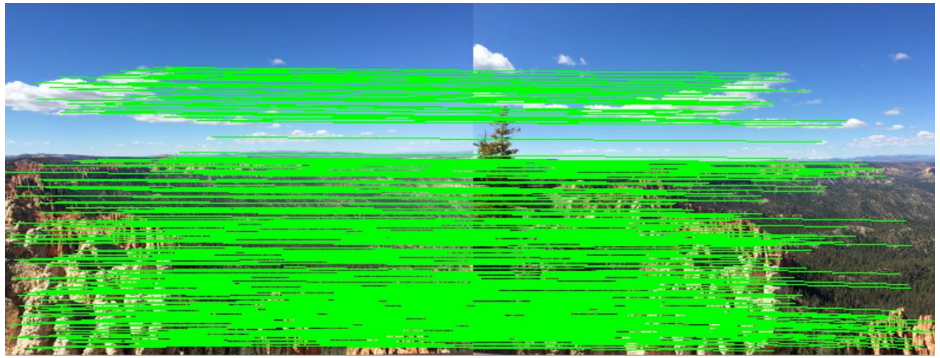
Image Matching

Matching -› Stitching -› Panorama

Image alignment gives panorama

Salient feature detection and description



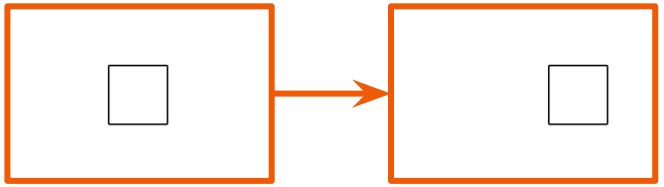
Matching gives point correspondences

Robust estimation of geometric transformation

Matching -› Localization, Mapping, Reconstruction

Translation

2 DoF



$$x' = x + t_x$$
$$y' = y + t_y$$

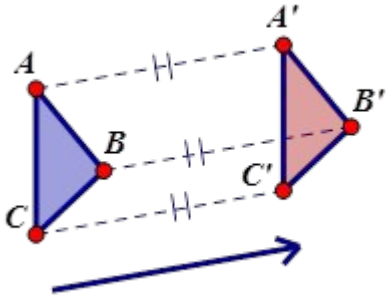
Matrix Form

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{I}_2 & \mathbf{t} \\ 0 \quad 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- 2 parameters: shift along x-axis and shift along y-axis
- Preserves *line orientation, length, angle, area, line parallelism, straight lines (3 points on the same line will map to the 3 points also lying on one line)*

Estimate Translation Minimally

- **Point correspondence (cspond)** — two points that are related by a transformation which we want to estimate
- How many point correspondences do we need to estimate translation?



$$t_x = x' - x$$

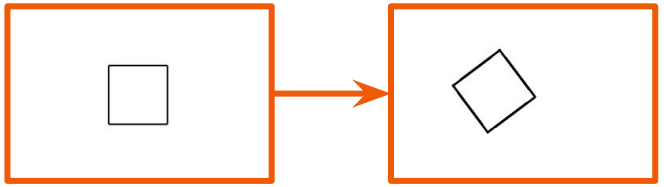
$$t_y = y' - y$$

- In general, 1 cspond provides 2 independent constraint equations*
- Minimal number of **csponds = ceil(DoF / 2)**

** if the arrangement of the total set of cspond's is not degenerate (can be verified geometrically)*

Rigid

3 DoF



$$x' = \cos\theta x - \sin\theta y + t_x$$
$$y' = \sin\theta x + \cos\theta y + t_y$$

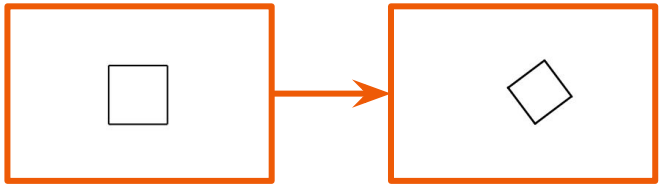
Matrix Form

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \mathrm{R}_\theta & \mathbf{t} \\ 0 \quad 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Also called Euclidean transformation
- 3 parameters: rotation angle, shift along x-axis and y-axis
- Preserves *length*, *angle*, *area*, *line parallelism, straight lines*.

Similarity

4 DoF



$$x' = s(\cos\theta x - \sin\theta y) + t_x$$
$$y' = s(\sin\theta x + \cos\theta y) + t_y$$

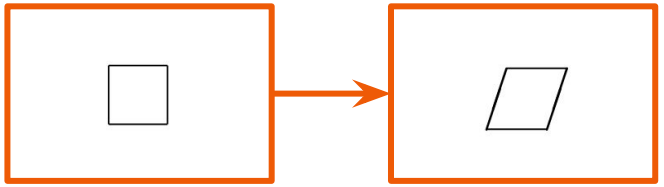
Matrix Form

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s\mathbf{R}_\theta & \mathbf{t} \\ 0 \quad 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Also called Euclidean transformation
- 4 parameters: scale factor, rotation angle, shift along x-axis and y-axis
- Preserves *angle*, *ratio of lengths* and *ratio of areas*, *line parallelism, straight lines*.

Affinity

6 DoF



$$x' = a_{11}x + a_{12}y + t_x$$
$$y' = a_{21}x + a_{22}y + t_y$$

Matrix Form

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ 0 \quad 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- 6 parameters interpreted differently depending on the decomposition
  e.g. Eigendecomposition or SVD
- **A** must be non-singular (invertible)
- Preserves *line parallelism* and *straight lines*.

$$\mathbf{A} = \mathbf{R}(\theta)\,\mathbf{R}(-\phi)\,\mathbf{D}\,\mathbf{R}(\phi)$$

Estimate Affinity Minimally: Task 1.1

Given **3 point cspond's** find an affine transformation

In: $(x_1, y_1) \rightarrow (x_1', y_1')$
$(x_2, y_2) \rightarrow (x_2', y_2')$
$(x_3, y_3) \rightarrow (x_3', y_3')$

Out: $\begin{bmatrix} \mathbf{A} & \mathbf{t} \\ 0 \quad 0 & 1 \end{bmatrix}$

What arrangement of the 3 point cspond's would be **degenerate** *i.e.* in which case 3 pairs of points are not enough to estimate the affinity?

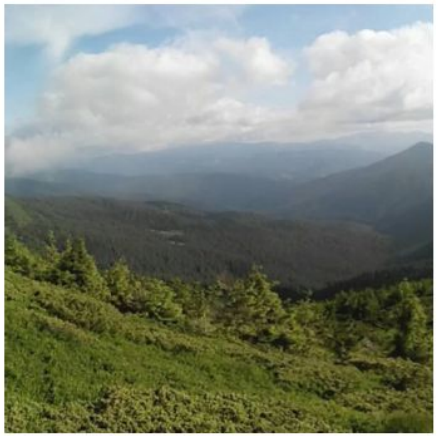
Estimate Affinity Minimally: Task 1.2 (coding)

Given **3 point cspond's** find an affine transformation, apply it to an image, compare with cv2.getAffineTransform

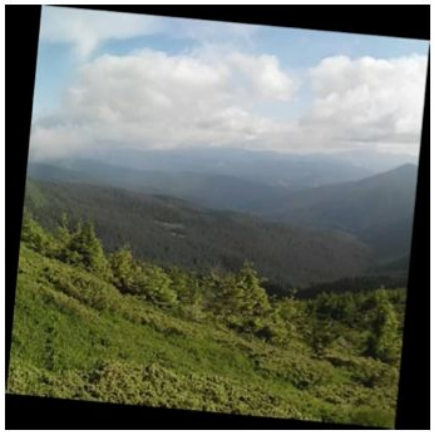
In: $(x_1, y_1) \rightarrow (x_1', y_1')$
$(x_2, y_2) \rightarrow (x_2', y_2')$
$(x_3, y_3) \rightarrow (x_3', y_3')$

Out: $\begin{bmatrix} \mathbf{A} & \mathbf{t} \\ 0 \quad 0 & 1 \end{bmatrix}$

In:

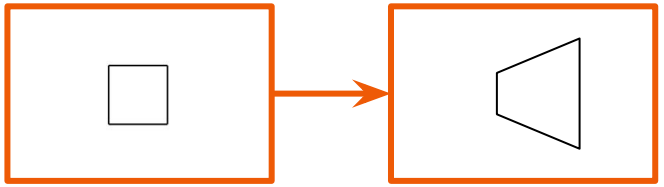


Out:



Homography

8 DoF



$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

Matrix Form

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \sim \mathtt{H} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Also called projective transformation
- **H** must be non-singular (invertible)
- Preserves *straight lines*.

Estimate Homography Minimally: Task 2.1

Given **4 point cspond's** find projective transformation.

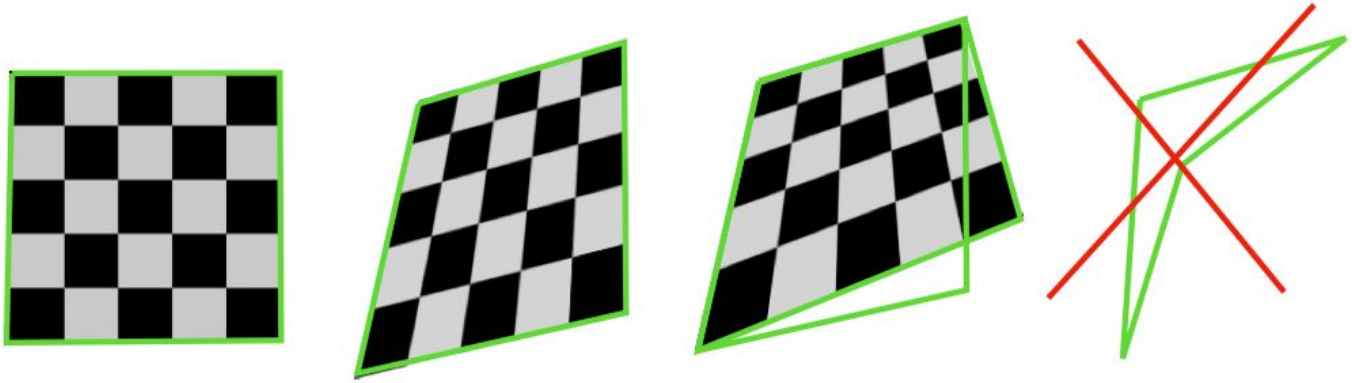
In: $(x_1, y_1) \rightarrow (x_1', y_1')$
$(x_2, y_2) \rightarrow (x_2', y_2')$
$(x_3, y_3) \rightarrow (x_3', y_3')$
$(x_4, y_4) \rightarrow (x_4', y_4')$

Out: H

What arrangement of the 4 point cspond's would be **degenerate** *i.e.* in which case 4 pairs of points are not enough to estimate the homography?

Homography Matrix

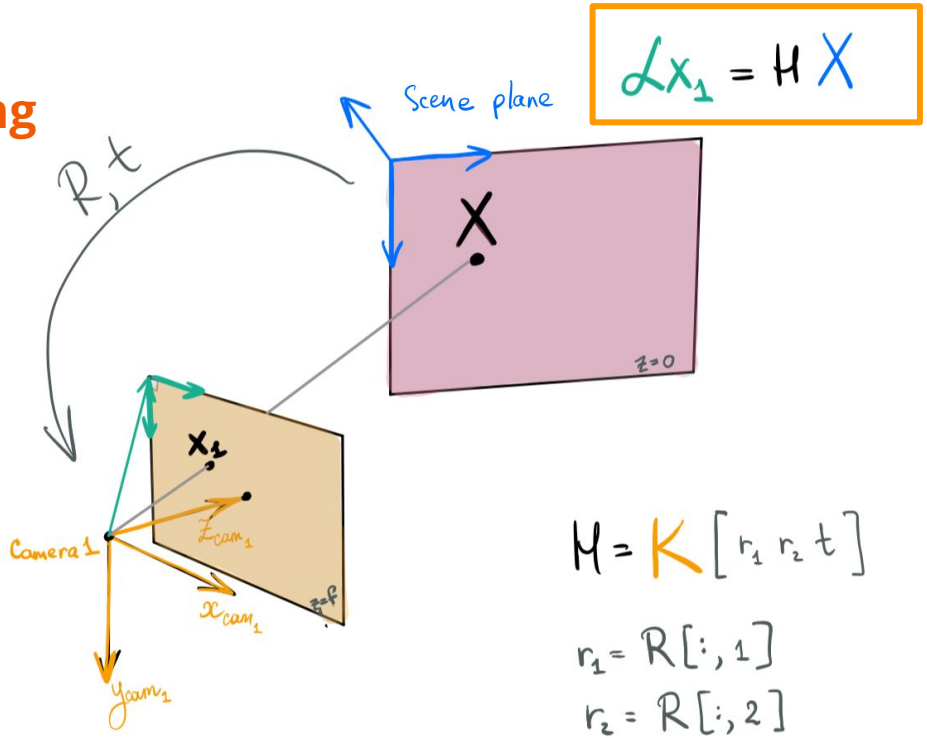
is defined by **4 noncollinear** points that **do not break convex polygons**



Assuming **a pinhole camera model**, any two images of **the same planar surface** in space are related by a **homography**

Homography between Scene Plane ‹-› Image Plane

Inverse of ***H*** is called a **rectifying homography**



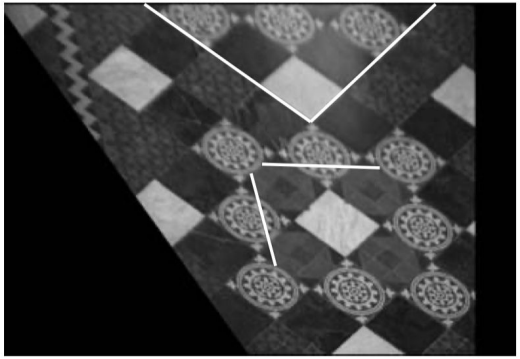
$$H = K [r_1\ r_2\ t]$$

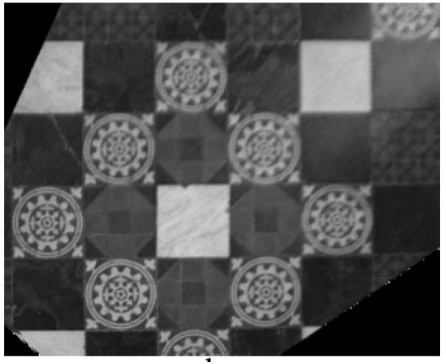
$r_1 = R[:, 1]$

$r_2 = R[:, 2]$

Scene Plane Rectification

Estimated **rectifying homography** is used to rectify scene planes





Homography: Task 2.2 (coding)

Given **4 point cspond's** find projective transformation, apply it to an image, compare with cv2.getPerspectiveTransform

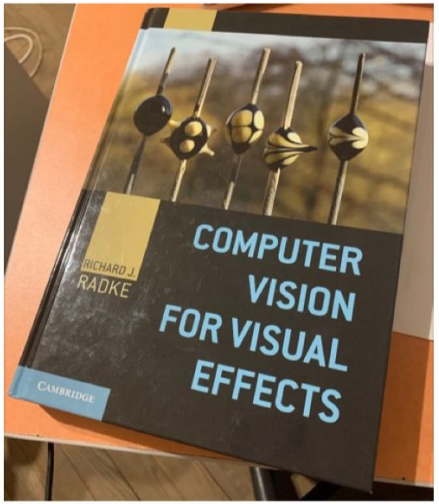
In: $(x_1, y_1) \rightarrow (x_1', y_1')$

$(x_2, y_2) \rightarrow (x_2', y_2')$

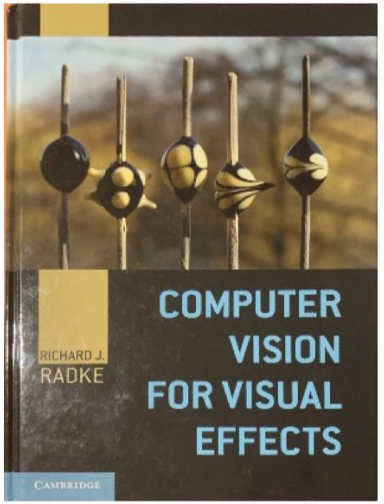
$(x_3, y_3) \rightarrow (x_3', y_3')$

$(x_4, y_4) \rightarrow (x_4', y_4')$

In:



Out:



Homography between Image Planes*

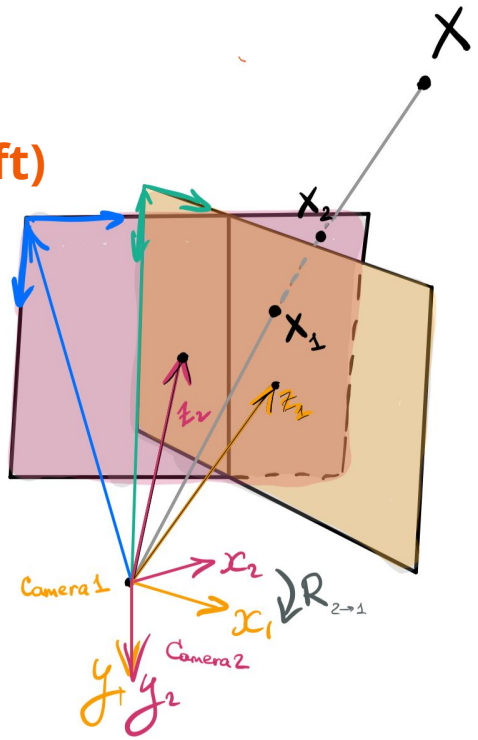
*Assume that the camera **can be only rotated (no shift)**

Let's align the world coordinate system with the first camera position and orientation

$$\alpha_1 \mathbf{x_1} = \mathrm{K}\left[\mathrm{I}|\mathbf{t}_0\right]\mathbf{X} \qquad \mathbf{t}_0 = \begin{pmatrix}0 & 0 & 1\end{pmatrix}^\top$$

Then, for the second frame, extrinsics consists of rotation only

$$\alpha_2 \mathbf{x_2} = \mathrm{K}\left[\mathrm{R}|\mathbf{t}_0\right]\mathbf{X}$$



Homography is a Conjugate Rotation

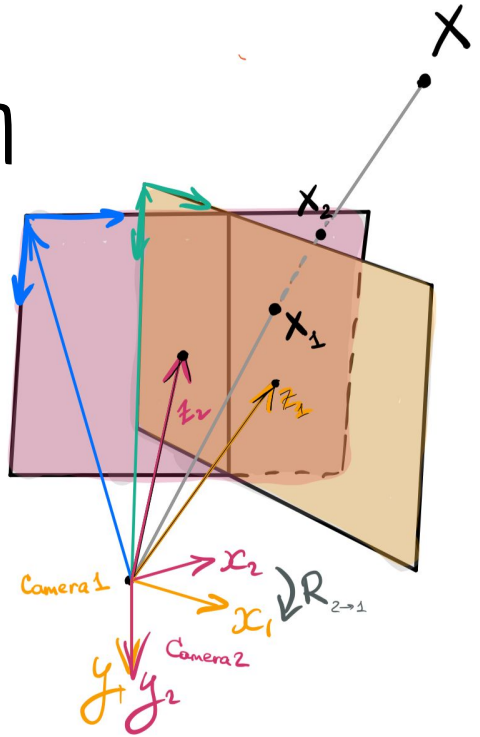
By combining the two equations we get

$$\alpha_1 \mathrm{K}^{-1}\mathbf{x_1} = \alpha_2 \mathrm{R}^{\top}\mathrm{K}^{-1}\mathbf{x_2}$$

or

$$\beta\mathbf{x_1} = \boxed{\mathrm{K}\mathrm{R}^{\top}\mathrm{K}^{-1}}\mathbf{x_2}$$

this transformation is called **conjugate rotation**



It is a special form of homography arising under the assumption we made. It is common for panoramas.

One can apply this homography to map all images onto one (**reference**) image plane — stitch them.

What if we do not have exact correspondences?

$$(x_i, y_i) \rightarrow (x_i', y_i') + \delta_i$$

How can we use more matches to make estimation error smaller?

What if we do not have exact correspondences?

$$(x_i, y_i) \rightarrow (x_i', y_i') + \delta_i$$

How can we use more matches to make estimation error smaller?

Use **SVD**

Direct Linear Transformation

Let's say we want to estimate homography matrix

If the j-th row of the matrix $\mathtt{H}$ is denoted by $\mathbf{h}^{j\mathsf{T}}$, then we may write

$$\mathtt{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}^{1\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{2\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{3\mathsf{T}}\mathbf{x}_i \end{pmatrix}.$$

Writing $\mathbf{x}'_i = (x'_i, y'_i, w'_i)^\mathsf{T}$, the cross product may then be given explicitly as

$$\mathbf{x}'_i \times \mathtt{H}\mathbf{x}_i = \begin{pmatrix} y'_i\mathbf{h}^{3\mathsf{T}}\mathbf{x}_i - w'_i\mathbf{h}^{2\mathsf{T}}\mathbf{x}_i \\ w'_i\mathbf{h}^{1\mathsf{T}}\mathbf{x}_i - x'_i\mathbf{h}^{3\mathsf{T}}\mathbf{x}_i \\ x'_i\mathbf{h}^{2\mathsf{T}}\mathbf{x}_i - y'_i\mathbf{h}^{1\mathsf{T}}\mathbf{x}_i \end{pmatrix}.$$

Direct Linear Transformation

Since $\mathbf{h}^{j\mathsf{T}}\mathbf{x}_i = \mathbf{x}_i^{\mathsf{T}}\mathbf{h}^j$ for $j = 1, \ldots, 3$, this gives a set of three equations in the entries of $\mathtt{H}$, which may be written in the form

$$\begin{bmatrix} \mathbf{0}^{\mathsf{T}} & -w_i'\mathbf{x}_i^{\mathsf{T}} & y_i'\mathbf{x}_i^{\mathsf{T}} \\ w_i'\mathbf{x}_i^{\mathsf{T}} & \mathbf{0}^{\mathsf{T}} & -x_i'\mathbf{x}_i^{\mathsf{T}} \\ -y_i'\mathbf{x}_i^{\mathsf{T}} & x_i'\mathbf{x}_i^{\mathsf{T}} & \mathbf{0}^{\mathsf{T}} \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}. \tag{4.1}$$

These equations have the form $\mathtt{A}_i\mathbf{h} = \mathbf{0}$, where $\mathtt{A}_i$ is a 3×9 matrix, and $\mathbf{h}$ is a 9-vector made up of the entries of the matrix $\mathtt{H}$,

$$\mathbf{h} = \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix}, \qquad \mathtt{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{4.2}$$

Direct Linear Transformation

If have an overdetermined case (more than 4 matches)

$$\min_{\mathbf{h}} \|\mathbf{A}\mathbf{h}\| \quad \text{subject to} \quad \|\mathbf{h}\| = 1$$

Identical to

$$\min_{\mathbf{h}} \frac{\|\mathbf{A}\mathbf{h}\|}{\|\mathbf{h}\|}$$

The solution is the (unit) eigenvector of $\boldsymbol{A^TA}$ with the least eigenvalue — thus we need to use **SVD** of $\boldsymbol{A}$.

What if we have outliers?

$$(x_i, y_i) \rightarrow (x_i', y_i') + \delta_i$$

$$\exists i : \|\delta_i\| > \Delta$$

Use **RANSAC:**

- Hypothesize: randomly pick **4** point csponds, compute exact homography
- Verify: compute the geometric errors on all point csponds

$$d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}_i', \hat{\mathbf{x}}_i')^2 \quad \text{subject to } \hat{\mathbf{x}}_i' = \hat{\mathbf{H}}\hat{\mathbf{x}}_i$$

- Verify: calculate the inliers (consensus set)
- Repeat. Choose a homography that maximizes the consensus set

OK, how do we get point correspondences?

OK, how do we get point correspondences?

We need features!

What is a Feature

- Find the exact location of the patches in the original image
- How many correct results can you find?

What is a Feature

- Find the exact location of the patches in the original image
- How many correct results can you find?
- A and B are flat surfaces and they are spread over a lot of area.

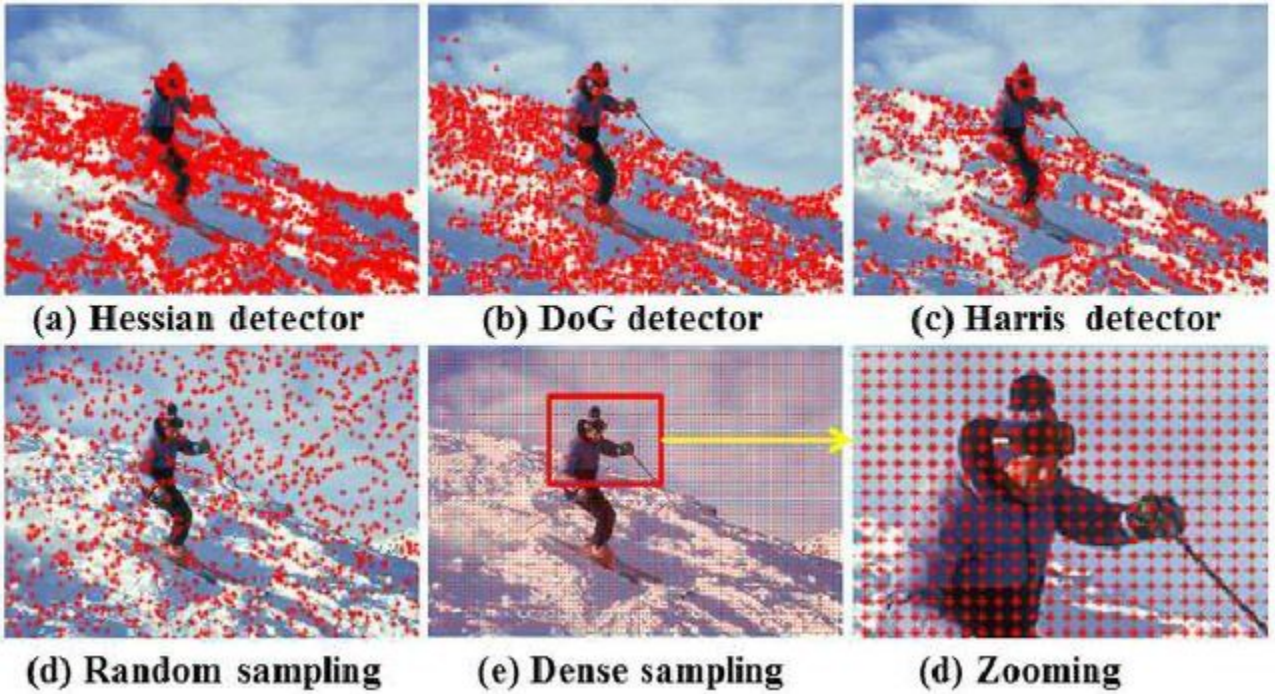
What is a Feature

- Find the exact location of the patches in the original image
- How many correct results can you find?
- A and B are flat surfaces and they are spread over a lot of area.
- C and D are edges of the building. Better, but an approximate location.

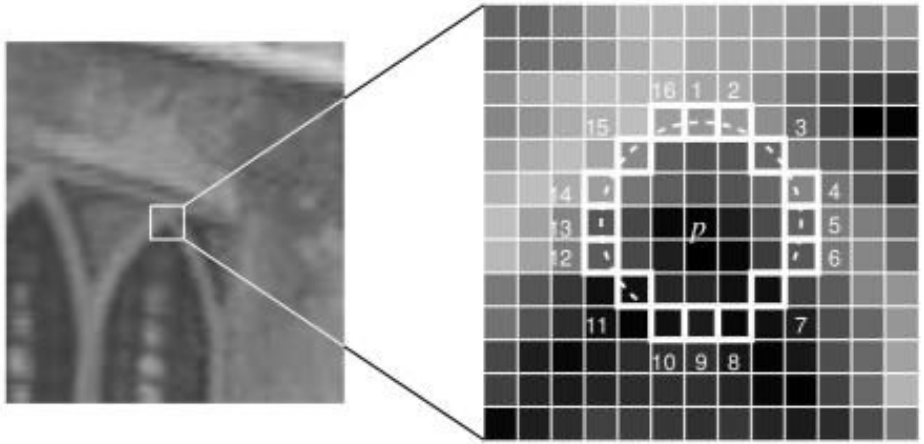
What is a Feature

- Find the exact location of the patches in the original image
- How many correct results can you find?
- A and B are flat surfaces and they are spread over a lot of area.
- C and D are edges of the building. Better, but an approximate location.
- E and F are some corners of the building, and can be easily found. Wherever you move this patch, it will look different. So they can be considered as **good features**.

Feature Detection



Features from Accelerated Segment Test (FAST)



- The pixel **x** is a corner if there exists a set of **12** contiguous pixels in the circle (of 16 pixels) which are all brighter than ***I(x)+t***, or all darker than ***I(x)−t***.
- There exists a high speed test
- See more: https://docs.opencv.org/3.4/df/d0c/tutorial_py_fast.html

Covariance and Invariance

The feature is **covariant** to a family of transformations — in case the image is transformed the feature (shape) is transformed in the same way.

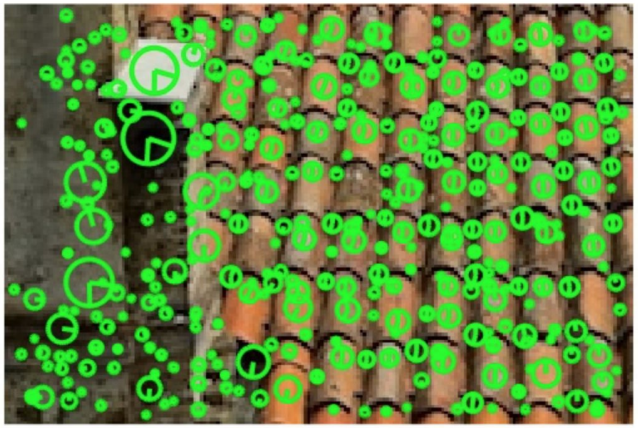
The feature is **invariant** to a family of transformations — in case the image is transformed the feature description doesn't change.

The latter is crucial for matching but is usually achieved through covariance. Covariance allows mapping features to normalized space.

Covariant Features

Rotational Covariance

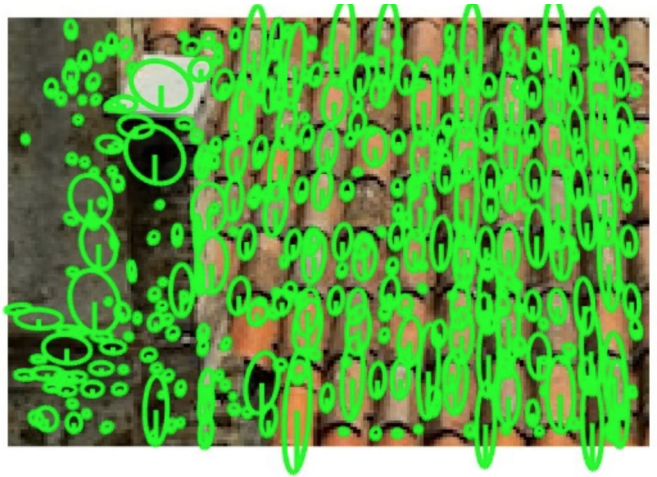
- Can be achieved by estimating the feature orientation



Features with orientation detection.

Affine Covariance

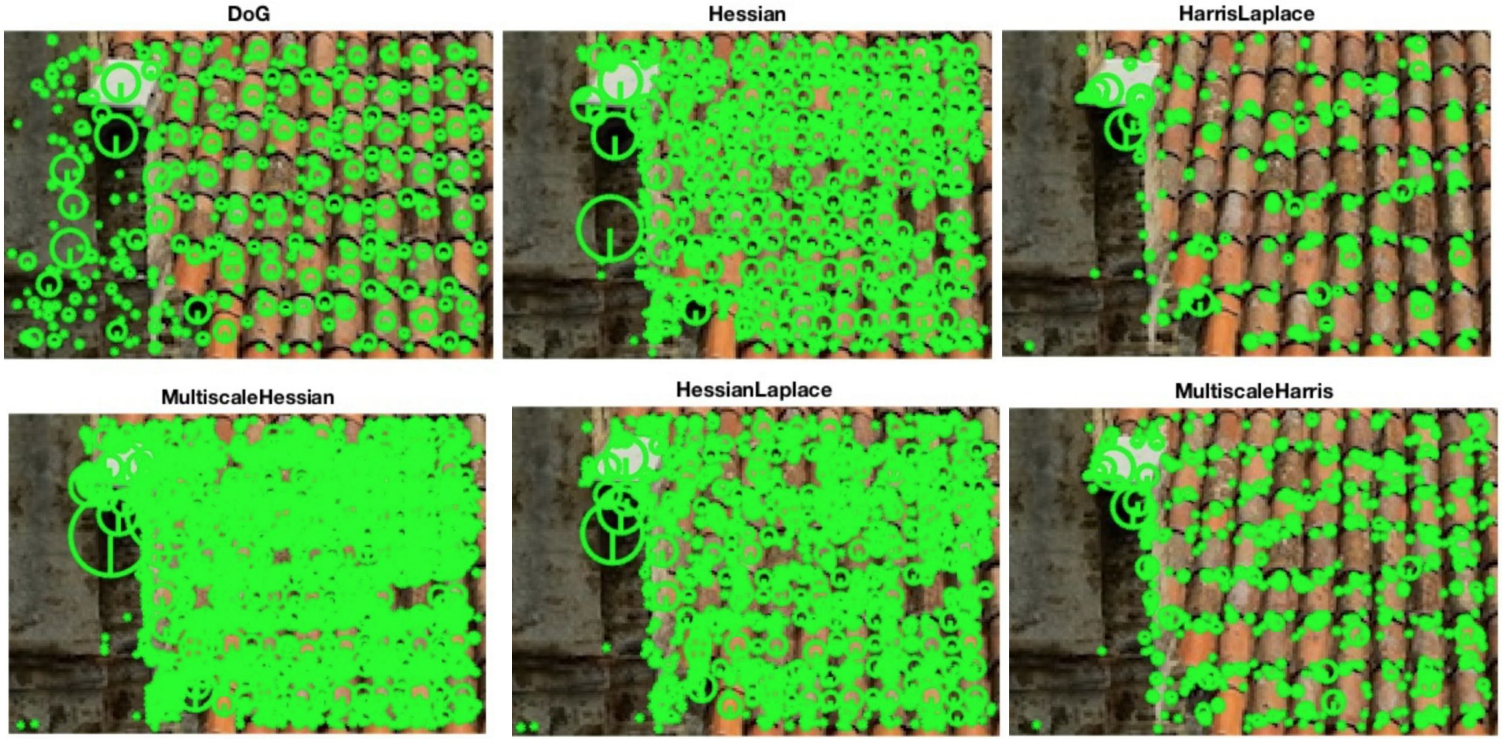
- Can be achieved by estimating the affine shape of an image region (**affine adaptation**)



Affinely adapted features.

http://www.vlfeat.org/overview/covdet.html

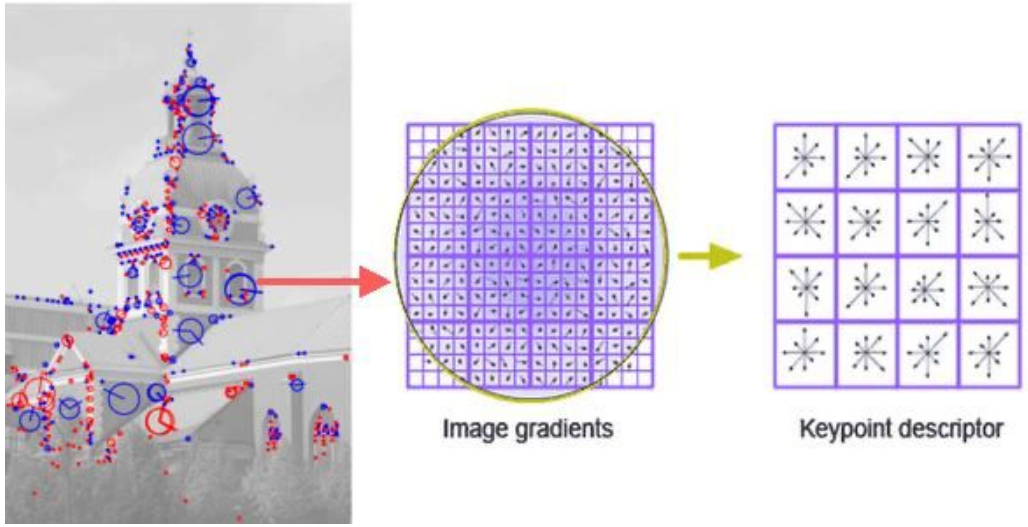
Covariant Features



http://www.vlfeat.org/overview/covdet.html

Feature Descriptors

Scale-Invariant Feature Transform (SIFT)
Speeded Up Robust Feature (SURF)

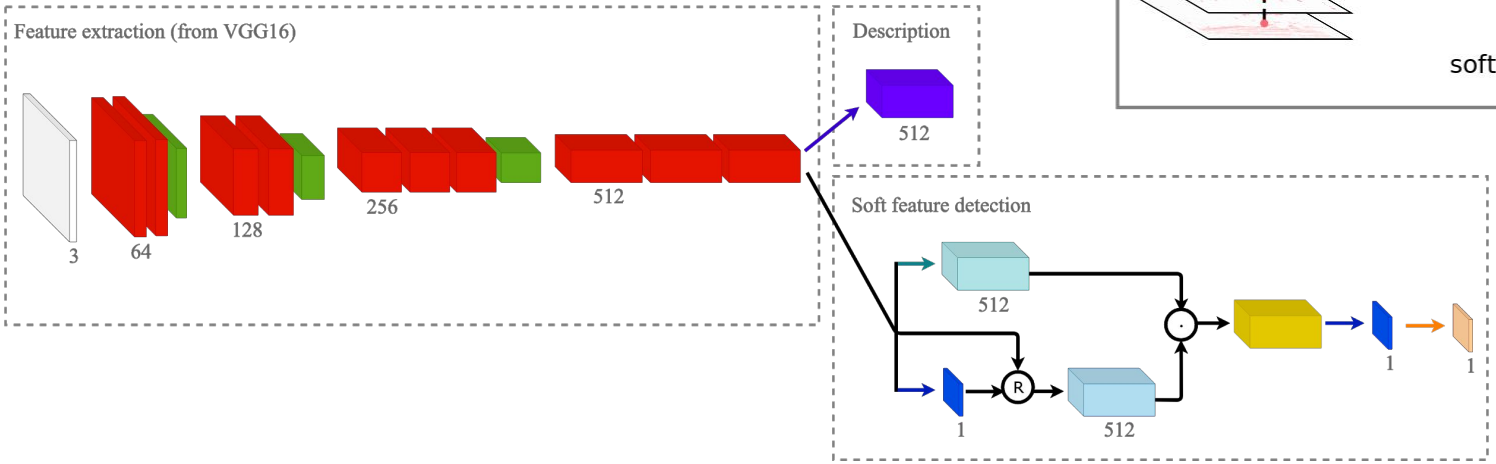


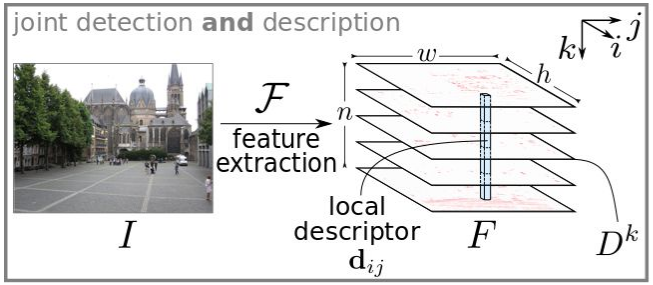
Convolutional Neural Networks

CNNs can do both detection and description simultaneously

e.g. **D2-Net**

Architecture





Output tensors

Convolution

Max Pooling

L2-Normalization
(channel-wise)

Soft Local-Maximum

Maximum
(channel-wise)

1

Soft Channel Selection

Multiplication
(element-wise)

Normalization
(image-level)

1

Operations

Data Flow

L2-Normalization
(channel-wise)

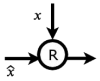
$$y_{ij}^k = \frac{x_{ij}^k}{\|x_{ij}\|_2}$$

Soft Local-Maximum

$$y_{ij}^k = \frac{\exp\left(x_{ij}^k\right)}{\sum_{(p,l)\in\mathcal{N}(i,j)} \exp\left(x_{pl}^k\right)}$$

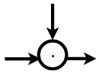
Maximum
(channel-wise)

$$y_{ij} = \max_t \left\{x_{ij}^t\right\}$$



Ratio-to-max

$$y_{ij}^k = \frac{x_{ij}^k}{\hat{x}_{ij}} \quad \left(\text{here } \hat{x}_{ij} = \max_t \left\{x_{ij}^t\right\}\right)$$



Multiplication
(element-wise)

$$y = x_1 \cdot x_2$$

Normalization
(image-level)

$$y_{ij} = \frac{x_{ij}}{\sum_{(p,l)} x_{pl}}$$

Inputs / Outputs

Input image

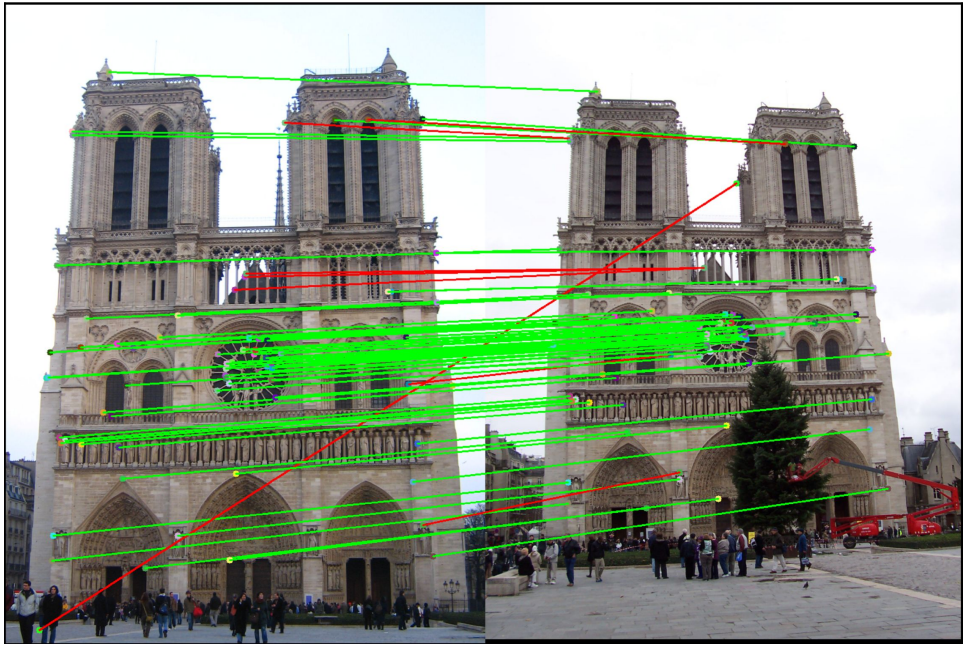
Soft Detection Scores

Local Descriptors

Feature Matching

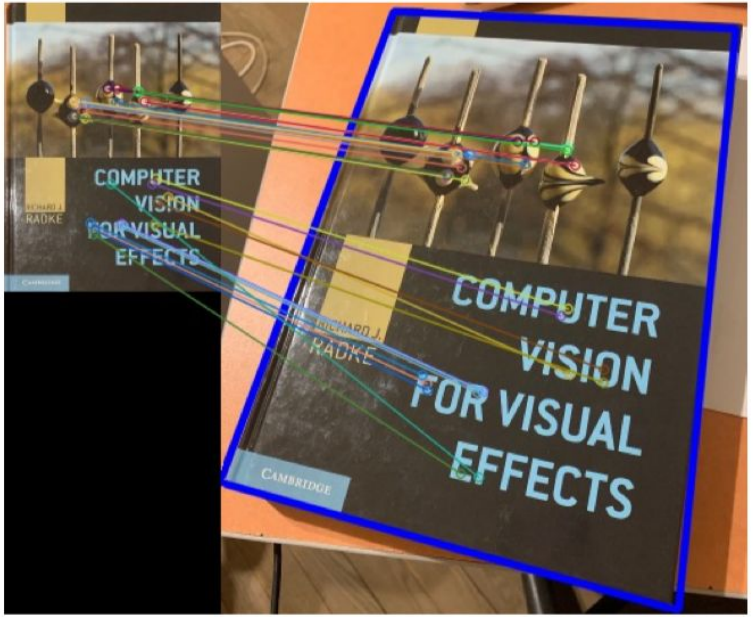
- Brute-Force Matcher
- Fast Library for Approximate Nearest Neighbors (FLANN) Matcher

Gives us desired point csond’s



Homography from Matched Features: Task 3.1

Given **2 images** of the **scene plane**, detect features, describe and match them (using OpenCV / PyTorch / etc), find homography transformation from matched features, compare with cv2.findHomography.



References

[1] Pajdla, Tomas. Elements of geometry for computer vision. FEE CTU, 2013

[2] Hartley, Richard, and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003

[3] Maksym Davydov. Computer Vision − Image correspondence, 2019

[4] OpenCV Documentation

[5] VLFeat Documentation