Feature Detection and Matching (2 of 2)

Lluís Garrido – lluis.garrido@ub.edu

Department of Analysis and Applied Mathematics
University of Barcelona

Computational Vision, November 2015

Local features

A feature detection and matching application is composed of the following stages:

1. Feature detection (extraction)
   - Search for interest points in the image: done!
2. Feature description
   - Describe the interest point in a compact and stable way: done!
3. Feature matching
   - Search for matching candidates in other images

Stage 3: Feature matching

How can we establish a matching between (SIFT) descriptors ?

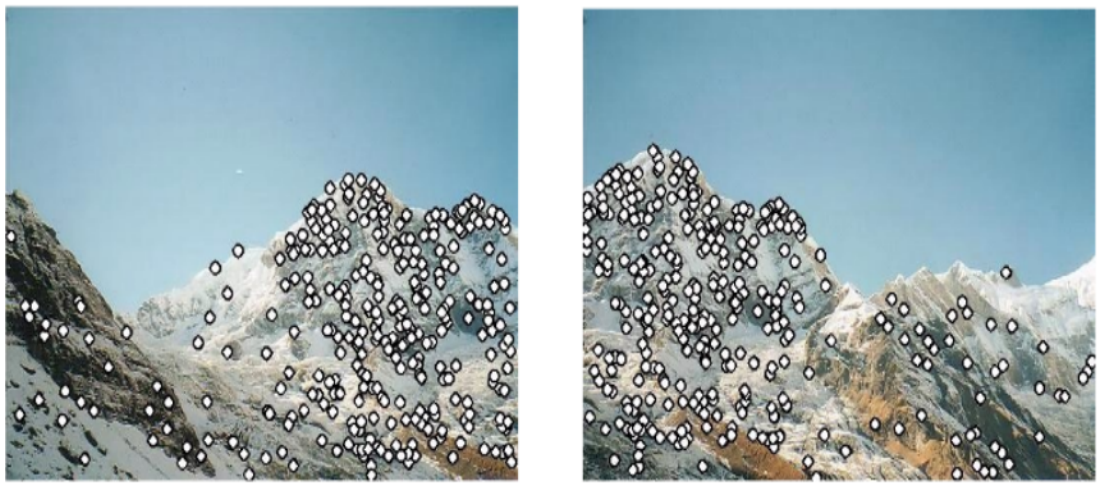
The matching strategy depends on the context of the problem

- Most features match: e.g. panorama creation
- Most features do not match: e.g. object recognition

Stage 3: Feature matching

Example: panorama creation.

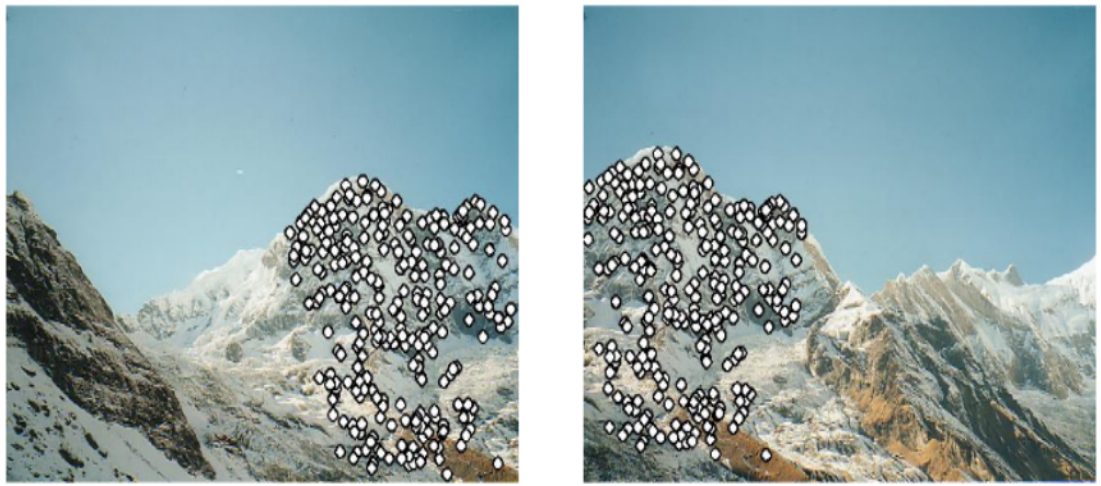
The two images show the SIFT keypoints extracted from each image.



Stage 3: Feature matching

Example: panorama creation.

Here we show the SIFT keypoints that have matched.



Stage 3: Feature matching

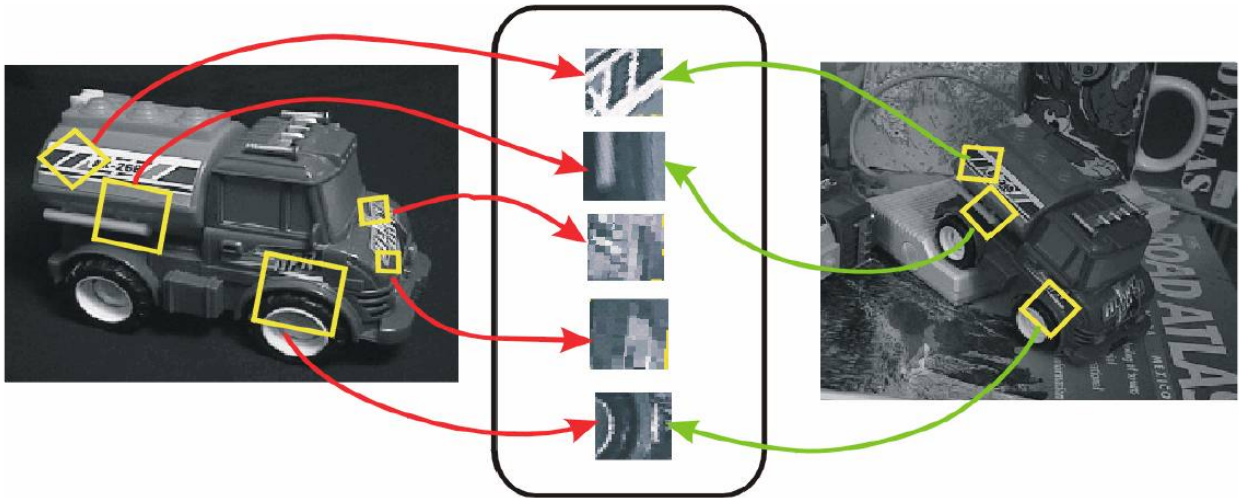
Example: panorama creation.

The matching can then be used to warp the second image over the first one in order to construct a panoramic image.

Stage 3: Feature matching

Example: object instance recognition

By extracting the SIFT keypoints from one image one is able to find an instance of the same object (from another viewpoint) in a second image.



Stage 3: Feature matching

We assume that feature descriptors have been designed to that Euclidean distance can be used for ranking potential matches.

What kind of strategies can we use to make the matching robust ? That is, what kind of algorithms can be devised in order to correctly match descriptors ? It's not easy!

- If two descriptors are present in both images we would like them to match at first rank.
- A descriptor may be present in one image (image A) but not in the second (image B). Our algorithm will always try to find a matching for the descriptor in image A even if it's not present in image B. The descriptor in image A should be ranked at the last positions!

Stage 3: Feature matching

We assume that feature descriptors have been designed to that Euclidean distance can be used for ranking potential matches.

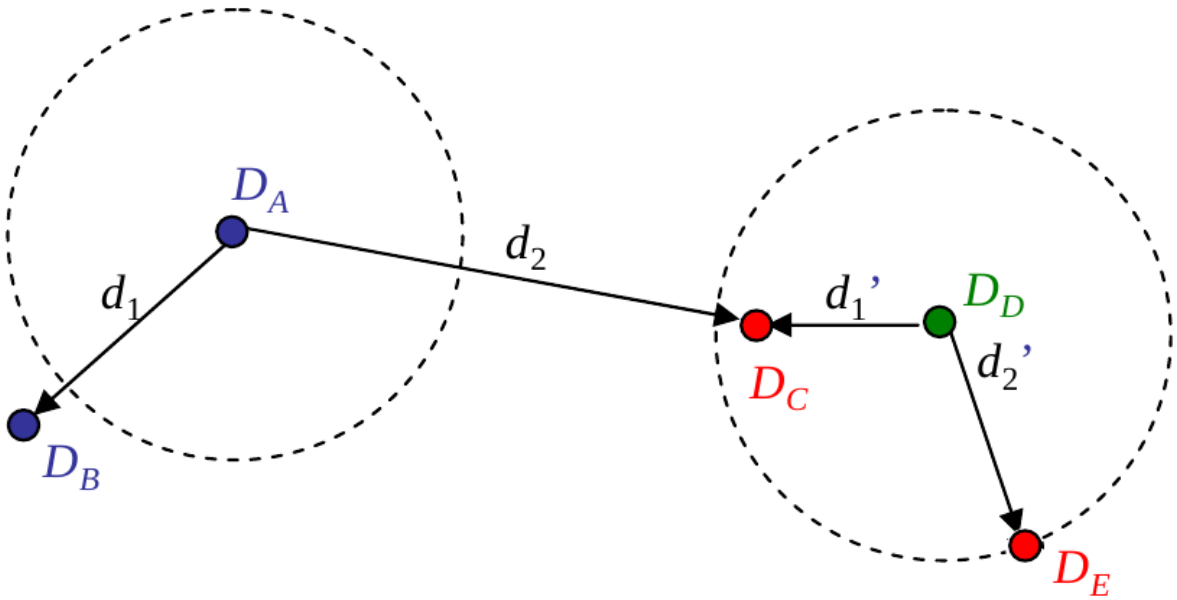
What kind of strategies can we use to make the matching robust ? That is, what kind of algorithms can be devised in order to correctly match descriptors ? It's not easy!

- If two descriptors are present in both images we would like them to match at first rank.
- A descriptor may be present in one image (image A) but not in the second (image B). Our algorithm will always try to find a matching for the descriptor in image A even if it's not present in image B. The descriptor in image A should be ranked at the last positions!

Stage 3: Feature matching

Euclidean distance

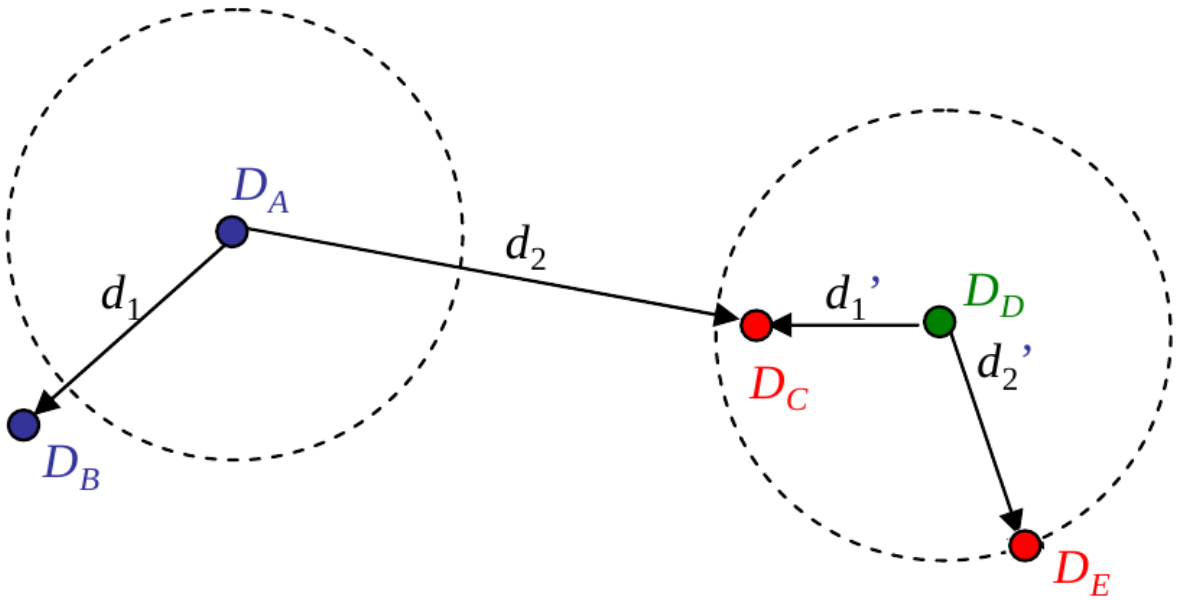
- We may use a threshold (dashed circle) to to select the candidate matches. The useful range range of thresholds can vary a lot depending on the kind of image.
- In the figure descriptor D_A is fails to match against descriptor D_B. In addition D_D is incorrectly matched against two descriptors: D_C and D_E (D_D should match only against one descriptor).



Stage 3: Feature matching

Euclidean distance

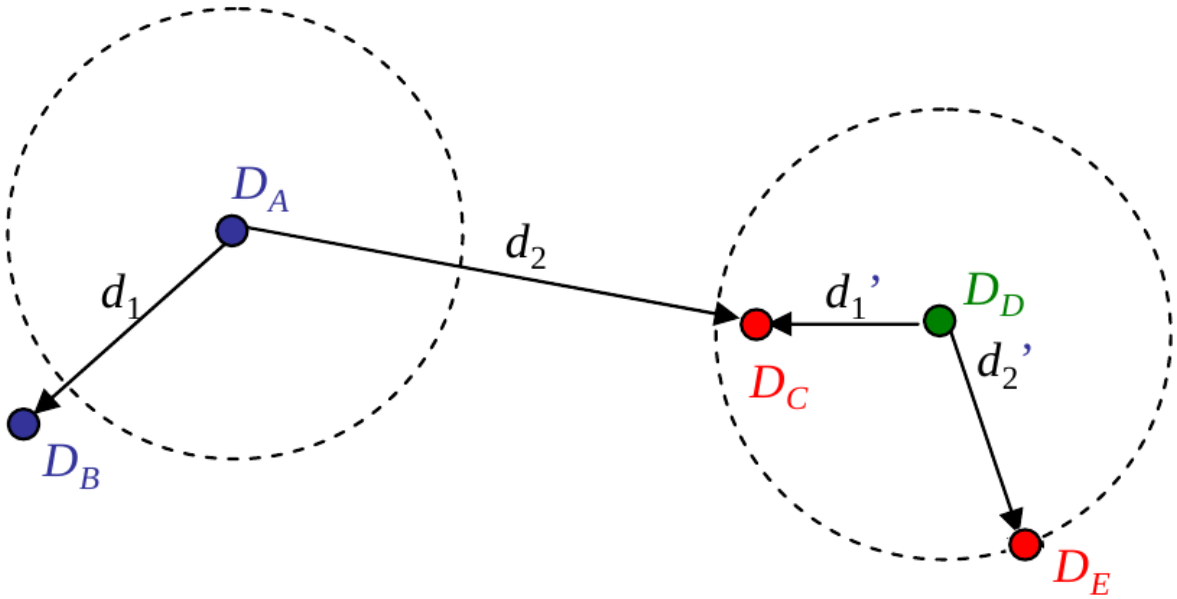
- We may use a threshold (dashed circle) to to select the candidate matches. The useful range range of thresholds can vary a lot depending on the kind of image.
- In the figure descriptor D_A is fails to match against descriptor D_B. In addition D_D is incorrectly matched against two descriptors: D_C and D_E (D_D should match only against one descriptor).



Stage 3: Feature matching

Nearest neighbor

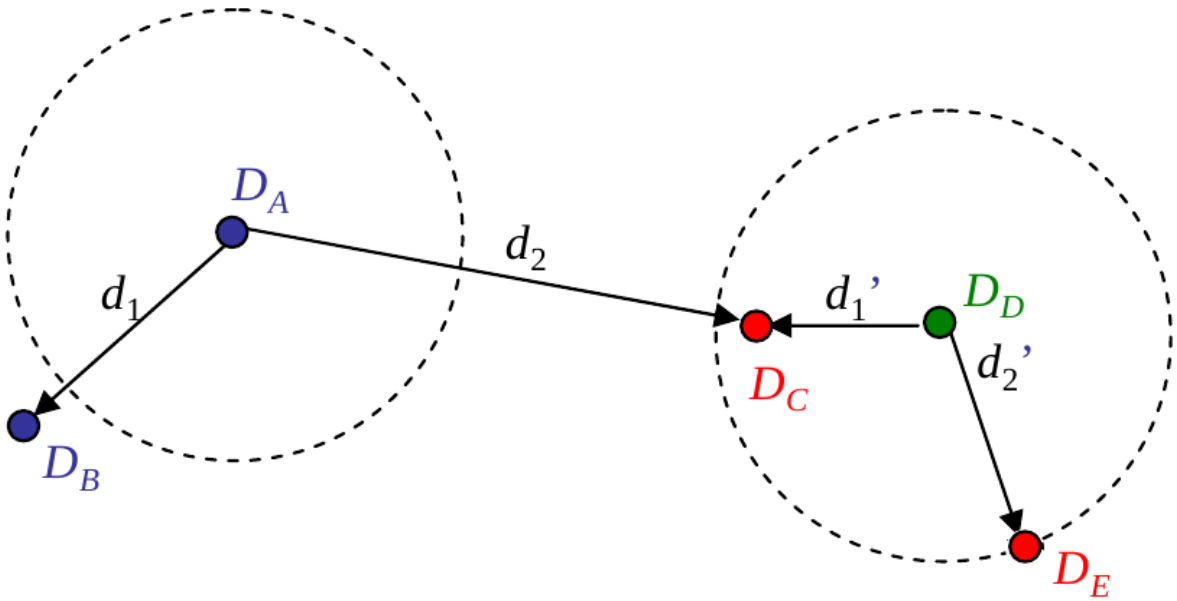
- Match to each descriptor its nearest neighbor according to some distance.
- Here D_A is correctly matched against D_B. D_D may be incorrectly matched against D_C (D_E may be the correct one).
- A threshold may be used to avoid false positives.



Stage 3: Feature matching

Nearest neighbor distance ratio

- Compare nearest neighbor distance (d_1) with the second nearest neighbor distance (d_2).
- Here d_1/d_2 is small, thus D_A matches with D_B. But d_1'/d_2' is large, thus D_D does not match D_C.



Stage 3: Feature matching

A feature detection and matching application is composed of the following stages:

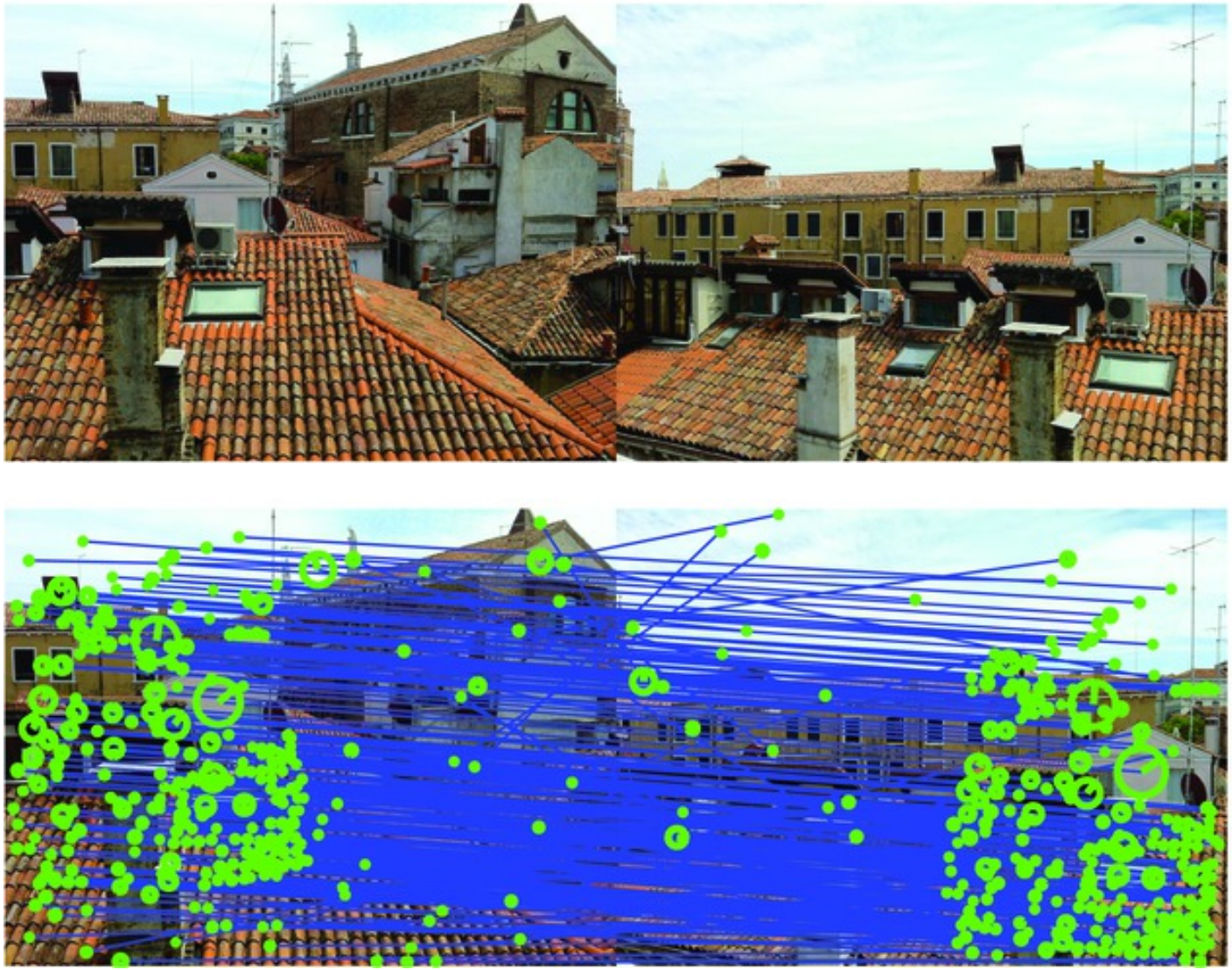
1. Feature detection
2. Feature description
3. Feature matching

The three stages are application dependent!

How can we apply this technique to point correspondences, for instance ?

Application: point correspondences

- How can we be robust with respect the false correspondences ?



Application: point correspondences

Take into account that

- Usually, if we try to match SIFT descriptors in an independent way we may obtain false correspondences.
- We may improve the correspondences by using higher level information about the scene.

In the next slides we will focus on the general problem of point correspondences with the particular application of panorama creation.

- Here we will assume that the whole image is transformed according to a particular motion model.
- In the general problem of point correspondences the motion model is applied only on the object to track.

Application: point correspondences

Take into account that

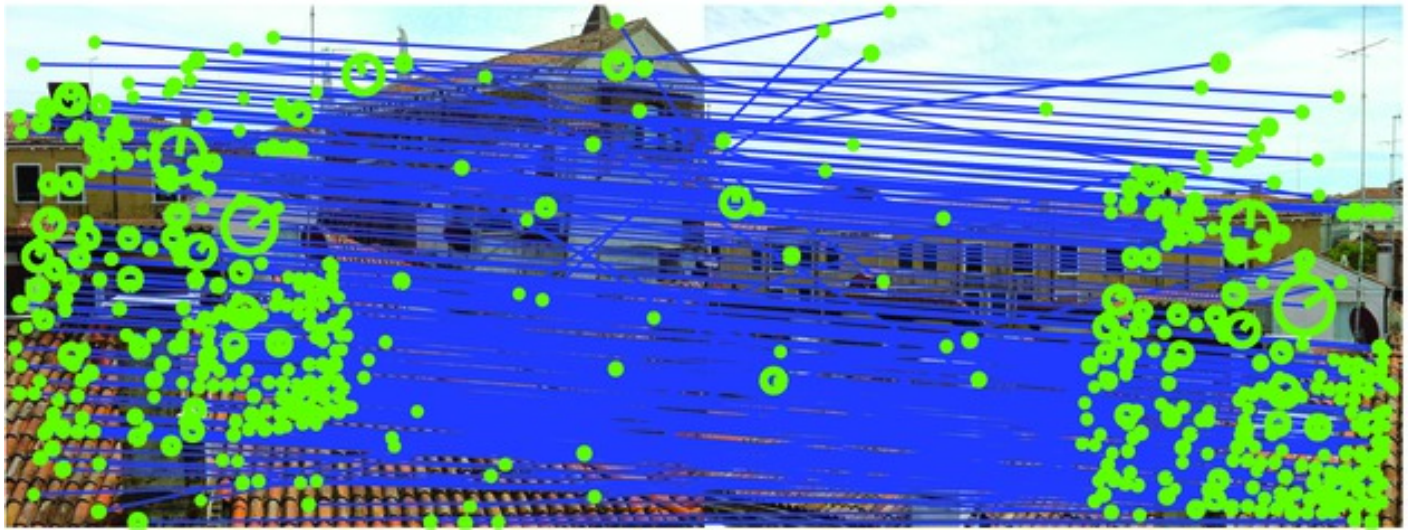
- Usually, if we try to match SIFT descriptors in an independent way we may obtain false correspondences.
- We may improve the correspondences by using higher level information about the scene.

In the next slides we will focus on the general problem of point correspondences with the particular application of panorama creation.

- Here we will assume that the whole image is transformed according to a particular motion model.
- In the general problem of point correspondences the motion model is applied only on the object to track.

Application: point correspondences

- Assume that a point $(x_{i,1}, x_{i,2})$ matches $(x'_{i,1}, x'_{i,2})$.



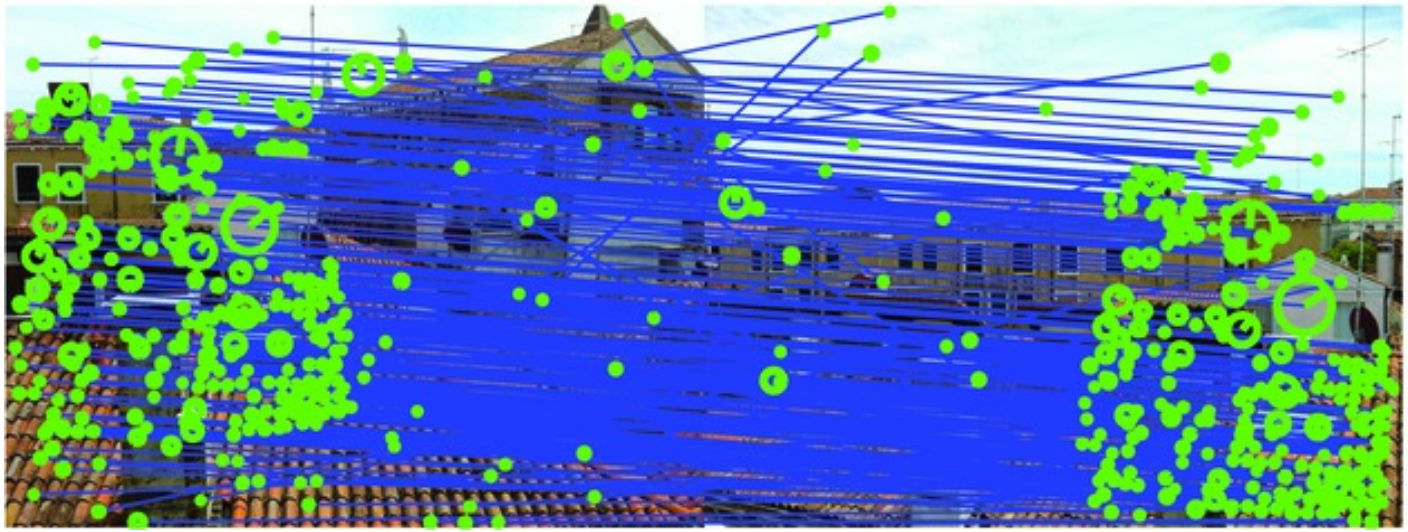
- We assume that every matched SIFT descriptors satisfies

$$(x_{i,1}, x_{i,2}) + (t_x, t_y) = (x'_{i,1}, x'_{i,2})$$

where $\mathbf{p} = (t_x, t_y)^T$ are the common translation parameters (i.e. constant for all matchings).

Application: point correspondences

- Assume that a point $(x_{i,1}, x_{i,2})$ matches $(x'_{i,1}, x'_{i,2})$.



- We assume that every matched SIFT descriptors satisfies

$$(x_{i,1}, x_{i,2}) + (t_x, t_y) = (x'_{i,1}, x'_{i,2})$$

where $\mathbf{p} = (t_x, t_y)^T$ are the common translation parameters (i.e. constant for all matchings).

Application: point correspondences

The problem is

1. Given a set of matched desccriptors (good and bad ones)

$$(x_{i,1}, x_{i,2})^T \quad \Leftrightarrow \quad (x'_{i,1}, x'_{i,2})^T$$

2. Given a translational model $\mathbf{p} = (t_x, t_y)^T$

Which are the best parameters $\mathbf{p}$ that model the correspondences between the matched descriptors ?

Ideally we would like that (exactly)

$$(x_{i,1}, x_{i,2}) + (t_x, t_y) = (x'_{i,1}, x'_{i,2})$$

but this is, in general, not possible since the model is just an approximation. Or, even if the model is exact, the sift positions may no be exact!

Application: point correspondences

The problem is

1. Given a set of matched desccriptors (good and bad ones)

$$(x_{i,1}, x_{i,2})^T \quad \Leftrightarrow \quad (x'_{i,1}, x'_{i,2})^T$$

2. Given a translational model $\mathbf{p} = (t_x, t_y)^T$

Which are the best parameters $\mathbf{p}$ that model the correspondences between the matched descriptors ?

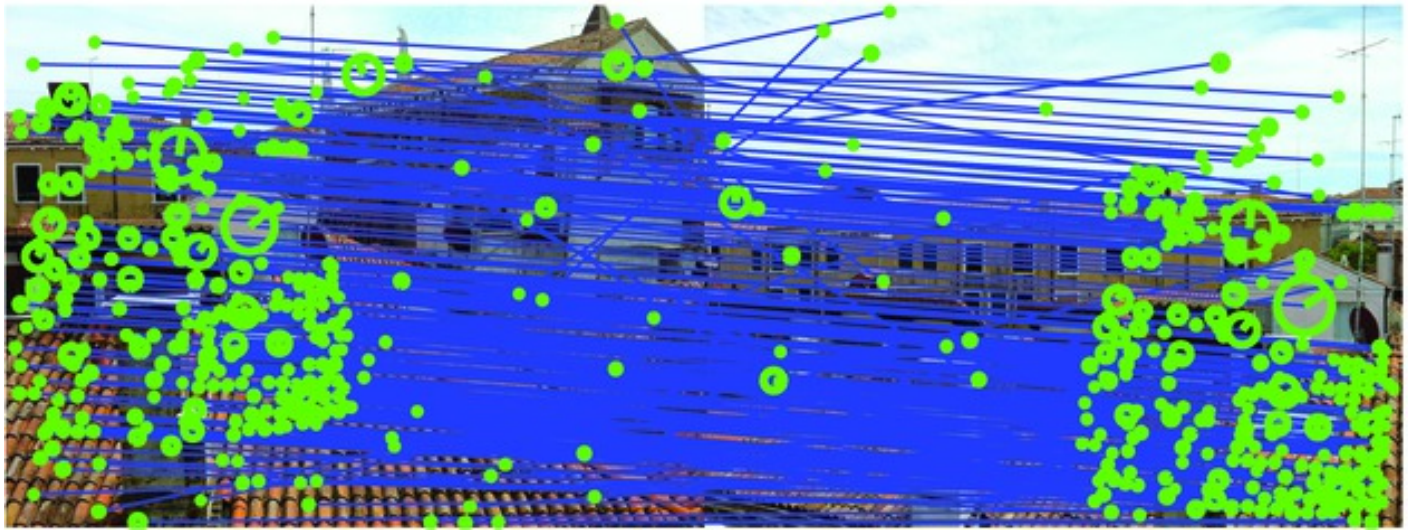
Ideally we would like that (exactly)

$$(x_{i,1}, x_{i,2}) + (t_x, t_y) = (x'_{i,1}, x'_{i,2})$$

but this is, in general, not possible since the model is just an approximation. Or, even if the model is exact, the sift positions may no be exact!

Application: point correspondences

In general, given any model and the matchings, how can we compute the best model that adapts to the correspondences ?

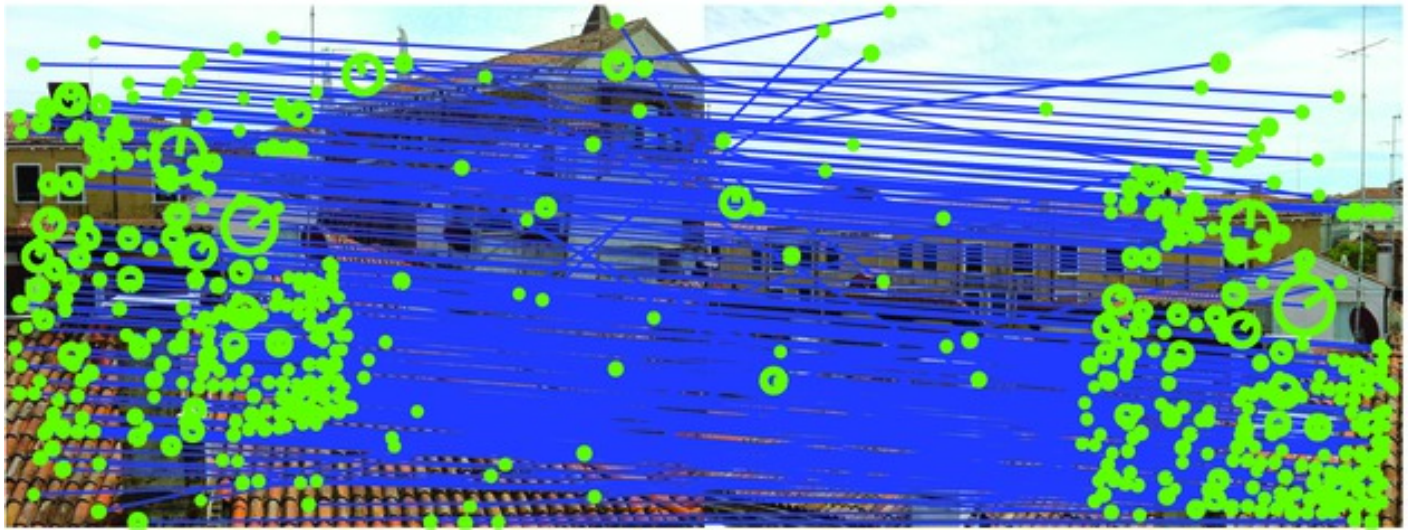


A simple way to proceed is to use least squares.

- We will see now a small tutorial with an example in 1D.
- We will then extend the problem to 2D in our case.

Application: point correspondences

In general, given any model and the matchings, how can we compute the best model that adapts to the correspondences ?

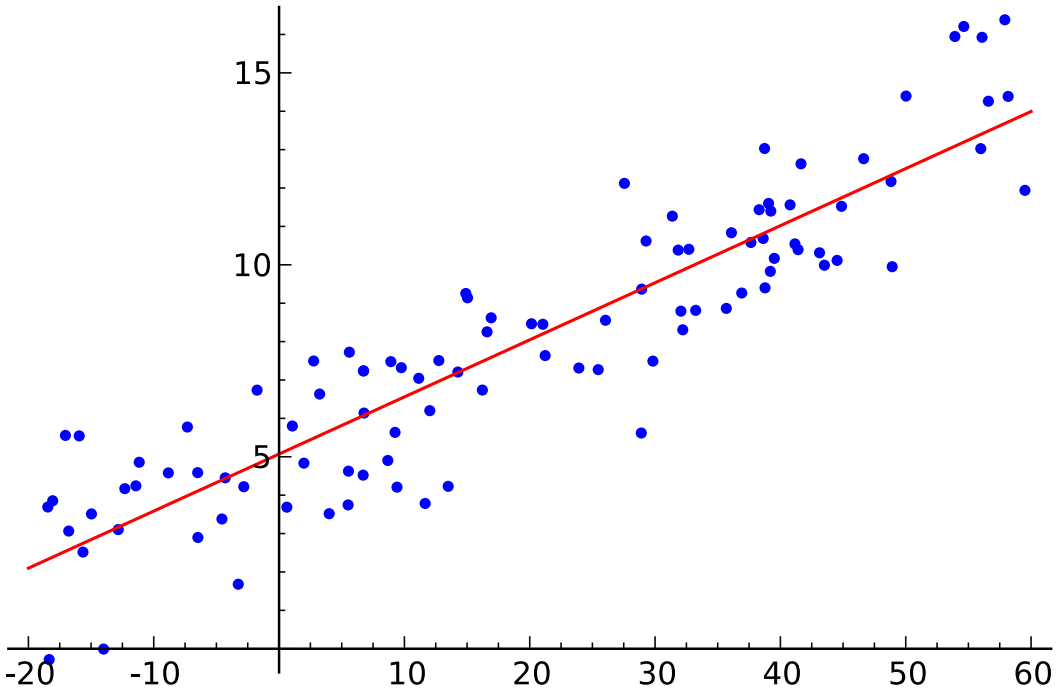


A simple way to proceed is to use least squares.

- We will see now a small tutorial with an example in 1D.
- We will then extend the problem to 2D in our case.

Tutorial: linear regression

Assume we want to model a set of points in the plane with a linear function, such as $f(x) = ax + b$.

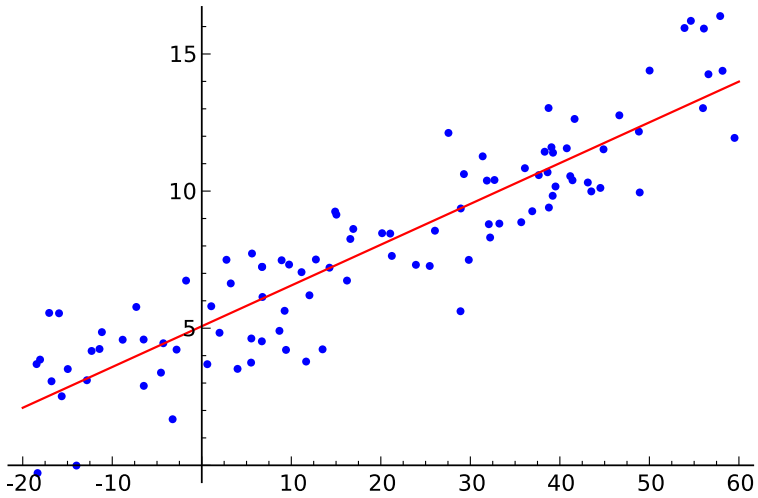


Given the data points $\{x_i, y_i\}$ with $i = 1 \ldots m$ (blue points) how can the problem be solved ?

Tutorial: linear regression

Our problem: approximate $\{x_i, y_i\}$ with the function $f(x) = ax + b$. Ideally, we would like the line passing through each data point

$$
\begin{aligned}
y_1 &= a x_1 + b \\
y_2 &= a x_2 + b \\
y_3 &= a x_3 + b \\
\vdots &\quad \vdots \\
y_m &= a x_m + b
\end{aligned}
$$

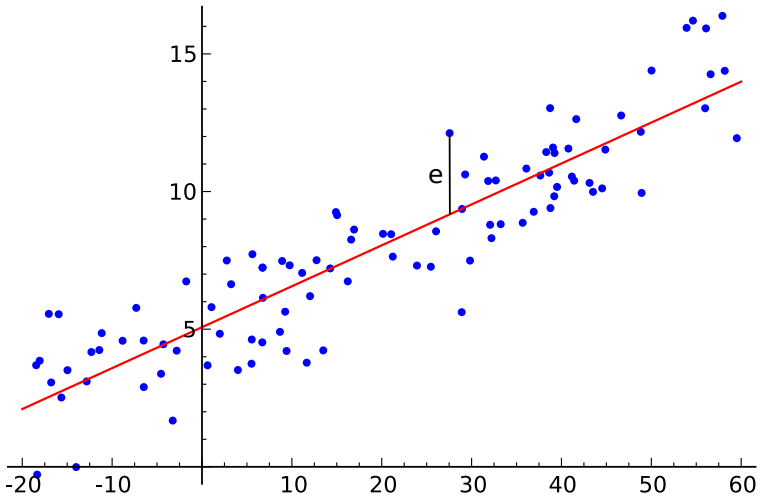


That's not possible, as you see! How do we proceed, then ?

Tutorial: linear regression

Assume $f(x) = ax + b$ is given (i.e. a and b are known). For a given point $\{x_i, y_i\}$, we may measure the error between the model and the real value

$$e_i = f(x_i) - y_i$$



We may measure the overall error by computing

$$E = \sum_{i=1}^{m} e_i^2 = \sum_{i=1}^{m} (f(x_i) - y_i)^2$$

Tutorial: linear regression

Given $\{x_i, y_i\}$ with $i = 1 \dots m$, and a model $f(x) = a\,x + b$ (a and b unknown) our objective is to compute the parameters a and b that minimizes

$$E = \sum_{i=1}^{m} e_i^2 = \sum_{i=1}^{m} (f(x_i) - y_i)^2$$

This is a least squares problem. This problem has a closed solution.

How can it be solved ? Just write down the equations of $e_i = f(x_i) - y_i$

$$\begin{aligned} a\,x_1 + b - y_1 &= e_1 \\ a\,x_2 + b - y_2 &= e_2 \\ &\vdots \quad\quad \vdots \\ a\,x_m + b - y_m &= e_m \end{aligned}$$

Tutorial: linear regression

Given $\{x_i, y_i\}$ with $i = 1 \ldots m$, and a model $f(x) = a\,x + b$ (a and b unknown) our objective is to compute the parameters a and b that minimizes

$$E = \sum_{i=1}^{m} e_i^2 = \sum_{i=1}^{m} (f(x_i) - y_i)^2$$

This is a least squares problem. This problem has a closed solution.

How can it be solved ? Just write down the equations of $e_i = f(x_i) - y_i$

$$\begin{aligned} a\,x_1 + b - y_1 &= e_1 \\ a\,x_2 + b - y_2 &= e_2 \\ &\vdots \quad\;\; \vdots \\ a\,x_m + b - y_m &= e_m \end{aligned}$$

The equations

$$\begin{aligned} a x_1 + b - y_1 &= e_1 \\ a x_2 + b - y_2 &= e_2 \\ &\vdots \quad \vdots \\ a x_m + b - y_m &= e_m \end{aligned}$$

can be written compactly as

$$\underbrace{\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\mathbf{x}} - \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{b}} = \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix}}_{\mathbf{e}} \quad \rightarrow \quad \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{e}$$

where $\mathbf{x}$ are the unknown parameters.

Our minimization problem

$$E = \sum_{i=1}^{m} e_i^2 = \sum_{i=1}^{m} (f(x_i) - y_i)^2$$

can be now written in matrix form as

$$E = \|\mathbf{e}\|^2 = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$$

The minimum of the previous equation is attained for

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

where $\mathbf{x} = (a, b)^T$ are the unknown parameters. The matrix $A^T A$ has size 2×2.

Our minimization problem

$$E = \sum_{i=1}^{m} e_i^2 = \sum_{i=1}^{m} (f(x_i) - y_i)^2$$

can be now written in matrix form as

$$E = \|\mathbf{e}\|^2 = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$$

The minimum of the previous equation is attained for

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

where $\mathbf{x} = (a, b)^T$ are the unknown parameters. The matrix $A^T A$ has size 2×2.

Application: point correspondences

Let us return to our original problem.

1. Assume we have a set of correspondences (good and bad ones) performed by SIFT.

$$(x_{i,1}, x_{i,2})^T \quad \Leftrightarrow \quad (x'_{i,1}, x'_{i,2})^T$$

2. We would like to compute the best (least squares) parameters using a translational model

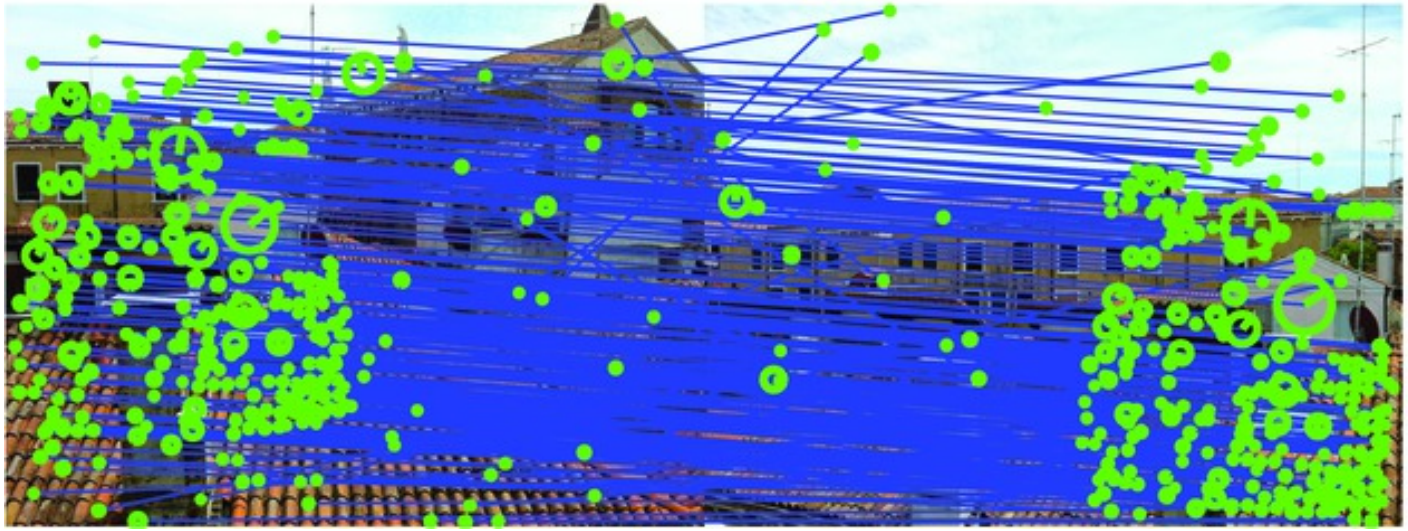
$$\mathbf{p} = (t_x, t_y)^T$$

The problem is similar to the example we have just seen.

Application: point correspondences

Let us define the displacement between two matched SIFT descriptors as

$$\mathbf{d}_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix} - \begin{pmatrix} x'_{i,1} \\ x'_{i,2} \end{pmatrix}$$



We would like to compute $\mathbf{p}$ by minimizing

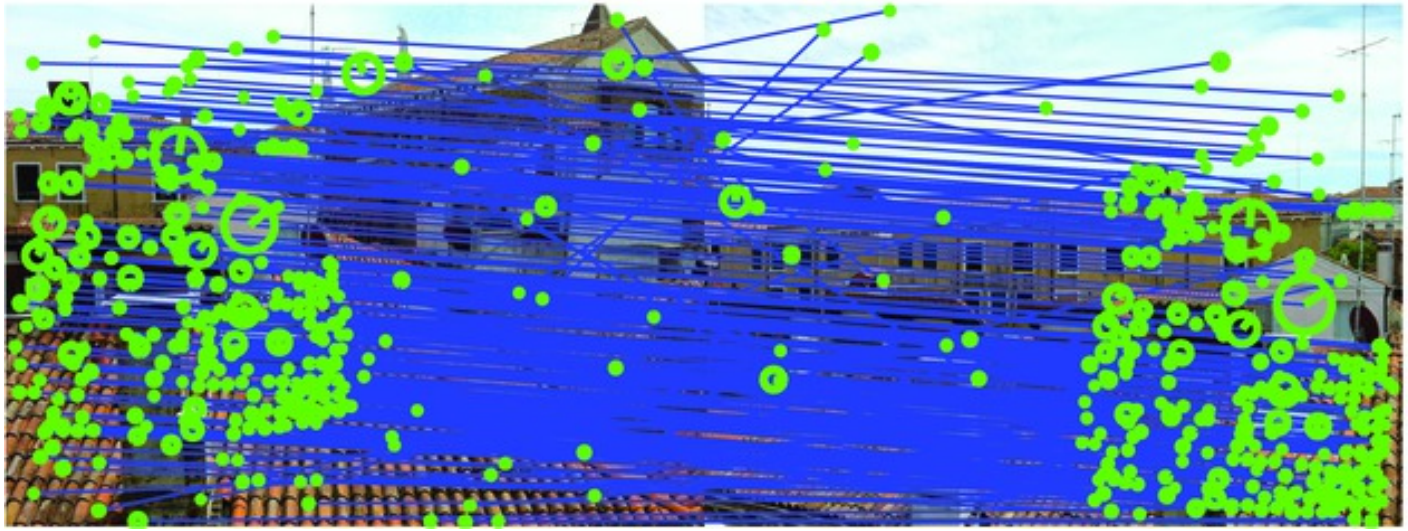
$$\sum_i \|\mathbf{p} - \mathbf{d}_i\|^2$$

where the sum is performed over all the correspondences.

Application: point correspondences

Let us define the displacement between two matched SIFT descriptors as

$$\mathbf{d}_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix} - \begin{pmatrix} x'_{i,1} \\ x'_{i,2} \end{pmatrix}$$



We would like to compute $\mathbf{p}$ by minimizing

$$\sum_i \|\mathbf{p} - \mathbf{d}_i\|^2$$

where the sum is performed over all the correspondences.

Application: point correspondences

To minimize the previous function

- We may follow the same procedure as in the tutorial (i.e. write down all the equations and try to express it in matrix form). This is left for you as exercise.
- We follow here a more simpler approach

Both will lead us to the same solution

We write

$$\sum_i \|\mathbf{p} - \mathbf{d}_i\|^2 = \sum_i \|\mathbf{J}\mathbf{p} - \mathbf{d}_i\|^2$$

where

$$\mathbf{J} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

we will assume that $\mathbf{J}$ may depend on i (to be seen later).

Application: point correspondences

To minimize the previous function

- We may follow the same procedure as in the tutorial (i.e. write down all the equations and try to express it in matrix form). This is left for you as exercise.
- We follow here a more simpler approach

Both will lead us to the same solution

We write

$$\sum_i \|\mathbf{p} - \mathbf{d}_i\|^2 = \sum_i \|\mathbf{J}\mathbf{p} - \mathbf{d}_i\|^2$$

where

$$\mathbf{J} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

we will assume that $\mathbf{J}$ may depend on i (to be seen later).

Application: point correspondences

Since $\mathbf{p}$ (the parameters) is constant for all i

$$\sum_i \|\mathbf{p} - \mathbf{d}_i\|^2 = \sum_i \|\mathbf{J}_i\mathbf{p} - \mathbf{d}_i\|^2 = \sum_i (\mathbf{J}_i\mathbf{p} - \mathbf{d}_i)^T (\mathbf{J}_i\mathbf{p} - \mathbf{d}_i)$$

$$\mathbf{p}^T \left(\sum_i \mathbf{J}_i^T \mathbf{J}_i \right) \mathbf{p} - 2\,\mathbf{p}^T \left(\sum_i \mathbf{J}_i^T \mathbf{d}_i \right) + \sum_i \|\mathbf{d}_i\|^2 =$$

$$\mathbf{p}^T \mathbf{A} \mathbf{p} - 2\,\mathbf{p}^T \mathbf{b} + c$$

where

$$\begin{aligned} \mathbf{A} &= \sum_i \mathbf{J}_i^T \mathbf{J}_i \\ \mathbf{b} &= \sum_i \mathbf{J}_i^T \mathbf{d}_i \end{aligned}$$

Application: point correspondences

The previous energy attains the minimum for

$$\mathbf{A}\mathbf{p} - \mathbf{b} = 0$$

That is, we need to solve the linear system of equations

$$\mathbf{A}\mathbf{p} = \mathbf{b}$$

where

$$\begin{aligned} \mathbf{A} &= \sum_i \mathbf{J}_i^T \mathbf{J}_i \\ \mathbf{b} &= \sum_i \mathbf{J}_i^T \mathbf{d}_i \end{aligned}$$

Application: point correspondences

For the case of a translational model

$$\mathbf{p} = (t_x, t_y)^T$$

the solution to the previous linear system of equations is the average over the observed displacements

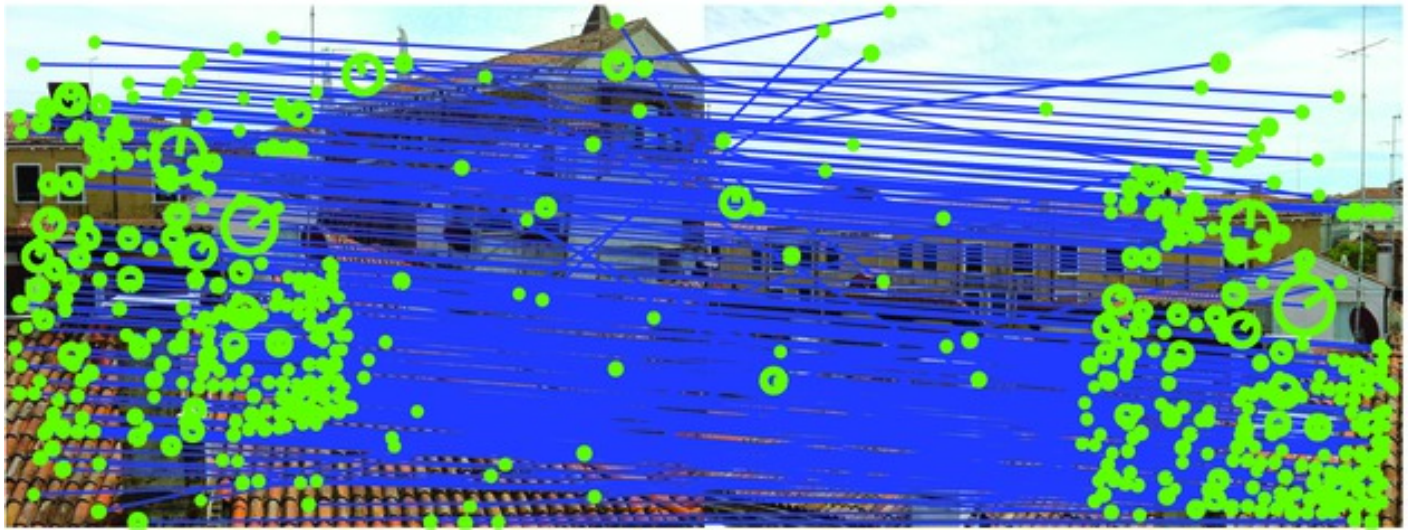
$$\mathbf{p} = \frac{1}{M} \sum_i \mathbf{d}_i$$

where M is the number of matched SIFT descriptors and

$$\mathbf{d}_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix} - \begin{pmatrix} x'_{i,1} \\ x'_{i,2} \end{pmatrix}$$

Application: point correspondences

For these correspondences



the resulting model is (only two correspondences are shown)

Application: point correspondences

By taking only the 10 best correspondences we obtain

which can be compared to the previous obtained model

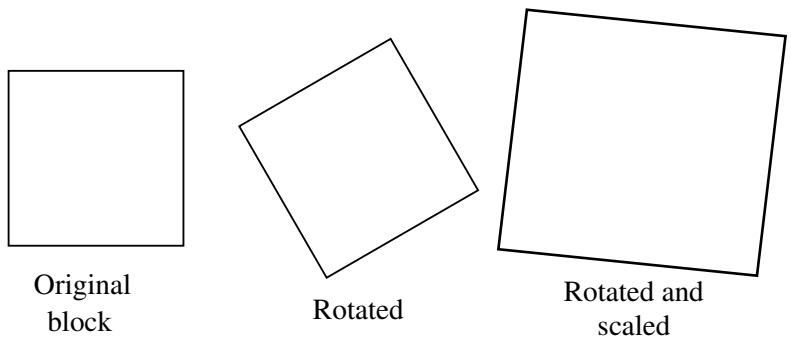
Application: point correspondences

With regard the model

- The translational model may be enough if the object (or image) to track has a translational motion.
- More complex models may be used, e.g. affine models.

$$\begin{pmatrix} x'_{i,1} \\ x'_{i,2} \end{pmatrix} = \underbrace{\begin{pmatrix} 1+a_1 & a_2 \\ a_3 & 1+a_4 \end{pmatrix}}_{\text{rotations,scale}} \begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix} + \underbrace{\begin{pmatrix} t_x \\ t_y \end{pmatrix}}_{\text{displacement}}$$

where $(x_{i,1}, x_{i,2})$ matches $(x'_{i,1}, x'_{i,2})$.



Application: point correspondences

Assume we have a set of correspondences between SIFT descriptors

$$(x_{i,1}, x_{i,2})^T \quad \Leftrightarrow \quad (x'_{i,1}, x'_{i,2})^T$$

We would like to approximate an affine model (instead of a translational one)

$$\begin{pmatrix} x'_{i,1} \\ x'_{i,2} \end{pmatrix} = \begin{pmatrix} 1+a_1 & a_2 \\ a_3 & 1+a_4 \end{pmatrix} \begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

How do we proceed ?

Application: point correspondences

We need to write the problem as before

$$\sum_i \|\mathbf{J}_i\mathbf{p} - \mathbf{d}_i\|^2$$

where $\mathbf{p}$ are the parameters to solve

$$\mathbf{p} = (a_1, a_2, a_3, a_4, t_x, t_y)^T$$

The original expression

$$\begin{pmatrix} x'_{i,1} \\ x'_{i,2} \end{pmatrix} = \begin{pmatrix} 1+a_1 & a_2 \\ a_3 & 1+a_4 \end{pmatrix} \begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

can be rewritten as

$$\underbrace{\begin{pmatrix} x'_{i,1} \\ x'_{i,2} \end{pmatrix} - \begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix}}_{\mathbf{d}_i} = \underbrace{\begin{pmatrix} x_{i,1} & x_{i,2} & 0 & 0 & 1 & 0 \\ 0 & 0 & x_{i,1} & x_{i,2} & 0 & 1 \end{pmatrix}}_{\mathbf{J}_i} \mathbf{p}$$

Application: point correspondences

We need to write the problem as before

$$\sum_i \|\mathbf{J}_i\mathbf{p} - \mathbf{d}_i\|^2$$

where $\mathbf{p}$ are the parameters to solve

$$\mathbf{p} = (a_1, a_2, a_3, a_4, t_x, t_y)^T$$

The original expression

$$\begin{pmatrix} x'_{i,1} \\ x'_{i,2} \end{pmatrix} = \begin{pmatrix} 1+a_1 & a_2 \\ a_3 & 1+a_4 \end{pmatrix} \begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

can be rewritten as

$$\underbrace{\begin{pmatrix} x'_{i,1} \\ x'_{i,2} \end{pmatrix} - \begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix}}_{\mathbf{d}_i} = \underbrace{\begin{pmatrix} x_{i,1} & x_{i,2} & 0 & 0 & 1 & 0 \\ 0 & 0 & x_{i,1} & x_{i,2} & 0 & 1 \end{pmatrix}}_{\mathbf{J}_i} \mathbf{p}$$

Application: point correspondences

To approximate an affine model to the matched SIFT descriptors we need to solve (as for the translational model)

$$\mathbf{A}\mathbf{p} = \mathbf{b}$$

where

$$\begin{aligned} \mathbf{A} &= \sum_i \mathbf{J}_i^T \mathbf{J}_i \\ \mathbf{b} &= \sum_i \mathbf{J}_i^T \mathbf{d}_i \end{aligned}$$

but with (this is different with respect the translational model)

$$\mathbf{J}_i = \begin{pmatrix} x_{i,1} & x_{i,2} & 0 & 0 & 1 & 0 \\ 0 & 0 & x_{i,1} & x_{i,2} & 0 & 1 \end{pmatrix}$$

Application: point correspondences

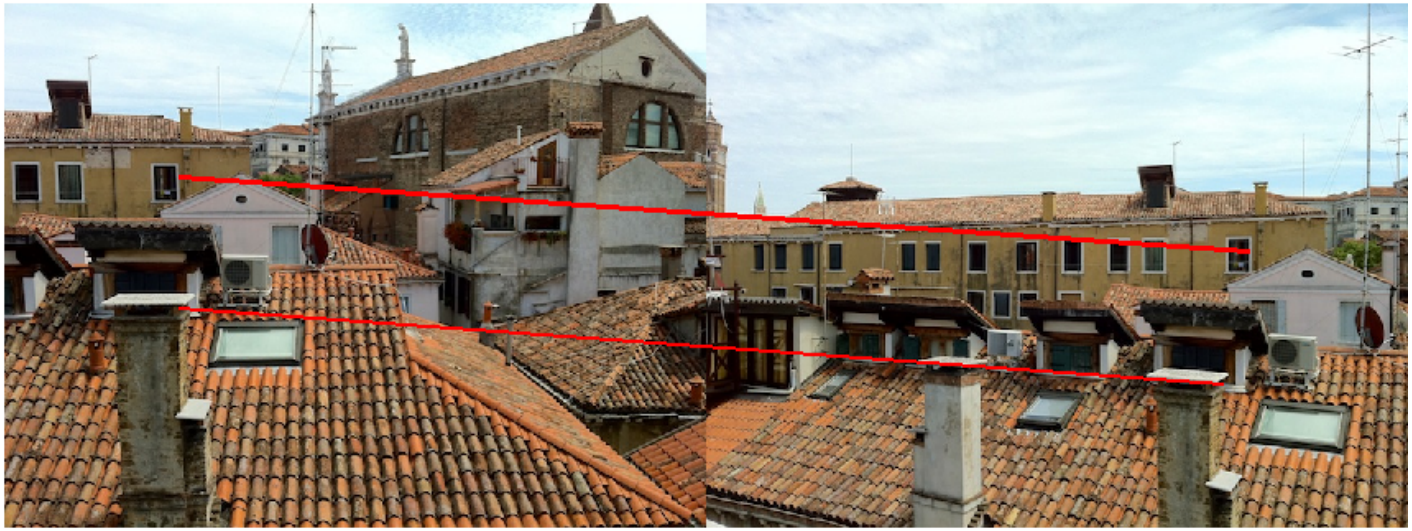
This is the result for the affine model using all correspondences

Note that

- The model is not a translation anymore (for instance, the two shown lines are not parallel).
- The model has been degraded by the outliers (bad correspondences).

Application: point correspondences

By taking only the 10 best correspondences we obtain



which can be compared to the previous obtained model

Application: point correspondences

Take into account that

- With the least squares method we assume that the noise follows a Gaussian distribution. Here the "noise" are the matchings that do not follow the motion model – called outliers –.
- More robust versions of least squares are required when the outliers degrade the model. This may happen if the number of outliers is large.

Other techniques are required. For instance, we may use

- Robust energies (e.g.. take the absolute value rather than the square of the error). Gradient descent techniques are then needed since no closed solution exists.
- Apply least squares but use alternative techniques rather than simply select e.g. the "10 best matchings." We'll see a method called RANSAC.

Application: point correspondences

A simple and effective way to proceed is the RANSAC method (Random Sample Consensus)

- Select a random subset of k correspondences.
- Estimate the model $\mathbf{p}$ for this random subset.
- Count, among all correspondences, the number of inliers that are within ε of their predicted location

$$\|\mathbf{J}_i\mathbf{p} - \mathbf{d}_i\| \leq \varepsilon$$

- The random selection process is repeated S times and the sample set with the largest number of inliers is kept as the final solution.

Summary

A feature detection and matching application is composed of the following stages:

1. Feature detection
2. Feature description
3. Feature matching

The three stages are application dependent!

Bibliography

- Szelisky, "Computer Vision: algorithms and applications"
- Lowe, "Distinctive image features from Scale-Invariant Keypoints"

Most of the images shown are have been taken from these two sources.