Introduction to Computer Vision

Lecture 1. Introduction to 3D reconstruction. Hough Transform. RANSAC

Sergey Zagoruyko

November 7, 2023

Skoltech
Skolkovo Institute of Science and Technology

Last lecture

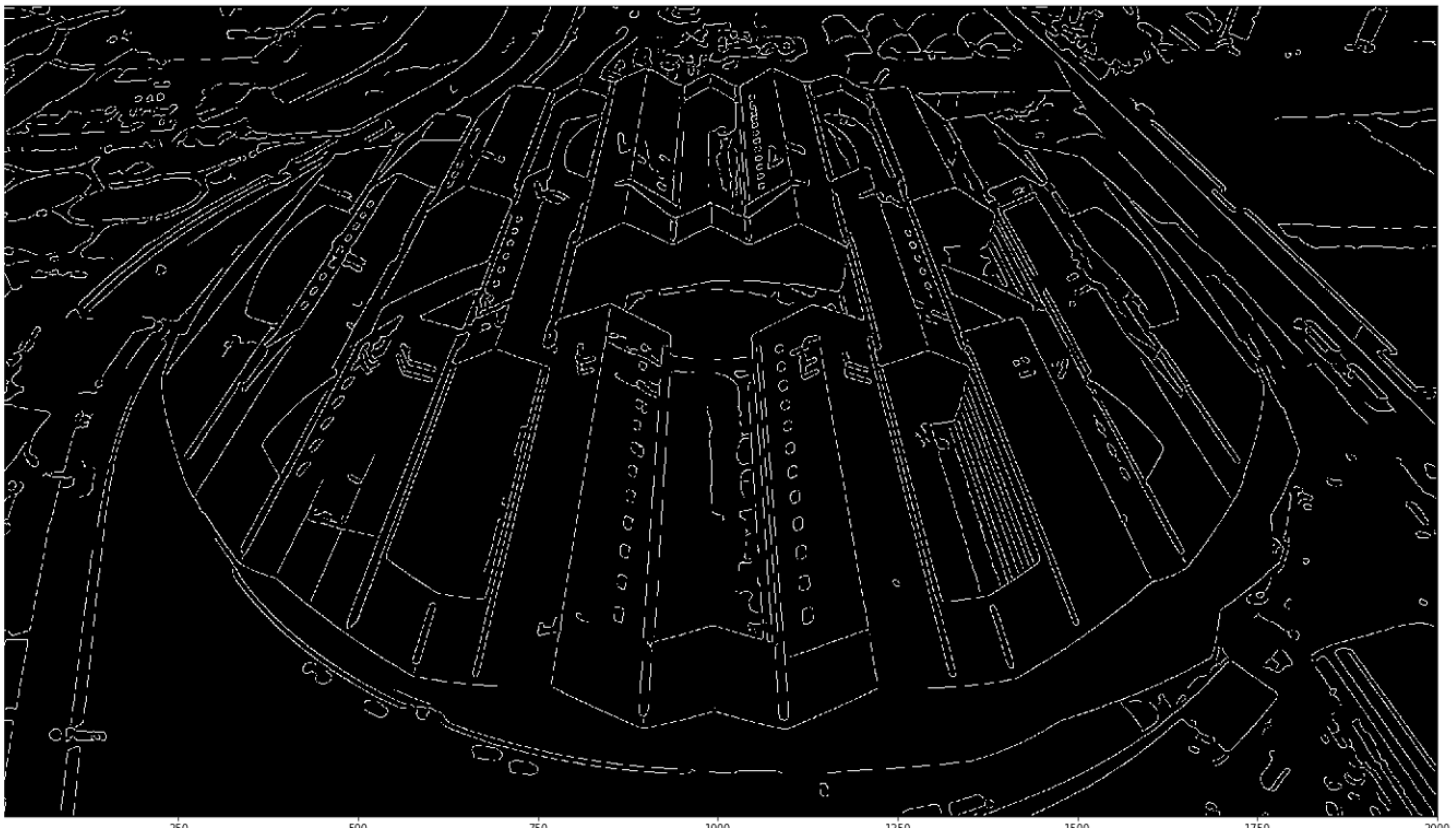
- Template matching
- Edge detection (Canny, Sobel)

This lecture

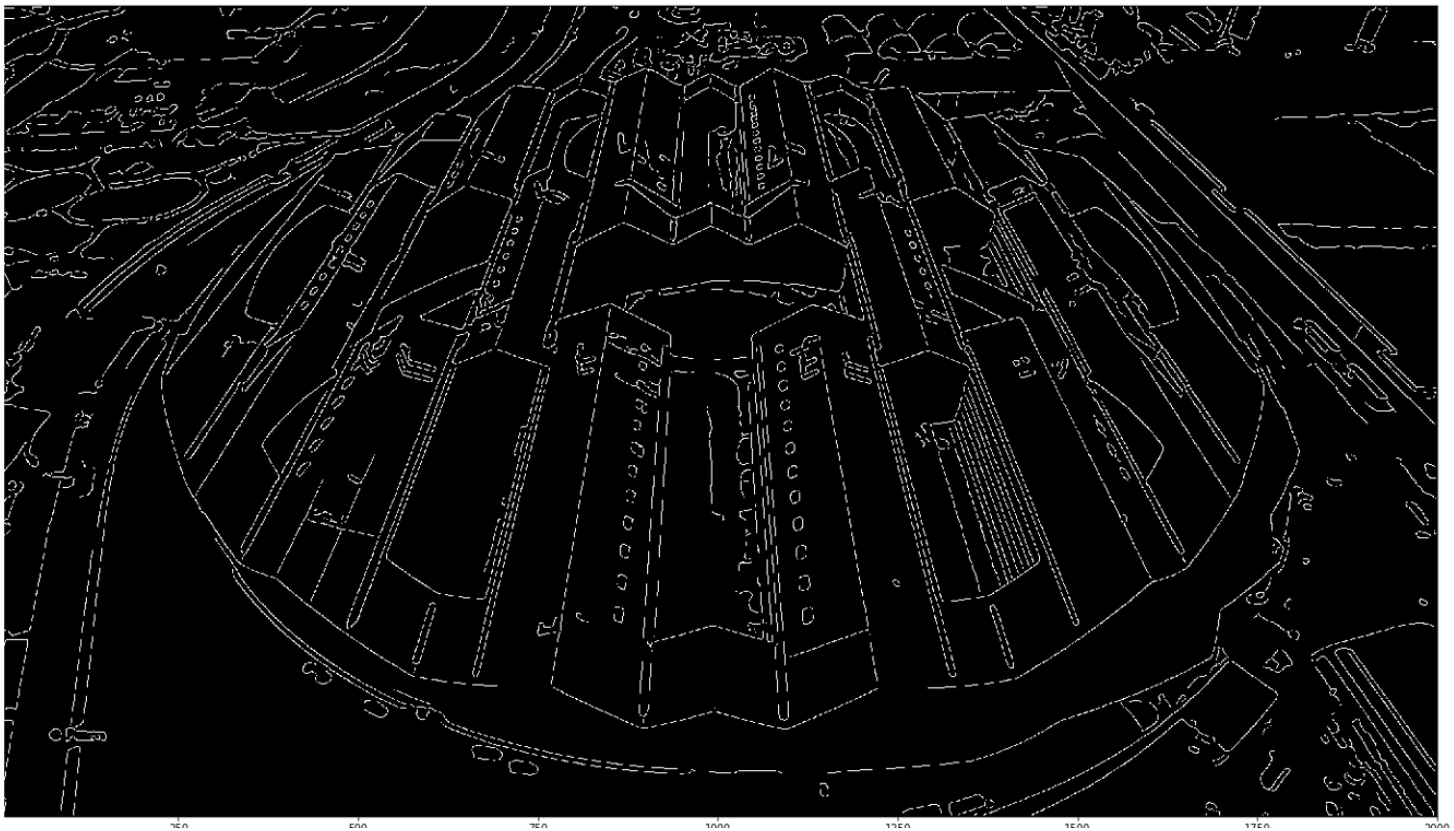
- Hough Transform
- RANSAC
- Intro to 3D reconstruction

Table of Contents

What can we extract from detected edges?

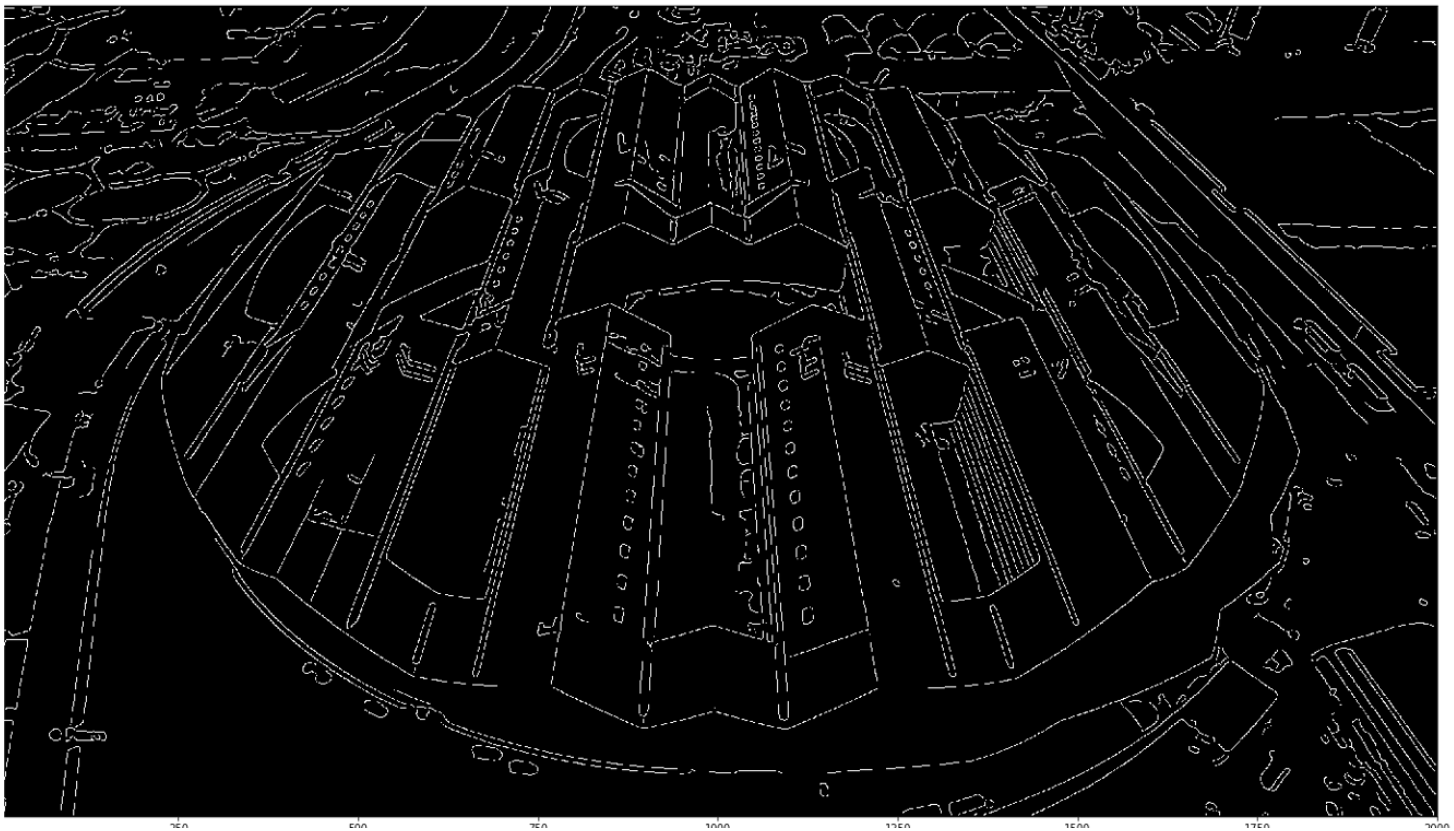


What can we extract from detected edges?



- Predefined templates (the lamp counter)
- Some figures like squares / circles

What can we extract from detected edges?

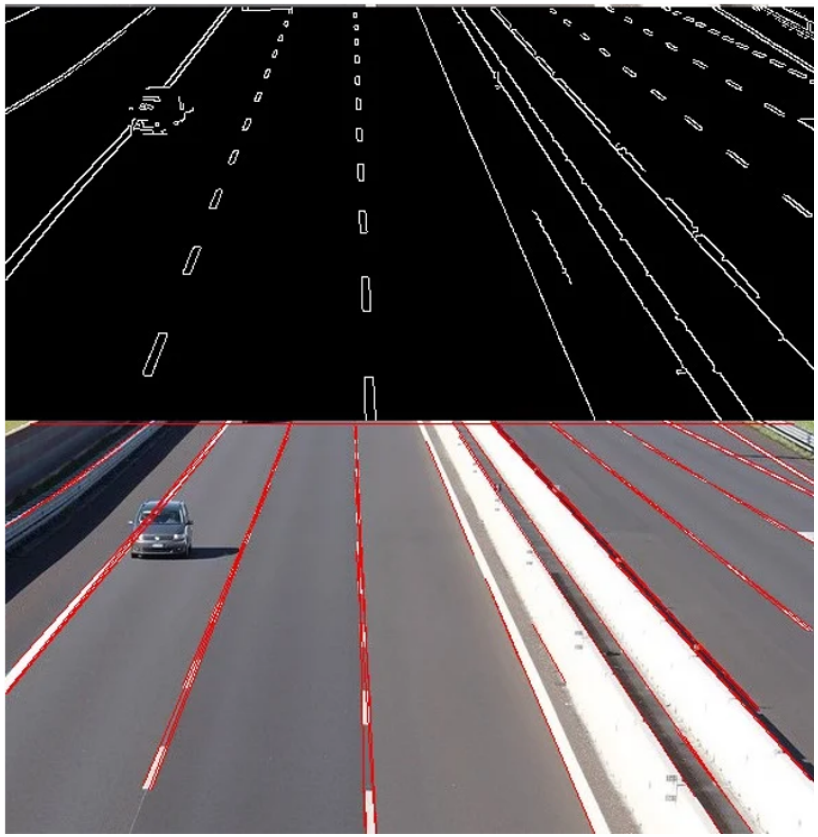


- Predefined templates (the lamp counter)
- Some figures like squares / circles
- Lines! How to get parametric equation of a line from edges?

Hough transform

Using Hough transform!

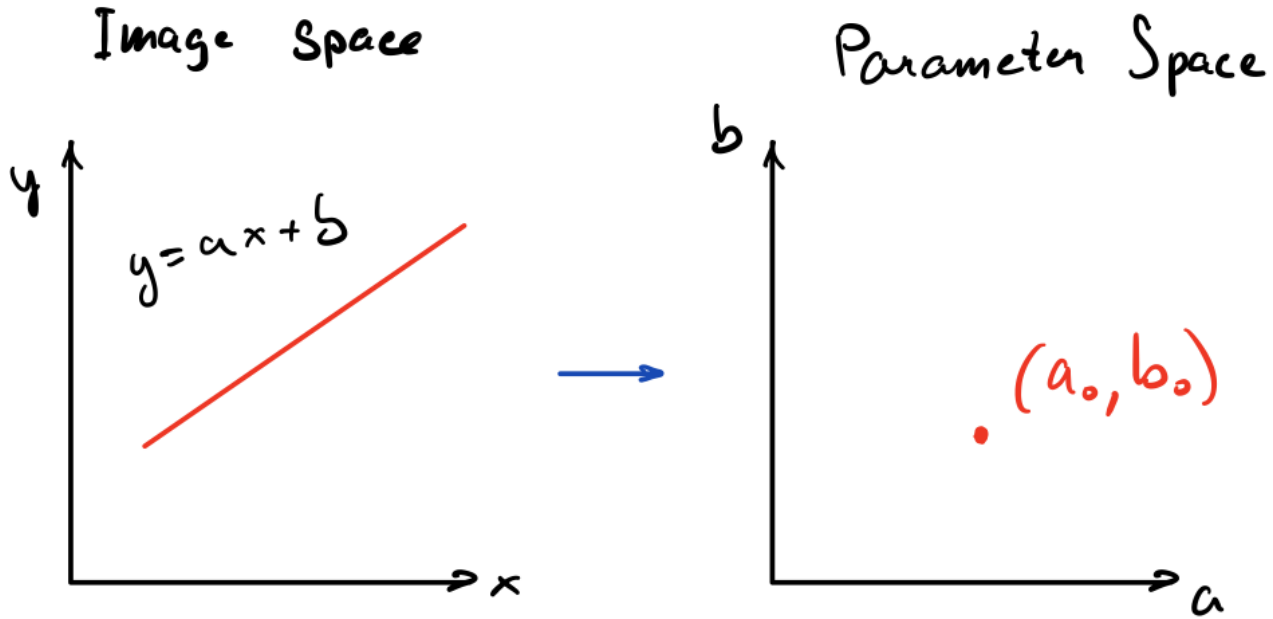
- Core idea: try to fit all possible lines and find ones with **good fit**
- Detected lines gives us rotation angle, view point reconstruction
- Robust detection under noise
- Can detect other structures (not only lines - e.g., circles) if their parametric equation is known



Source: `https://learnopencv.com/hough-transform-with-opencv-c-python/`

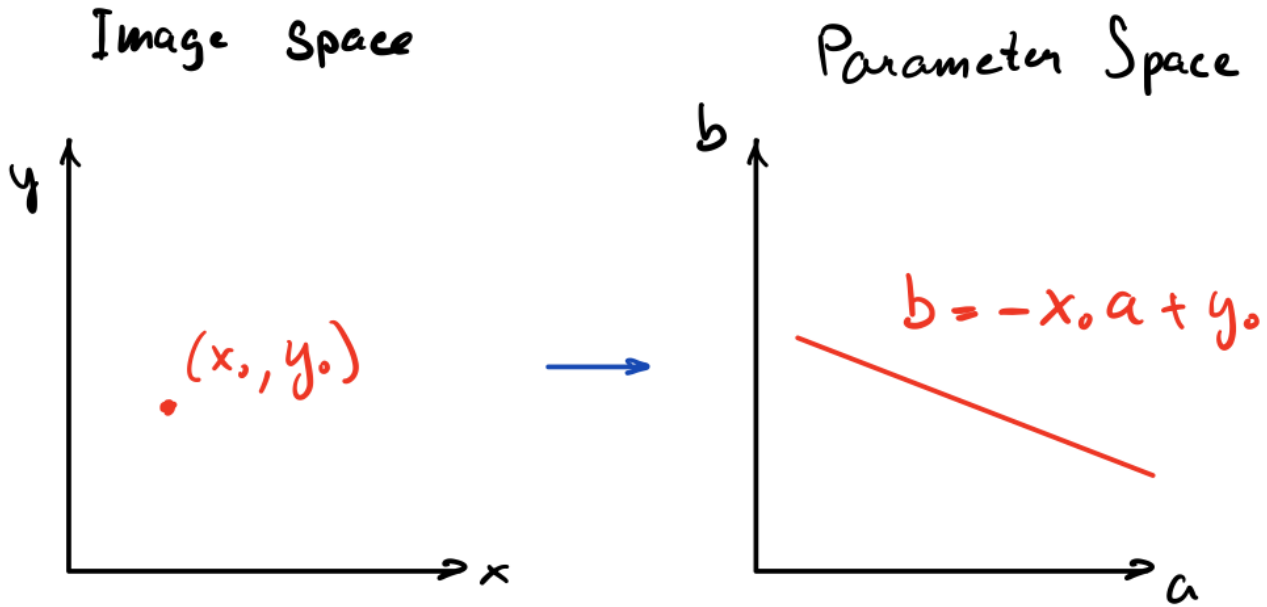
Hough transform

- Consider a point of known coordinates (x_i, y_i).
- Straight lines that pass that point: $y_i = a \cdot x_i + b$
- Equation can be rewritten as follows: $b = -a \cdot x_i + y_i$
- Line $y = a_0 \cdot x + b_0$ gives us a point in (a, b) space



Hough transform

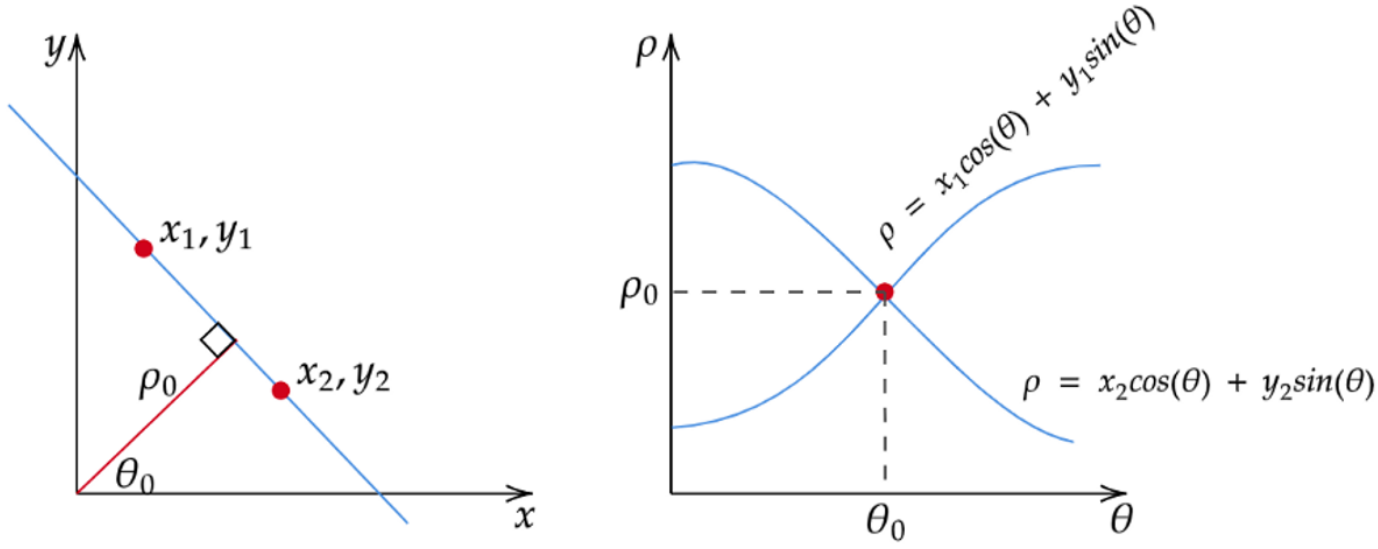
- Consider a point of known coordinates (x_i, y_i).
- Straight lines that pass that point: $y_i = a \cdot x_i + b$
- Equation can be rewritten as follows: $b = -a \cdot x_i + y_i$
- Line $y = a_0 \cdot x + b_0$ gives us a point in (a, b) space
- Point (x_i, y_i) gives us line in (a, b)-space



Hough transform

An alternative parametrization for a linear Hough transform:

$$x \cos\theta + y \sin\theta = \rho$$

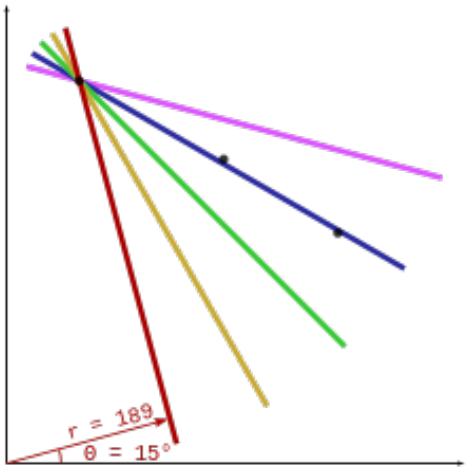


Source: `https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549`

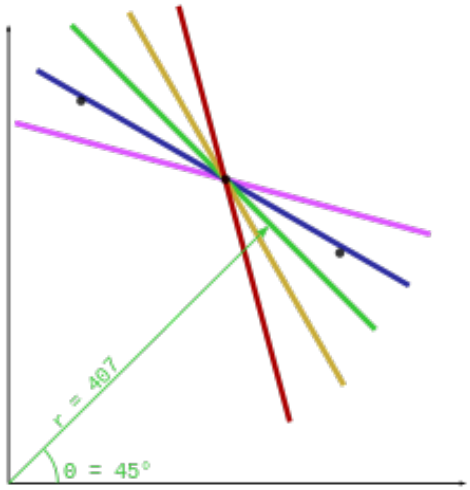
Hough transform

An alternative parametrization for a linear Hough transform:

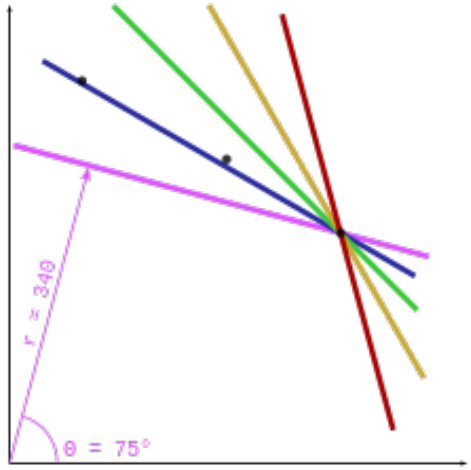
$$x \cos\theta + y \sin\theta = \rho$$



θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4



θ	r
15	318.5
30	376.8
45	407.3
60	409.8
75	385.3

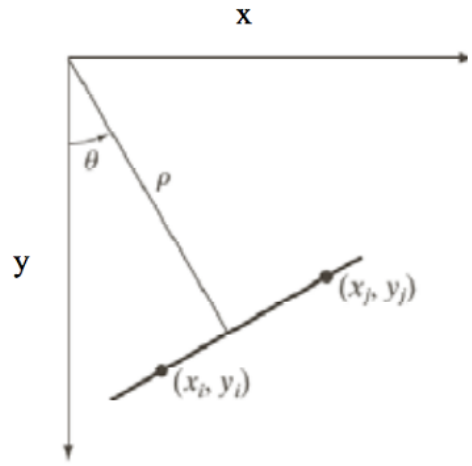


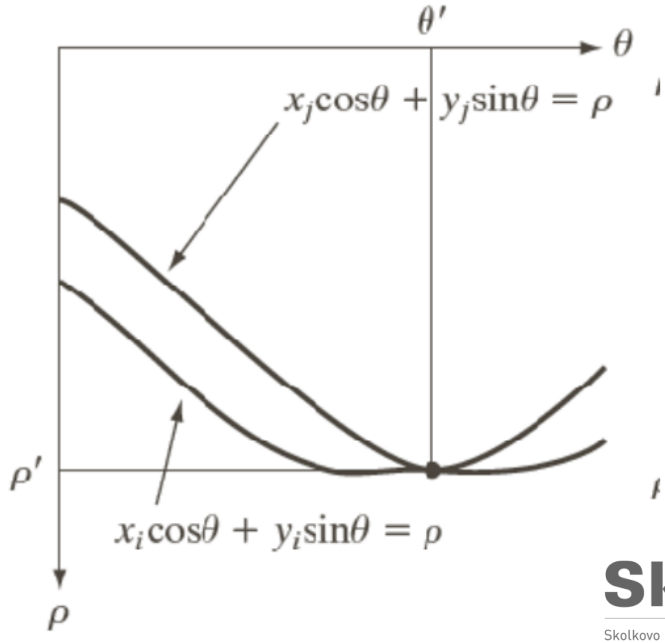
θ	r
15	419.0
30	443.6
45	438.4
60	402.9
75	340.1

Source: Wikipedia

Hough transform

- A vertical line will have $\theta = 0$ and ρ equal to the intercept with the x-axis.
- A horizontal line will have $\theta = 90$ and ρ equal to the intercept with the y-axis.

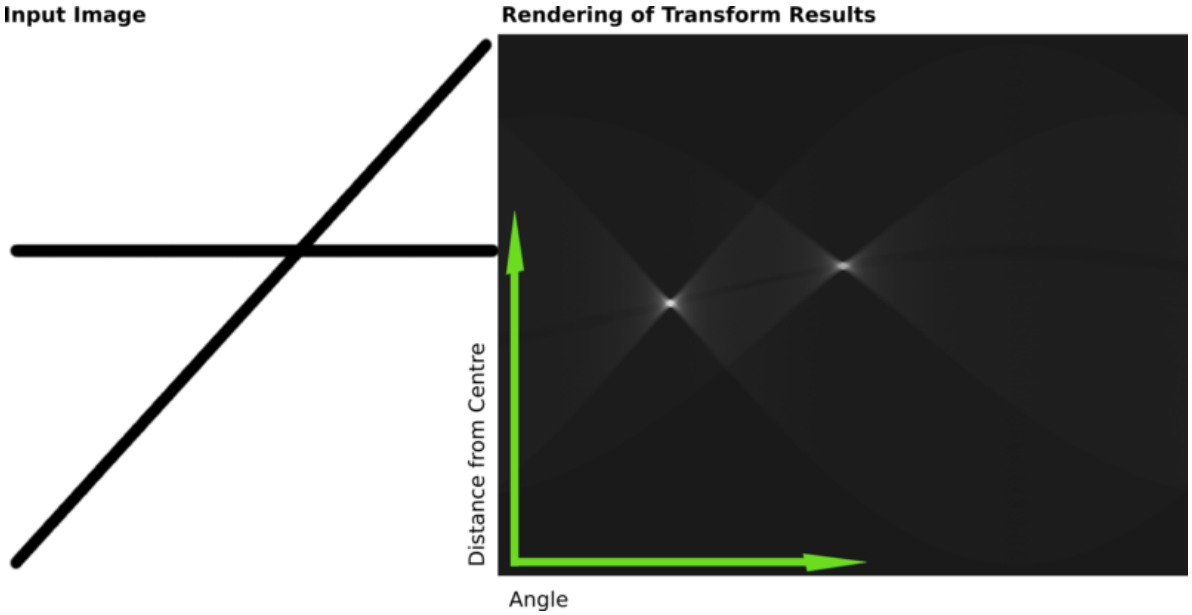




Skoltech
Skolkovo Institute of Science and Technology

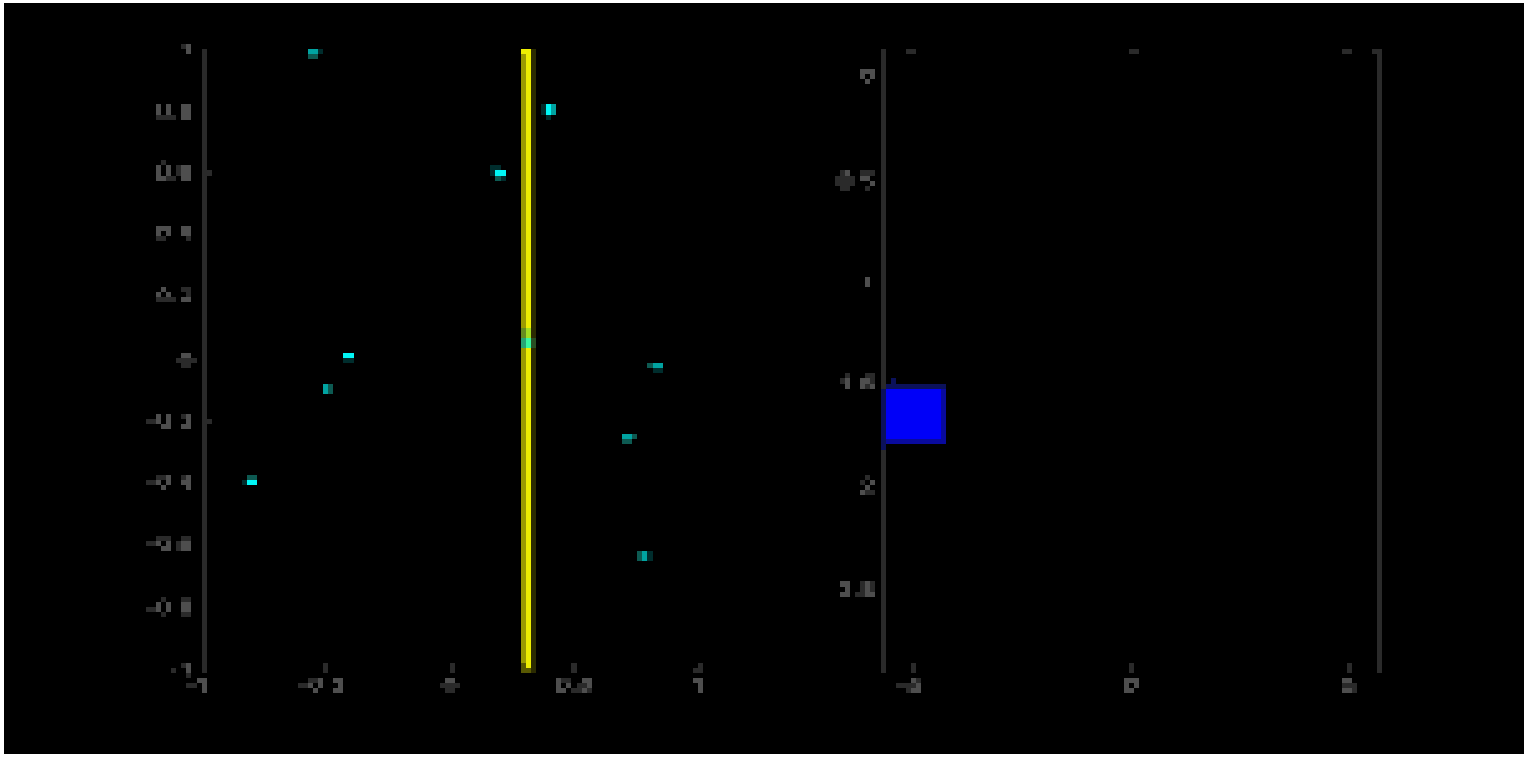
Hough transform

1. Detect edges, e.g. via Canny
2. For every line calculate the intersection with the edges image (= get intensity of all pixels in (a, b) or (r, θ) space)
3. Filter too similar lines (optional)
4. Select best lines (= pixels with high intensity values)



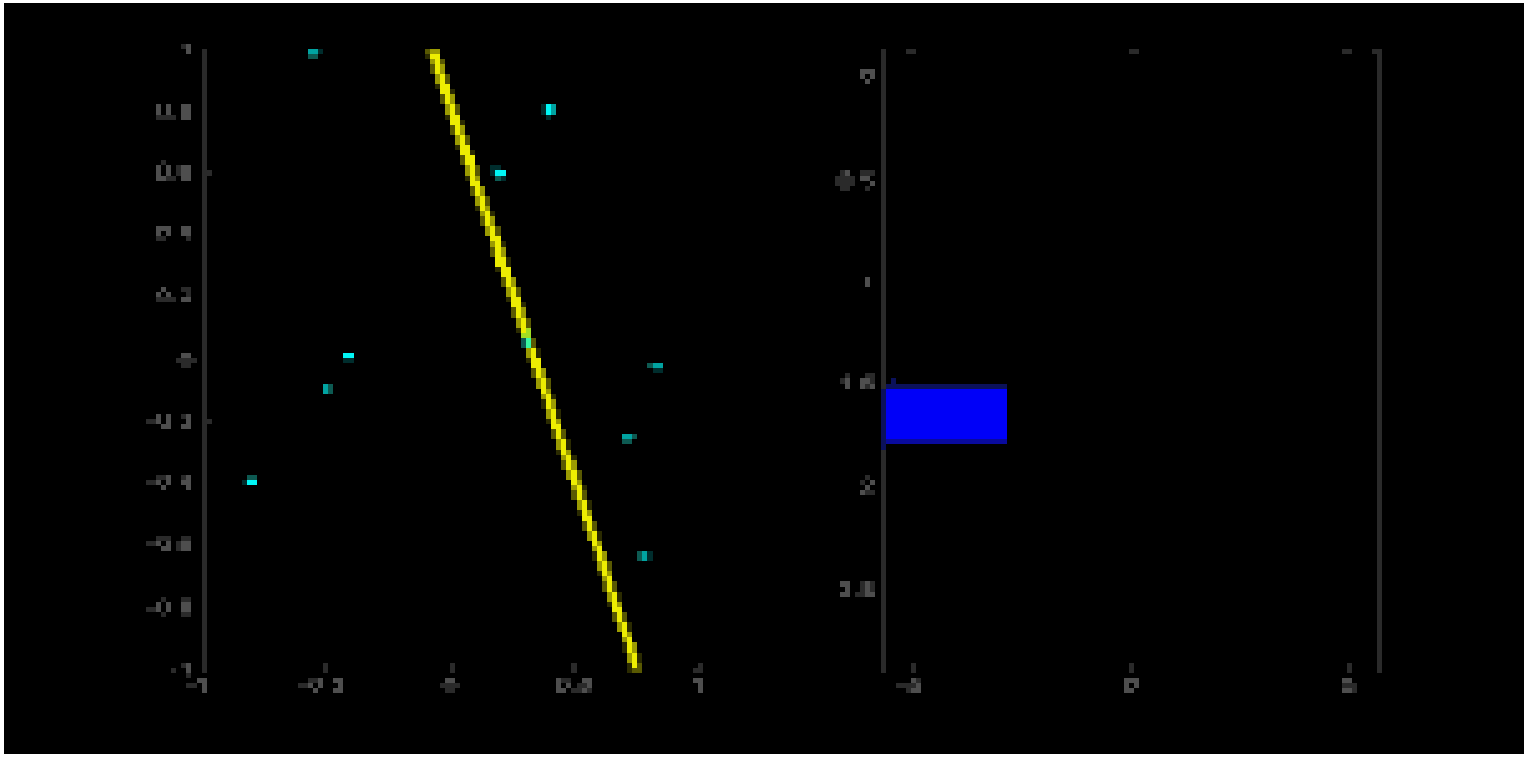
Source: Wikipedia

Hough transform



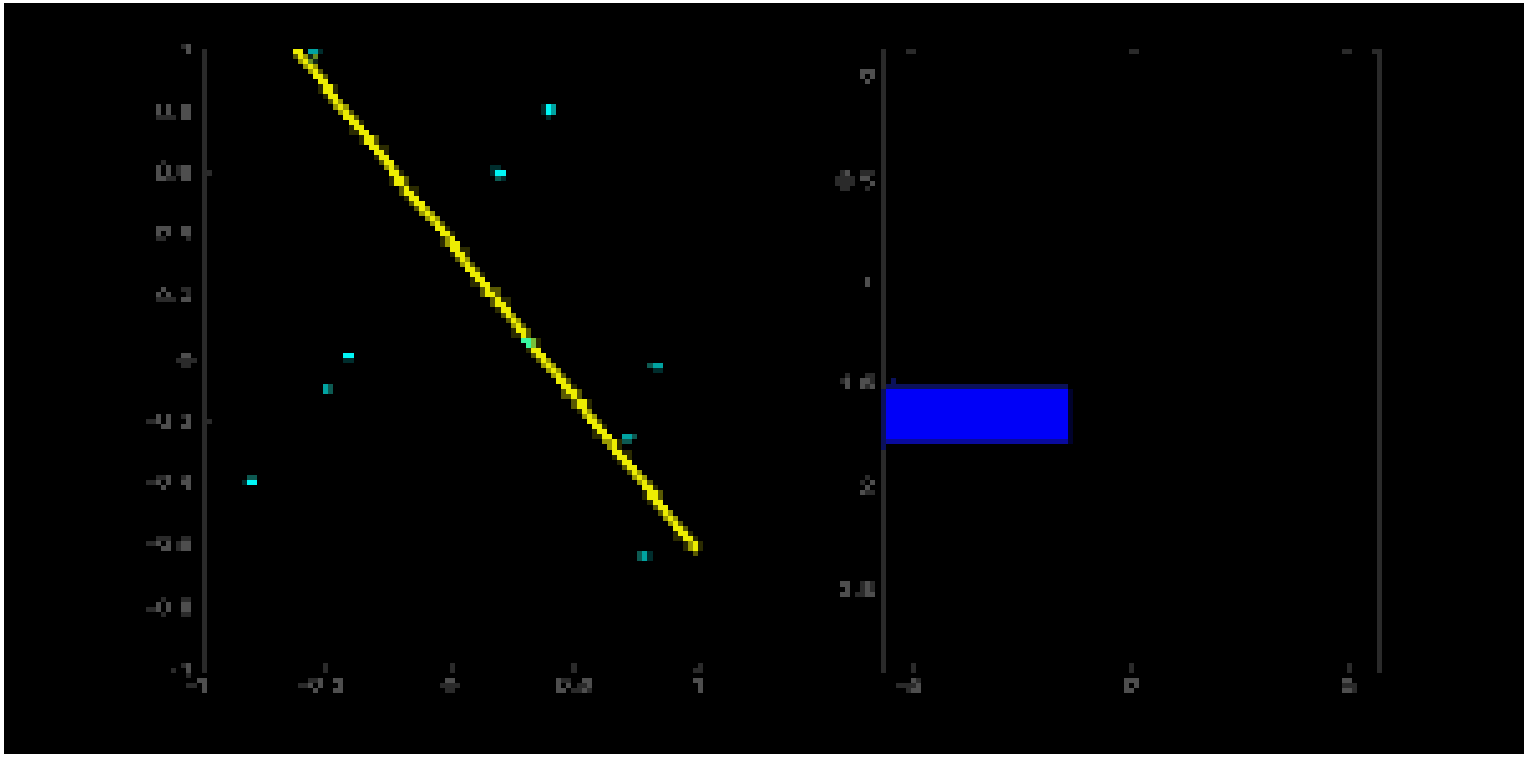
Source: `https://homepages.inf.ed.ac.uk/amos/hough.html`

Hough transform



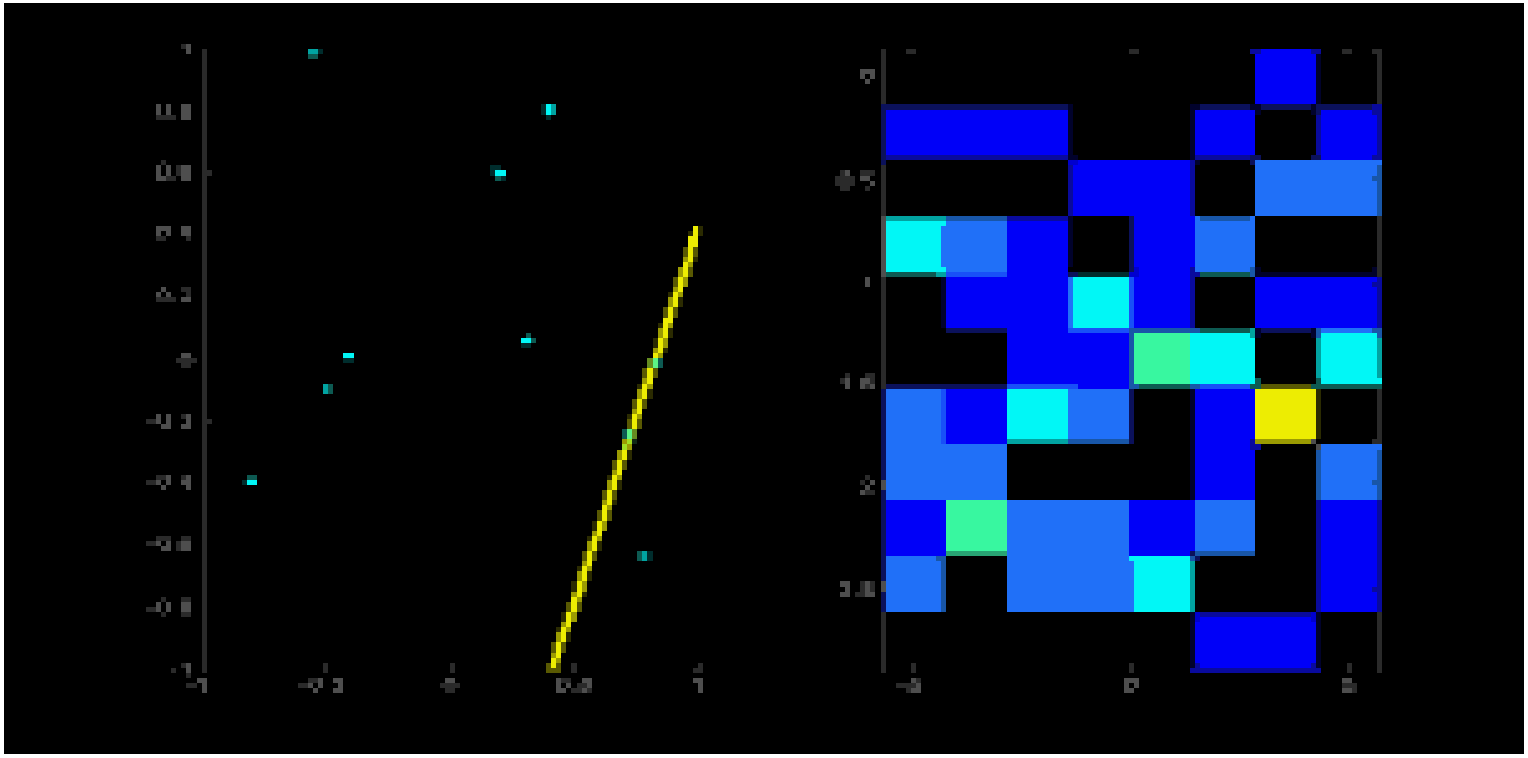
Source: `https://homepages.inf.ed.ac.uk/amos/hough.html`

Hough transform



Source: `https://homepages.inf.ed.ac.uk/amos/hough.html`

Hough transform



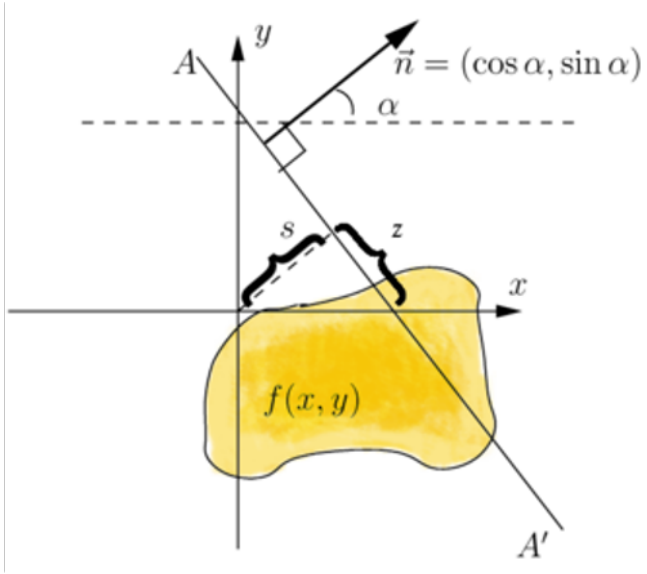
Source: `https://homepages.inf.ed.ac.uk/amos/hough.html`

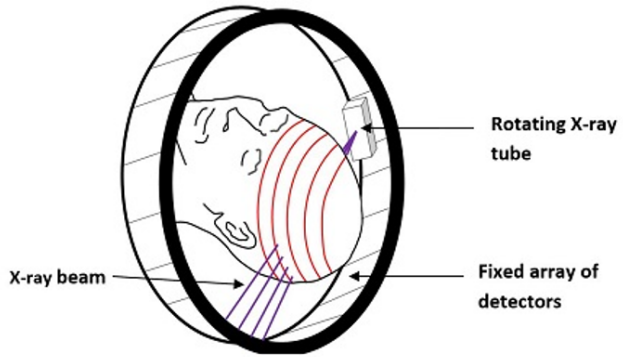
Hough transform parameterization

An alternative parametrization for a linear Hough transform:

$$x \cos\theta + y \sin\theta = \rho$$

Is it possible to solve an inverse problem?





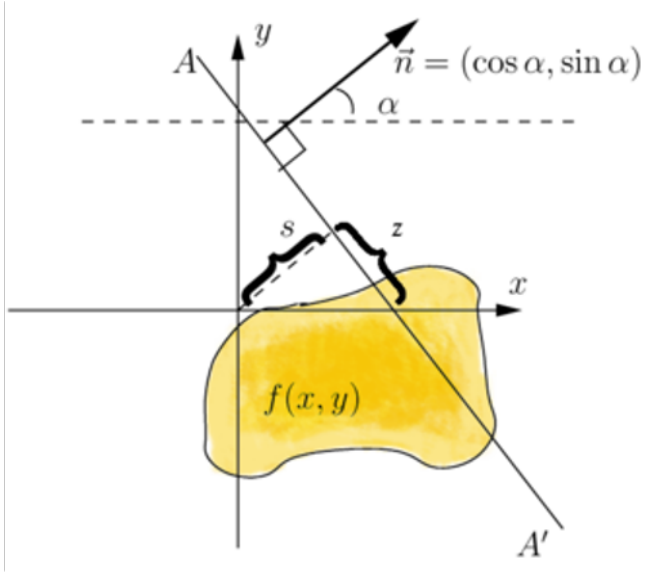
Source: Wikipedia

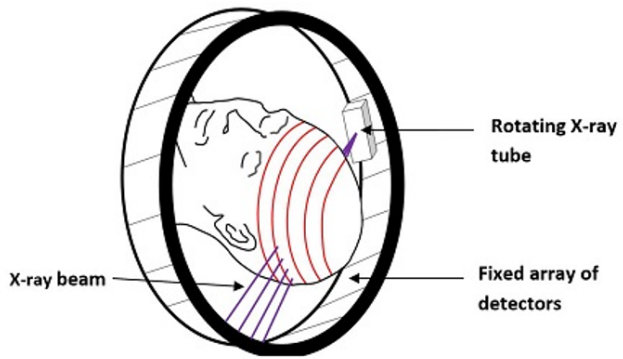
Hough transform parameterization

An alternative parametrization for a linear Hough transform:

$$x \cos\theta + y \sin\theta = \rho$$

Is it possible to solve an inverse problem? Can it be useful? Yes, in computed tomography!
The math behind this problem is the inverse Radon transformation. In some sense, Hough can be considered as an engineering approach to calculate discrete Radon transform.





Source: Wikipedia

Hough transform

- Straight lines that pass that point: $y_i = a \cdot x_i + b$
- Equation can be rewritten as follows: $b = -a \cdot x_i + y_i$
- An alternative parametrization for a linear Hough transform:

$$x \cos\theta + y \sin\theta = \rho$$

- A circular Hough transform:

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

Hough transform

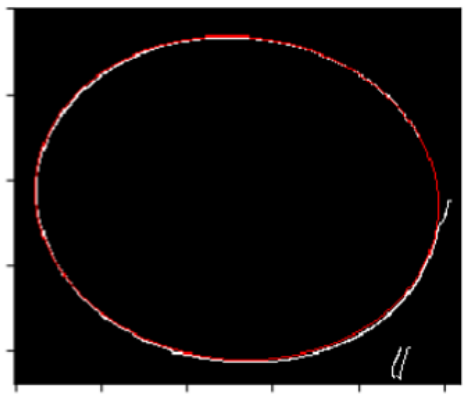
- Straight lines that pass that point: $y_i = a \cdot x_i + b$
- Equation can be rewritten as follows: $b = -a \cdot x_i + y_i$
- An alternative parametrization for a linear Hough transform:

$$x \cos\theta + y \sin\theta = \rho$$

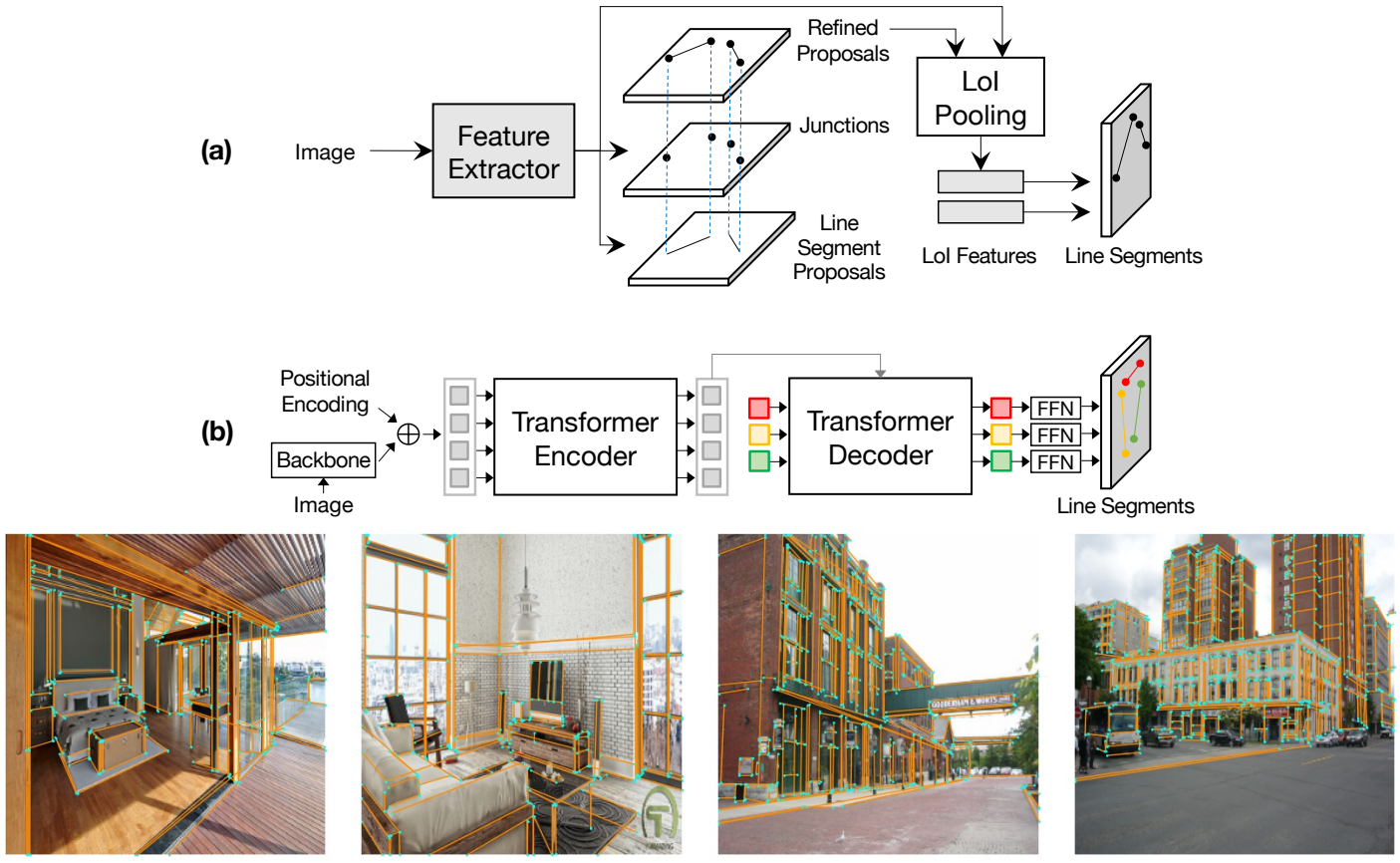
- A circular Hough transform:

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

- It's also possible to look for ellipses using skimage



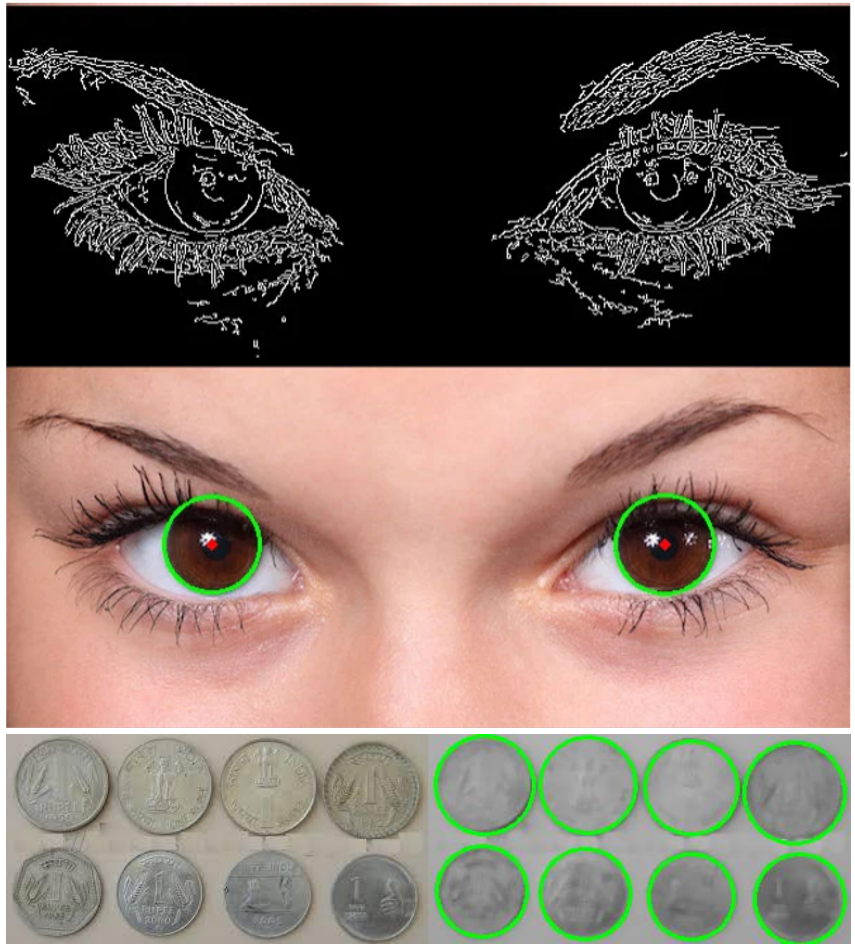
Modern line detection



Xu et al., Line Segment Detection Using Transformers without Edges

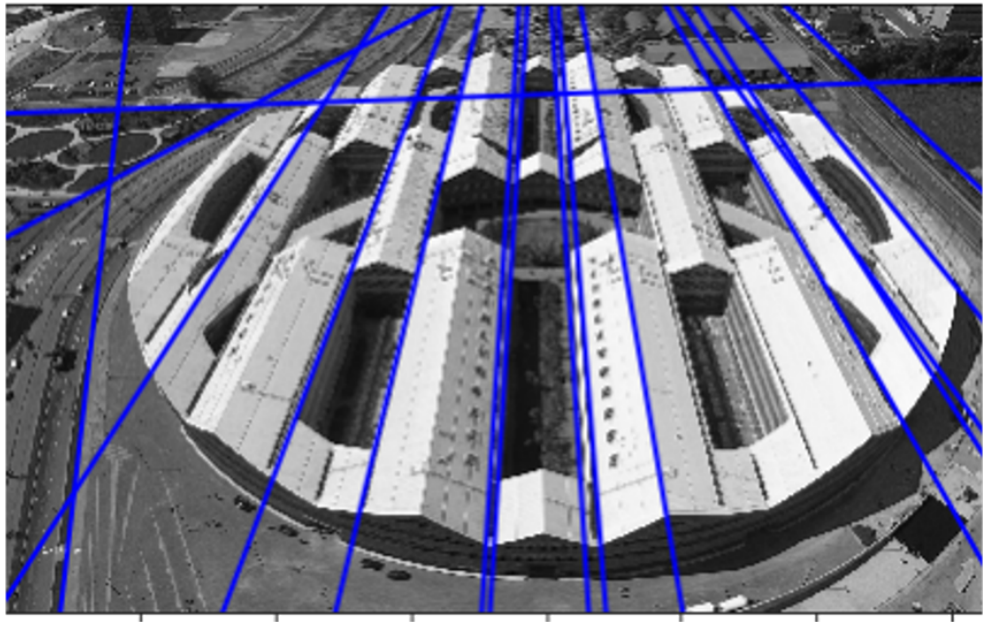
Hough transform

- A circular Hough transform:

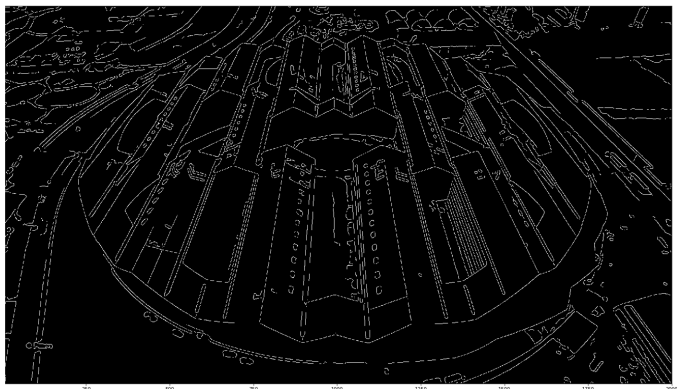


Source: `https://learnopencv.com/hough-transform-with-opencv-c-python/`

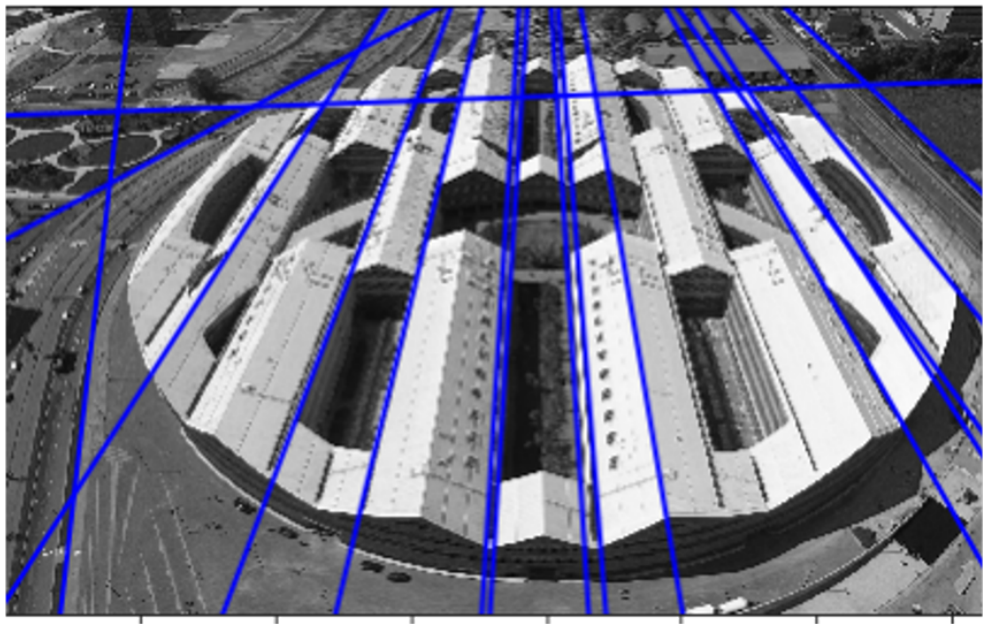
Output of Hough transform



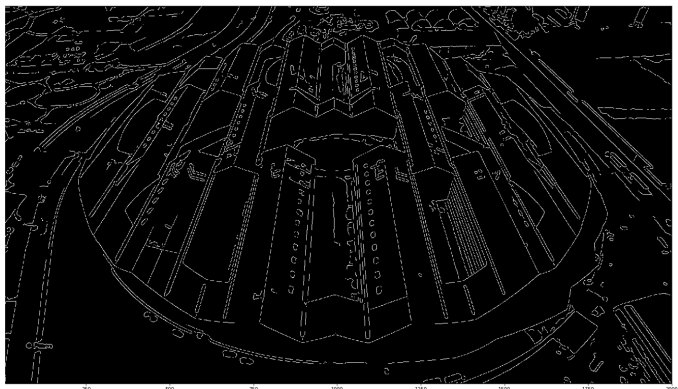
Output of Hough transform



Please name a group of "similar" lines

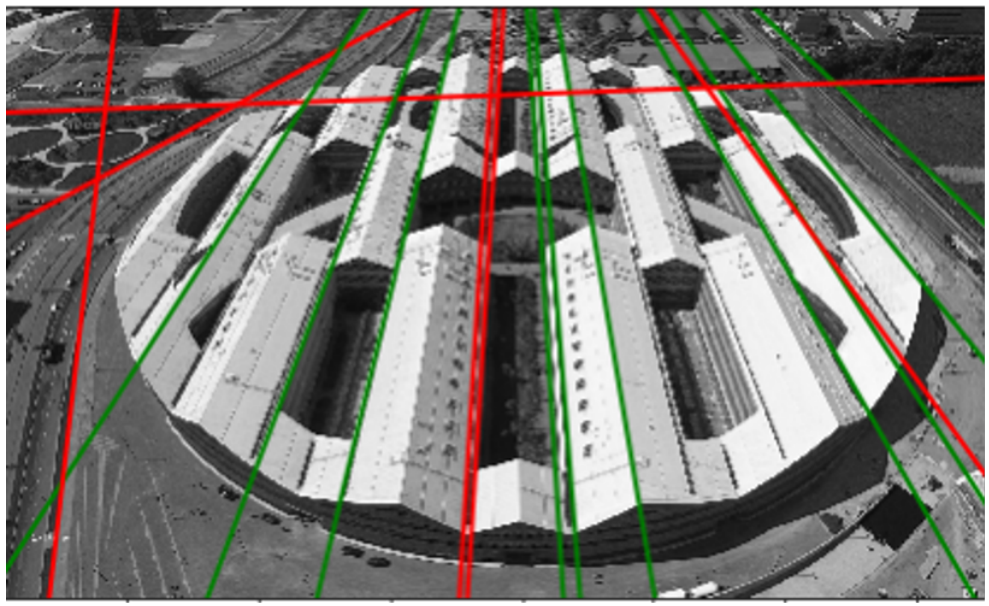


Output of Hough transform

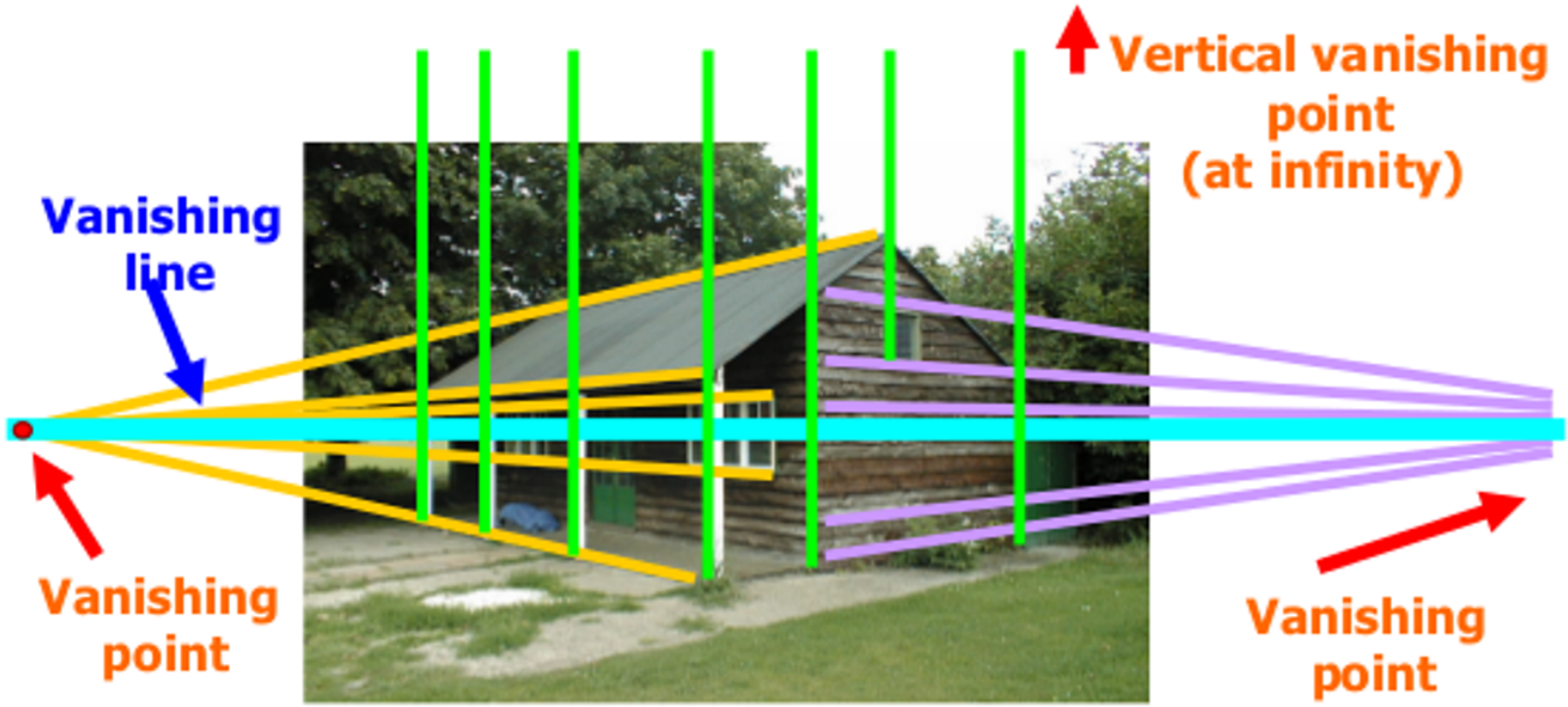


Please name a group of "similar" lines

Green lines are parallel...
Or not?

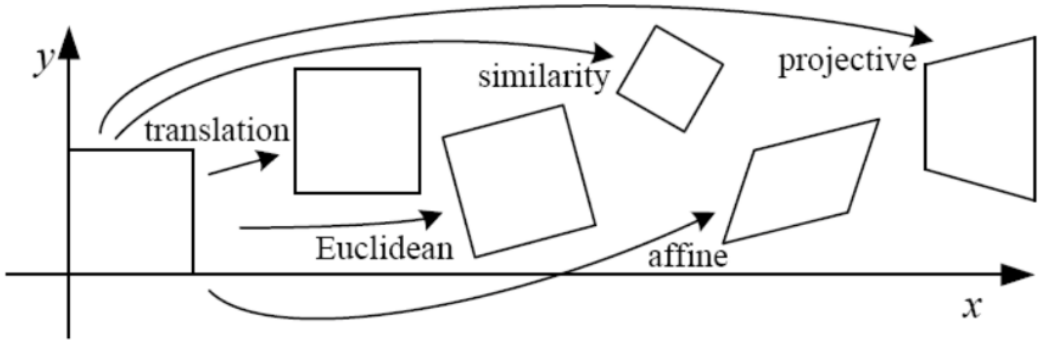


Linear transforms

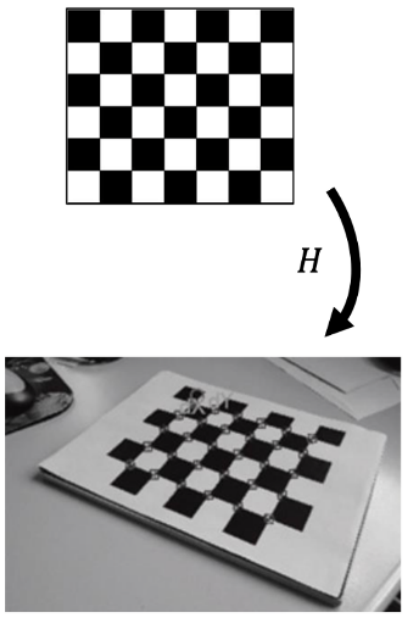


How can we make yellow / purple lines parallel? Can we do it using a linear transform?

Linear transforms



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	



This matrix is called **Homography**

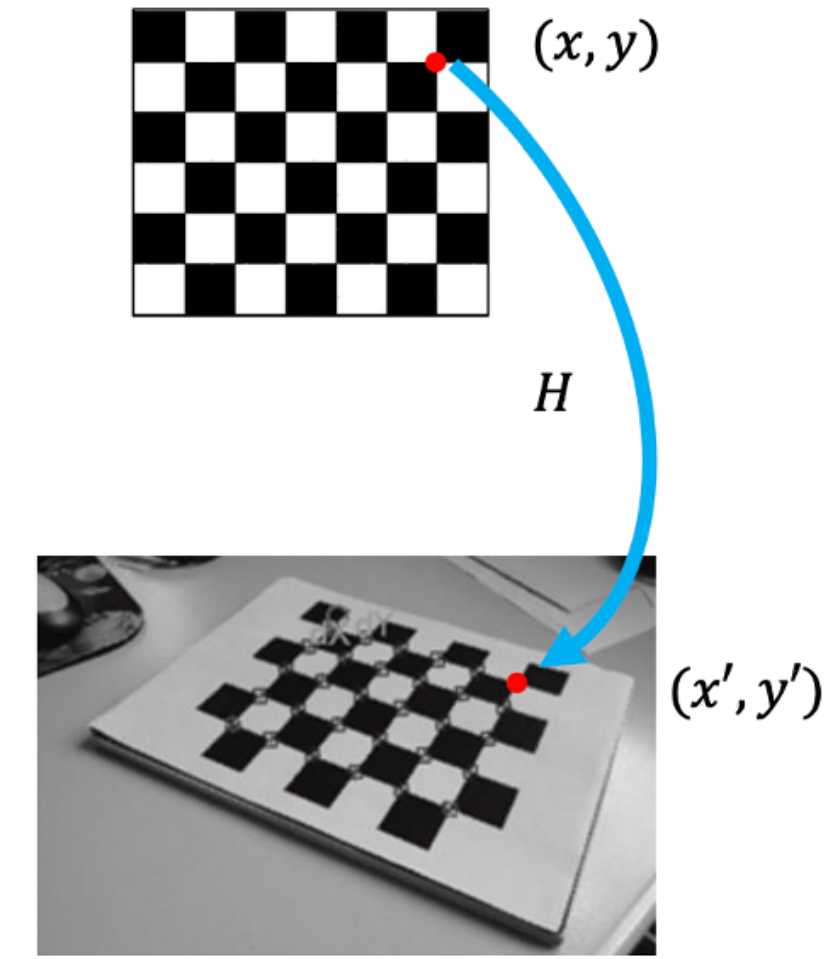
Projective transform

- Point (x, y) is transformed into (x', y') as follows:

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$
$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

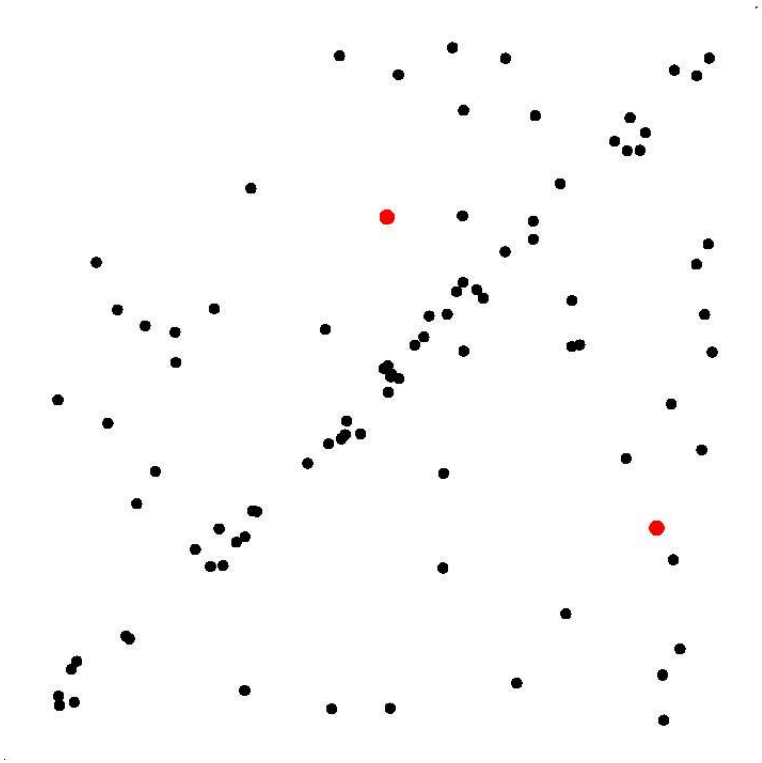
- Coordinates:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



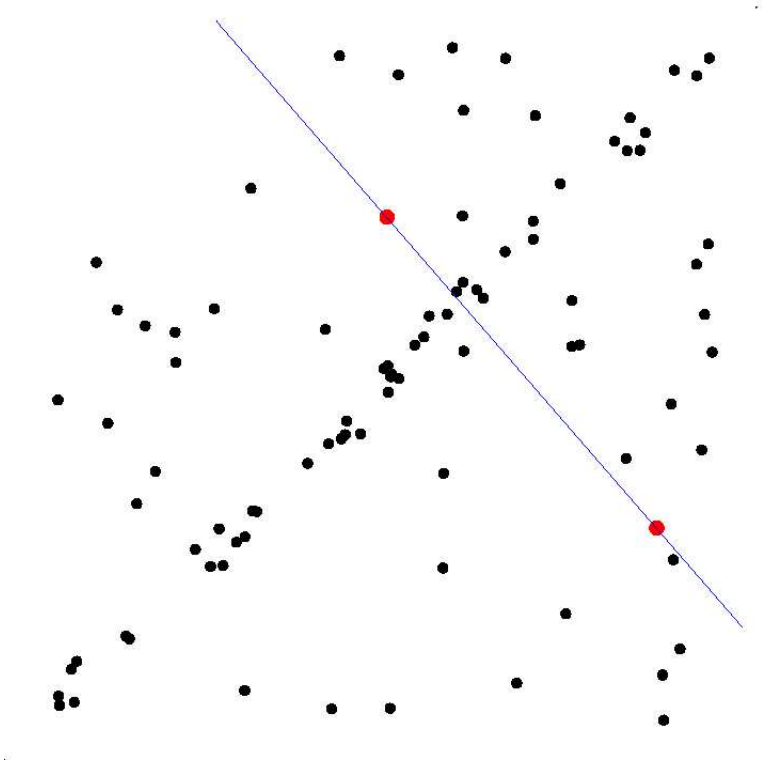
RANSAC

RANSAC



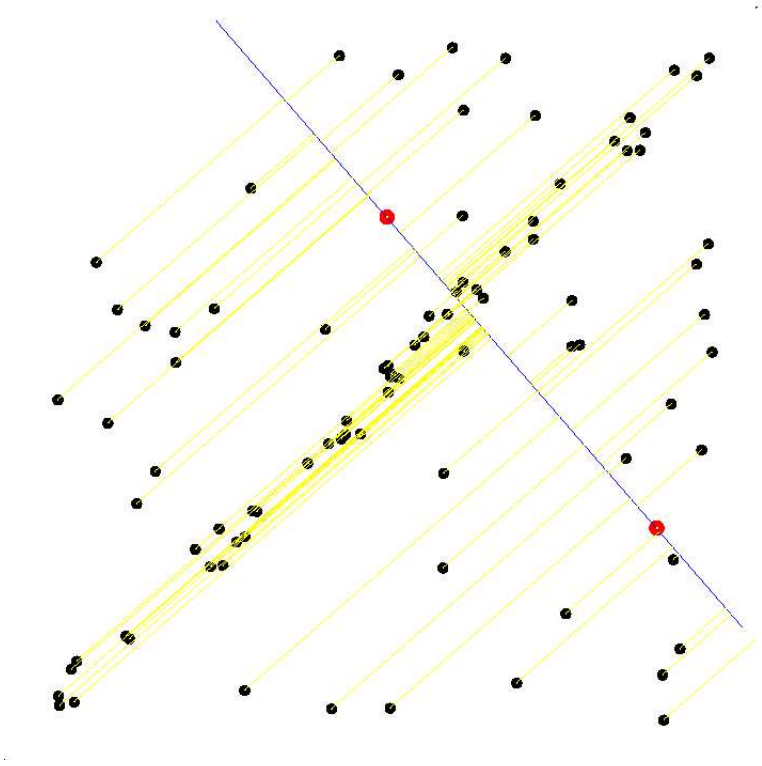
- We make a hypothesis: we choose two random points.

RANSAC



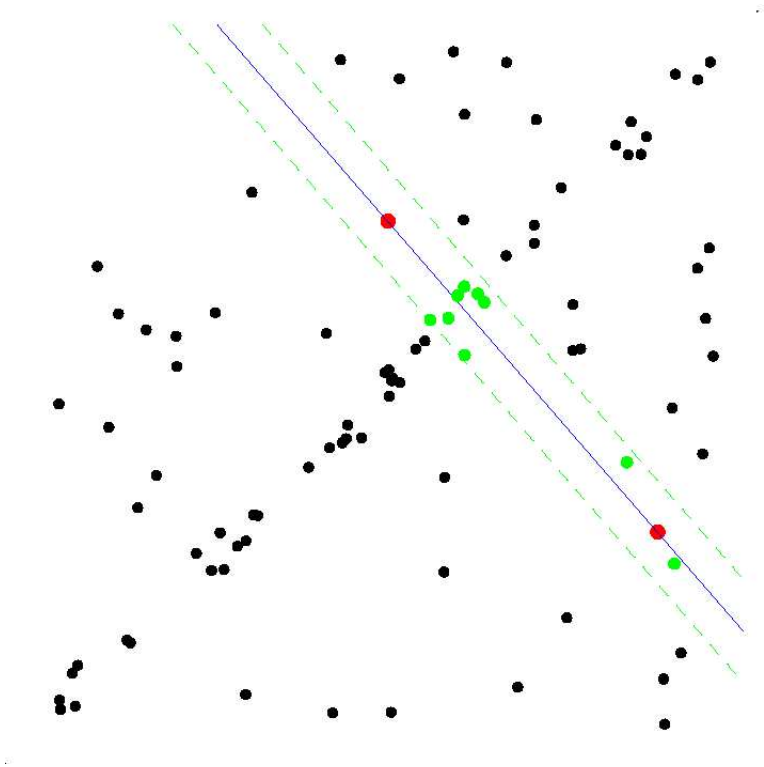
- We make a hypothesis: we choose two random points.
- We select a model that fits the data.

RANSAC



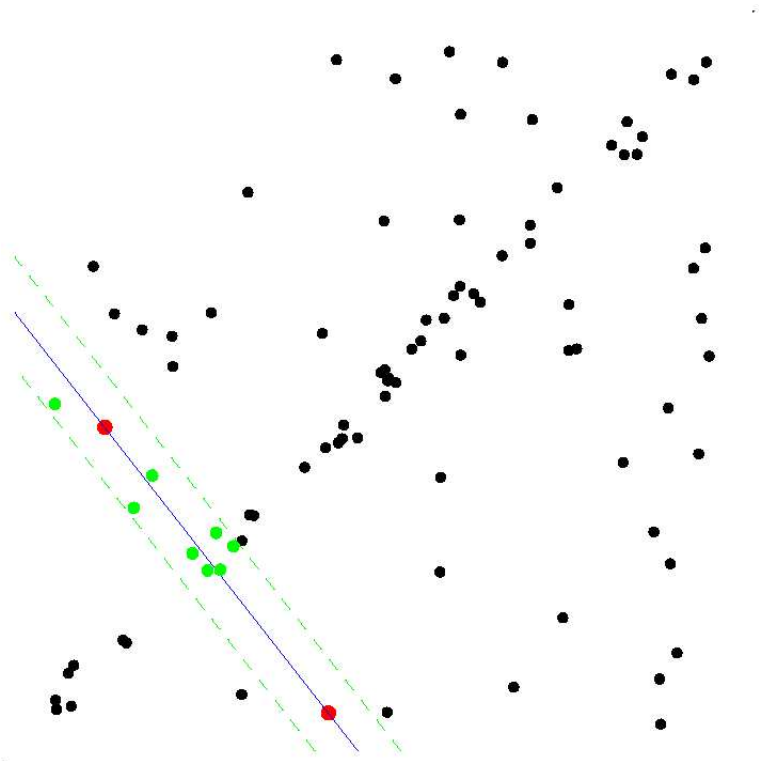
- We make a hypothesis: we choose two random points.
- We select a model that fits the data.
- We calculate the error for each point.

RANSAC



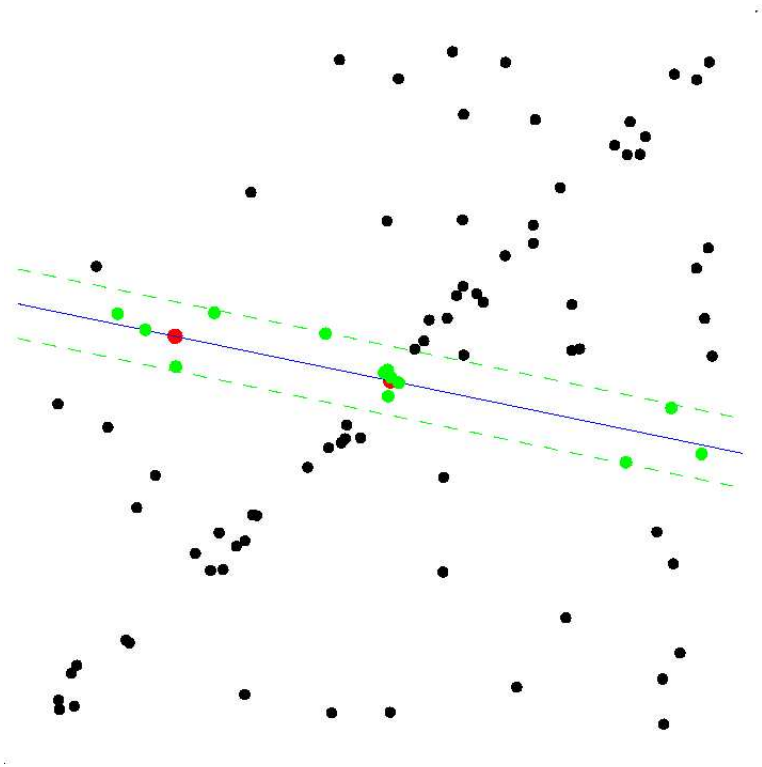
- We make a hypothesis: we choose two random points.
- We select a model that fits the data.
- We calculate the error for each point.
- We choose the points that confirm the current hypothesis.

RANSAC



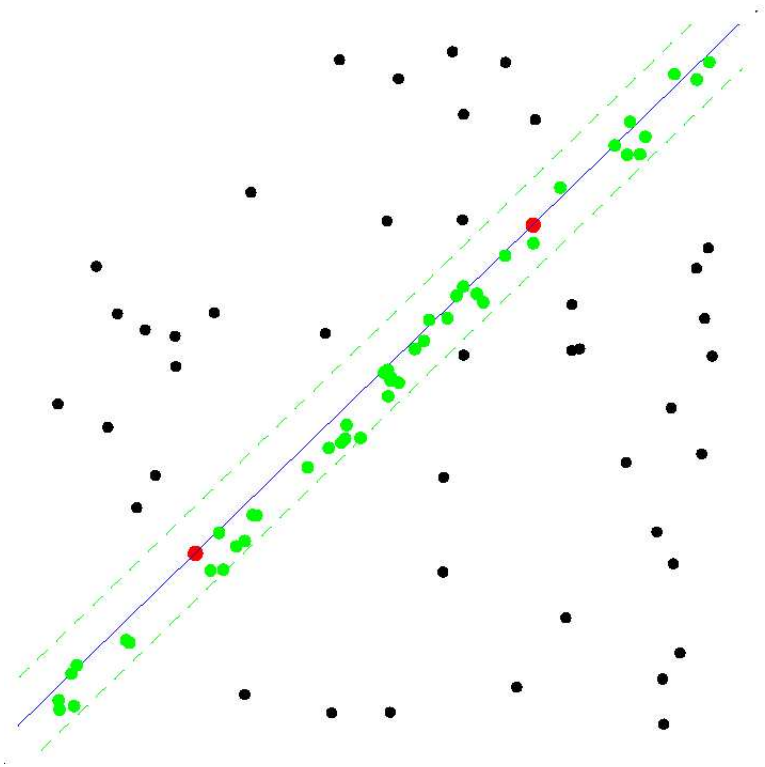
- We make a hypothesis: we choose two random points.
- We select a model that fits the data.
- We calculate the error for each point.
- We choose the points that confirm the current hypothesis.
- We repeat from step 1.

RANSAC



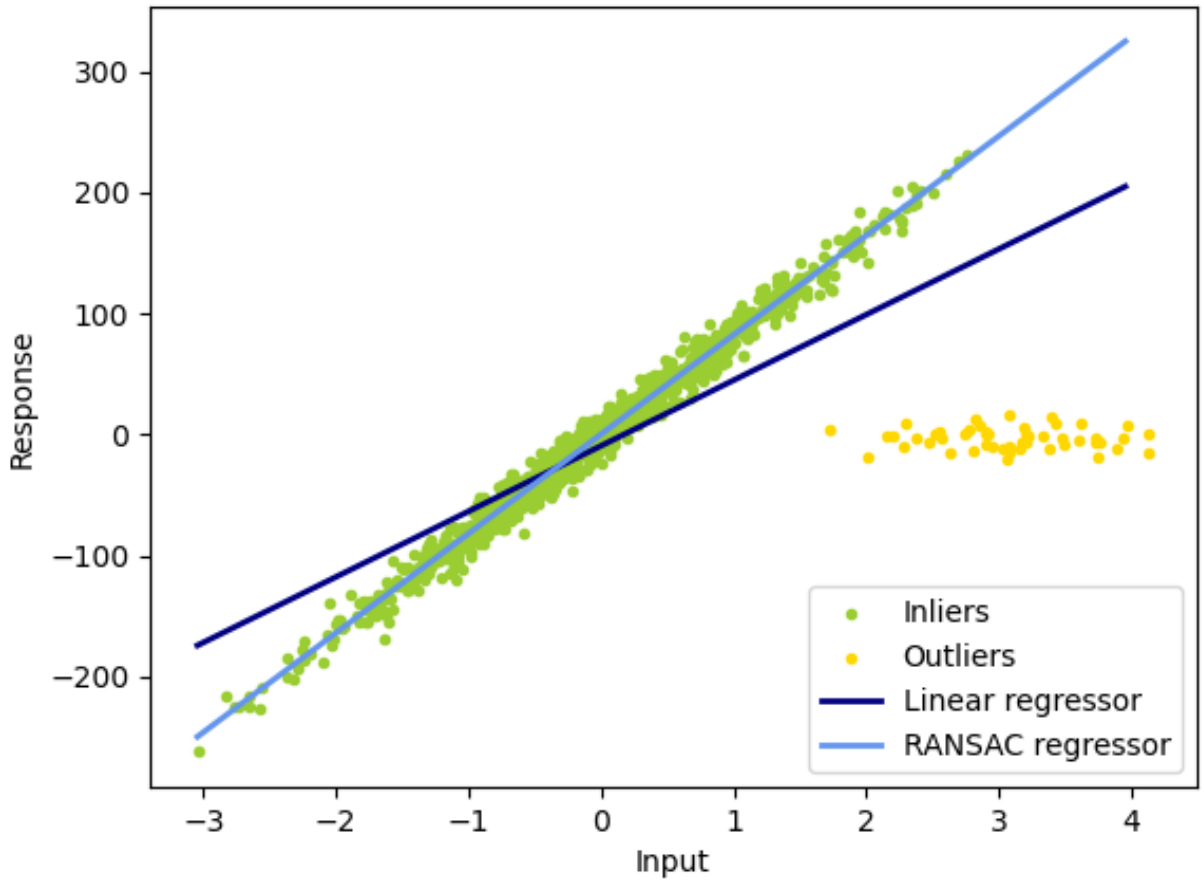
- We make a hypothesis: we choose two random points.
- We select a model that fits the data.
- We calculate the error for each point.
- We choose the points that confirm the current hypothesis.
- We repeat from step 1.
- We select the set of points with the maximum number of inliers.

RANSAC



- We make a hypothesis: we choose two random points.
- We select a model that fits the data.
- We calculate the error for each point.
- We choose the points that confirm the current hypothesis.
- We repeat from step 1.
- We select the set of points with the maximum number of inliers.
- We calculate the model using all the inliers.

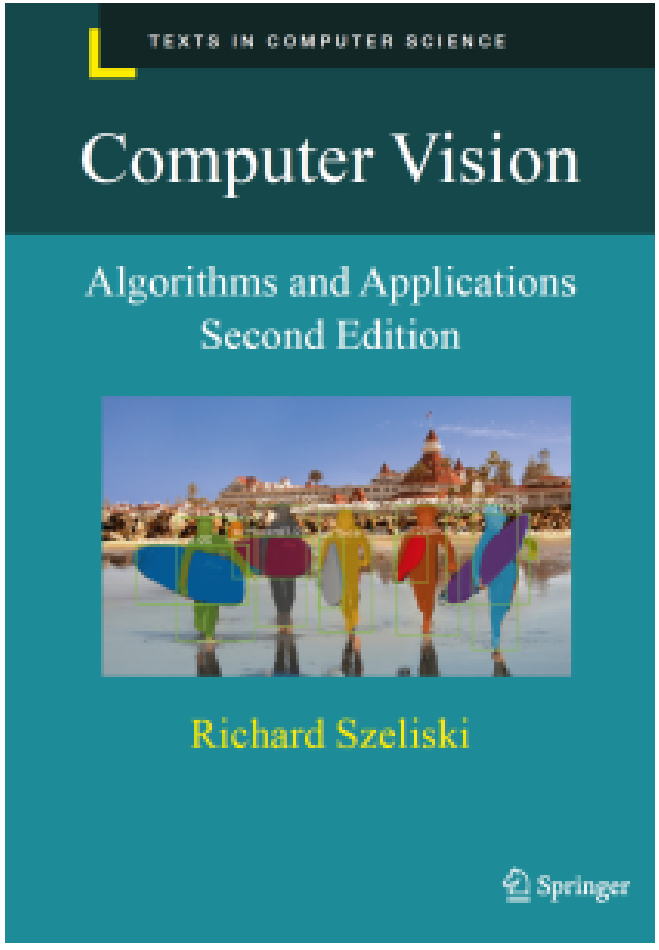
RANSAC



Source: `https://scikit-learn.org/stable/auto_examples/linear_model/plot_ransac.html`

Table of Contents

Reading

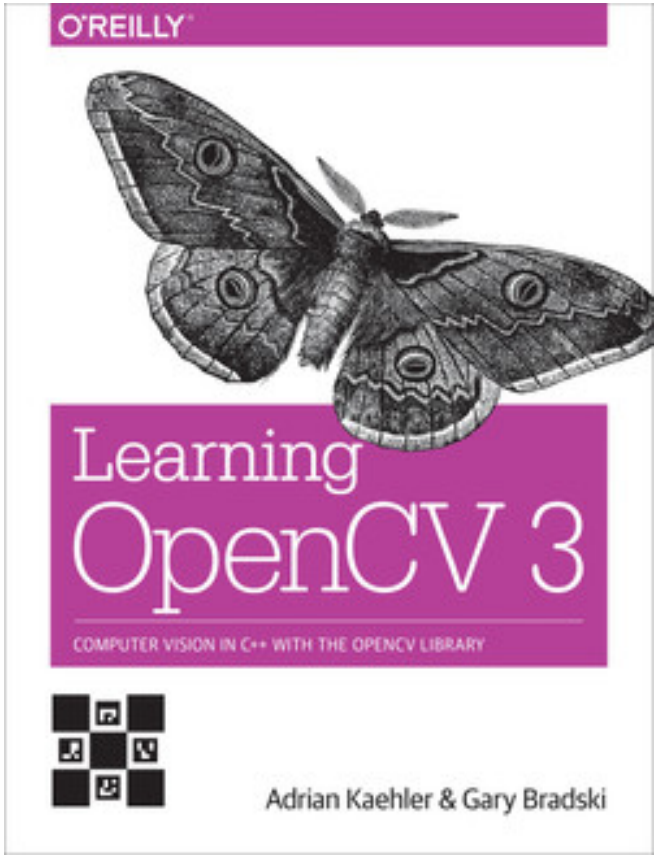


The book is available online

https://szeliski.org/Book

2nd edition

Reading



OpenCV

https://www.oreilly.com

In Russian https://habr.com/ru/companies/intel/articles/507382/

Software

- **OpenCV**: `https://opencv.org/` - a huge collection of various computer vision and 3D reconstruction methods.
- **OpenMVG**: `https://github.com/openMVG/openMVG` - a library for 3D reconstruction.
- **COLMAP**: `https://colmap.github.io/` - a library for 3D reconstruction.

https://grail.cs.washington.edu/rome/

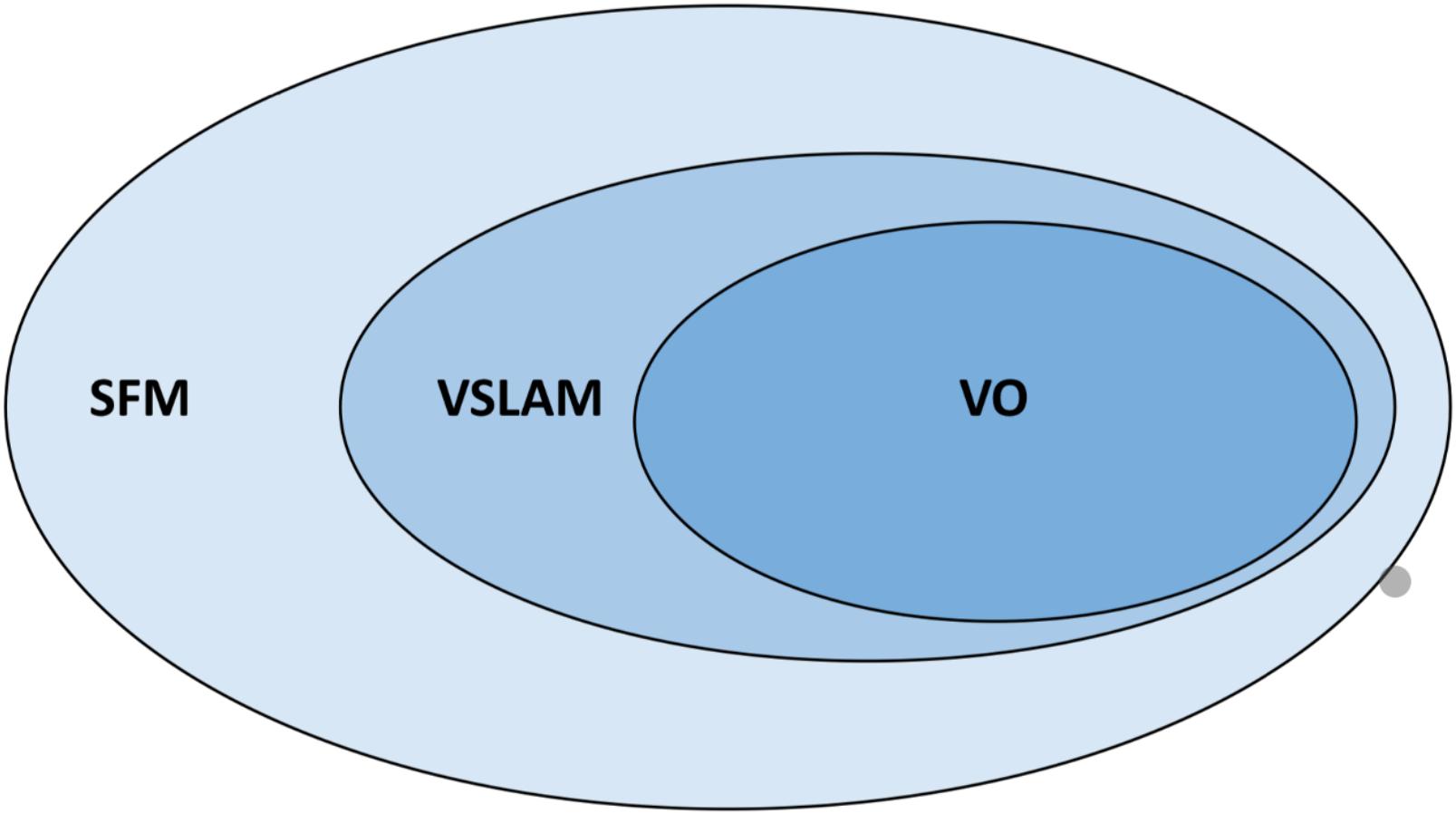
Credits

Materials from the course on Visual Odometry by **Davide Scaramuzza** were used.

- `https://rpg.ifi.uzh.ch/teaching.html`

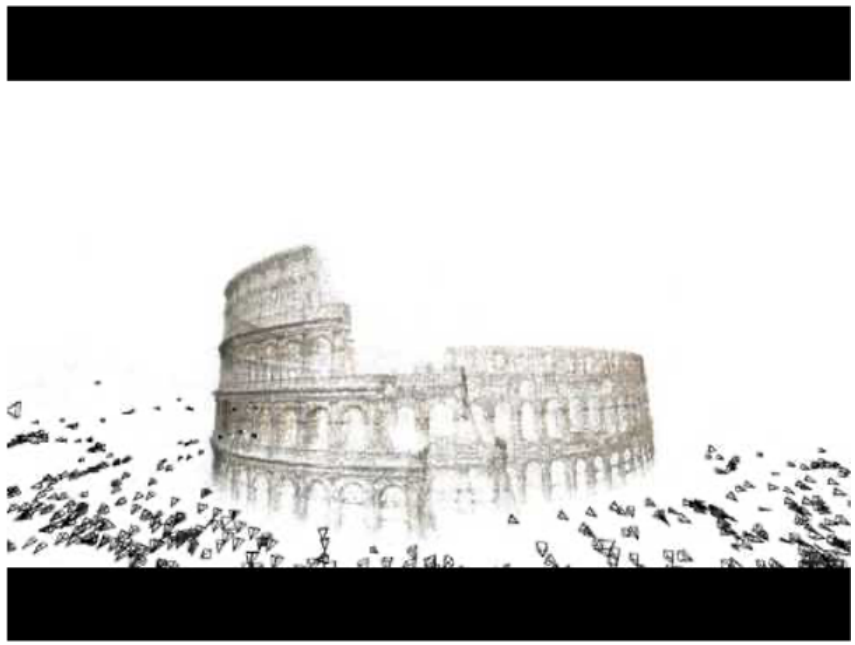
`https://www.youtube.com/watch?v=fIoydrCbBAM`

3D reconstruction terminology



Structure from Motion (SfM)

SfM in the general case works on unstructured sets of images.



Reconstruction from **3 million images** from Flickr.com on a cluster of 250 computers, 24 hours of computation

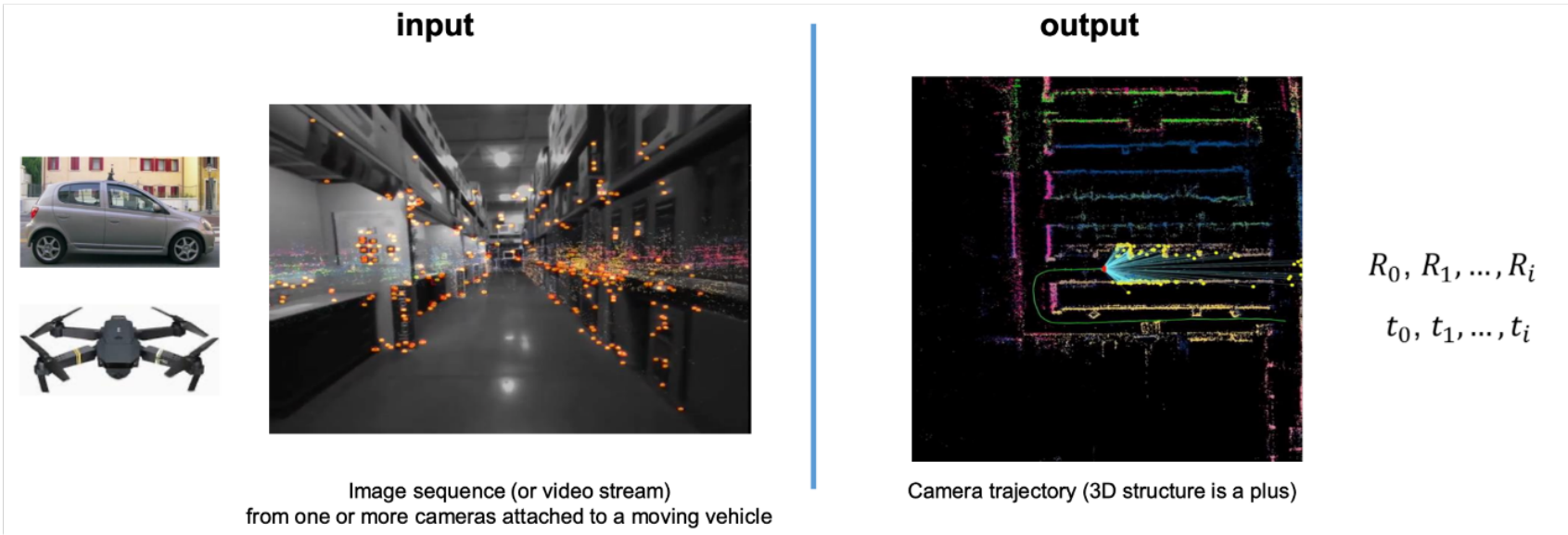
Paper: Building Rome in a Day

State of the art software: COLMAP

https://www.youtube.com/watch?v=kxtQqYLRaSQ

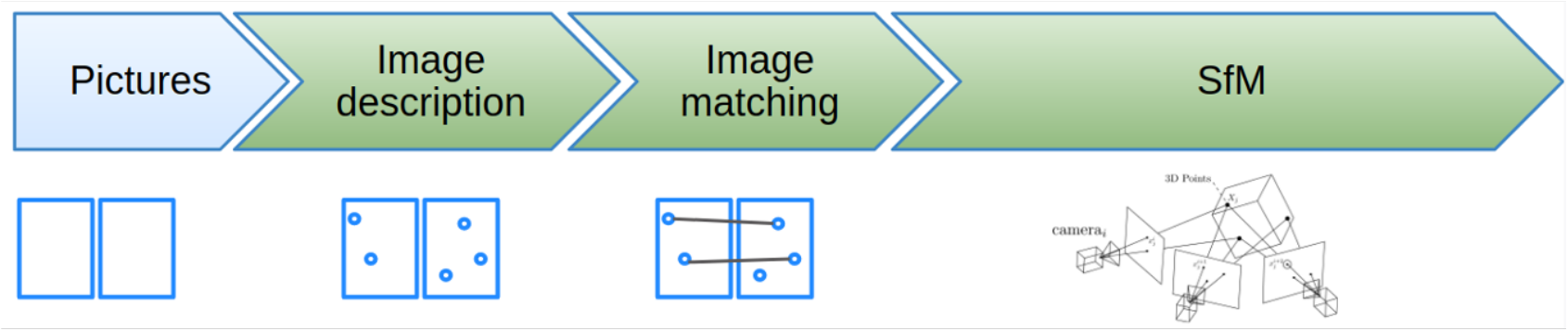
Visual Odometry, VO

Visual odometry - the process of incrementally computing the robot's coordinates by analyzing changes in sensors.



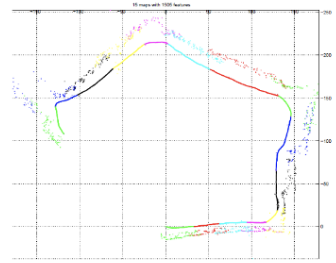
VO Vs SfM

- VO - a particular case of SfM
- VO computes new robot coordinates sequentially as frames come in
- (Sometimes SfM is used as a synonym for VO)

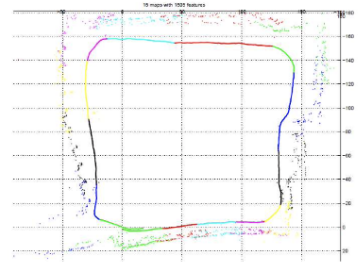


VO Vs Visual SLAM (VSLAM)

- Visual Odometry
  - Per-frame computation of positions
  - Does not guarantee global loop closure, only local consistency
- Visual SLAM:
  - Visual Simultaneous Localization and Mapping
  - SLAM + Visual Odometry + Loop Detection & Closure



Visual odometry



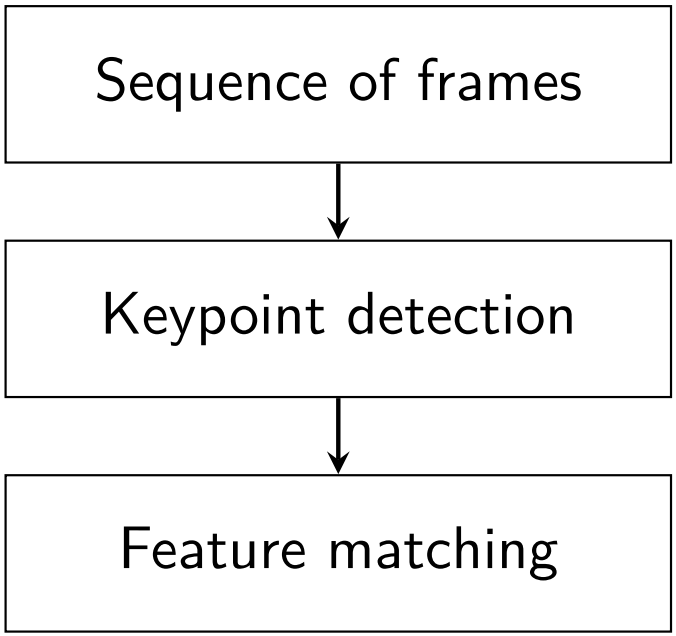
Visual SLAM

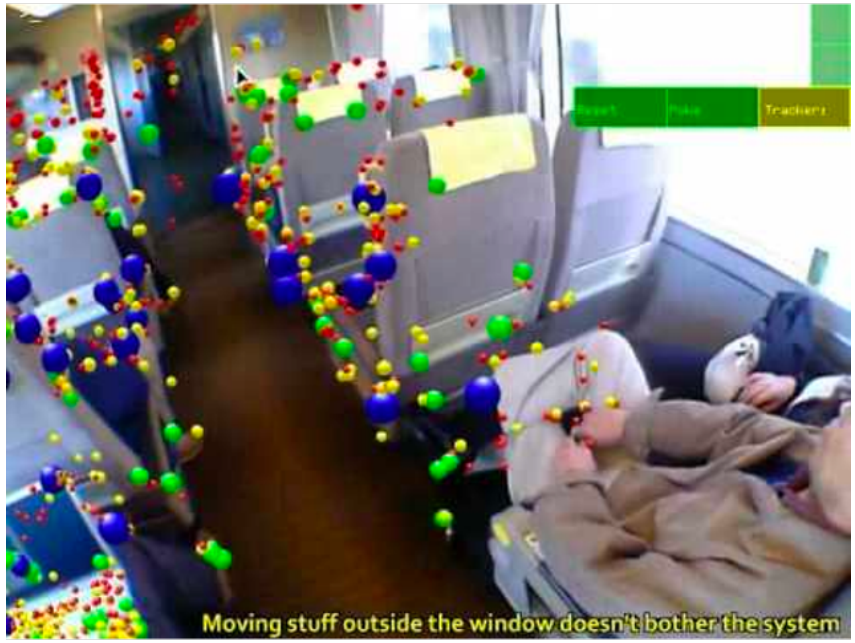
Image courtesy of [Clemente et al., RSS'07]

VO Odometry overview

Sequence of frames

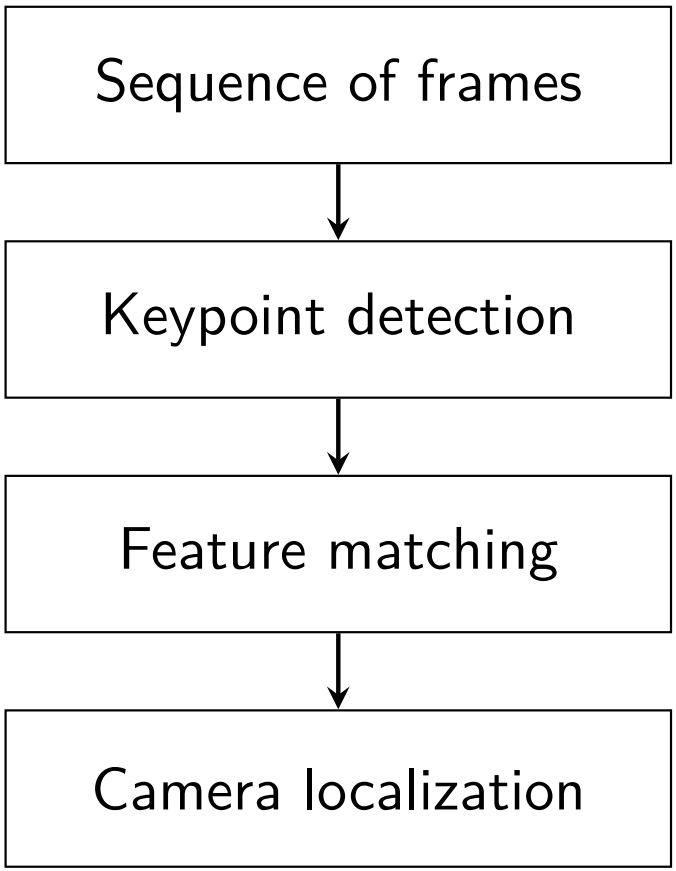
VO Odometry overview



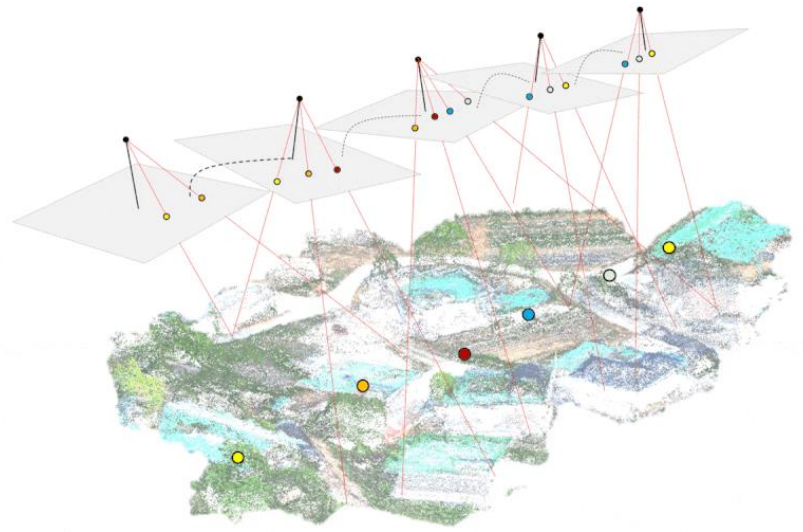


keypoints are tracked across frames

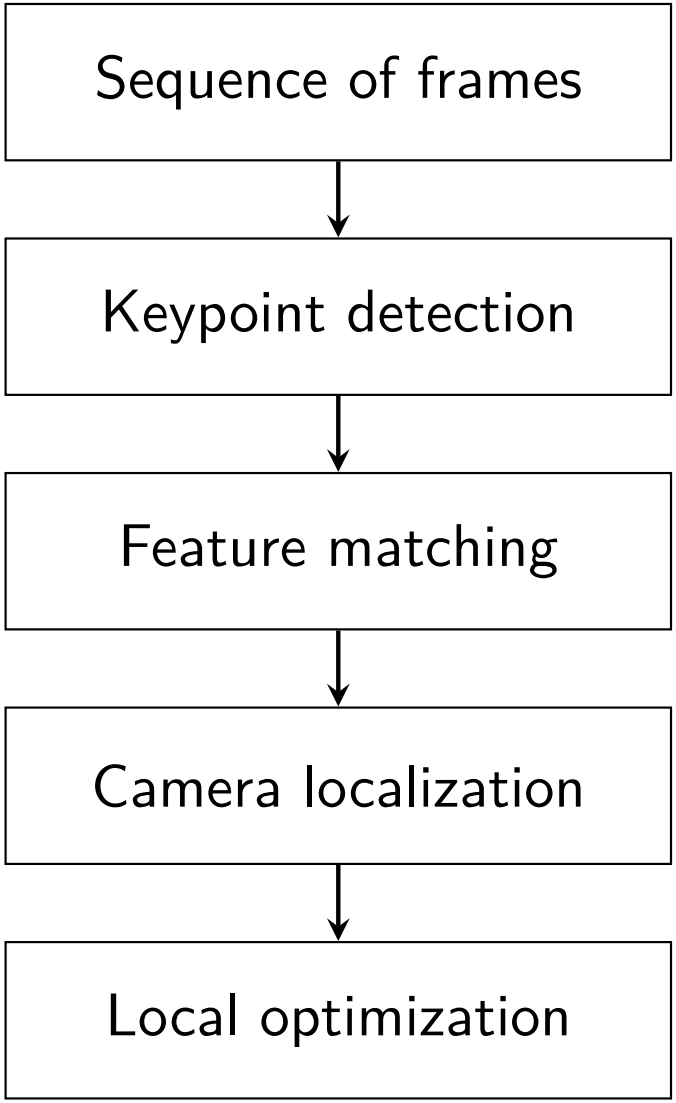
VO Odometry overview



R,T = ?



VO Odometry overview



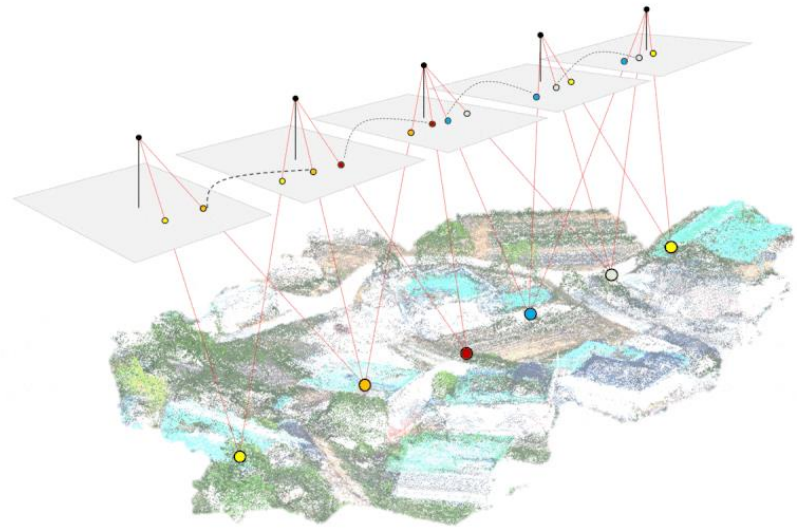


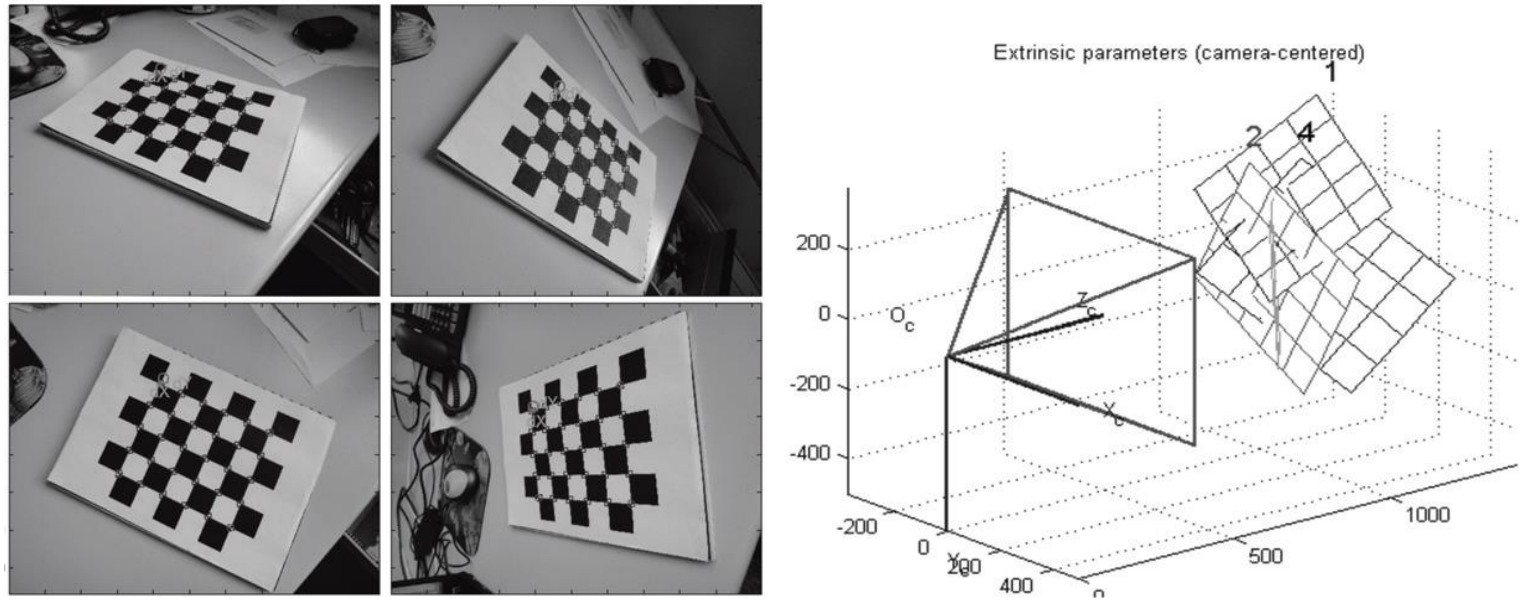
Table of Contents

Why calibrate?

The simplest way to determine the position of a camera in space:

- Intrinsic parameters: the camera's internal parameters **K** (focal length, axis offset, principal point, pixel tilt angle) and distortion parameters
- Extrinsic parameters: the camera's position parameters in space **R, T**

Camera calibration



Zhang, A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000

Direct Linear Transform (DLT)

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|\,T] \cdot \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix} \rightarrow$$

$$\rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_u & 0 & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix}$$

$$\rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_u & 0 & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

Direct Linear Transform (DLT)

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|\,T] \cdot \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix} \rightarrow$$

$$\rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_u & 0 & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix}$$

$$\rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

Direct Linear Transform (DLT)

Translation from the pattern coordinate system to the image coordinate system.

$$u = \frac{\lambda u}{\lambda} = \frac{h_1^T \cdot P}{h_3^T \cdot P} \rightarrow (h_1^T - u_i h_3^T) \cdot P_i = 0$$
$$v = \frac{\lambda v}{\lambda} = \frac{h_2^T \cdot P}{h_3^T \cdot P} \rightarrow (h_2^T - v_i h_3^T) \cdot P_i = 0$$

$$\rightarrow \begin{matrix} P_i^T \cdot h_1 + 0 \cdot h_2^T - u_i P_i^T \cdot h_3^T = 0 \\ 0 \cdot h_1^T + P_i^T \cdot h_2 - v_i P_i^T \cdot h_3^T = 0 \end{matrix} \rightarrow \begin{bmatrix} P_i^T & 0^T & -u_1 P_i^T \\ 0^T & P_i^T & -v_1 P_i^T \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ \dots & \dots & \dots \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ 0 \end{bmatrix} \rightarrow \mathbf{Q} \cdot \mathbf{H} = 0$$

Direct Linear Transform (DLT)

$$\mathbf{Q} \cdot \mathbf{H} = 0$$

Minimum number of points required for existence of a unique solution:

- Matrix Q(2n x 9) must have rank 8
- Each point provides 2 independent equations
- Therefore, at least 4 non-collinear points are required

Solution for n $>=$ 4:

Direct Linear Transform (DLT)

$$\mathbf{Q} \cdot \mathbf{H} = 0$$

Minimum number of points required for existence of a unique solution:

- Matrix Q(2n x 9) must have rank 8
- Each point provides 2 independent equations
- Therefore, at least 4 non-collinear points are required

Solution for n $>=$ 4:

- SVD

Direct Linear Transform (DLT)

Is it possible to obtain K, R, and T from the homography matrix?

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} a_u & 0 & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

Direct Linear Transform (DLT)

Is it possible to obtain K, R, and T from the homography matrix?

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} a_u & 0 & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

- Yes, it is possible! You need a minimum of 2 images.

Direct Linear Transform (DLT)

Is it possible to obtain K, R, and T from the homography matrix?

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} a_u & 0 & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

- Yes, it is possible! You need a minimum of 2 images.
- In practice, 20-50 photographs give the best results.

Direct Linear Transform (DLT)

Is it possible to obtain K, R, and T from the homography matrix?

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} a_u & 0 & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

- Yes, it is possible! You need a minimum of 2 images.
- In practice, 20-50 photographs give the best results.
- Matrices R and T are different for each viewpoint.

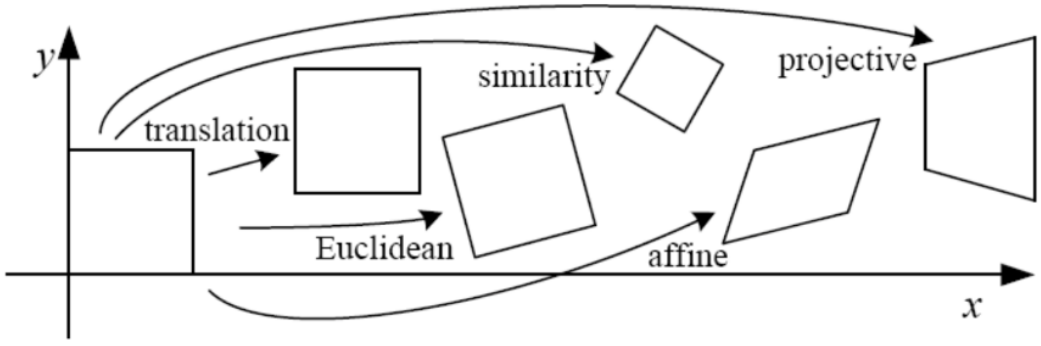
Direct Linear Transform (DLT)

Is it possible to obtain K, R, and T from the homography matrix?

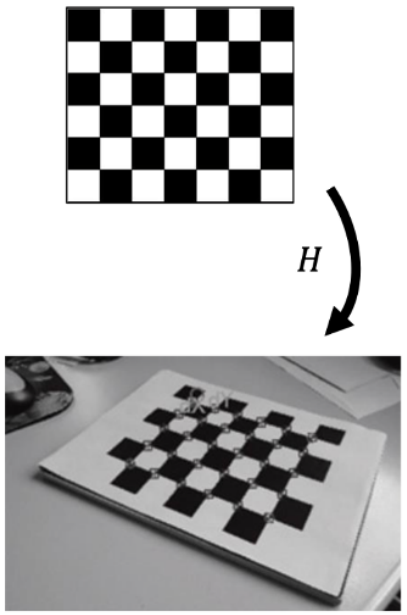
$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} a_u & 0 & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

- Yes, it is possible! You need a minimum of 2 images.
- In practice, 20-50 photographs give the best results.
- Matrices R and T are different for each viewpoint.
- Matrix K remains unchanged.

Linear transforms



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	



This matrix is called **Homography**

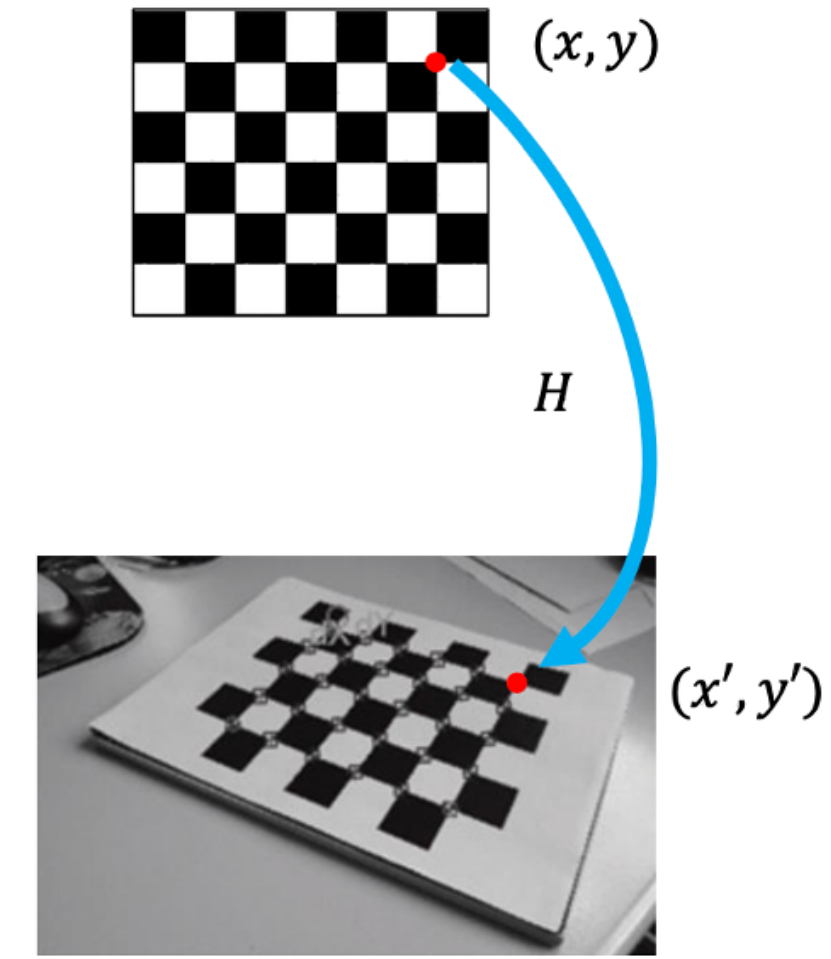
Projective transform

- Point (x, y) is transformed into (x', y') as follows:

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$
$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

- Coordinates:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

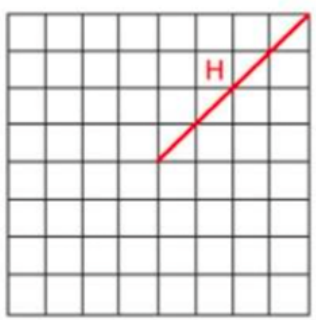


Distortion

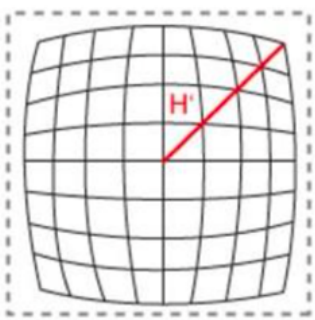
- Radial distortion - caused by the lens
- Tangential distortion - caused by lens displacement relative to the sensor

Radial distortion | Tangential distortion

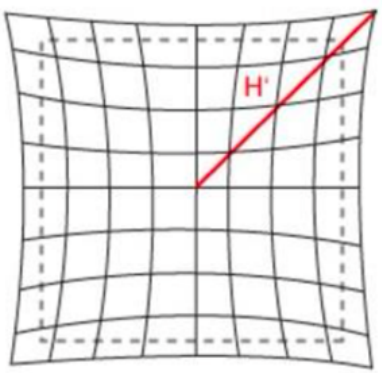
$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} 2k_4(u - u_0)(v - v_0) + k_5(r^2 + 2(u - u_0)^2) \\ k_4(r^2 + 2(v - v_0)^2 + 2k_5(u - u_0)(v - v_0) \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$



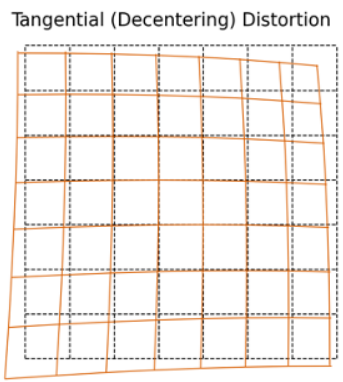
No distortion



Negative radial distortion
(Barrel distortion)
$k_1 < 0$

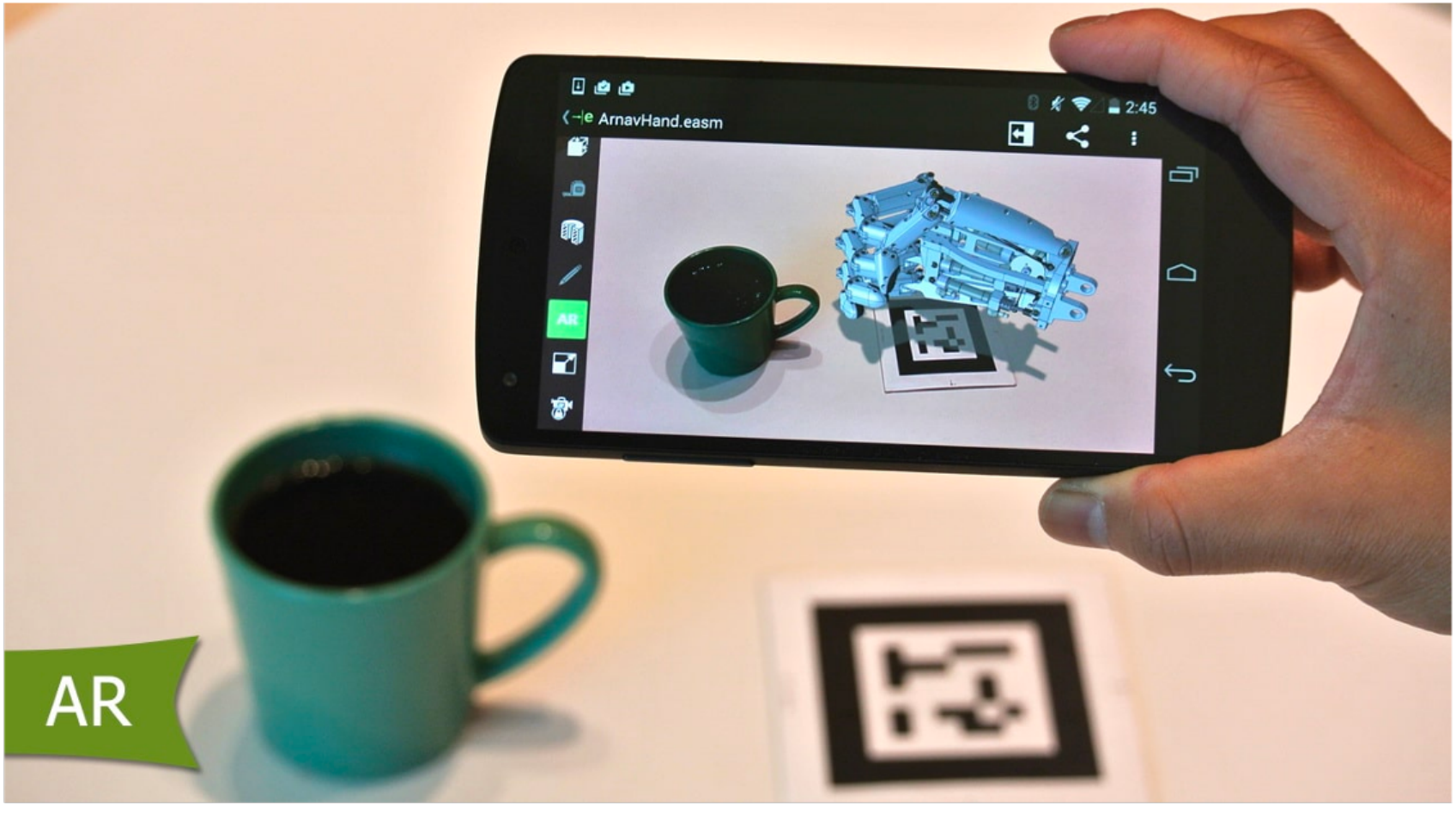


Positive radial distortion
(Pincushion distortion)
$k_1 > 0$



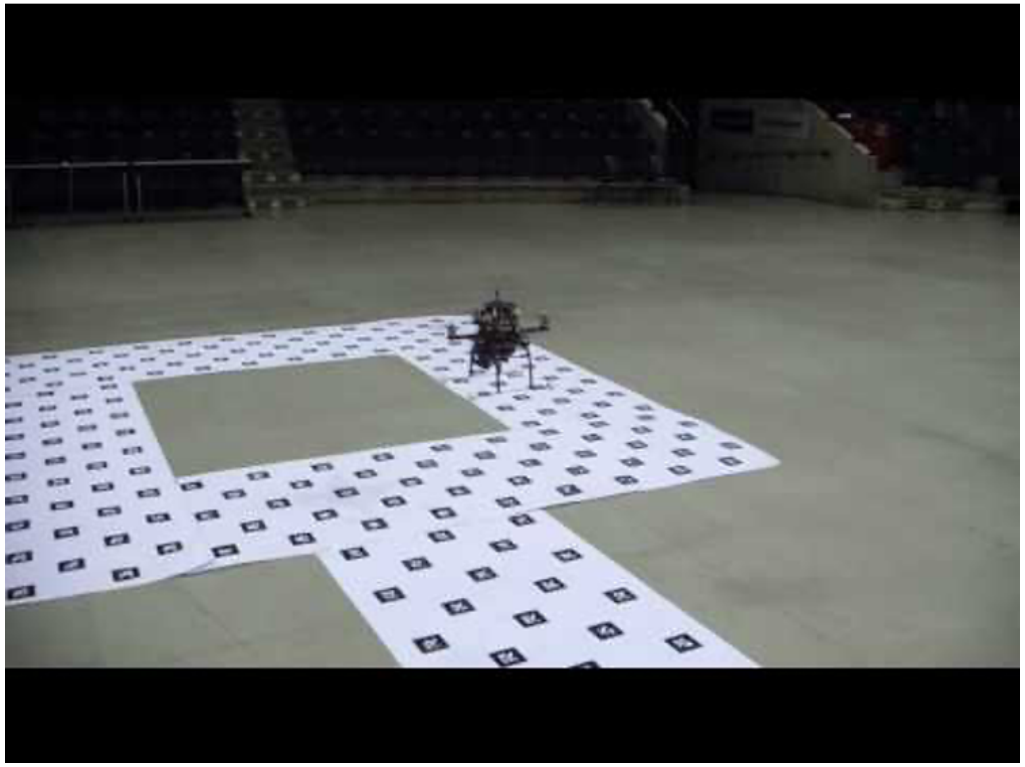
Calibration applications

Augmented Reality (AR) applications require a camera calibration.



Calibration applications

Navigation using AR tags:



https://www.youtube.com/watch?v=83YR15vf718