Robot Perception Pt. II

Octavio Arriaga
octavio.arriaga@dfki.de

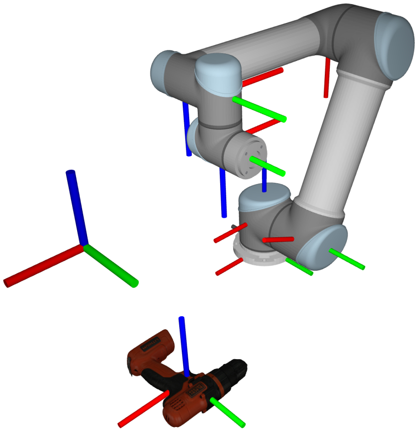
github.com/oarriaga/paz

December 17, 2021

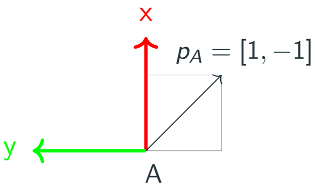
1. Rotation matrices
2. Camera calibration
3. Image processing
4. Object recognition

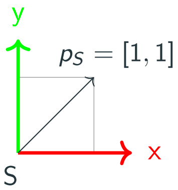
Reference frames

Why do we need reference frames?



Rotation matrices





We want to build a machine (function) that

takes a point in A coordinates and outputs same point in S coordinates.

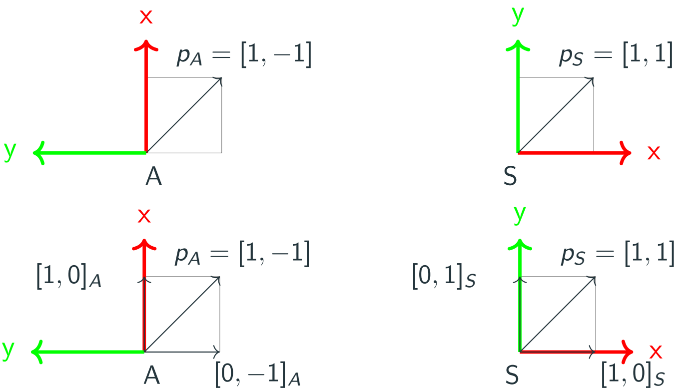
We call this machine:

$$R_{SA}$$

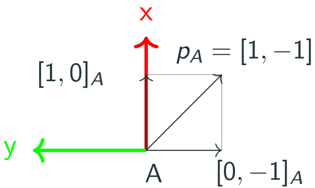
And it operates like this:

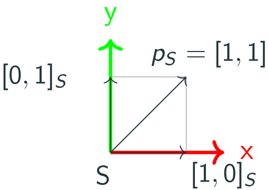
$$p_S = R_{SA}(p_A)$$

Rotation matrices



Rotation matrices





We want to find the function that

$$p_S = R_{SA}(p_A)$$

$$\begin{aligned}
\begin{bmatrix} 0 \\ 1 \end{bmatrix}_S &= \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}_{SA} \begin{bmatrix} 1 \\ 0 \end{bmatrix}_A \\
\begin{bmatrix} 1 \\ 0 \end{bmatrix}_S &= \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}_{SA} \begin{bmatrix} 0 \\ -1 \end{bmatrix}_A
\end{aligned} \tag{1}$$

Rotation matrices

Let's do the first point first

$$\begin{bmatrix}0\\1\end{bmatrix}_S = \begin{bmatrix}r_{11} & r_{12}\\ r_{21} & r_{22}\end{bmatrix}_{SA} \begin{bmatrix}1\\0\end{bmatrix}_A$$

$$\begin{bmatrix}0\\1\end{bmatrix}_S = 1\begin{bmatrix}r_{11}\\ r_{21}\end{bmatrix} + 0\begin{bmatrix}r_{12}\\ r_{22}\end{bmatrix}$$

$$\begin{bmatrix}0\\1\end{bmatrix}_S = \begin{bmatrix}r_{11}\\ r_{21}\end{bmatrix}$$

Thus

$$r_{11} = 0$$

$$r_{21} = 1$$

Rotation matrices

Let's do now the second point

$$\begin{bmatrix}1\\0\end{bmatrix}_S = \begin{bmatrix}r_{11} & r_{12}\\ r_{21} & r_{22}\end{bmatrix}_{SA} \begin{bmatrix}0\\-1\end{bmatrix}_A$$

$$\begin{bmatrix}1\\0\end{bmatrix}_S = 0\begin{bmatrix}r_{11}\\ r_{21}\end{bmatrix} - 1\begin{bmatrix}r_{12}\\ r_{22}\end{bmatrix}$$

$$\begin{bmatrix}1\\0\end{bmatrix}_S = -1\begin{bmatrix}r_{12}\\ r_{22}\end{bmatrix}$$

Thus

$$r_{12} = -1$$

$$r_{22} = 0$$

Rotation matrices

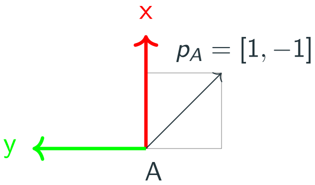
Putting all $r_{11} = 0$, $r_{21} = 1$, $r_{12} = -1$, $r_{22} = 0$ we have:

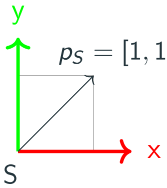
$$R_{SA} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Running now our (unit) test

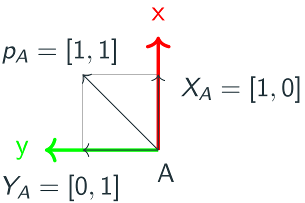
$$\begin{bmatrix} x_S \\ y_S \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}_{SA} \begin{bmatrix} 1 \\ -1 \end{bmatrix}_A$$

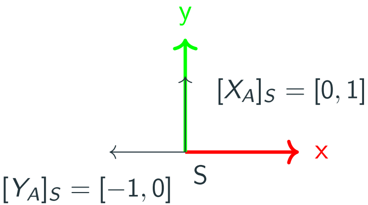
By matrix multiplication $x_S = 1$ and $y_S = 1$. Which is $p_S = [1, 1]$





Rotation matrices





From what we learned we can build matrices quicker

$$\begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}_{SA} \begin{bmatrix} 1 \\ 0 \end{bmatrix}_A = \begin{bmatrix} r_{11} \\ r_{21} \end{bmatrix} := [X_A]_S \qquad \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}_{SA} \begin{bmatrix} 0 \\ 1 \end{bmatrix}_A = \begin{bmatrix} r_{12} \\ r_{22} \end{bmatrix} := [Y_A]_S$$

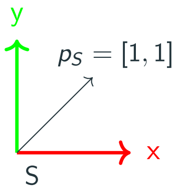
$$R_{SA} = \begin{bmatrix} \uparrow & \uparrow \\ [X_A]_S & [Y_A]_S \\ \downarrow & \downarrow \end{bmatrix}$$

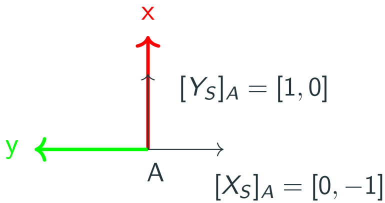
$$R_{SA} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Send unit impulses in A and seeing how they project in S

Rotation matrices

We have built R_{SA}. Now let's build R_{AS}





$$R_{AS} = \begin{bmatrix} \uparrow & \uparrow \\ [X_S]_A & [Y_S]_A \\ \downarrow & \downarrow \end{bmatrix}$$

$$R_{AS} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Rotation matrices

Let's look at both matrices

$$R_{SA} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$R_{AS} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

We observe that

$$R_{SA} = R_{AS}^T$$

Or the same as

$$R_{AS} = R_{SA}^T$$

Moreover we see that

$$R_{SA}R_{AS} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} := I$$

Why?

Rotation matrices

We have that

$$R_{SA} = \begin{bmatrix} \uparrow & \uparrow \\ [X_A]_S & [Y_A]_S \\ \downarrow & \downarrow \end{bmatrix}$$

Thus

$$R_{AS}R_{SA} = R_{SA}^T R_{SA} = \begin{bmatrix} \leftarrow [X_A]_S \rightarrow \\ \leftarrow [Y_A]_S \rightarrow \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow \\ [X_A]_S & [Y_A]_S \\ \downarrow & \downarrow \end{bmatrix}$$

Which is

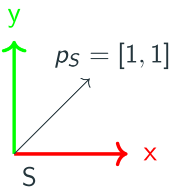
$$R_{SA}^T R_{SA} = \begin{bmatrix} [X_A]_S \cdot [X_A]_S & [X_A]_S \cdot [Y_A]_S \\ [Y_A]_S \cdot [Y_A]_S & [Y_A]_S \cdot [Y_A]_S \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

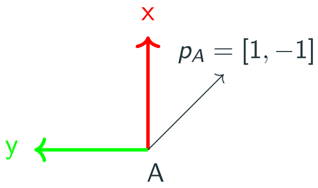
Where $\cdot$ is the dot product and we use these (easy to prove) facts:

- The dot product of a normalized vector with itself is 1.
- The dot product of two orthogonal vectors is 0.

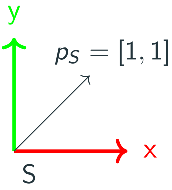
Passive vs. Active

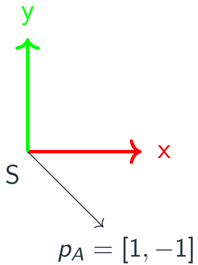
Passive: Space remains the same but the frame of reference changes.





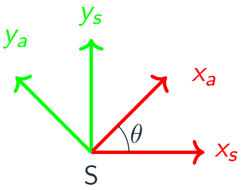
Active: Frame of reference remains the same but space changes.





Rotation matrices

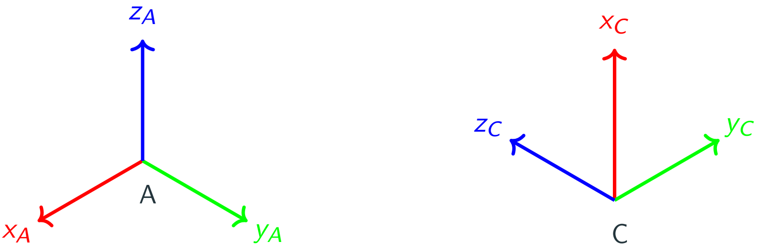
Generic rotation matrix for 2D



$$R_{SA} = \begin{bmatrix} \uparrow & \uparrow \\ [X_A]_S & [Y_A]_S \\ \downarrow & \downarrow \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Rotation matrices

Extension to 3D



Let's build R_{AC}:

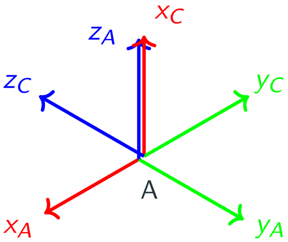
$$R_{AC} = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ [X_C]_A & [Y_C]_A & [Z_C]_A \\ \downarrow & \downarrow & \downarrow \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}$$

Making our test

$$\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Extension to translations

We have only rotated frames, and computed coordinates in rotated frames.

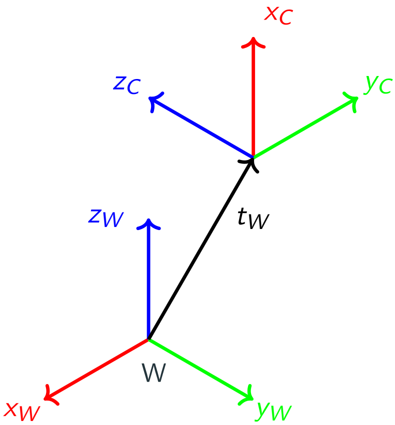


But we can also translate frames.

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & tx \\ r_{21} & r_{22} & r_{23} & ty \\ r_{31} & r_{31} & r_{33} & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}_{CA}$$

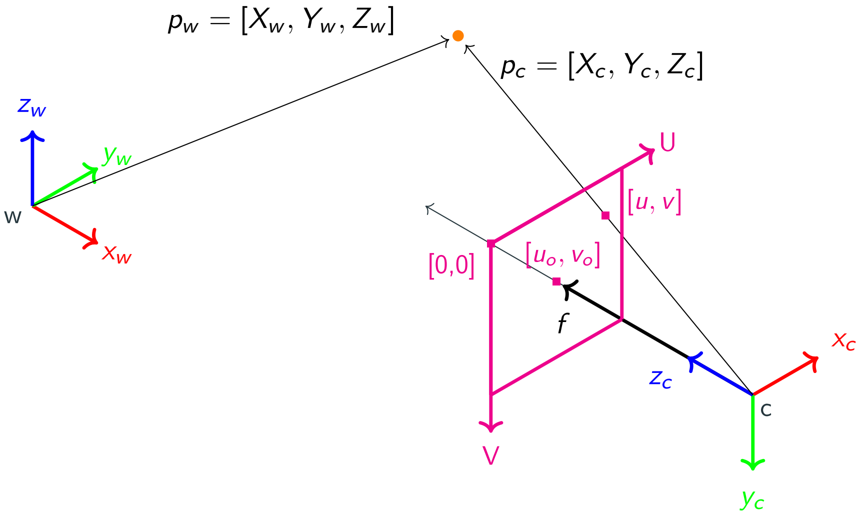
Extension to translations

What is $t = [t_x, t_y, t_z]$?



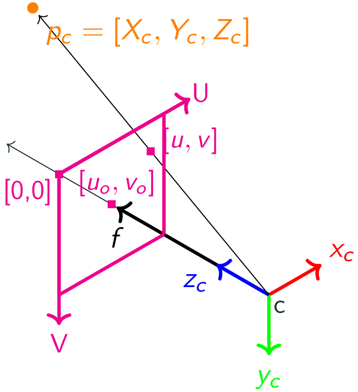
$$\begin{bmatrix} R_{CW} & R_{CW}(-t_W) \\ \mathbf{0}_{3\times 1} & 1 \end{bmatrix}_{CW}$$

Camera calibration



Camera calibration

From our previous lecture we know:



The pinhole camera model projects p_c to the image plane using:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = K \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

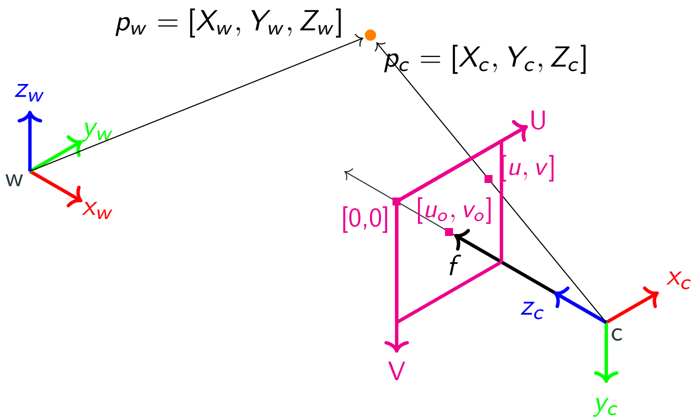
Camera calibration

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = K \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

To use homogenous points ($[X_c, Y_c, Z_c, 1]$) we extend K with a 0's column

$$[K|0] = \begin{bmatrix} k_u f & 0 & u_0 & 0 \\ 0 & k_v f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = [K|0] = \begin{bmatrix} k_u f & 0 & u_0 & 0 \\ 0 & k_v f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$



From what we know now of affine transformations we can easily compute

$$A_{cw} = \begin{bmatrix} R_{cw} & T_{cw} \\ \mathbf{0} & 1 \end{bmatrix} \quad (2)$$

Putting all together we have intrinsics and extrinsics camera parameters

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u f & 0 & u_0 & 0 \\ 0 & k_v f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & tx \\ r_{21} & r_{22} & r_{23} & ty \\ r_{31} & r_{31} & r_{33} & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}_{CW} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = [K_{\text{intrinsic}}|0] M_{\text{extrinsic}}$$

How do we obtain the internal and external parameters?

The main steps are:

- Obtain pairs of $[X_w, Y_w, Z_w]$ and $[u, v]$.
- Minimize the error between pairs.

There are multiple algorithms and openCV has an out-of-the-box solution.

How does it look in openCV?

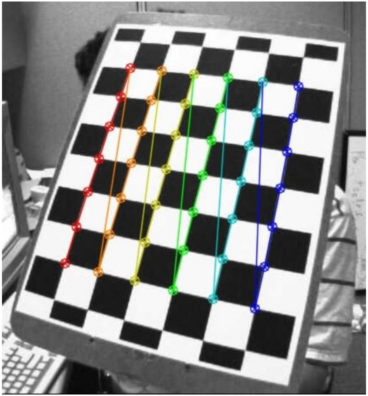


Figure: Calibration pattern

Correspondence between a known object dimensions in 3D space and easy to located coordinates in 2D image space.

Applications of camera calibration:

- Distance measurement
- Pose estimation
- Drawing
- Image rectification

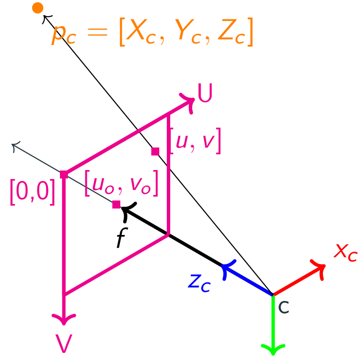
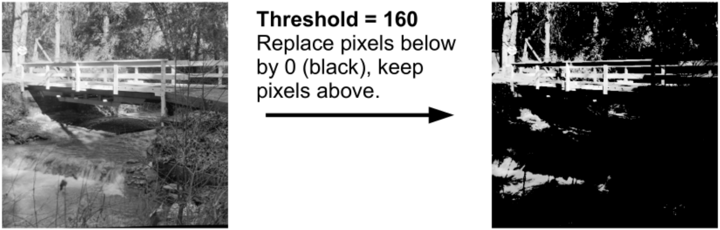


Image processing

From our previous lecture:

- Higher-level image processing might require binary images
    - We can apply a threshold operation
- Per-color threshold operations can help us create classifiers.
    - "How much red does our image has?"



We can also apply convolution operations to images

Convolutions apply the same operation in image patches

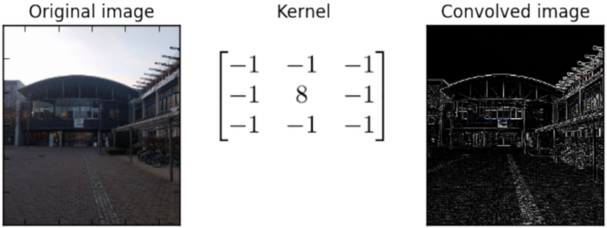


Figure: Convolution operation for image processing.

Image processing

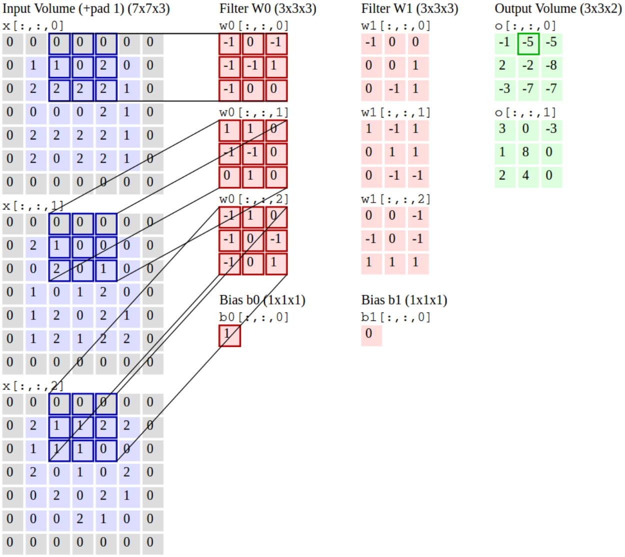


Figure: Extended convolution operation

Image processing

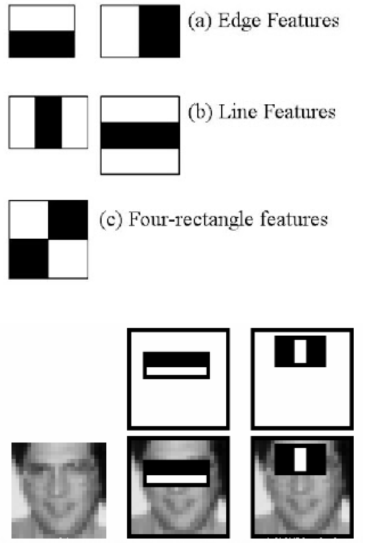


Figure: Haar features

Hough transform

If we we have a data point (x_i, y_i) and a linear model:

$$y_i = mx_i + b$$

A point can be explained by an infinite amount of lines $(m, b) \in R \times R$

$$b = -x_i m + y_i$$

Vote in parameter space using an accumulator function.

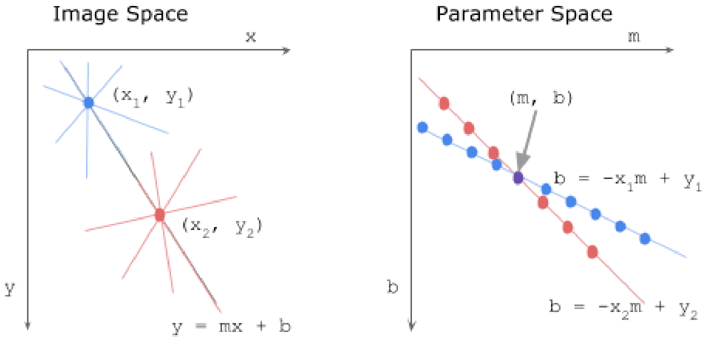


Figure: Hough space (source)

Hough transform

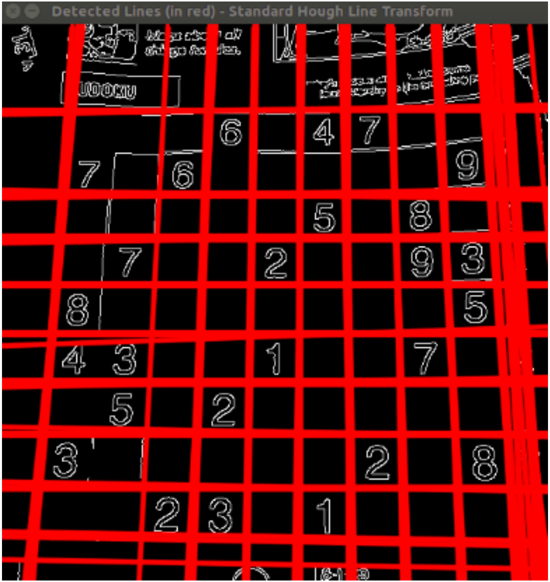


Figure: Application of Hough transform

Hough transform

If we we have a data point (x_i, y_i) and a circle model:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

A point can be explained as circles $(a, b) \in R \times R$

$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

Vote in parameter space using an accumulator function.

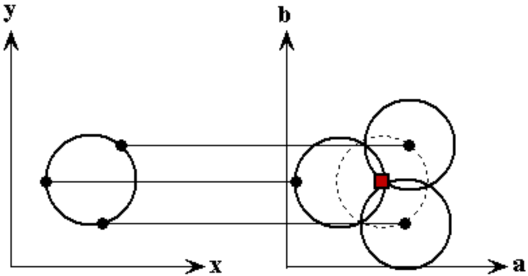


Figure: Hough space (source)

Object recognition

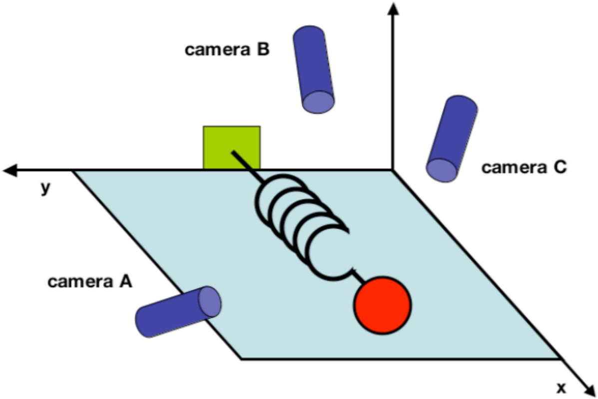


Figure: Experiment setup [Shlens 2014]

Object recognition

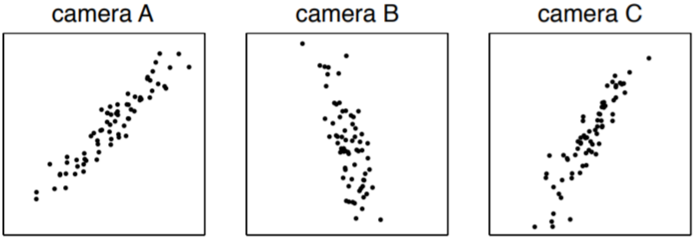


Figure: Experiment output [Shlens 2014]

Naive base coordinate description of a single sample

$$X = \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \\ x_C \\ y_C \end{bmatrix}$$

Object recognition

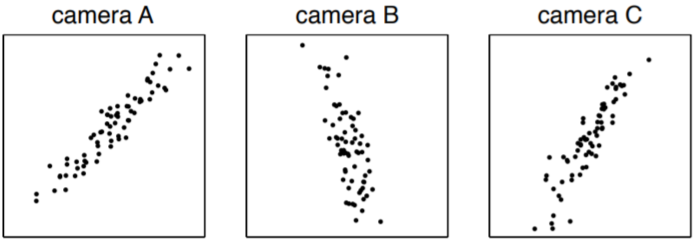


Figure: Experiment output [Shlens 2014]

However we have multiple samples:

$$\mathbf{X} = \begin{bmatrix} x_A^1 & x_A^2 & \dots & x_A^n \\ y_A^1 & y_A^2 & \dots & y_A^n \\ x_B^1 & x_B^2 & \dots & x_B^n \\ y_B^1 & y_B^2 & \dots & y_B^n \\ x_C^1 & x_C^2 & \dots & x_C^n \\ y_C^1 & y_C^2 & \dots & y_C^n \end{bmatrix}$$

Object recognition

We know that we only need one dimension.

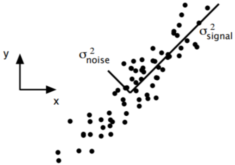


Figure: Signal to noise [Shlens 2014]

Moreover orthogonality can represent noise.

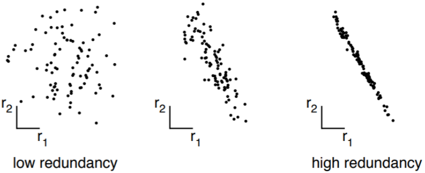


Figure: Redudancy [Shlens 2014]

Object recognition

We can transform our data $\mathbf{X}$

$$\mathbf{X} = \begin{bmatrix} x_A^1 & x_A^2 & \dots & x_A^n \\ y_A^1 & y_A^2 & \dots & y_A^n \\ x_B^1 & x_B^2 & \dots & x_B^n \\ y_B^1 & y_B^2 & \dots & y_B^n \\ x_C^1 & x_C^2 & \dots & x_C^n \\ y_C^1 & y_C^2 & \dots & y_C^n \end{bmatrix}$$

We can compute the redundancy (linear correlation) in the following way:

$$C = \frac{1}{n}\mathbf{X}\mathbf{X}^T$$

Object recognition

The explicit values of matrix C are as follows:

$$C = \begin{bmatrix} x_A^1 & x_A^2 & \dots & x_A^n \\ y_A^1 & y_A^2 & \dots & y_A^n \\ x_B^1 & x_B^2 & \dots & x_B^n \\ y_B^1 & y_B^2 & \dots & y_B^n \\ x_C^1 & x_C^2 & \dots & x_C^n \\ y_C^1 & y_C^2 & \dots & y_C^n \end{bmatrix} \begin{bmatrix} x_A^1 & y_A^1 & x_B^1 & y_B^1 & x_C^1 & y_C^1 \\ x_A^2 & y_A^2 & x_B^2 & y_B^2 & x_C^1 & y_C^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_A^n & y_A^n & x_B^n & y_B^n & x_C^1 & y_C^1 \end{bmatrix}$$

$$C = \begin{bmatrix} \mathbf{x}_A \cdot \mathbf{x}_A & \mathbf{x}_A \cdot \mathbf{y}_A & \mathbf{x}_A \cdot \mathbf{x}_B & \mathbf{x}_A \cdot \mathbf{y}_B & \mathbf{x}_A \cdot \mathbf{x}_C & \mathbf{x}_A \cdot \mathbf{y}_C \\ \mathbf{y}_A \cdot \mathbf{x}_A & \mathbf{y}_A \cdot \mathbf{y}_A & \mathbf{y}_A \cdot \mathbf{x}_B & \mathbf{y}_A \cdot \mathbf{y}_B & \mathbf{y}_A \cdot \mathbf{x}_C & \mathbf{y}_A \cdot \mathbf{y}_C \\ \mathbf{x}_B \cdot \mathbf{x}_A & \mathbf{x}_B \cdot \mathbf{y}_A & \mathbf{x}_B \cdot \mathbf{x}_B & \mathbf{x}_B \cdot \mathbf{y}_B & \mathbf{x}_B \cdot \mathbf{x}_C & \mathbf{x}_B \cdot \mathbf{y}_C \\ \mathbf{y}_B \cdot \mathbf{x}_A & \mathbf{y}_B \cdot \mathbf{y}_A & \mathbf{y}_B \cdot \mathbf{x}_B & \mathbf{y}_B \cdot \mathbf{y}_B & \mathbf{y}_B \cdot \mathbf{x}_C & \mathbf{y}_B \cdot \mathbf{y}_C \\ \mathbf{x}_C \cdot \mathbf{x}_A & \mathbf{x}_C \cdot \mathbf{y}_A & \mathbf{x}_C \cdot \mathbf{x}_B & \mathbf{x}_C \cdot \mathbf{y}_B & \mathbf{x}_C \cdot \mathbf{x}_C & \mathbf{x}_C \cdot \mathbf{y}_C \\ \mathbf{y}_C \cdot \mathbf{x}_A & \mathbf{y}_C \cdot \mathbf{y}_A & \mathbf{y}_C \cdot \mathbf{x}_B & \mathbf{y}_C \cdot \mathbf{y}_B & \mathbf{y}_C \cdot \mathbf{x}_C & \mathbf{y}_C \cdot \mathbf{y}_C \end{bmatrix}$$

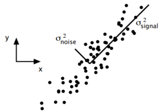
Object recognition

We want to find a base in which all diagonal are maximized.

$$\sigma = \frac{\sum_i (x_i - \mu_x)^2}{n} \tag{3}$$

and all off diagonal elements are zero (how much x tells me about y).

$$cov(x, y) = \sum_i \frac{(x_i - \mu_x)(y_i - \mu_y)}{n} \tag{4}$$



High correlation with x and y, imply I can discard one and use only the other.

Object recognition

We want a new basis where:

- Point to maximum variance (look where there is more change).
- Minimize linear correlation between values (remove redundancy).
- Basis vectors are orthogonal

In other words we want to diagonalize C.

Since the diagonal elements are variance and off-diagonal are covariance.

Object recognition

So we want a matrix P such that $\hat{C}$ is diagonal (P is a change of basis)

$$\hat{C} = \frac{1}{n}(PX)(PX)^T$$

$$\hat{C} = \frac{1}{n}(P(XX^T)P^T$$

$$\hat{C} = P(\frac{1}{n}XX^T)P^T$$

$$\hat{C} = PCP^T$$

If we choose P to be equal to the eigenvectors of C i.e. E

$$P = E$$

Then $\hat{C}$ will be a diagonal matrix. Eigenvectors satisfy

$$PE_i = \lambda E_i$$

$$\hat{C} = PCP^T \tag{5}$$

Assume

$$CE_i = \lambda_i E_i \tag{6}$$

If we multiply by a basis vector $e_i = [0, \ldots, 1, \ldots, 0]$

$$\hat{C} e_i = PCP^T e_i \tag{7}$$

(The trick) if we choose of P to be

$$P = \begin{bmatrix} \leftarrow E_1 \rightarrow \\ \vdots \\ \leftarrow E_n \rightarrow \end{bmatrix} \tag{8}$$

Then

$$P^T e_i = E_i \tag{9}$$

Thus substituting 9 into 7

$$\hat{C} e_i = PCE_i = P\lambda_i E_i \tag{10}$$

From our eigenvector assumption

$$\hat{C}e_i = P\lambda_i E_i \tag{11}$$

$$\hat{C}e_i = \lambda_i P E_i \tag{12}$$

From our previous assumption in equation 9 we have $e_i = PE_i$ thus:

$$\hat{C}e_i = \lambda_i e_i \tag{13}$$

Thus $\hat{C}e_i$ is a diagonal matrix (since all values not i are zero)

Mea culpa: If C is orthogonal then is orthogonally diagonziable i.e $P^T P = I$. In other words the vectors E_i are orthogonal

Object recognition

So the final result is that if we want diagonalize C

$$C = \begin{bmatrix} \mathbf{x}_A \cdot \mathbf{x}_A & \mathbf{x}_A \cdot \mathbf{y}_A & \mathbf{x}_A \cdot \mathbf{x}_B & \mathbf{x}_A \cdot \mathbf{y}_B & \mathbf{x}_A \cdot \mathbf{x}_C & \mathbf{x}_A \cdot \mathbf{y}_C \\ \mathbf{y}_A \cdot \mathbf{x}_A & \mathbf{y}_A \cdot \mathbf{y}_A & \mathbf{y}_A \cdot \mathbf{x}_B & \mathbf{y}_A \cdot \mathbf{y}_B & \mathbf{y}_A \cdot \mathbf{x}_C & \mathbf{y}_A \cdot \mathbf{y}_C \\ \mathbf{x}_B \cdot \mathbf{x}_A & \mathbf{x}_B \cdot \mathbf{y}_A & \mathbf{x}_B \cdot \mathbf{x}_B & \mathbf{x}_B \cdot \mathbf{y}_B & \mathbf{x}_B \cdot \mathbf{x}_C & \mathbf{x}_B \cdot \mathbf{y}_C \\ \mathbf{y}_B \cdot \mathbf{x}_A & \mathbf{y}_B \cdot \mathbf{y}_A & \mathbf{y}_B \cdot \mathbf{x}_B & \mathbf{y}_B \cdot \mathbf{y}_B & \mathbf{y}_B \cdot \mathbf{x}_C & \mathbf{y}_B \cdot \mathbf{y}_C \\ \mathbf{x}_C \cdot \mathbf{x}_A & \mathbf{x}_C \cdot \mathbf{y}_A & \mathbf{x}_C \cdot \mathbf{x}_B & \mathbf{x}_C \cdot \mathbf{y}_B & \mathbf{x}_C \cdot \mathbf{x}_C & \mathbf{x}_C \cdot \mathbf{y}_C \\ \mathbf{y}_C \cdot \mathbf{x}_A & \mathbf{y}_C \cdot \mathbf{y}_A & \mathbf{y}_C \cdot \mathbf{x}_B & \mathbf{y}_C \cdot \mathbf{y}_B & \mathbf{y}_C \cdot \mathbf{x}_C & \mathbf{y}_C \cdot \mathbf{y}_C \end{bmatrix}$$

We should choose the basis vectors E_i to describe our new base

$$P = \begin{bmatrix} \leftarrow E_1 \rightarrow \\ \vdots \\ \leftarrow E_n \rightarrow \end{bmatrix} \tag{14}$$

Object recognition

Principal component analysis (PCA) algorithm:

- Get data
- Compute data mean
- Subtract mean from data
- Compute covariance matrix (C).
- Find base the minimizes correlation, looks for maximum variance.
- Remove base vectors with small variance
- Project data to new base

What is a vector?

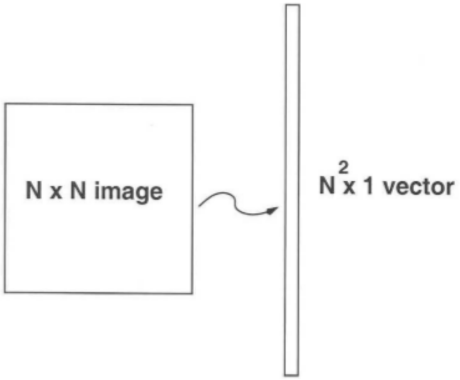


Figure: Image to vector

Object recognition

Get a large amount of faces and compute the mean face

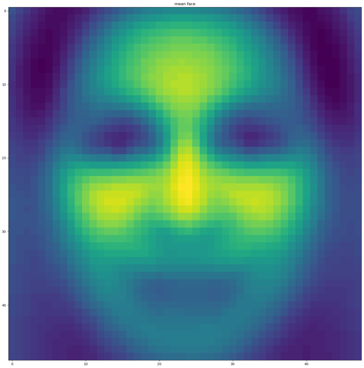
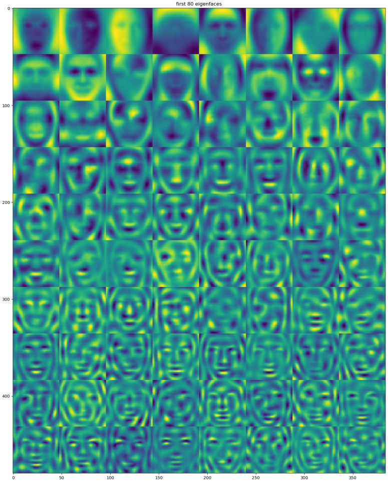


Figure: Mean face

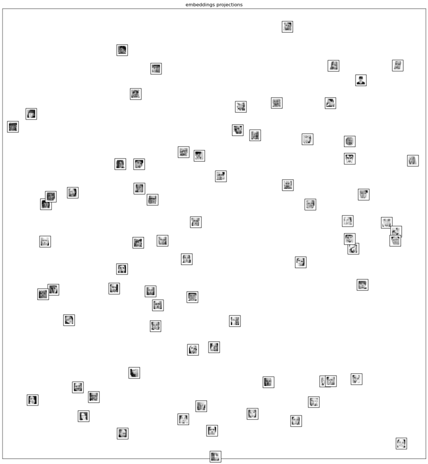
Object recognition

Subtract mean face from all faces and compute eigenvectors (eigenfaces) of the covariance matrix.



Object recognition

Now project any face to the eigenspace

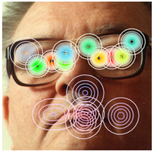


Additional material

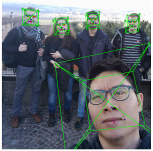
- Computer Vision: Algorithms and Applications, 2nd ed.
- First principles of computer vision
- Deep Learning with Python

https://github.com/oarriaga/paz

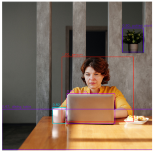
Examples implemented in PAZ



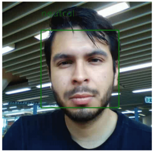
(a) Probabilistic kps



(b) Head-pose est.

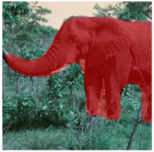


(c) Object detection



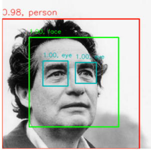
(d) Emotion recog.

(e) Keypoint est.



(f) Inst. segmentation

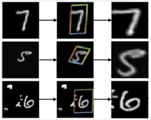
(g) Keypoint discovery



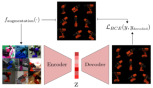
(h) Haar Cascades

(i) Pose estimation

(j) Face recognition



(k) Attention



(l) Implicit pose

Figure: PAZ examples