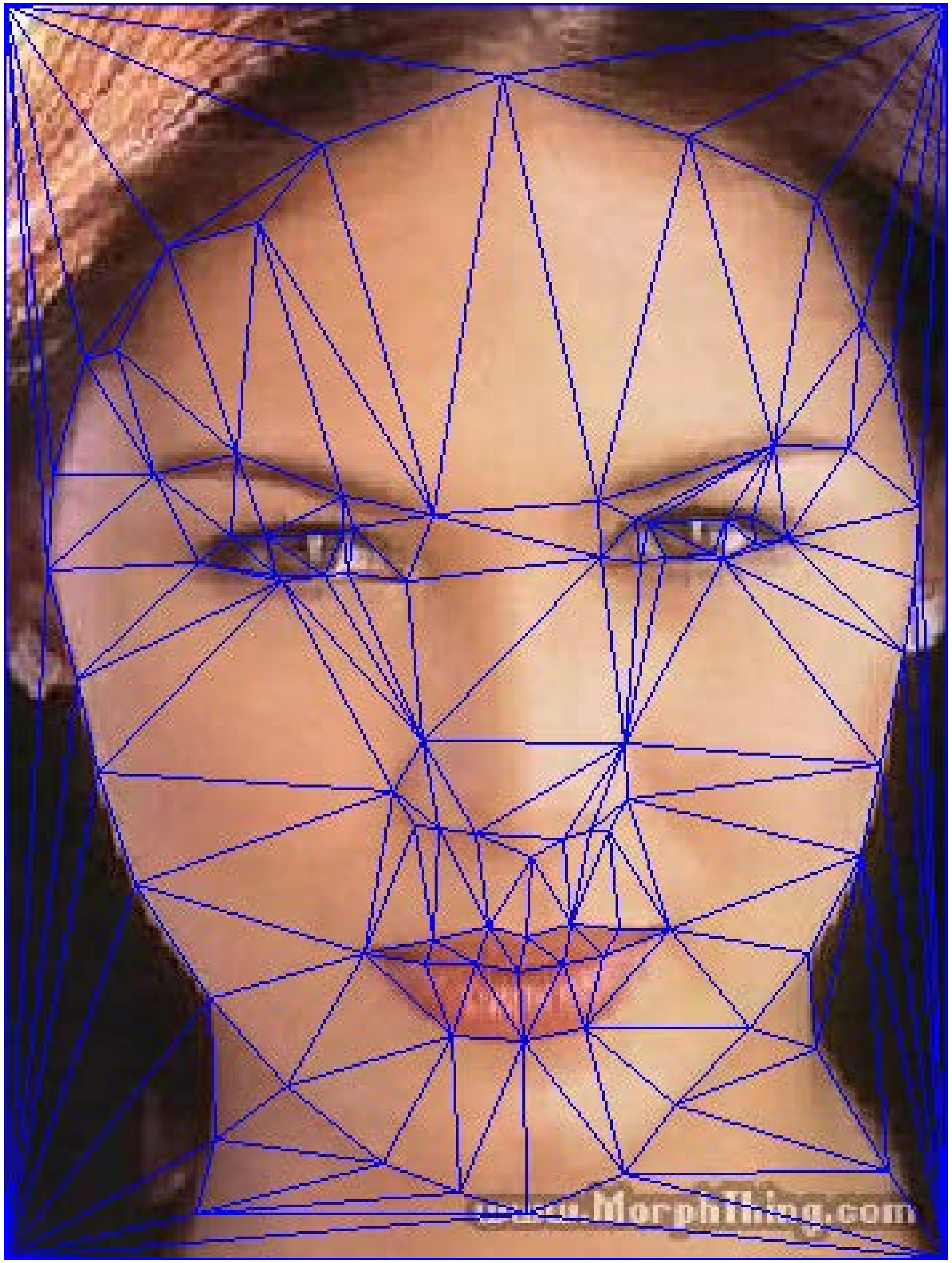


Mark Corresponding Points

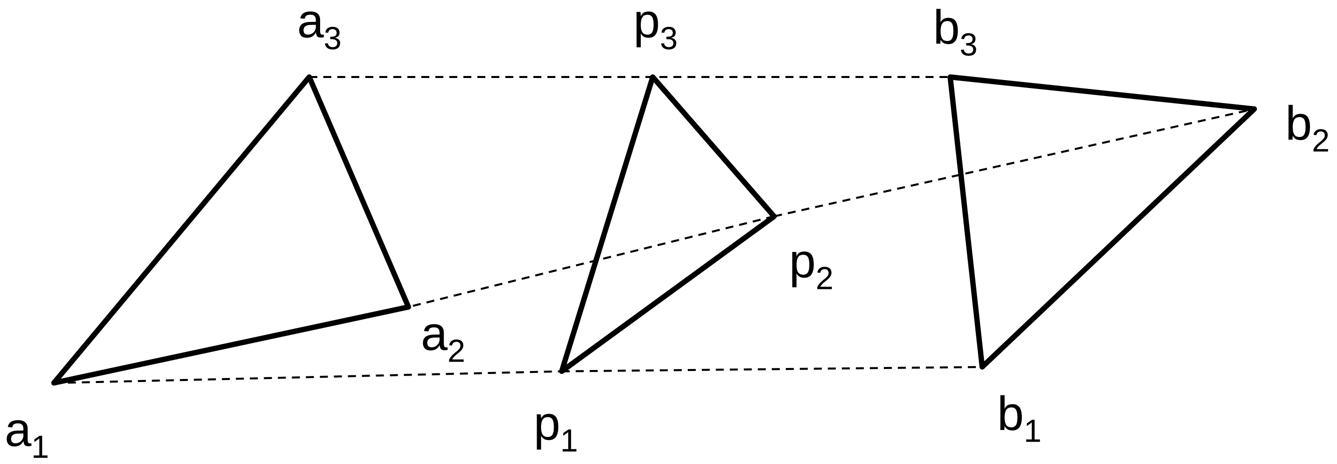


`cpselect`

Find Delaunay Triangulation

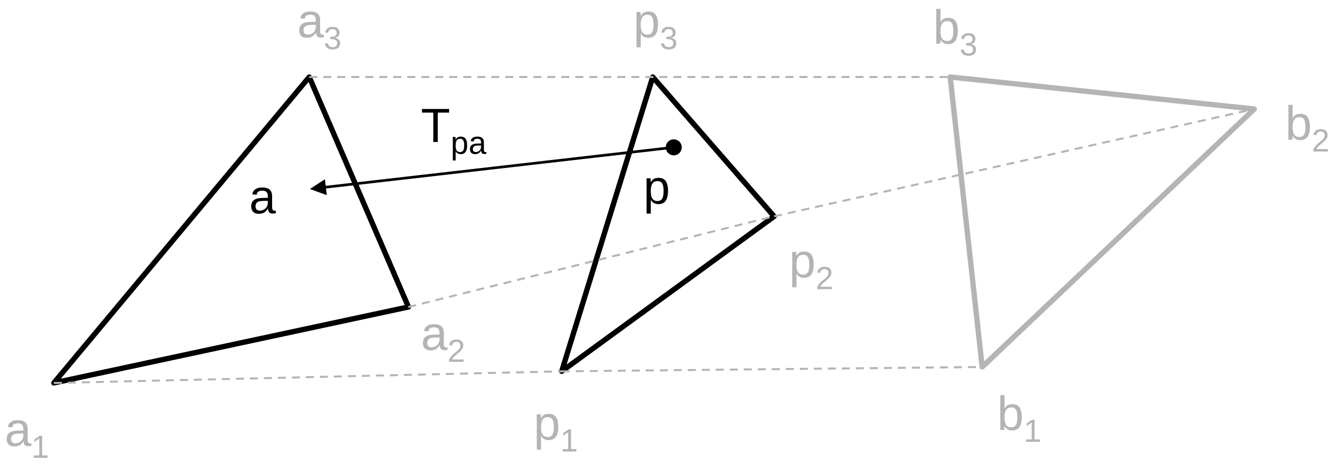


Do Triangle Interpolation



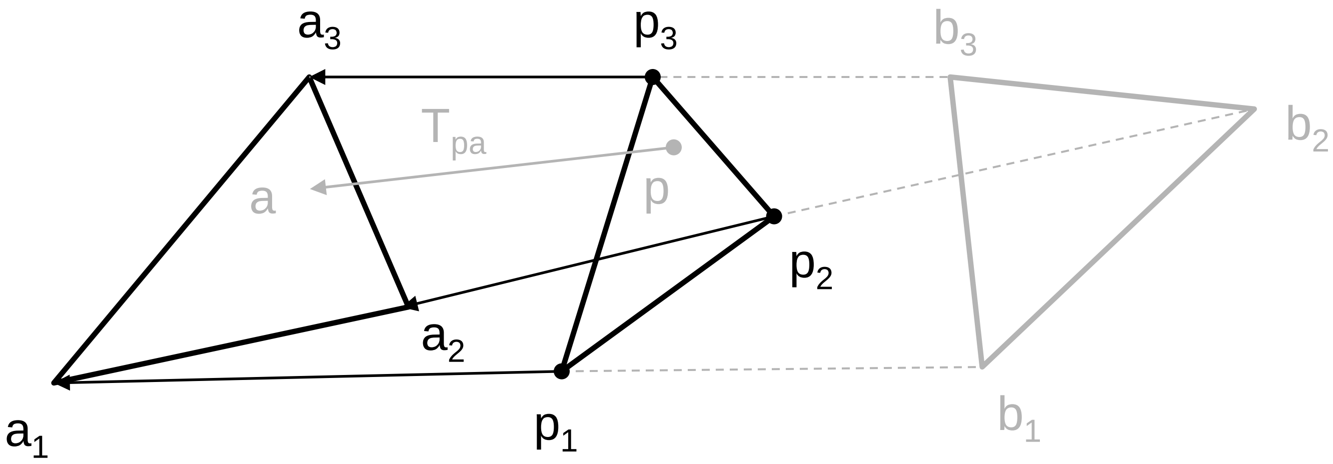
$$p_i = (1 - t)a_i + tb_i$$

Affine Transformation



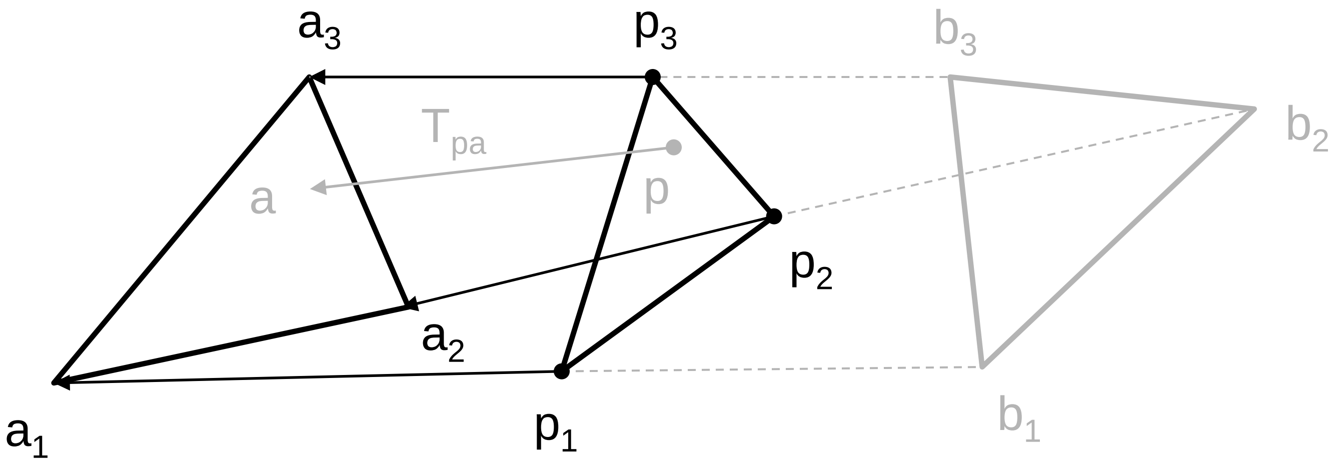
$$\begin{pmatrix} x_a \\ y_a \\ 1 \end{pmatrix} = T_{pa} \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix}$$

Constraints



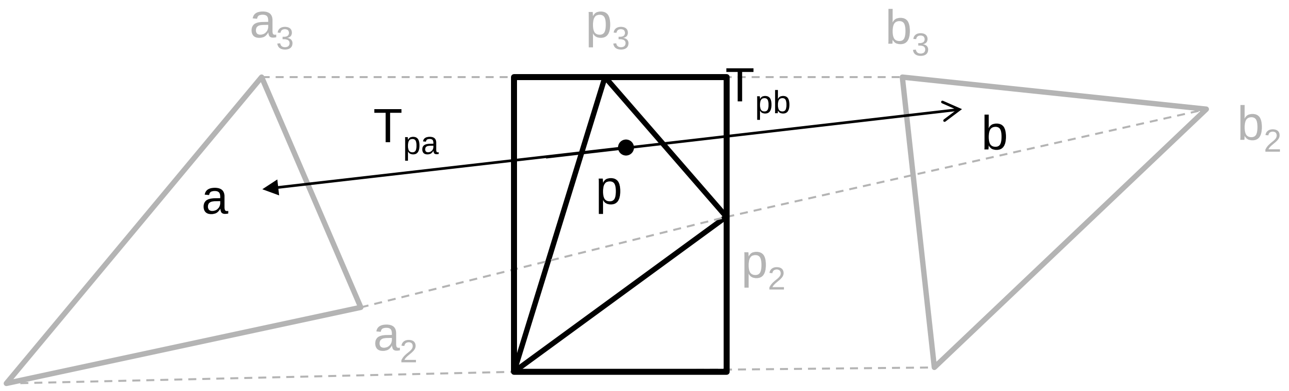
$$\begin{pmatrix} x_{a_i} \\ y_{a_i} \\ 1 \end{pmatrix} = T_{pa} \begin{pmatrix} x_{p_i} \\ y_{p_i} \\ 1 \end{pmatrix}$$

Calculate Affine Matrix



$$T_{pa} = \begin{vmatrix} x_{a1} & x_{a2} & x_{a3} \\ y_{a1} & y_{a2} & y_{a3} \\ 1 & 1 & 1 \end{vmatrix} \begin{vmatrix} x_{p1} & x_{p2} & x_{p3} \\ y_{p1} & y_{p2} & y_{p3} \\ 1 & 1 & 1 \end{vmatrix}^{-1}$$

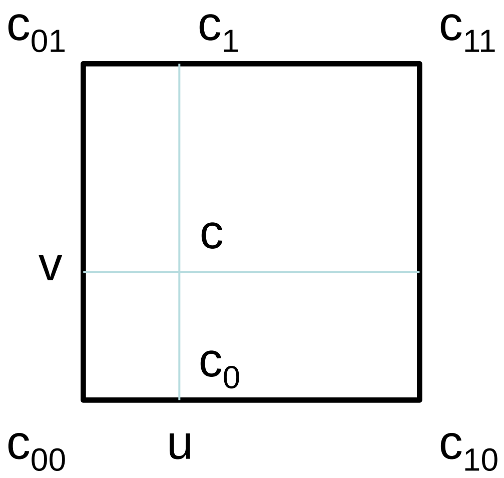
Do Triangle Rasterization



For each point in bounding box

1. Inside triangle test (we provide `rasterize`)
2. Affine transform to source and target triangles
3. Lookup colors using bilinear interpolation
4. Blend using interpolation parameter

Bilinear Interpolation



$$
\begin{aligned}
c_0 &= (1-u)\,c_{00} + u\,c_{10} \\
c_1 &= (1-u)\,c_{01} + u\,c_{11} \\
c &= (1-v)\,c_0 + v\,c_1
\end{aligned}
$$

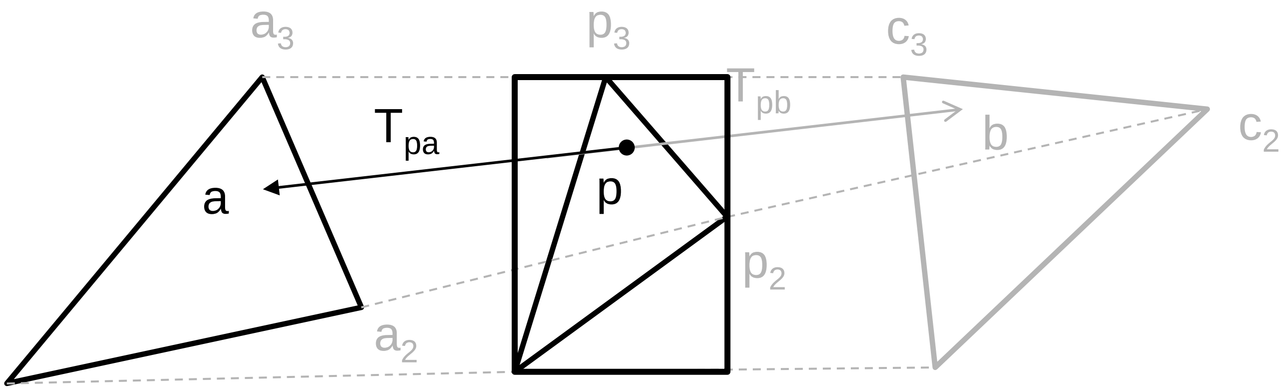
Video Assembly

- Create movie in Matlab
  - Convert image to frame with `im2frame`.
  - Play movie with `movie`.
  - Save movie by creating an avi file with `avifile` and add frames with `addframe`. Don’t forget to close (with `close`) the file.
- Create movie outside Matlab
  - Save images with `imwrite`.
  - Assemble the images with *VirtualDub* (Windows), *SequImago* (Mac) or *ffmpeg*.

Summary

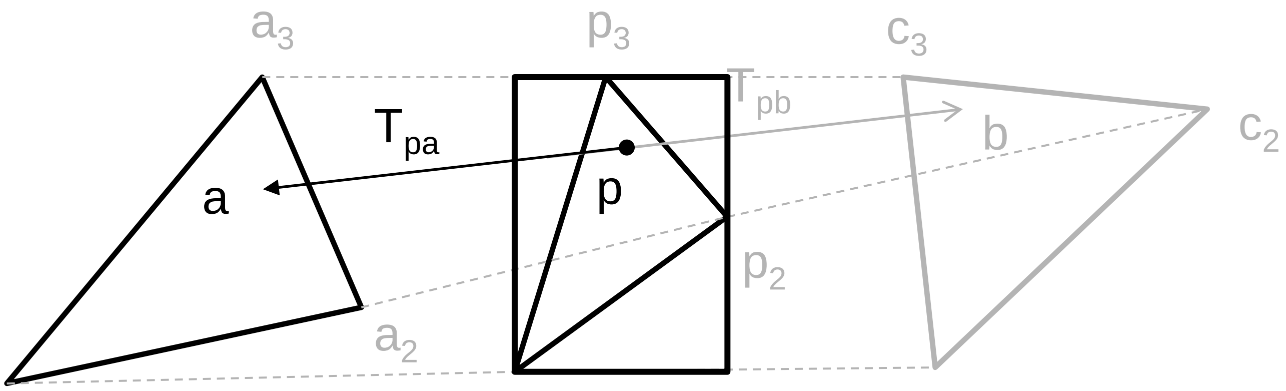
- Load source and target images
- Mark corresponding points: `cpselect`
- Find Delauney triangulation: `delauney`
- For each frame
  - Define interpolation parameters
  - For each triangle
    - Find affine transformations
    - Find pixels in triangle: `rasterize`
    - Color found pixels using the affine transformation. Try to avoid loops here!
- Assemble video: `im2frame`, `avifile`, `addframe`

No ”for each point” loop



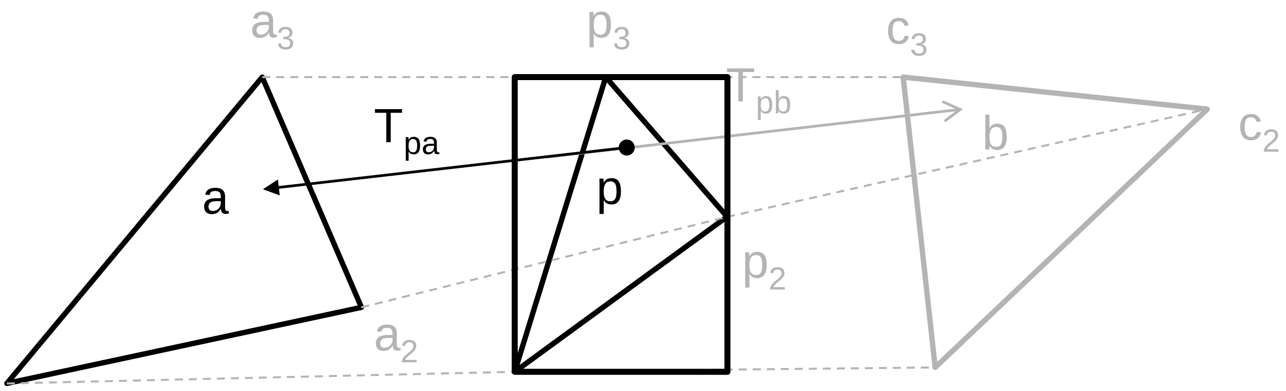
$$\begin{pmatrix} x_a \\ y_a \\ 1 \end{pmatrix} = T_{pa} \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix}$$

No ”for each point” loop



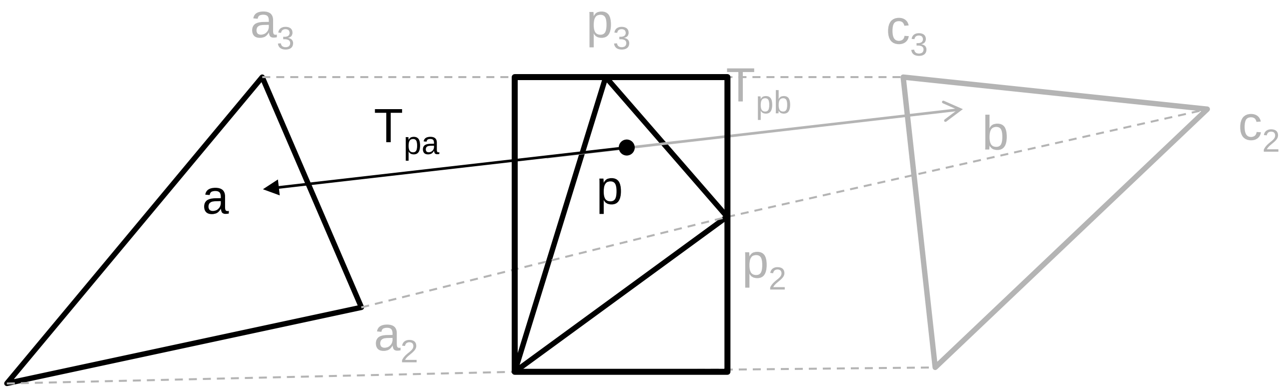
$$\begin{pmatrix} \boxed{x_a} \\ y_a \\ 1 \end{pmatrix} = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix} \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix}$$

No ”for each point” loop



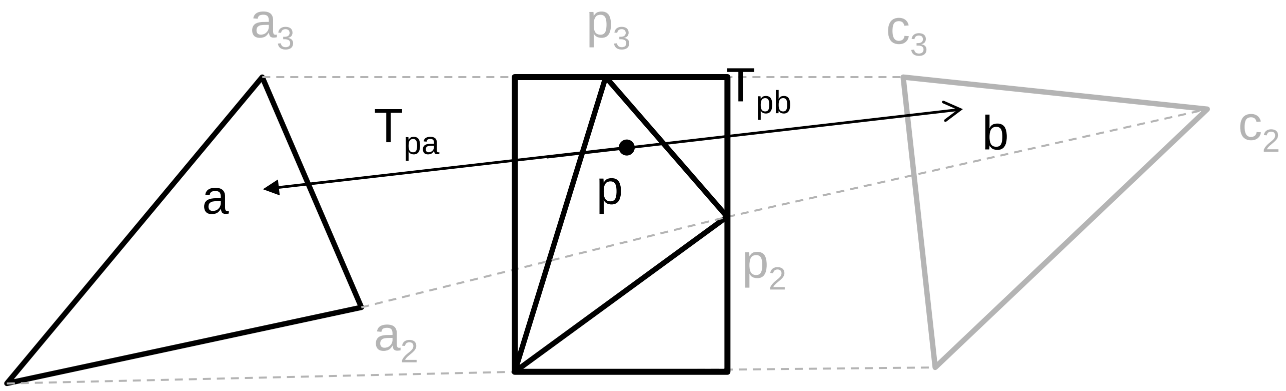
$$\begin{pmatrix}\boxed{x_a}\\ y_a\\ 1\end{pmatrix} = \begin{pmatrix}\boxed{\begin{matrix}t_{11} & t_{12} & t_{13}\end{matrix}}\\ \begin{matrix}t_{21} & t_{22} & t_{23}\end{matrix}\\ \begin{matrix}t_{31} & t_{32} & t_{33}\end{matrix}\end{pmatrix}\begin{pmatrix}\boxed{\begin{matrix}x_p\\ y_p\\ 1\end{matrix}}\end{pmatrix}$$

No ”for each point” loop



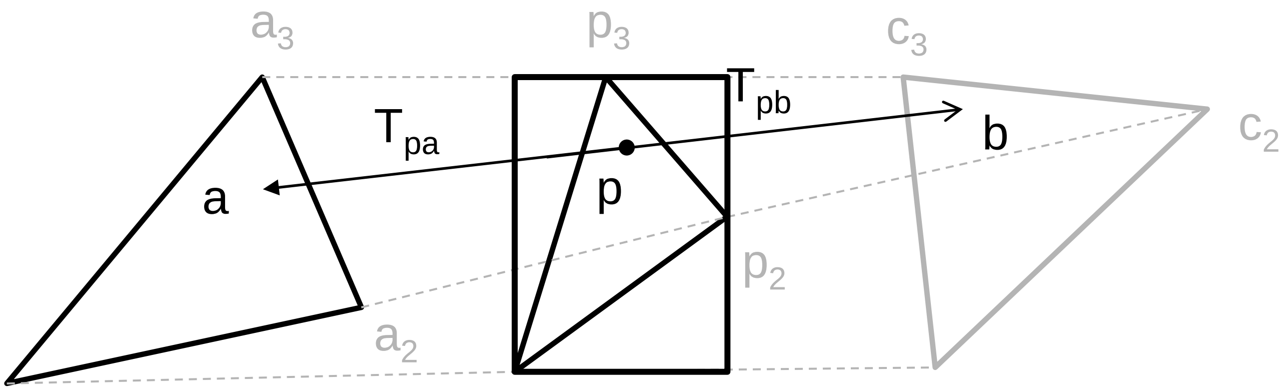
$$\begin{pmatrix} x_a \\ y_a \\ 1 \end{pmatrix} = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix} \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix}$$

No ”for each point” loop



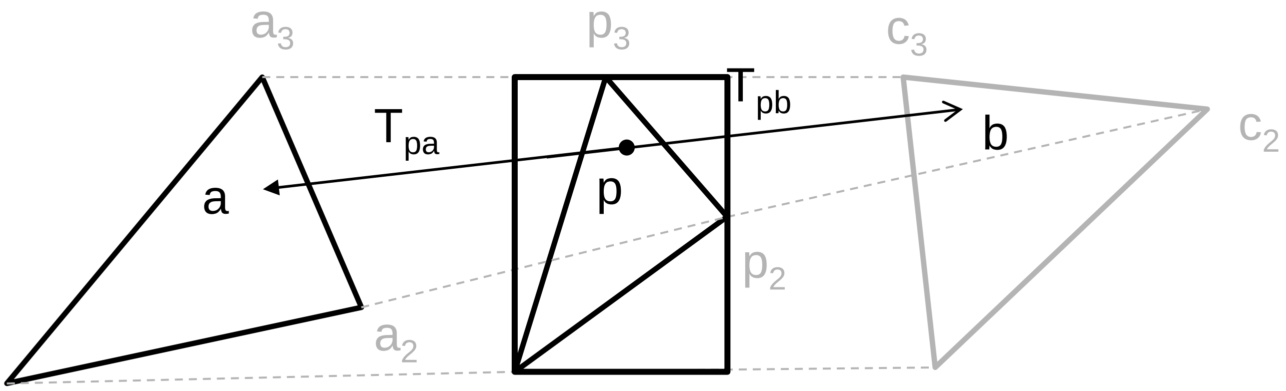
$$\begin{pmatrix} \boxed{x_{a1}} & x_{a2} \\ y_{a1} & x_{a2} \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} \boxed{t_{11}} & \boxed{t_{12}} & \boxed{t_{13}} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix} \begin{pmatrix} \boxed{x_{p1}} & x_{p2} \\ \boxed{y_{p1}} & x_{p2} \\ \boxed{1} & 1 \end{pmatrix}$$

No ”for each point” loop



$$\begin{pmatrix} x_{a1} & x_{a2} \\ y_{a1} & x_{a2} \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix} \begin{pmatrix} x_{p1} & x_{p2} \\ y_{p1} & x_{p2} \\ 1 & 1 \end{pmatrix}$$

No ”for each point” loop



$$\begin{pmatrix} x_{a1} & x_{a2} \\ y_{a1} & x_{a2} \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix} \begin{pmatrix} x_{p1} & x_{p2} \\ y_{p1} & x_{p2} \\ 1 & 1 \end{pmatrix}$$

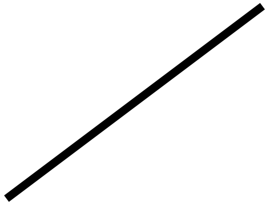
Parallelization

- Call `matlabpool`
- Replace `for` with `parfor`

- 8x speed up with 8 cores
- Constraints:
  - Outer most for loop only
  - Each process (loop iteration) needs to be independent

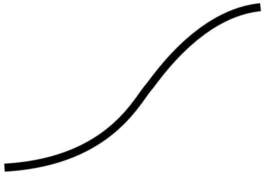
Define Interpolation Parameter

Linear ramp



$$t = \frac{index - 1}{count - 1}$$

Cosine ramp



$$t' = \frac{1 - \cos \pi t}{2}$$

Debug Tipps

- Use only imshow instead of real frames
- Only one triangle
- Show delauney triangulation in each frame.

Demo

More Examples

Image Rectification

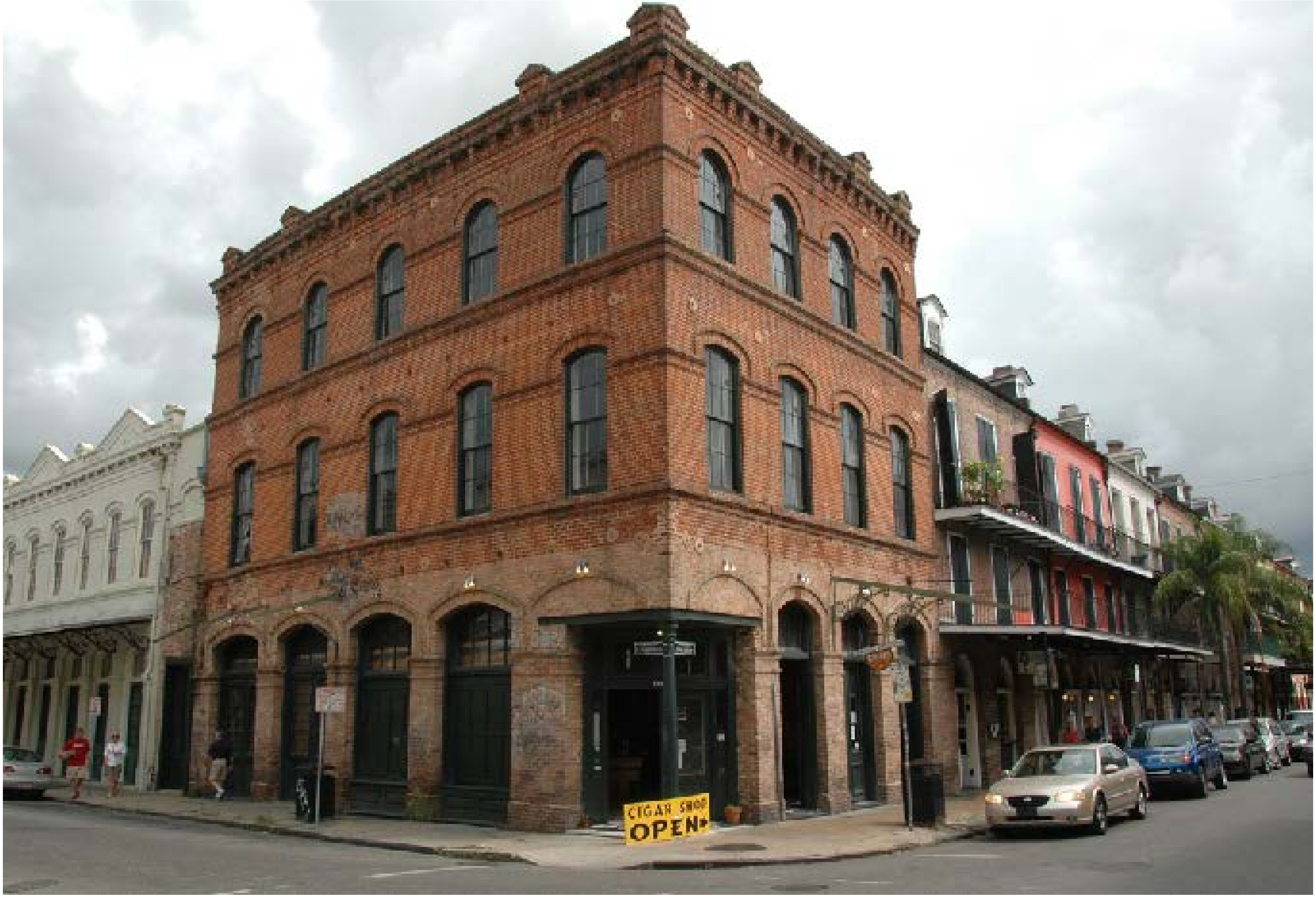


Image Rectification

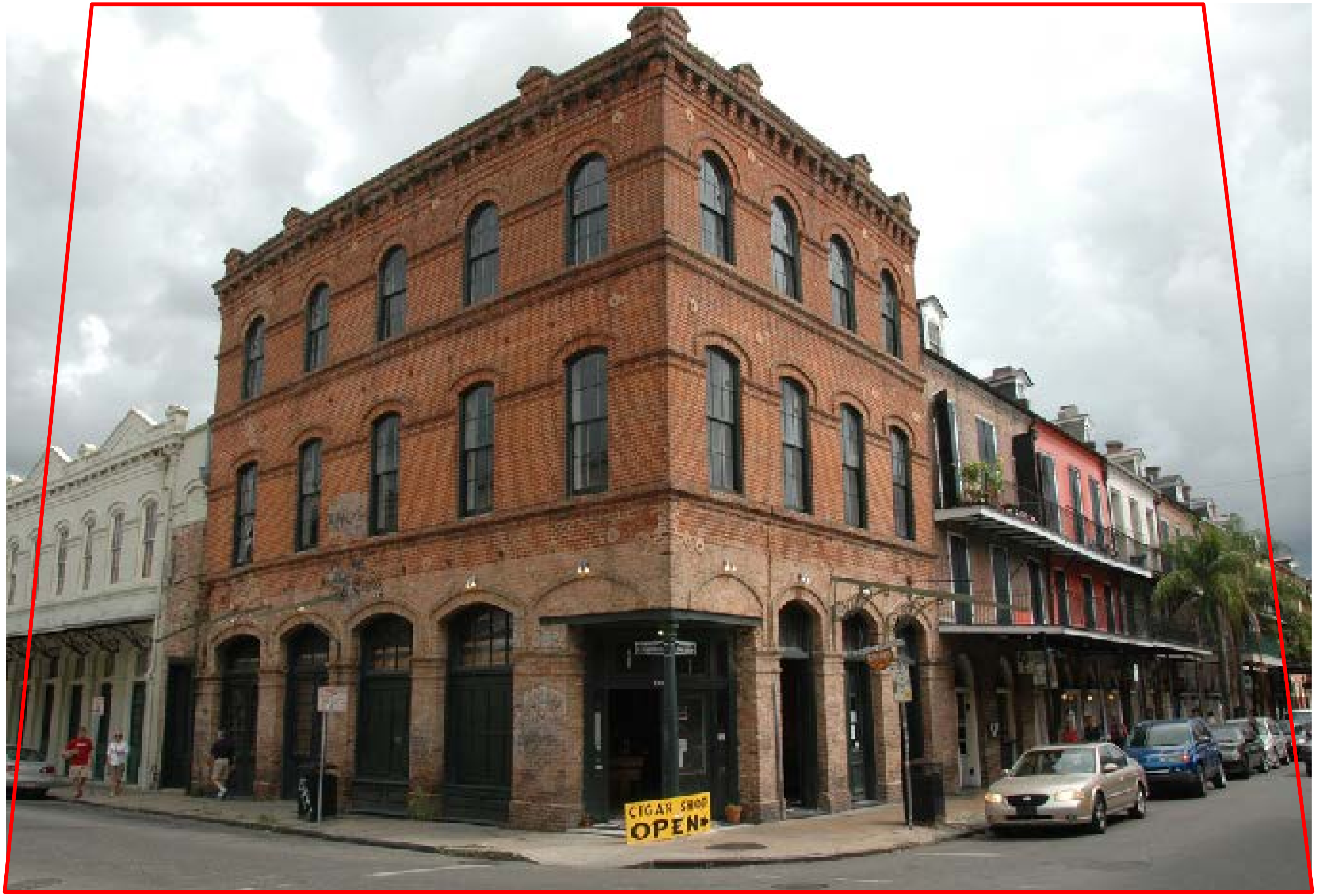


Image Rectification

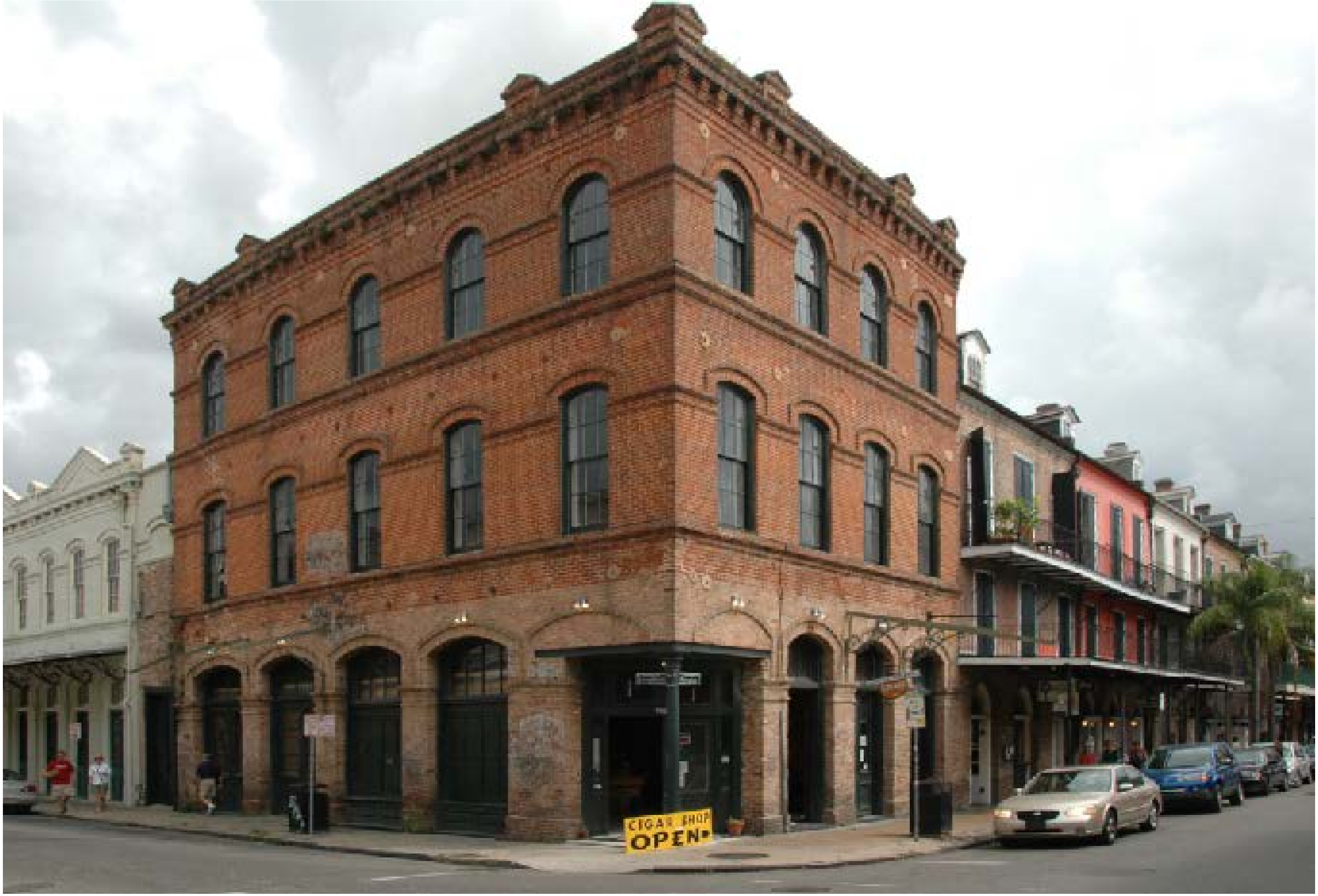
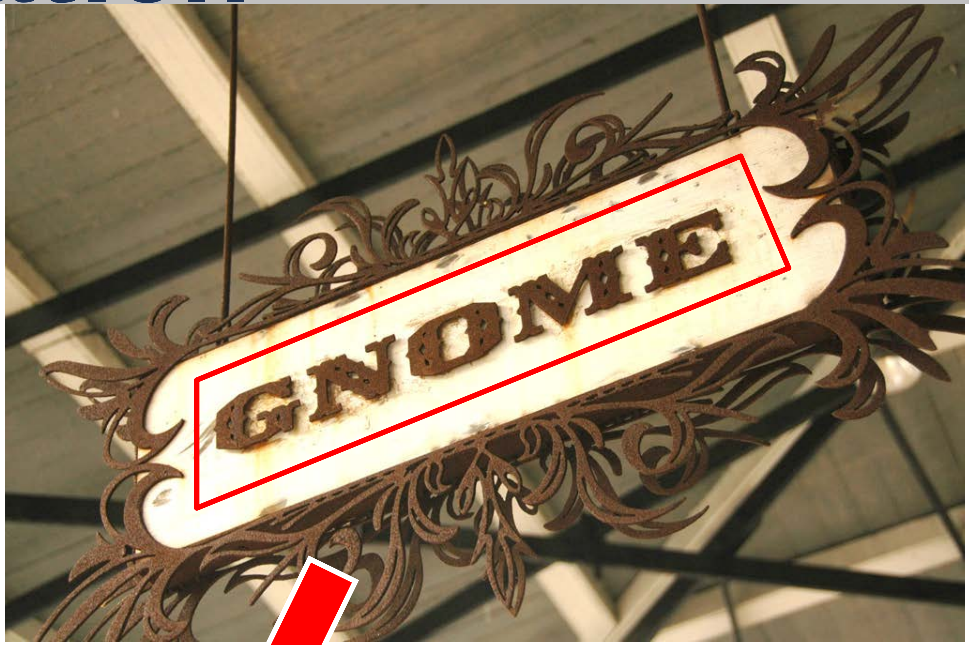


Image Rectification



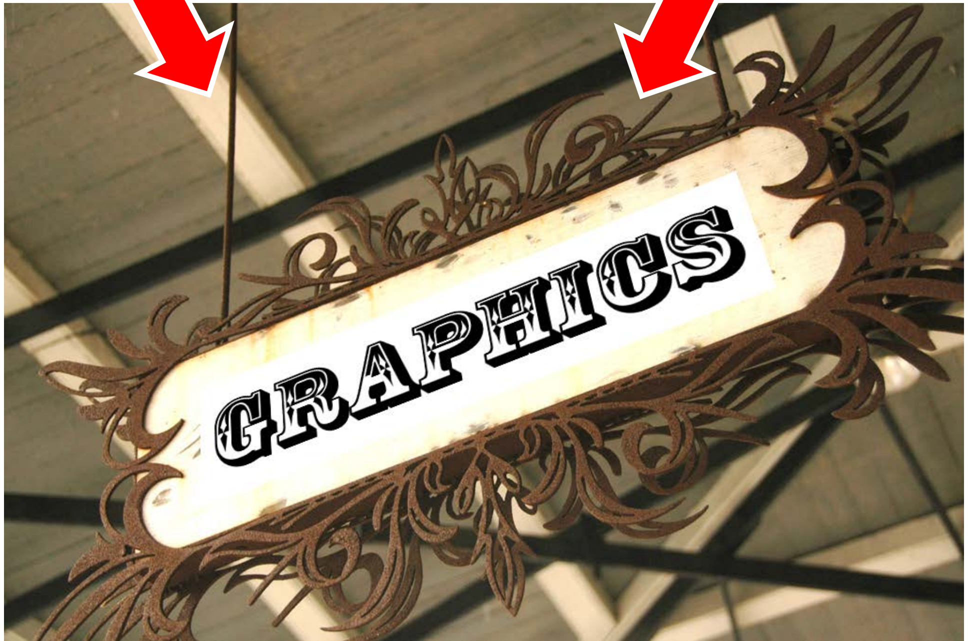
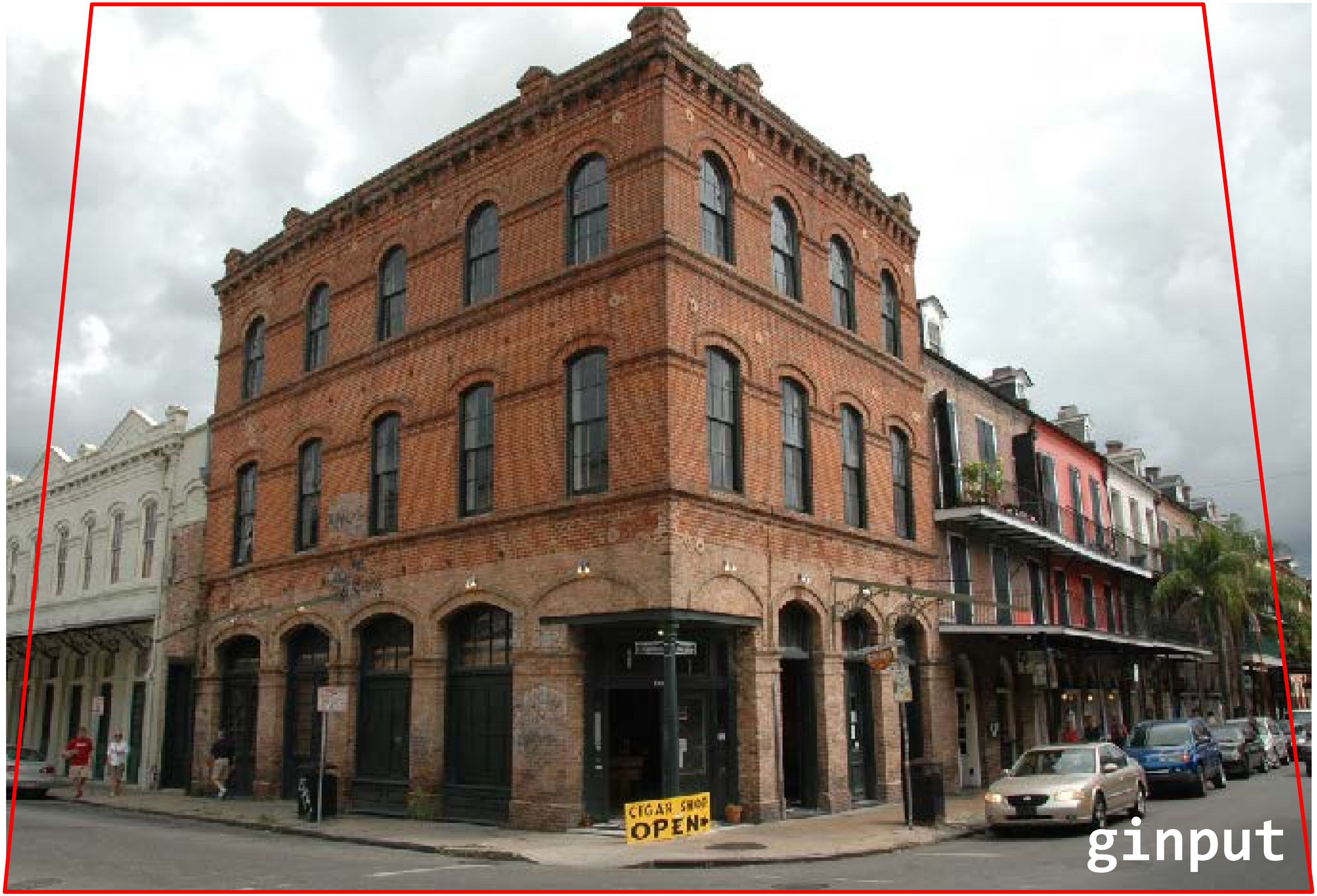


Image Rectification

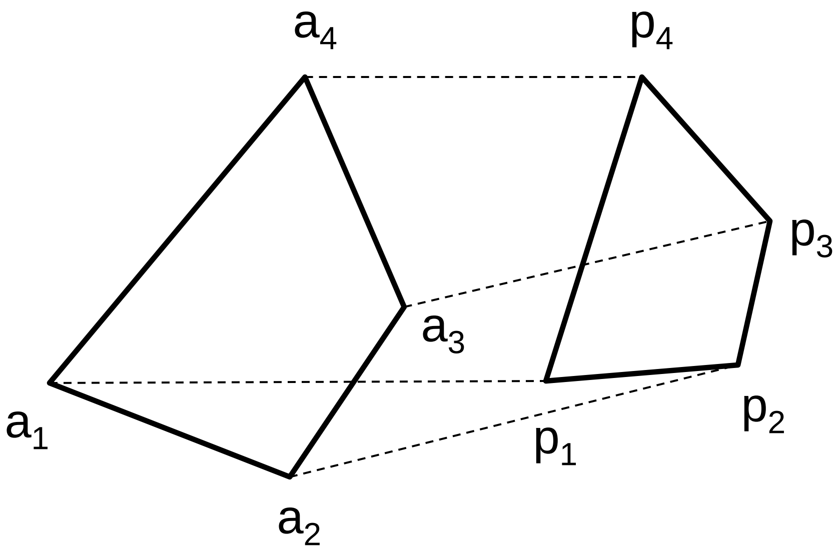


ginput

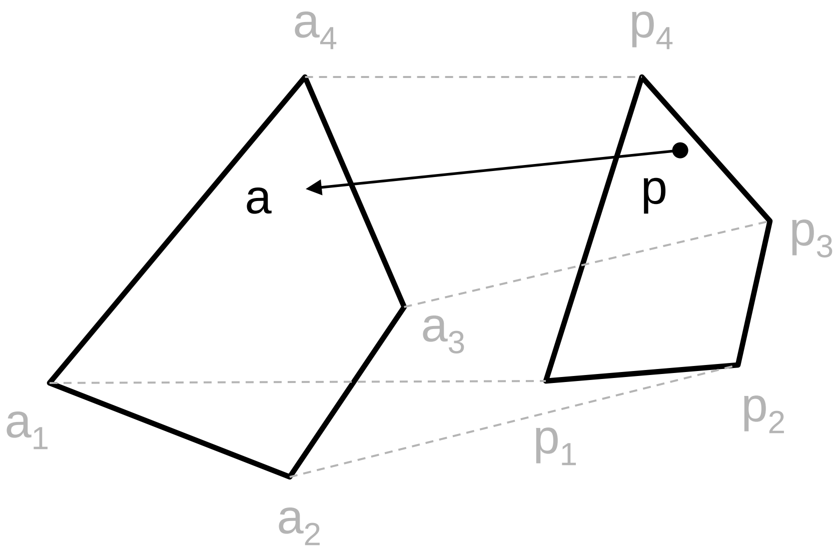
Give clear instruction

- Describe the order the points need to be selected as figure title or something similar OR
- The application is able to find out the order by itself.

4 Corresponding Points

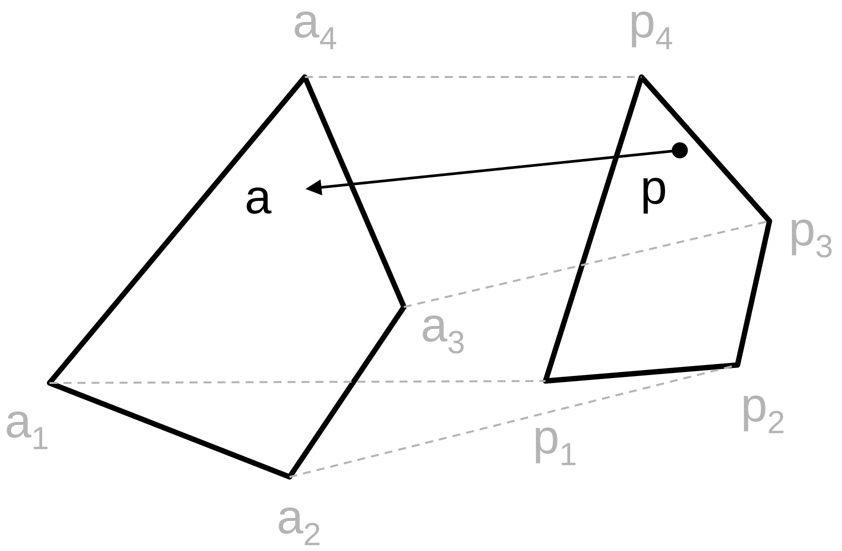


Homography Transform



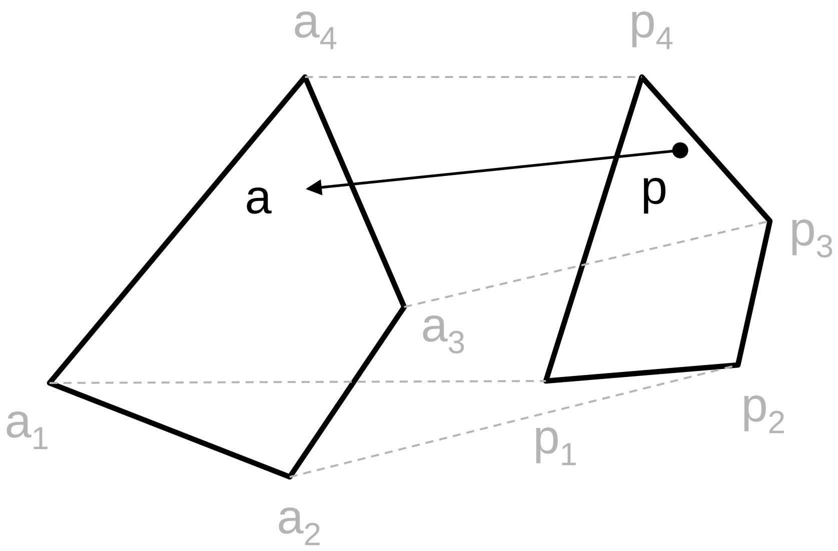
$$\begin{pmatrix} a_x a_z \\ a_y a_z \\ a_z \end{pmatrix} = H_{pa} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix}$$

Homography Matrix



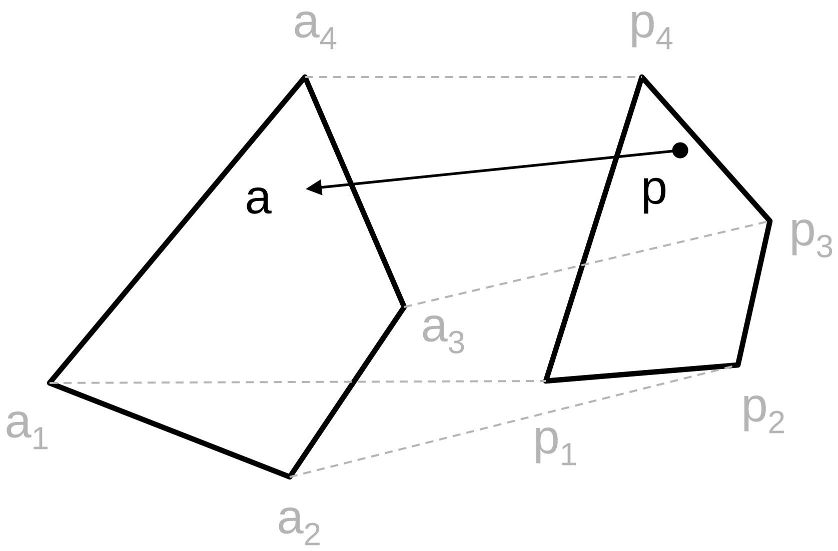
$$\begin{pmatrix} a_x a_z \\ a_y a_z \\ a_z \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix}$$

Homography Matrix



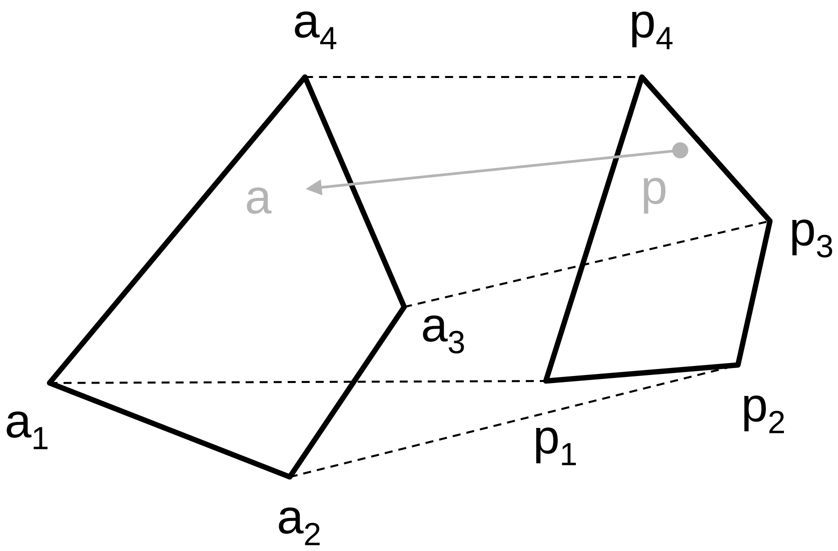
$$\begin{pmatrix} a_x a_z \\ a_y a_z \\ a_z \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & \boxed{1} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix}$$

Homography Matrix



$$\begin{pmatrix} -p_x & -p_y & -1 & 0 & 0 & 0 & a_x p_x & a_x p_y & a_x \\ 0 & 0 & 0 & -p_x & -p_y & -1 & a_y p_x & a_y p_y & a_y \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{32} \\ 1 \end{pmatrix} = 0$$

Homography Matrix



$$\begin{pmatrix} -p_{1x} & -p_{1y} & -1 & 0 & 0 & 0 & a_{1x}p_{1x} & a_{1x}p_{1y} & a_{1x} \\ 0 & 0 & 0 & -p_{1x} & -p_{1y} & -1 & a_{1y}p_{1x} & a_{1y}p_{1y} & a_{1y} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -p_{4x} & -p_{4y} & -1 & a_{4y}p_{4x} & a_{4y}p_{4y} & a_{4y} \end{pmatrix} \begin{pmatrix} h_{11} \\ \vdots \\ h_{32} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

Singular Value Decomposition

- Solve the equation system
  - singular value decomposition (svd)

$$A = USV \quad h = V_{:,9}$$

  - or backslash operator \

$$\begin{pmatrix} -p_{1x} & -p_{1y} & -1 & 0 & 0 & 0 & a_{1x}p_{1x} & a_{1x}p_{1y} & a_{1x} \\ 0 & 0 & 0 & -p_{1x} & -p_{1y} & -1 & a_{1y}p_{1x} & a_{1y}p_{1y} & a_{1y} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -p_{4x} & -p_{4y} & -1 & a_{4y}p_{4x} & a_{4y}p_{4y} & a_{4y} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{32} \\ h_{33} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

Rasterize Image

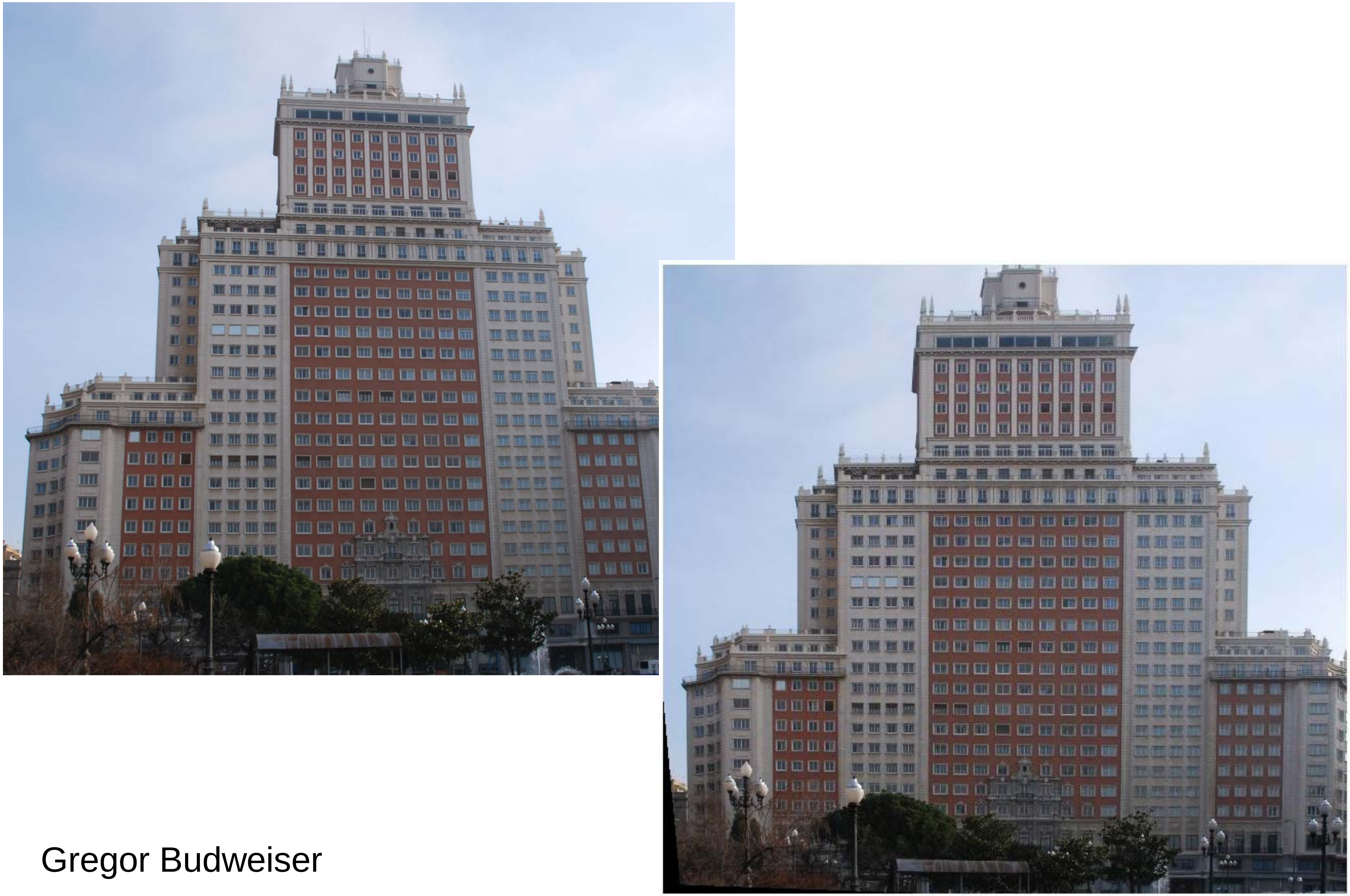
For every pixel

1. Transform using homography to source image
2. Lookup colors using bilinear interpolation
3. Color pixel

Summary

- Load source image(s)
- Select 4 corresponding points: `ginput`
- Find homography between quadriliterals
- Rasterize image

Previous years' Top Results



Gregor Budweiser

Previous years' Top Results

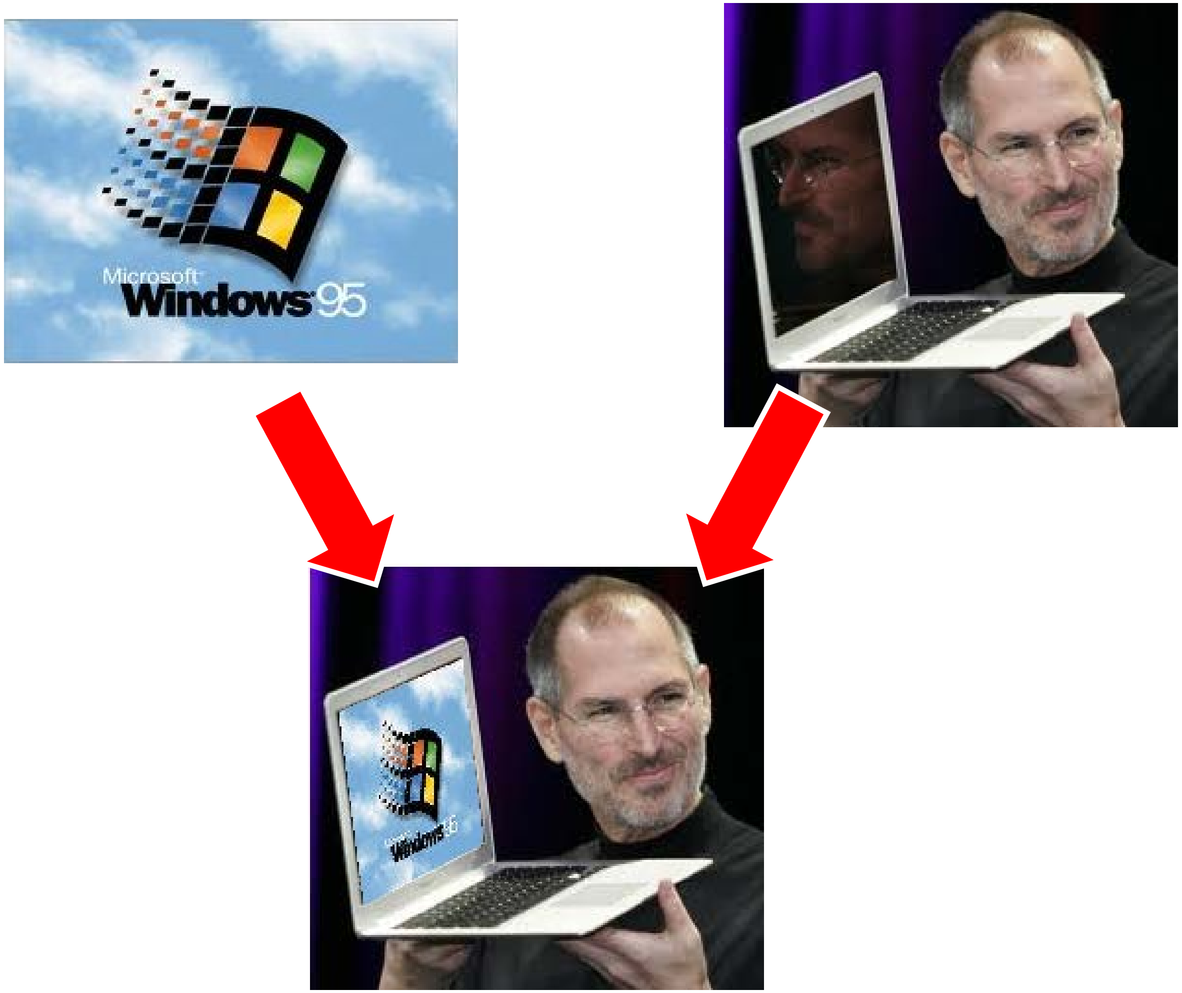
Daniel Frey

Previous years' Top Results

Tiziano Portenier

Previous years' Top Results

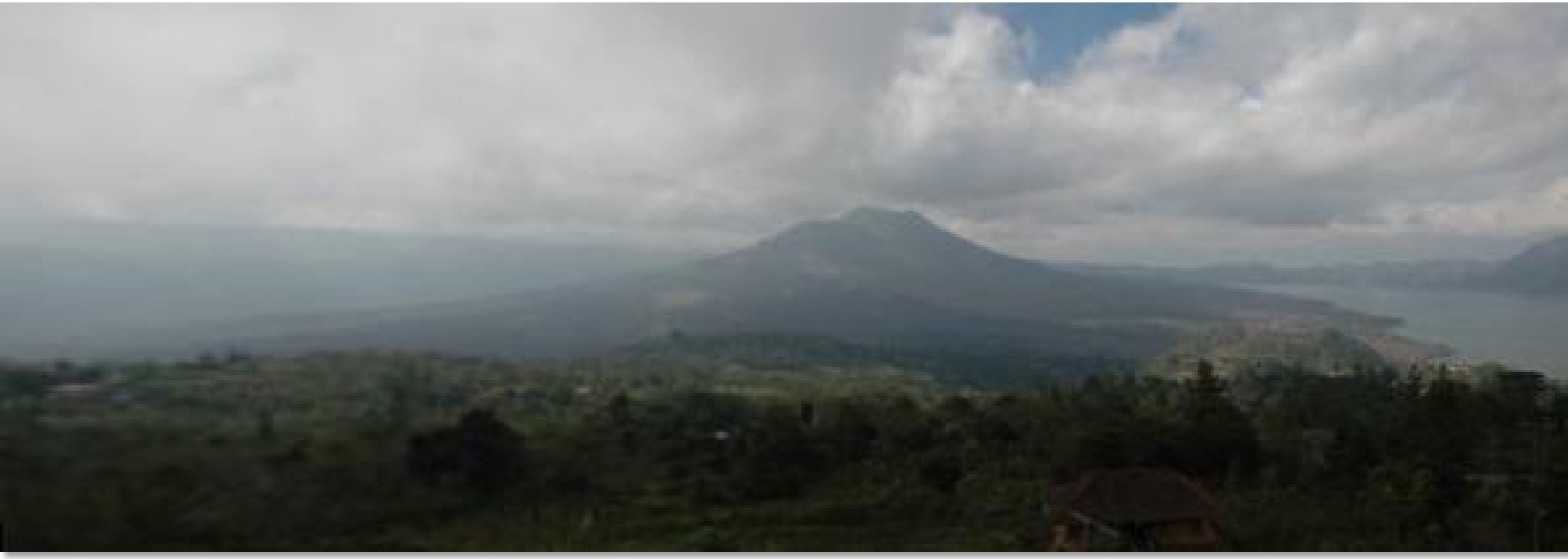
Previous years' Top Results



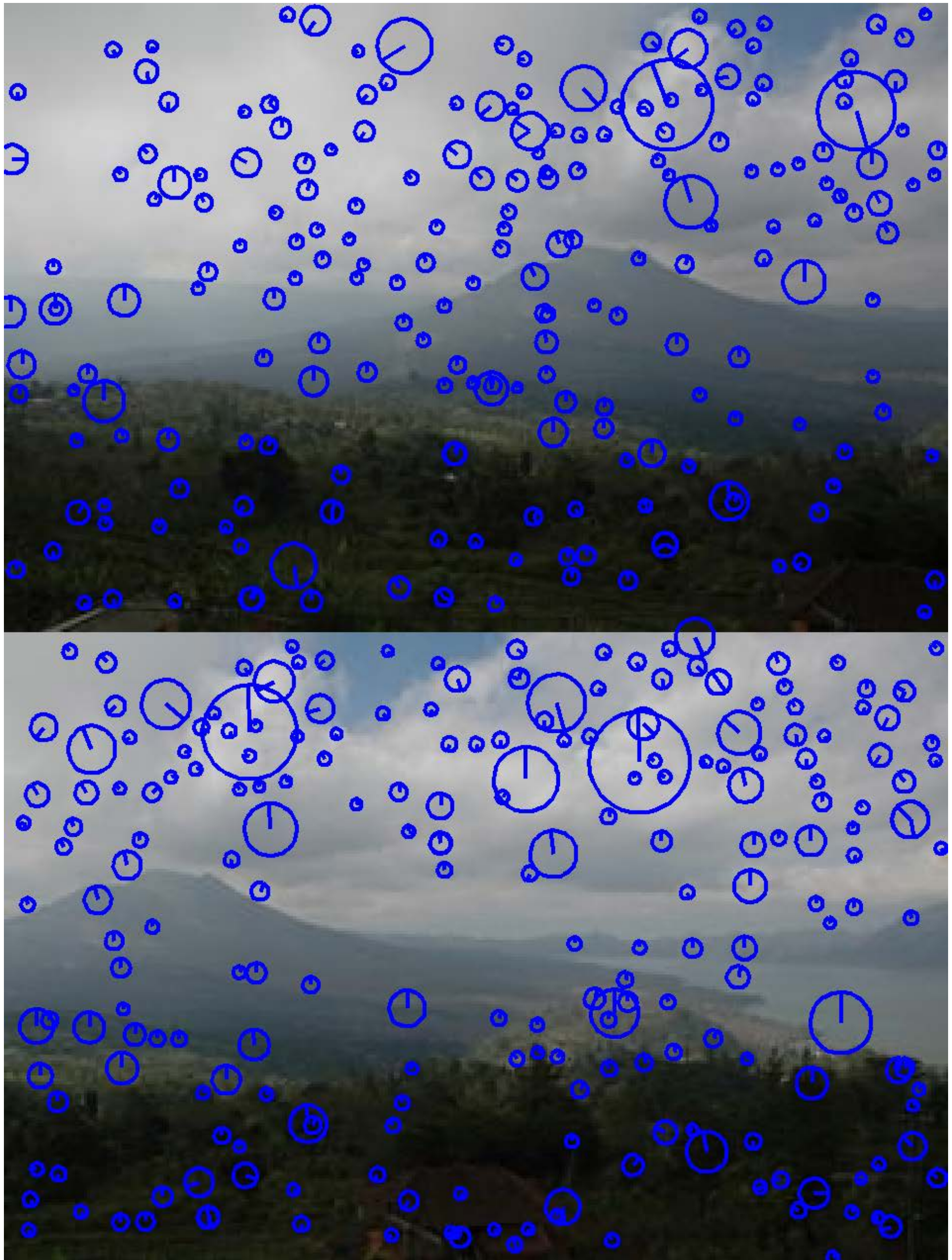
Steffen Schumann

Panorama Stitching

Panorama Stitching

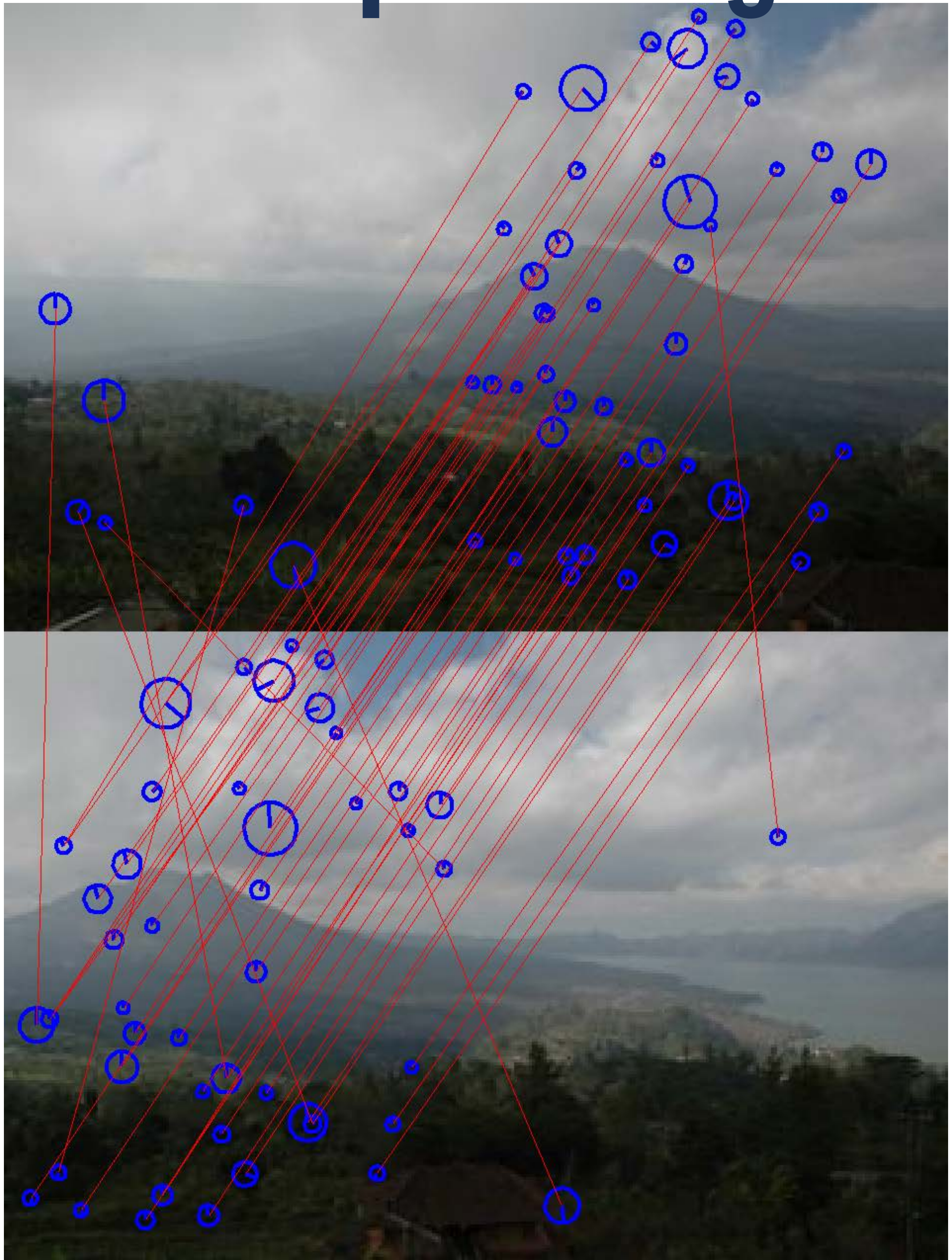


Feature Points



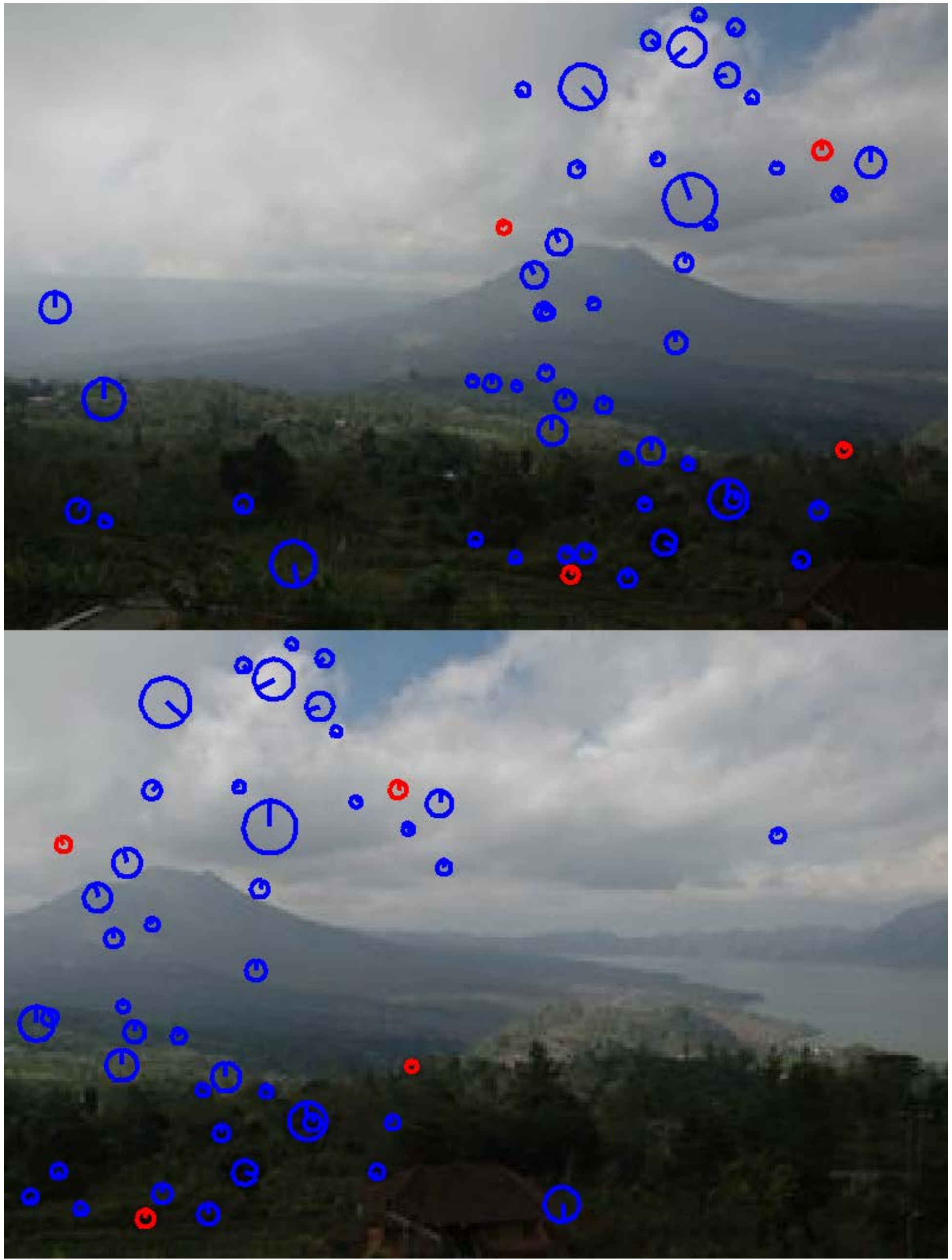
vl_sift

Corresponding Points



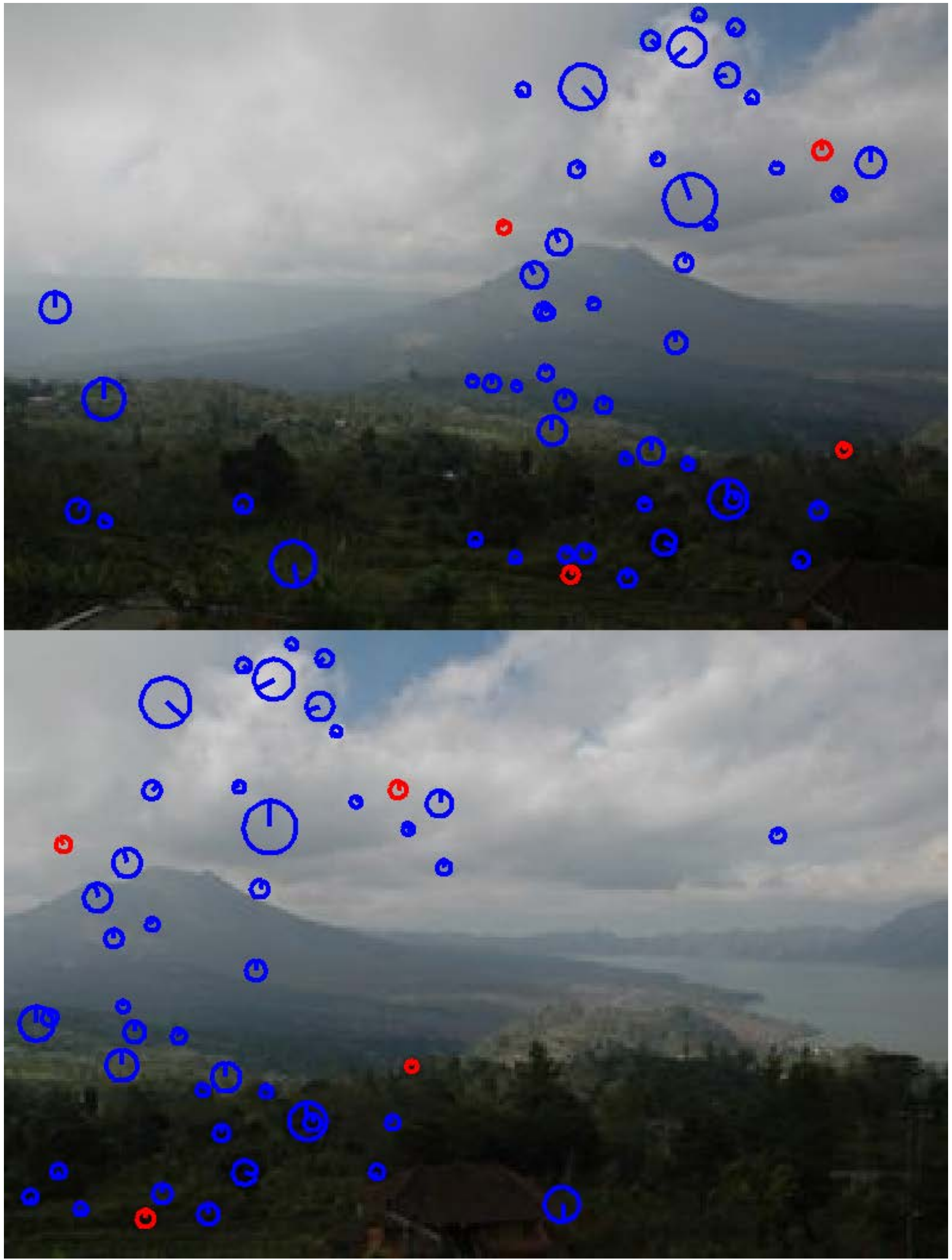
vl_ucbmatch

RANSAC



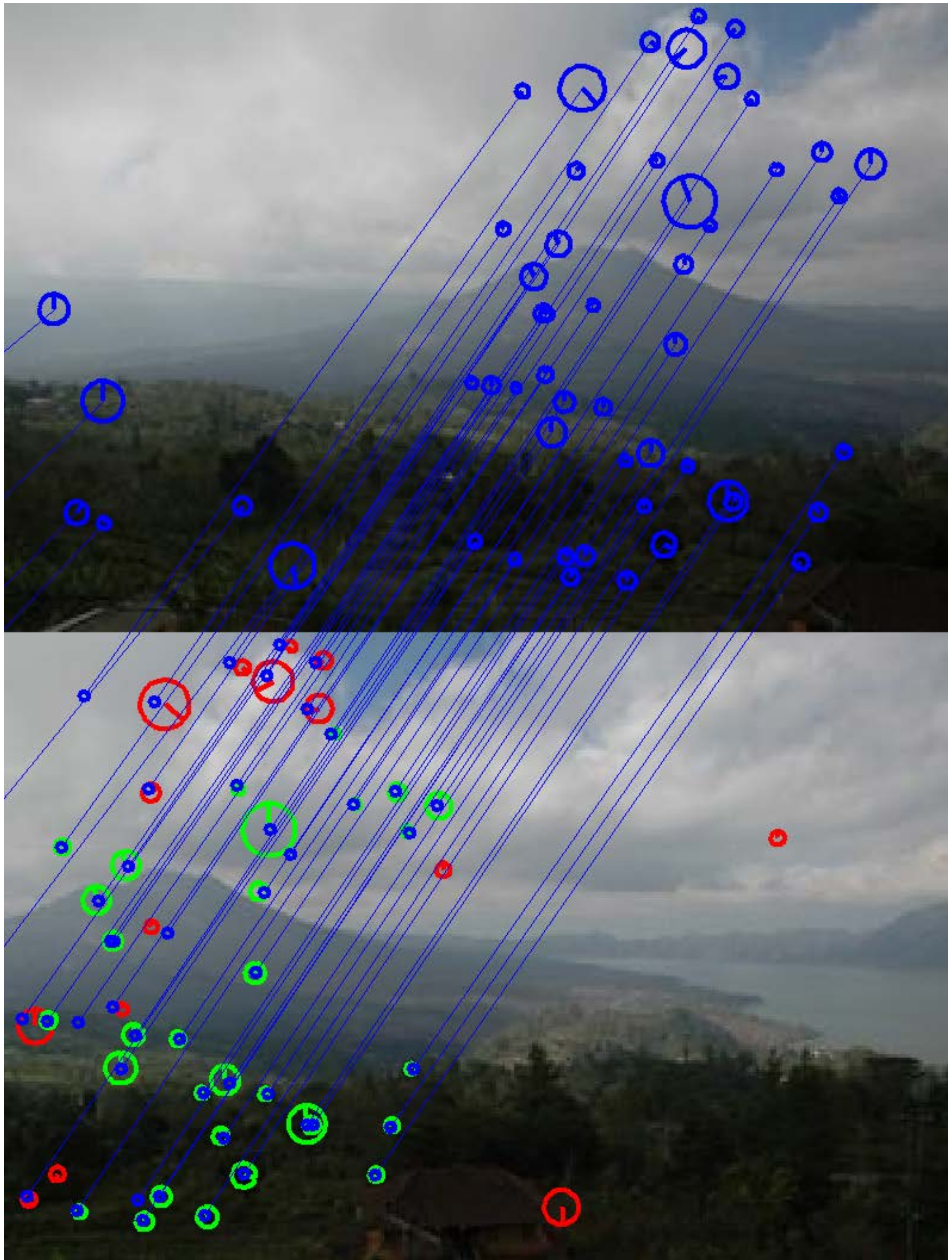
1. Choose 4 random corresponding points

RANSAC



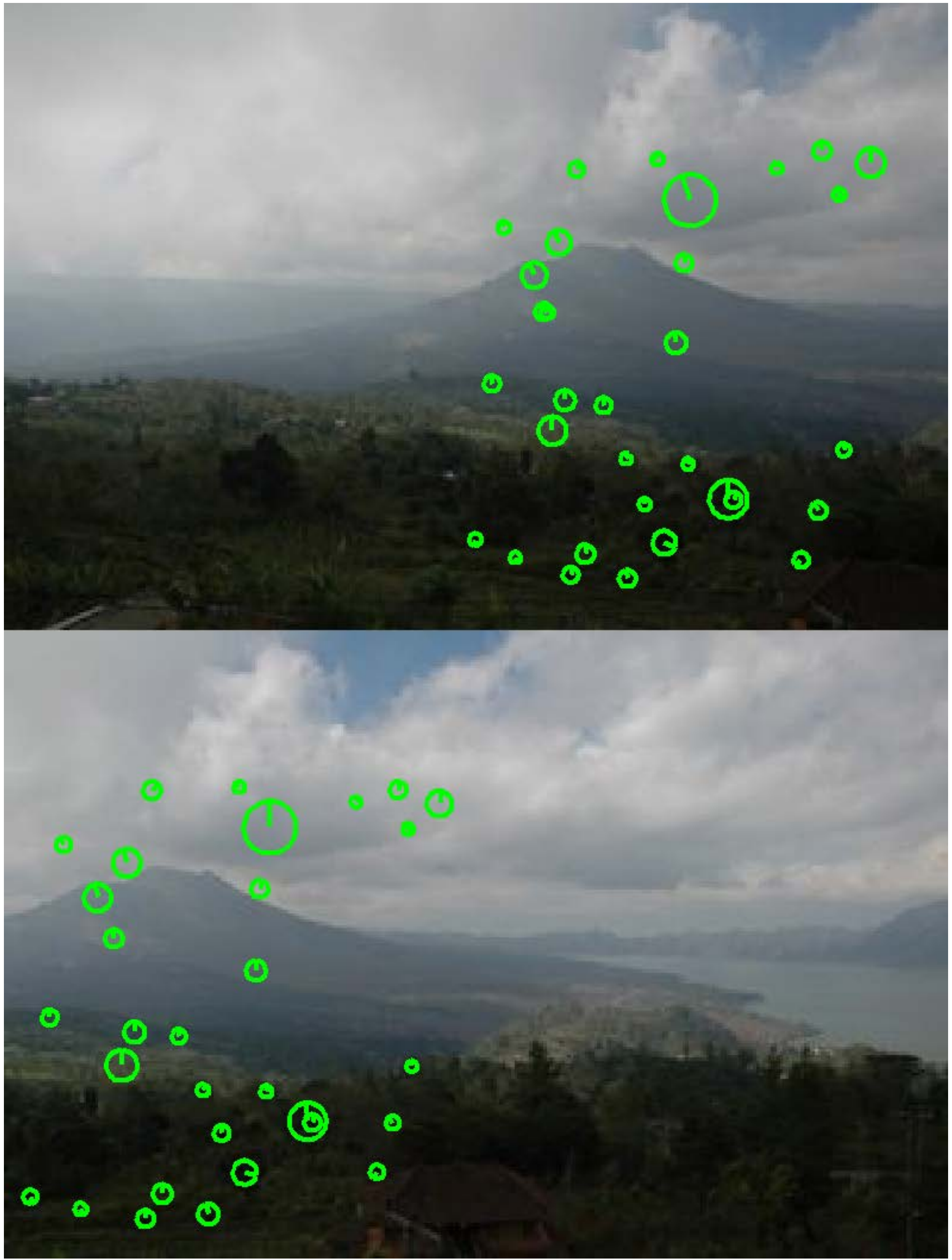
1. Choose 4 random corresponding points
2. Find homography

RANSAC



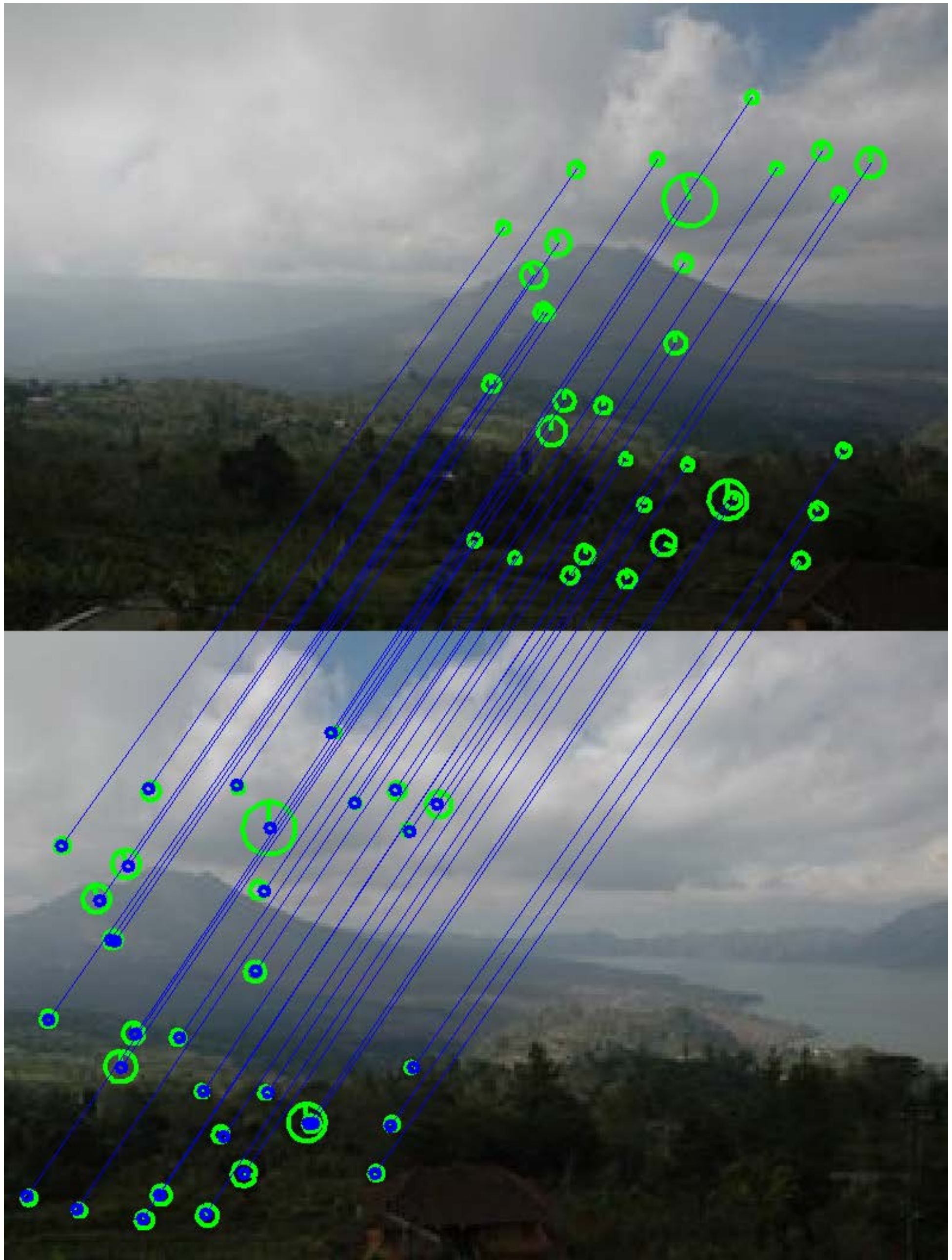
1. Choose 4 random corresponding points
2. Find homography
3. Transform points to other image using homography

RANSAC



1. Choose 4 random corresponding points
2. Find homography
3. Transform points to other image using homography
4. Count inliers based on Euclidean distance

RANSAC



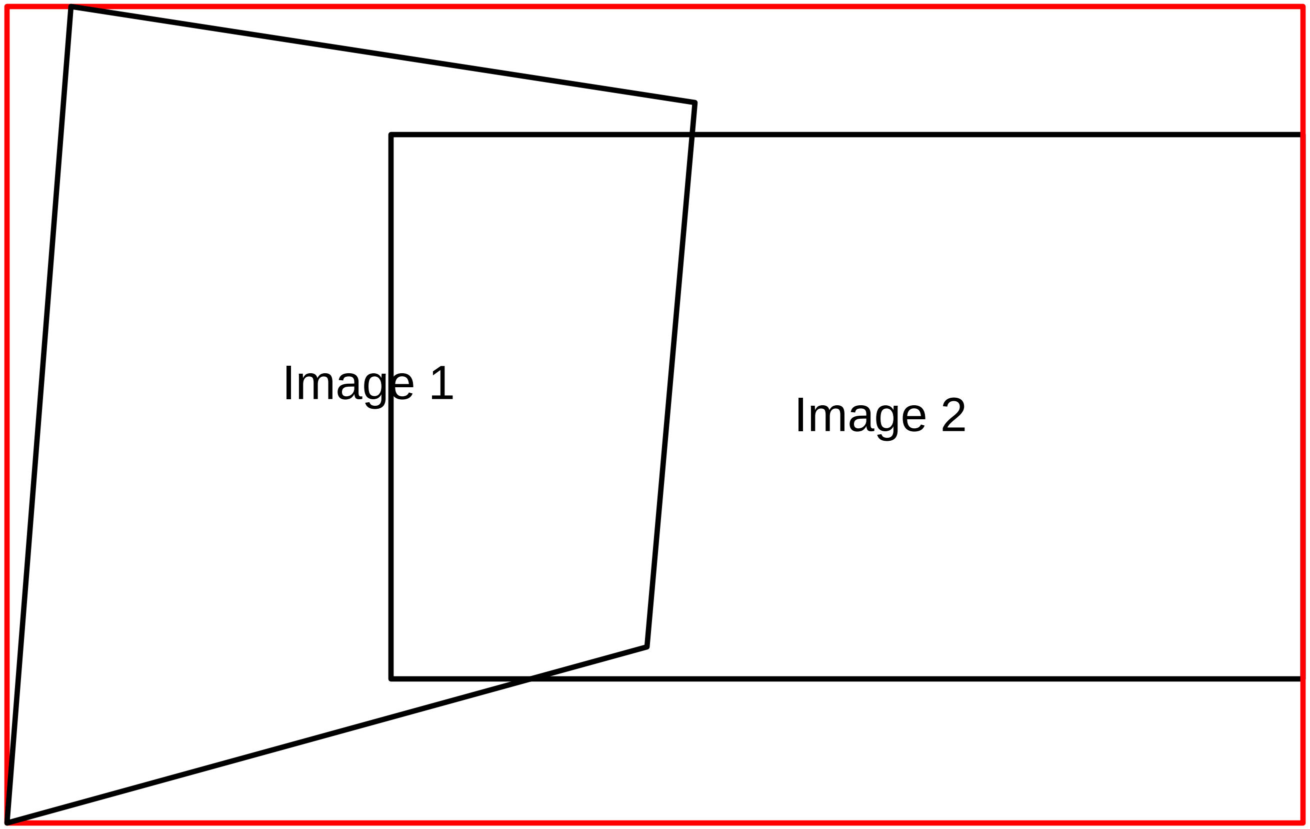
1. Choose 4 random corresponding points
2. Find homography
3. Transform points to other image using homography
4. Count inliers based on Euclidean distance
5. Keep homography with most inliers

RANSAC

For many iterations

1. Choose four random corresponding points
2. Find homography
3. Transform using homography points from one image to the other
4. Count inliers based on Euclidean distance
5. Keep homography with max. Inliers

Image Size



Rasterize Image

For every pixel

1. Transform to source image using homography
2. Bounding box test
3. Lookup colors using bilinear interpolation
4. Blend image 1/image 2

Image Blending

Image Blending

- How to blend two images?

Image Blending

- How to blend two images?

```
mask =
     1     1     1     1     1     1     1     1
     1     2     2     2     2     2     2     1
     1     2     3     3     3     3     2     1
     1     2     3     3     3     3     2     1
     1     2     2     2     2     2     2     1
     1     1     1     1     1     1     1     1
```

Image Blending

- How to blend two images?

```
mask =

     1     1     1     1     1     1     1     1
     1     2     2     2     2     2     2     1
     1     2     3     3     3     3     2     1
     1     2     3     3     3     3     2     1
     1     2     2     2     2     2     2     1
     1     1     1     1     1     1     1     1

mask = mask ./ max(max(mask));
figure
imshow(mask);
```

imshow(mask);

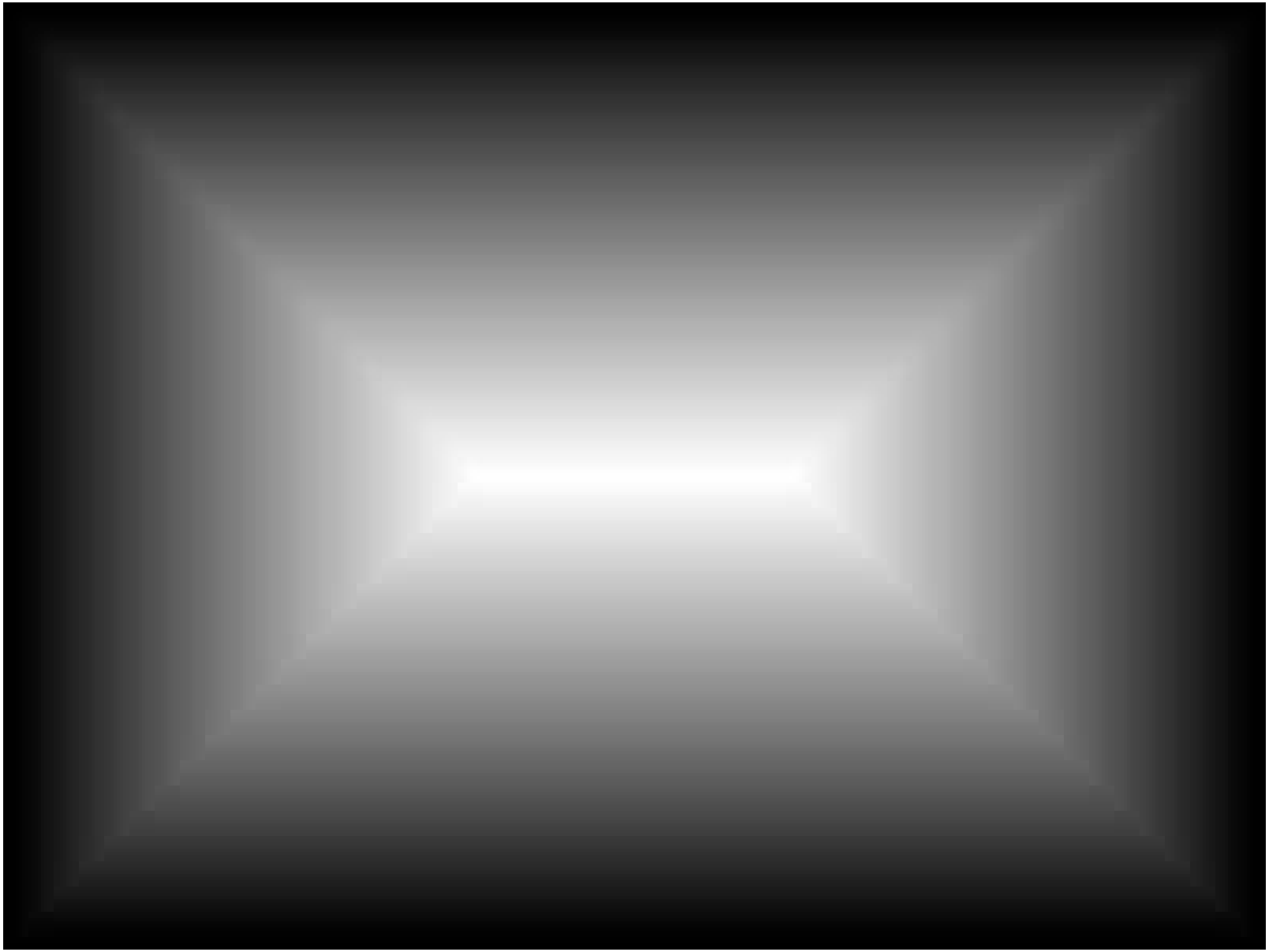
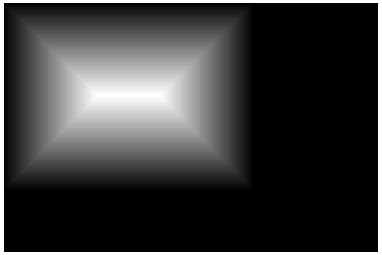
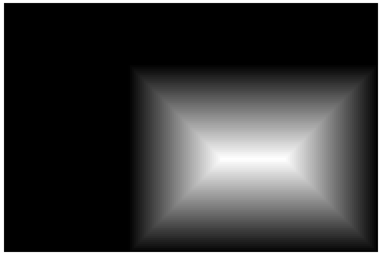


Image Blending





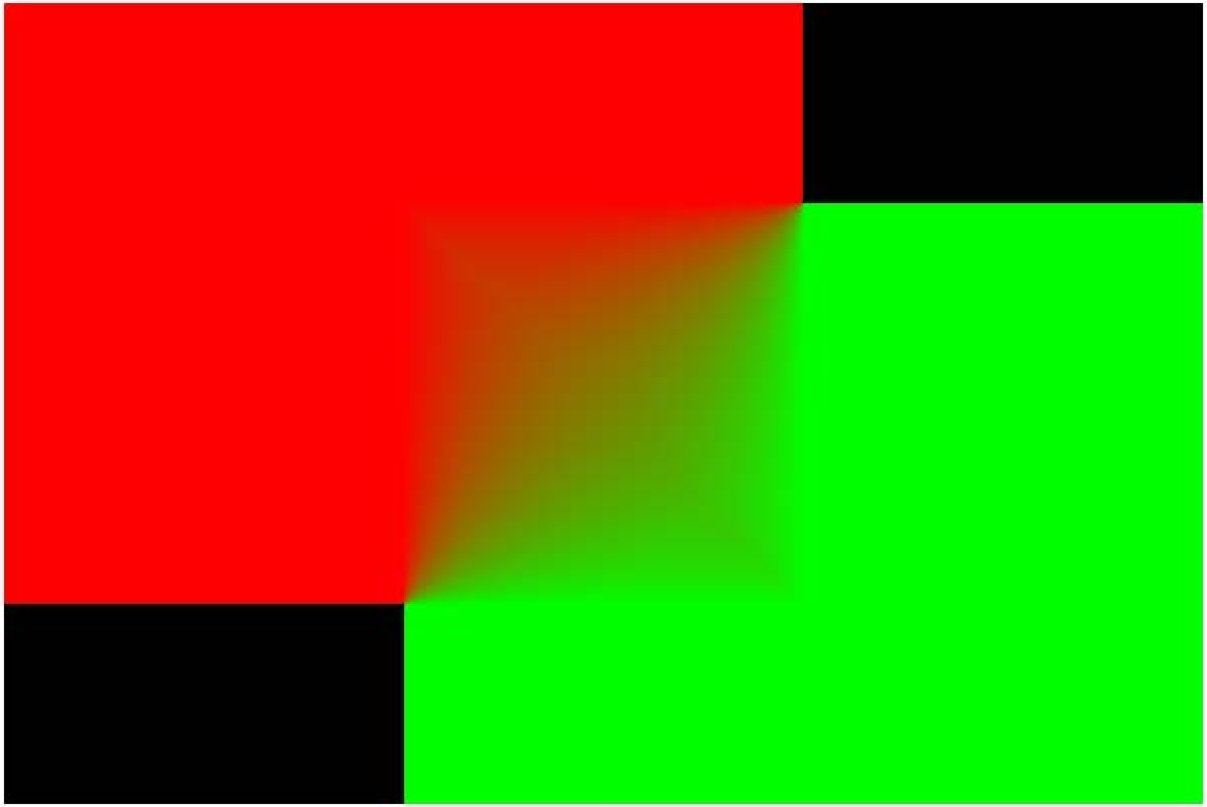
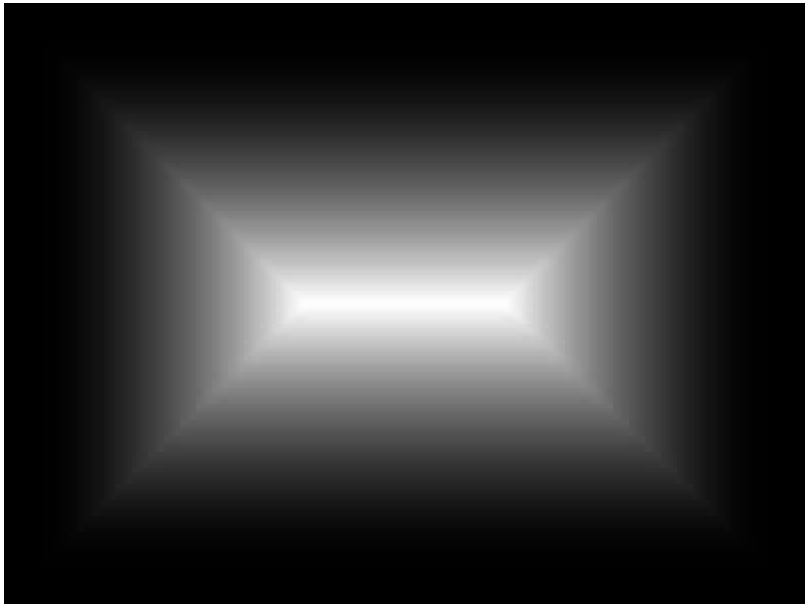


Image Blending

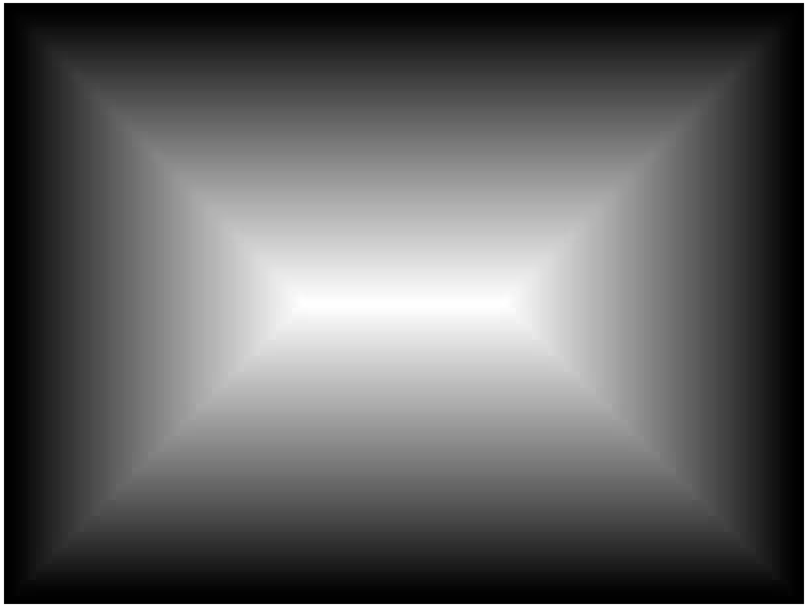
```
mask = mask .* mask;
mask =
     1     1     1     1     1     1     1     1
     1     4     4     4     4     4     4     1
     1     4     9     9     9     9     4     1
     1     4     9     9     9     9     4     1
     1     4     4     4     4     4     4     1
     1     1     1     1     1     1     1     1

mask = mask ./ max(max(mask));
figure
imshow(mask);
```

imshow(mask);



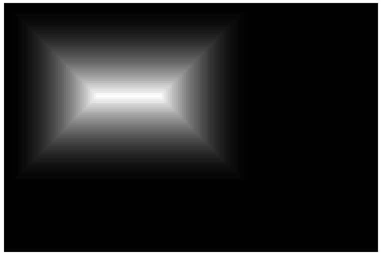
quadratic mask

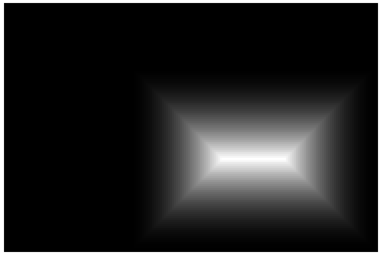


linear mask

Image Blending

quadratic mask





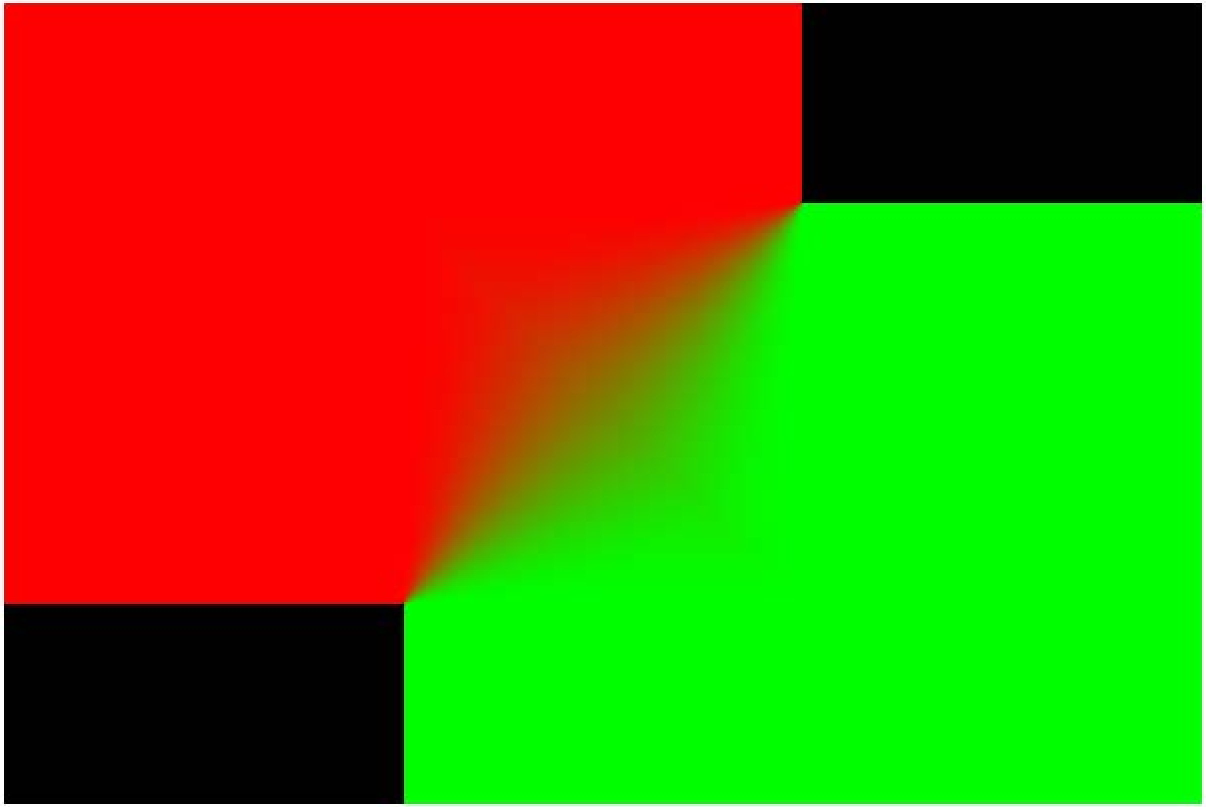
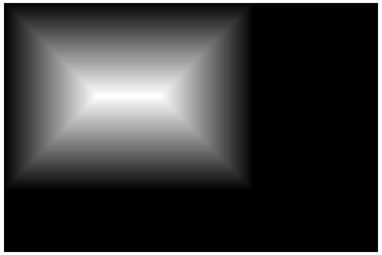
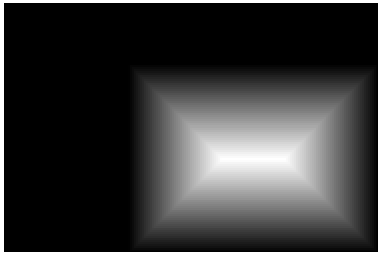
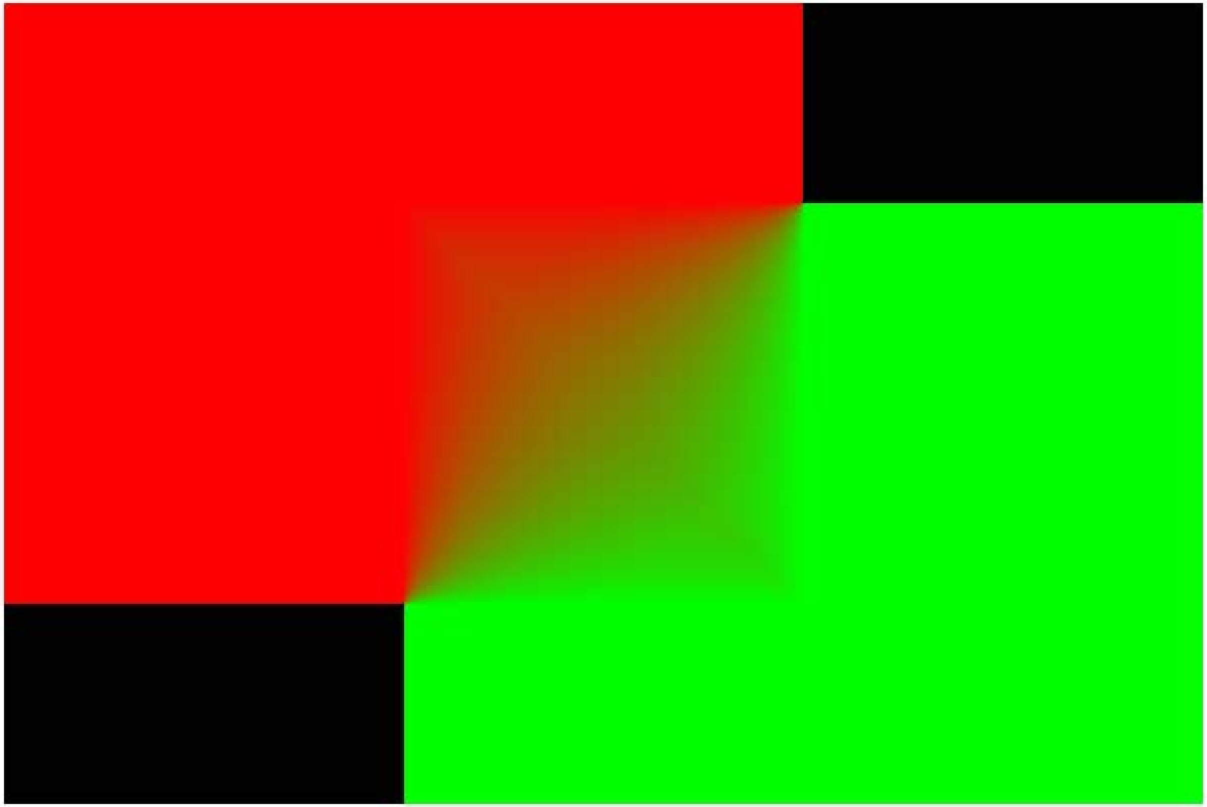


Image Blending

linear
mask



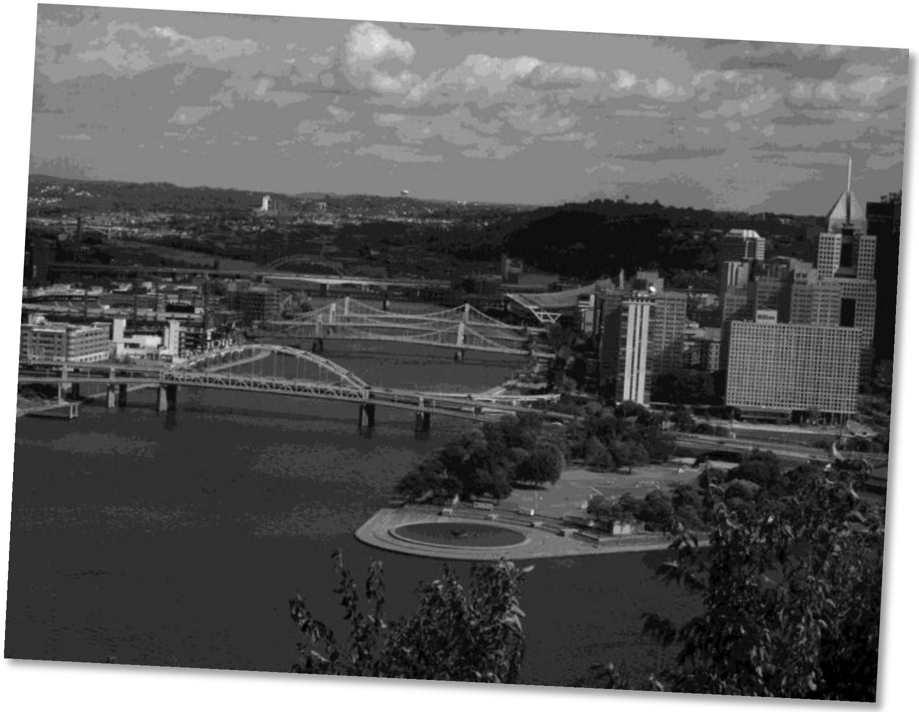




Summary

1. Load two images to stitch together
2. Extract feature points: `vl_sift`
3. Find corresponding points: `vl_ucbmatch`
4. Find best corresponding points: RANSAC
5. Find best homography for max. inliers
6. Calculate image size and offsets to sources
7. Rasterize image and crop

Previous years' Top Results



Marco Manzi

Previous years' Top Results

Marco Manzi

Previous years' Top Results

Michael Pfeuti

Previous years' Top Results

Michael Pfeuti