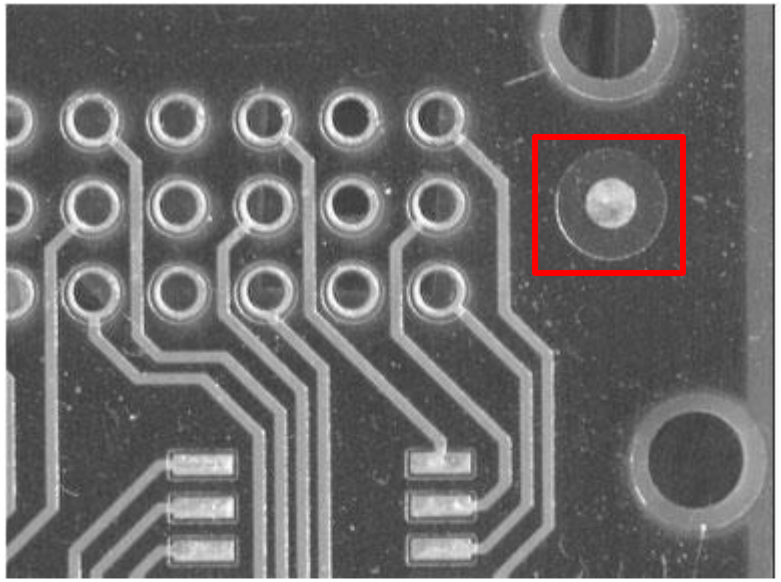
Instance Detection

Luigi Di Stefano (luigi.distefano@unibo.it)

Instance-level Object Detection (1)

- The *Instance-level Object Detection* problem occurs in countless applications and can be formulated as follows. Given a reference image (aka *model* image) of a specific object, determine whether the object is present or not in the image under analysis (aka *target* image) and, in case of detection, estimate the *pose* of the object.



- Depending on the application, the pose may often be given by a translation, a roto-translation or a similarity (roto-translation plus scale change).

Instance-level Object Detection (2)

- **The problem has a number of diverse facets: the sought object may appear only once or multiple times in the target image, we may be interested in detecting a number of diverse objects, each of which, in turn, may appear once of multiple times. For the sake of simplicity, and without much loss of generality, we will refer mainly to the basic setting: detecting a single object that may appear once in the target image. Generalization to the other variants turns out more often than not straightforward.**

- **Typical nuisances to be dealt with are intensity changes, occlusions and clutter. Moreover, computational efficiency is a major requirement in most practical applications.**

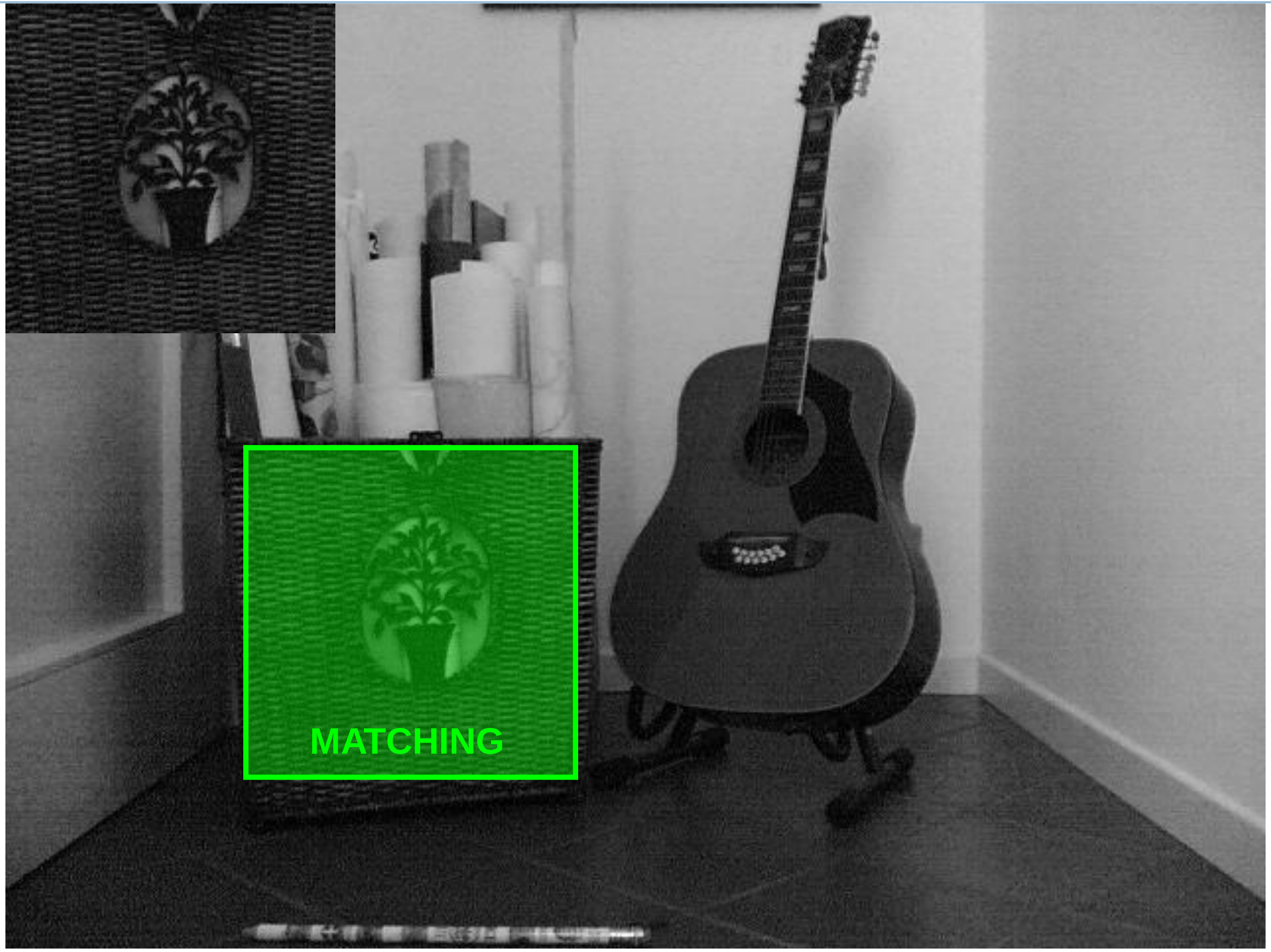
- **This problem is characterized by a <u>limited variability</u> as the assumption deals with the appearance of the object being captured by a single model image (i.e. roughly planar objects or no view-point changes) and the pose is typically either a *2D translation* or a *2D roto-translation* or a *similarity*.**

- **Given the limited variability, the problem can been addressed successfully by <u>classical computer vision</u> techniques, the major applications dealing mainly with the space of <u>industrial vision</u>.**

- **Instead, *Category-level Object Detection* aims at detecting certain kind of object(s) (e.g. cars, pedestrians..) regardless of their appearance and pose. Due to the <u>high-variability</u>, this problem is addressed by <u>machine/deep learning</u> techniques.**

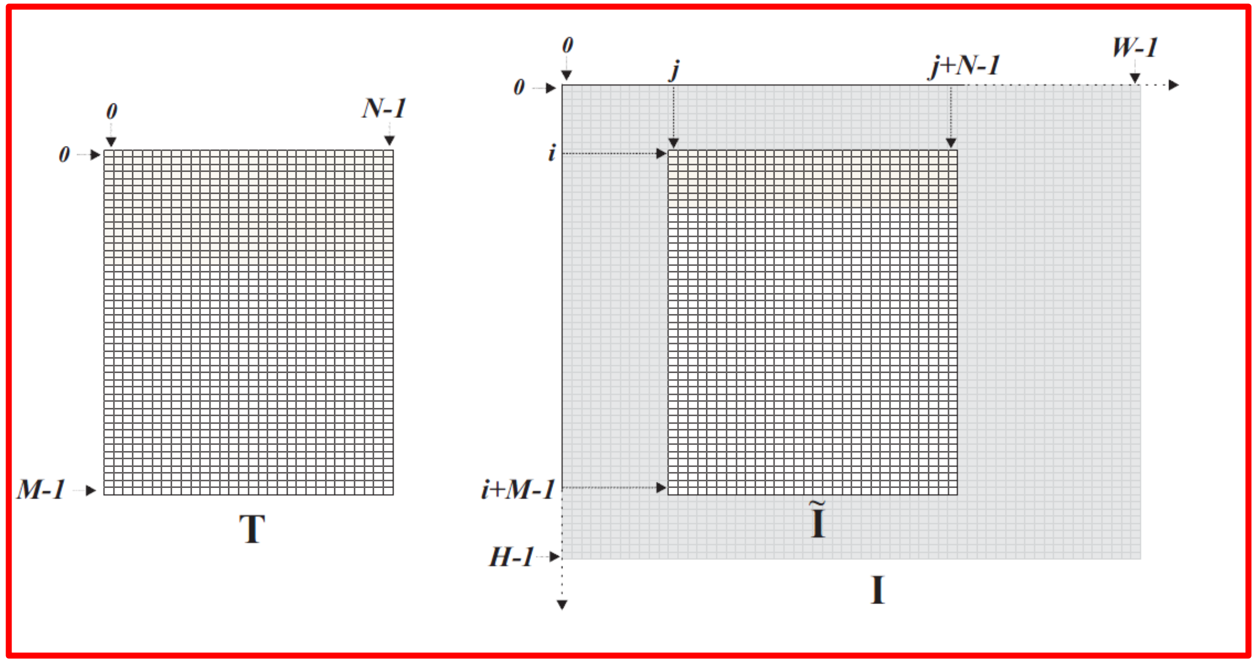
Template/Pattern Matching

The model image is slid across the target image to be compared at each pixel position to an equally sized window by means of a suitable (dis)similarity function



The global (minimum) maximum of the (dis)similarity function determines <u>whether</u> and <u>where</u> the object has been detected.

(Dis)Similarity Functions (1)



$$SSD(i,j)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(I(i+m,j+n)-T(m,n)\right)^2$$

Sum of Squared Differences

Both $\tilde{I}(i,j)$ (i.e. the window at position *(i, j)* of the target image having the same size as *T*) as well as *T* can be thought of as *M·N*-dimensional vectors (*flattening*). Accordingly, the SSD represents the squared L2 (Euclidean) norm of their difference.

(Dis)Similarity Functions (2)

$$SAD(i,j)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|I(i+m,j+n)-T(m,n)\right|$$

Sum of Absolute Differences

The SAD represents the L1 norm of the difference between vectors *Ĩ(i,j)* e *T*.

$$NCC(i,j)=\frac{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}I(i+m,j+n)\cdot T(m,n)}{\sqrt{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}I(i+m,j+n)^2}\cdot\sqrt{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}T(m,n)^2}}$$

Normalised Cross-Correlation

$$NCC(i,j)=\frac{\tilde{\mathbf{I}}(i,j)\cdot\mathbf{T}}{\left\|\tilde{\mathbf{I}}(i,j)\right\|\cdot\left\|\mathbf{T}\right\|}=\frac{\left\|\tilde{\mathbf{I}}(i,j)\right\|\cdot\left\|\mathbf{T}\right\|\cdot\cos\theta}{\left\|\tilde{\mathbf{I}}(i,j)\right\|\cdot\left\|\mathbf{T}\right\|}=\cos\theta \longrightarrow \tilde{\mathbf{I}}(i,j)=\alpha\cdot\mathbf{T}$$

(Invariant to linear intensity changes)

The NCC represents the cosine of the angle between vectors *Ĩ(i,j)* e *T*.

(Dis)Similarity Functions (3)

Zero-Mean Normalised Cross-Correlation (Correlation Coefficient)

$$\mu(\tilde{I}) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} I(i+m, j+n) \qquad \mu(T) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} T(m,n)$$

The NCC is computed after subtraction of the means:

$$I(i+m, j+n) \rightarrow \left(I(i+m, j+n) - \mu(\tilde{I})\right) \qquad T(m,n) \rightarrow \left(T(m,n) - \mu(T)\right)$$

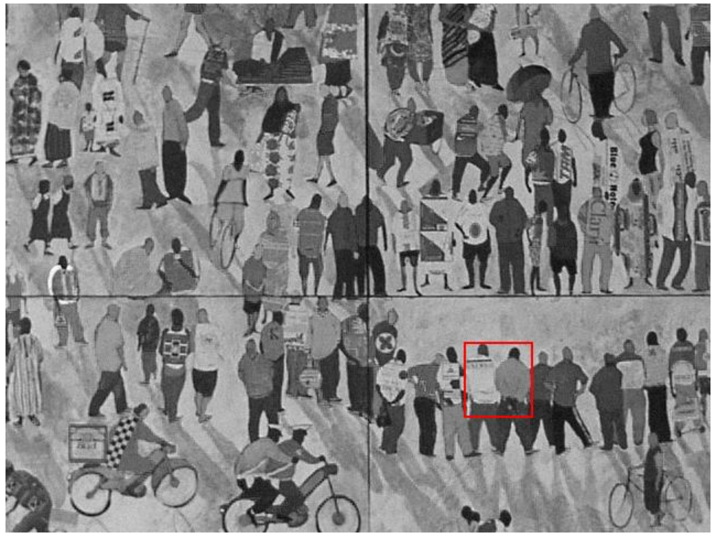
$$ZNCC(i,j) = \frac{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(I(i+m, j+n) - \mu(\tilde{I})\right)\cdot\left(T(m,n) - \mu(T)\right)}{\sqrt{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(I(i+m, j+n) - \mu(\tilde{I})\right)^2}\cdot\sqrt{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(T(m,n) - \mu(T)\right)^2}}$$

$$\longrightarrow \quad \tilde{\mathbf{I}}(i,j) = \alpha\cdot\mathbf{T} + \beta$$

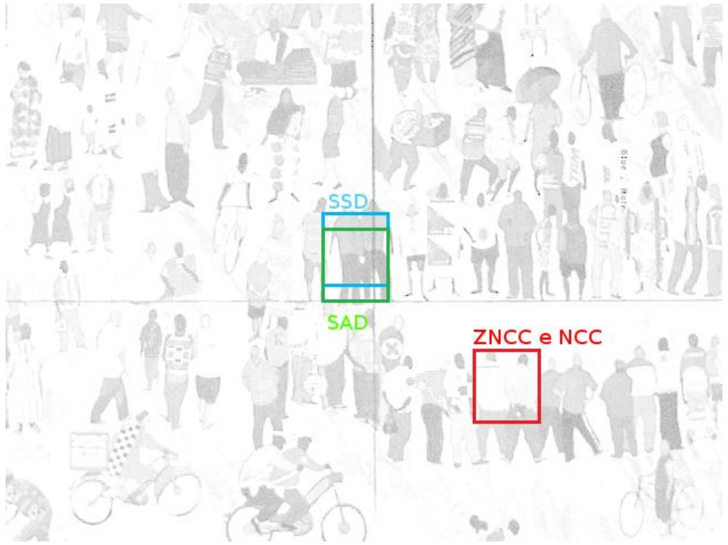
(Invariant to affine intensity changes)

OpenCV: cv2.matchTemplate()
OpenCV: Template Matching

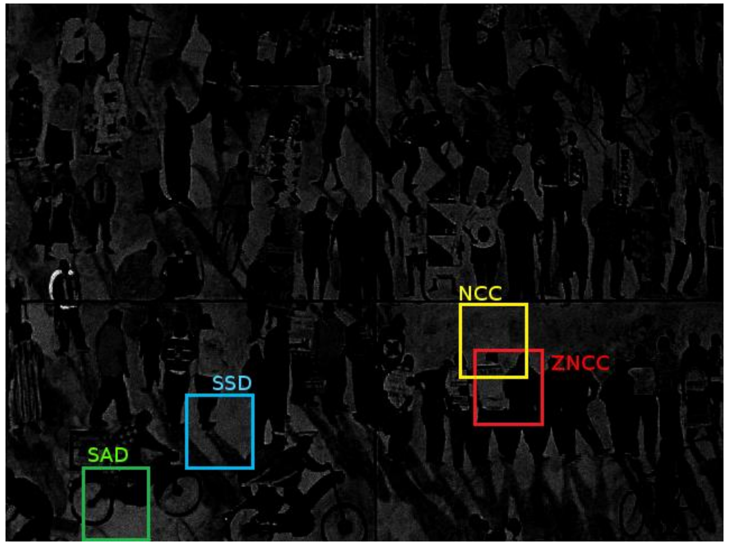
Comparison under significant intensity changes



Much brighter



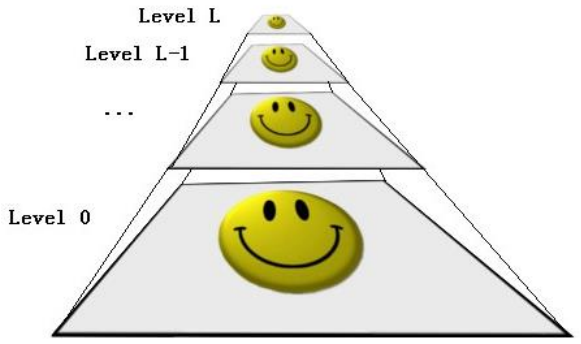
Much darker



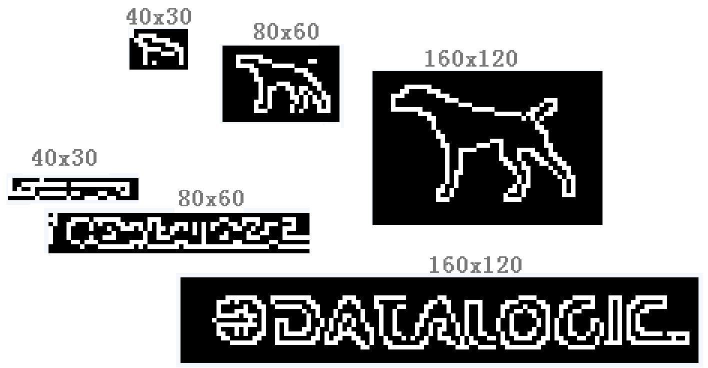
ZNCC turns out a similarity function very robust to intensity changes !

Fast Template/Pattern Matching

- **Template/pattern matching may turn out exceedingly slow whenever the model and/or target images have a large size (i.e. computational complexity is *O(M×N×W×H)*).**
- **The simplest and most popular approach to speed-up the computation concerns deploying an *image pyramid*:**



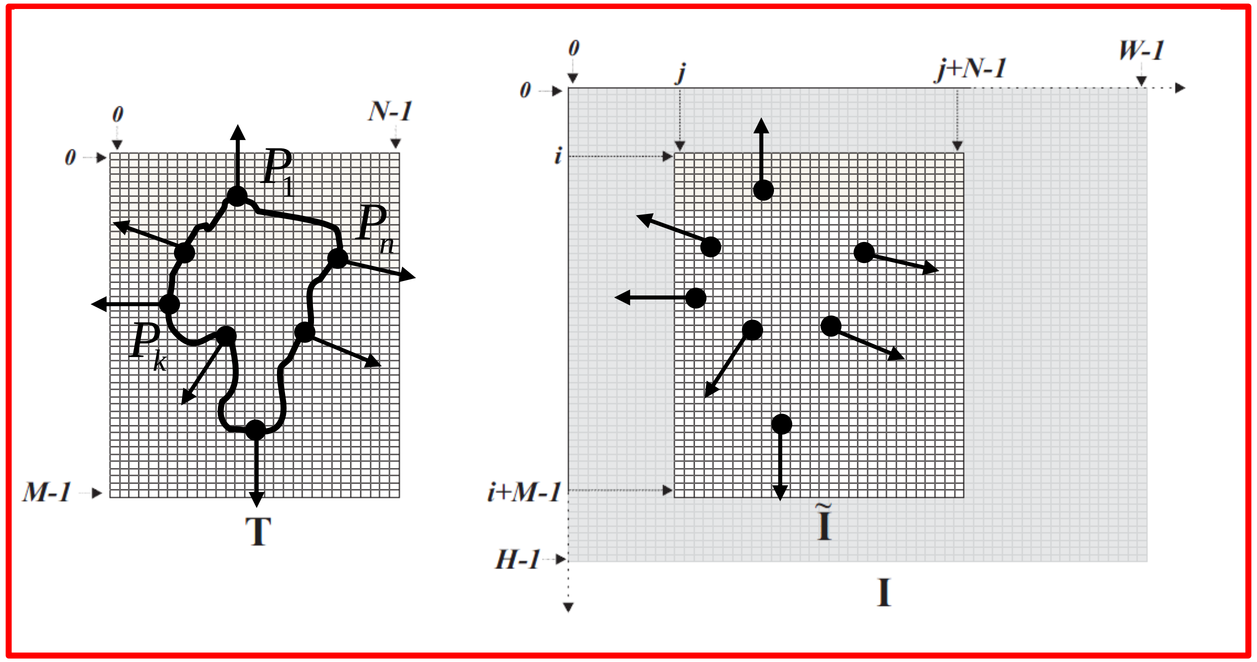
Very fast approximate approach, though the number of levels needs to be chosen carefully (and empirically) to avoid loss of information detrimental to detection. A few main exact approaches are presented in Appendix 1.



- **Smoothing and sub-sampling both the model and target image (typically 1/2 on both sides at each level).**
- **Full search at top level and then local refinements traversing back the pyramid down to original image.**
- **To handle rotation and/or scale changes one would need to probe rotated and/or scaled model images, possibly while deploying an image pyramid to speed-up the process.**

Shape-based Matching

- Shape-based Matching (Steger, 2002) may be thought of as an edge-based template/pattern matching approach.

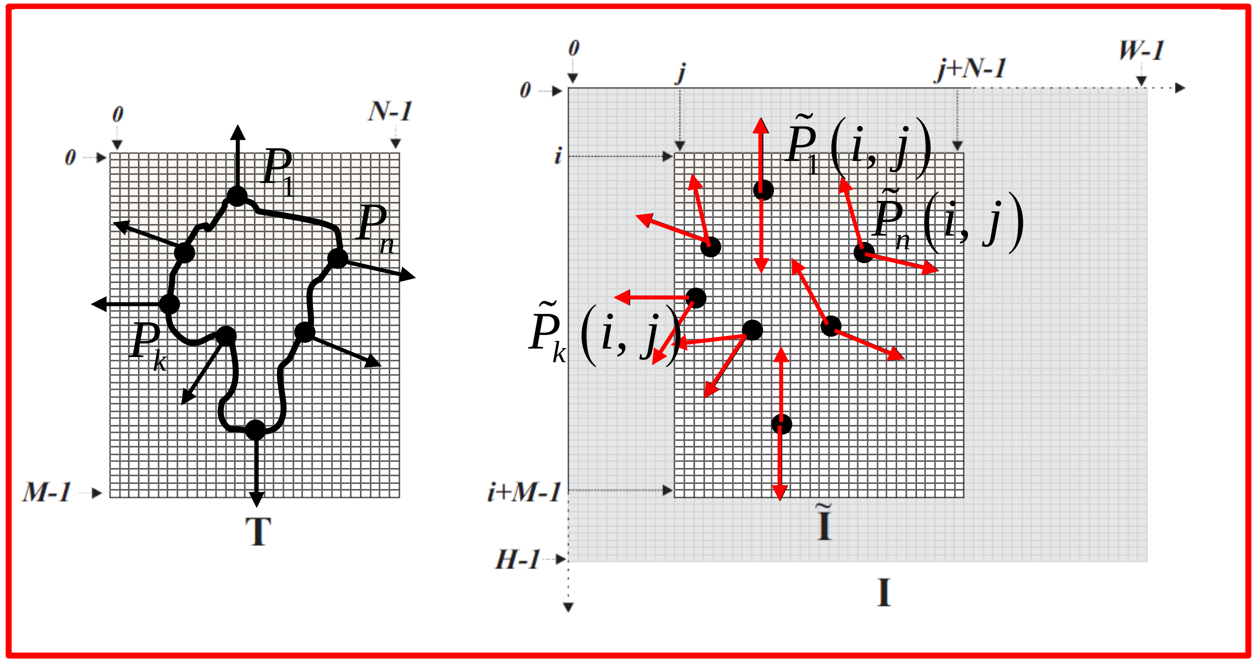


- First, a set of control points, P_k, is extracted from the model image by an Edge Detection operation (e.g. Sobel) and the gradient direction at each P_k is recorded.

~

Shape-based Matching

- **Shape-based Matching (Steger, 2002) may be thought of as an edge-based template/pattern matching approach.**



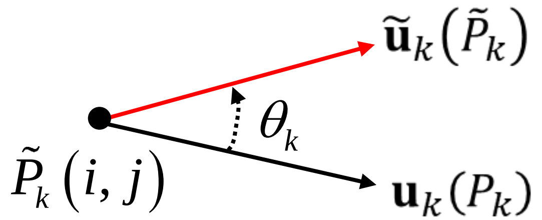
- **First, a set of control points, P_k, is extracted from the model image by an Edge Detection operation (e.g. Sobel) and the gradient direction at each P_k is recorded.**
- **Then, at each position *(i,j)* of the sliding window within the target image, the recorded gradient directions associated with control points are compared to those at their corresponding image points, $\tilde{P}_k(i,j)$, in order to compute a similarity function.**

Similarity Function

$$\mathbf{G}_k(P_k) = \begin{bmatrix} I_x(P_k) \\ I_y(P_k) \end{bmatrix}, \mathbf{u}_k(P_k) = \frac{1}{\|\mathbf{G}_k(P_k)\|} \begin{bmatrix} I_x(P_k) \\ I_y(P_k) \end{bmatrix}, k = 1..n$$

$$\tilde{\mathbf{G}}_k(\tilde{P}_k) = \begin{bmatrix} I_x(\tilde{P}_k) \\ I_y(\tilde{P}_k) \end{bmatrix}, \tilde{\mathbf{u}}_k(\tilde{P}_k) = \frac{1}{\|\tilde{\mathbf{G}}_k(\tilde{P}_k)\|} \begin{bmatrix} I_x(\tilde{P}_k) \\ I_y(\tilde{P}_k) \end{bmatrix}, k = 1..n$$

$$S(i,j) = \frac{1}{n}\sum_{k=1}^{n} \mathbf{u}_k(P_k) \cdot \tilde{\mathbf{u}}_k(\tilde{P}_k)$$



- **The above defined similarity function spans the interval [-1; 1]. It takes its maximum value when all the gradients at the control points in the current window of the target image are perfectly aligned to those at the control points of the model image.**
- **Choosing a detection threshold, S_{min}, can be thought of as specifying the fraction of model points which must be seen in the image to trigger a detection.**

More robust similarity functions

- **Certain application settings call for invariance to global inversion of contrast polarity along object's contours, as the object may appear either darker or brighter than the background in the target image. This kind of invariance can be achieved by a slight modification to the similarity function defined previously:**

$$S(i,j) = \frac{1}{n}\left|\sum_{k=1}^{n} \mathbf{u}_k(P_k)\cdot\tilde{\mathbf{u}}_k\left(\tilde{P}_k\right)\right| = \frac{1}{n}\left|\sum_{k=1}^{n}\cos\theta_k\right|$$

- **The following function is even more robust due to the ability to withstand local contrast polarity inversions:**

$$S(i,j) = \frac{1}{n}\sum_{k=1}^{n}\left|\mathbf{u}_k(P_k)\cdot\tilde{\mathbf{u}}_k\left(\tilde{P}_k\right)\right| \quad = \frac{1}{n}\sum_{k=1}^{n}|\cos\theta_k|$$

Speeding-up the search

Call-Out: given the chosen detection threshold *S_{min}* and the already computed *partial* similiarity, *$S_p(i,j)$*, it is possible to check whether the computation of the similarity function *$S(i,j)$* needs to continue or not:

$$S_p(i,j) = \sum_{k=1}^{p} \mathbf{u}_k(P_k) \cdot \tilde{\mathbf{u}}_k(\tilde{P}_k) \longrightarrow S(i,j) \le \frac{1}{n}\Big(S_p(i,j) + (n-p)\Big)$$

$$S(i,j) < S_{min} \longrightarrow \frac{1}{n}\Big(S_p(i,j) + (n-p)\Big) < S_{min}$$

$$S_p(i,j) + (n-p) < n \cdot S_{min} \longrightarrow S_p(i,j) < n \cdot S_{min} + (p-n)$$

If the condition holds, the computation of *S(i,j)* is terminated as the similarity score at (i,j) is certainly below the chosen detection threshold.

Main Properties

• Steger's similarity function is based on gradient directions only: as such it is invariant to affine intensity changes.

• With Shape-based Matching edges (i.e. control points) are detected in the model image (*T*) only, never in the target image (*I*). This renders the method <u>very robust wrt the potential fragility of carrying out edge detection on the target image</u> (e.g. how to set thresholds under varying lighting conditions ?)

• In principle, the similarity function is potentially robust to occlusion. Let us assume that a fraction of *T* is occluded in *I* and that gradient orientations are uniformly distributed across the occluding surface (as it would be the case, e.g., should the occluder be texture-less). Thus, the cosines of the angles between the gradients at the control points and their corresponding image points are uniformly distributed alike, such that the expected value of the sum of the contributions to the similarity function due to occluded control points is null. Accordingly, as already mentioned, the detection threshold determines the degree of occlusion that one wishes to withstand in the addressed application.

• In practise, a small threshold related to gradient magnitude at image points is enforced, so as to ensure nullifying the contributions due to uniformly textured occluders.

<u>Additional information on the Shape-based Matching Algorithm can be found in Appendix 2</u>

The Hough Transform

- The Hough Transform (HT) enables to detect objects having a shape that can be expressed by an equation (e.g. lines, circles, ellipses..) based on projection of the input data into a suitable space referred to as *parameter* or *Hough* space.

- The HT turns a *global* detection problem into a *local* one.

- The HT is usually applied after an edge detection process (i.e. the actual input data consist of the edge pixels extracted from the original image).

- The HT is robust to noise and allows for detecting the sought shape even though it is partially occluded into the image (up to a certain user-selectable degree of occlusion).

- The HT was invented to detect lines and later extended to other analytical shapes (circle, ellipses) as well as to arbitrary shapes (*Generalized Hough Transform - GHT*).

- The GHT principle is widely deployed also within object detection pipelines relying on local invariant features such as, e.g., SIFT.

Basic Principle (1)

- **We will start with the basic HT formulation for lines. Let's consider the equation of a line:**

$$y - mx - c = 0$$

- **In the usual image space interpretation of the line equation the parameters $(\hat{m}, \hat{c})$ are fixed**

$$y - \hat{m}x - \hat{c} = 0$$

so that the equation represents the mapping from point $(\hat{m}, \hat{c})$ of the parameter space to the image points belonging to the line.

- **However, we may instead fix $(\hat{m}, \hat{c})$**

$$\hat{y} - m\hat{x} - c = 0$$

so that the equation represents the mapping from image point $(\hat{x}, \hat{y})$ to the locus of the parameter space providing all the lines through the image point. Such a locus is itself a line.

Basic Principle (2)

- **Accordingly, if we consider two image points P_1,P_2 and map both into the parameter space, we get two lines intersecting at the parameter space point representing the image line through P_1,P_2 :**

$$\begin{cases} \hat{y}_1 - m\hat{x}_1 - c = & 0 \\ \hat{y}_2 - m\hat{x}_2 - c = & 0 \end{cases} \Longrightarrow \begin{cases} m = & \frac{\hat{y}_2 - \hat{y}_1}{\hat{x}_2 - \hat{x}_1} \\ c = & \frac{\hat{x}_2\hat{y}_1 - \hat{x}_1\hat{y}_2}{\hat{x}_2 - \hat{x}_1} \end{cases}$$

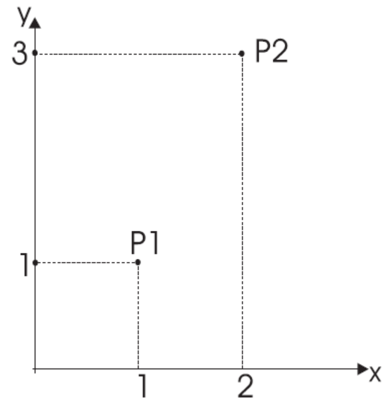
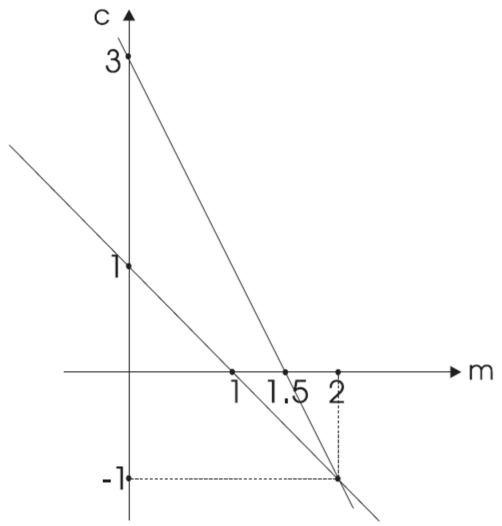


Image Space

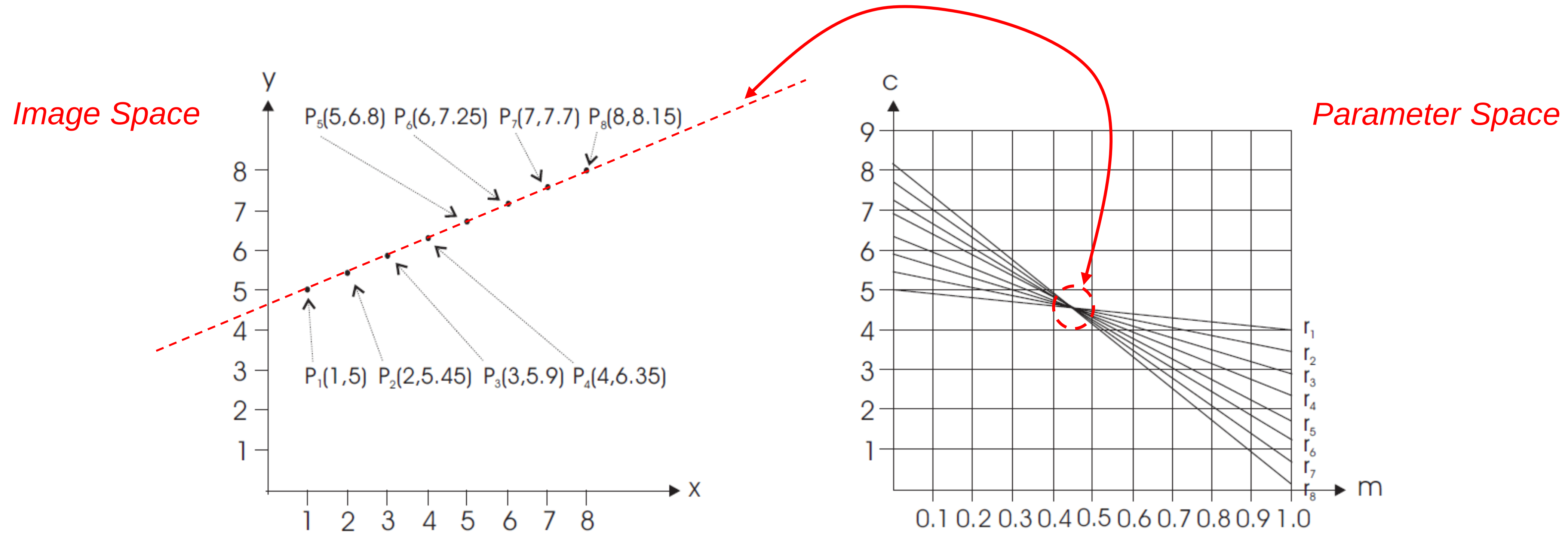


Parameter Space

- **More generally, if we map *n* image points we get as many intersections as *n(n-1)/2* (i.e. the number of lines through the *n* image points)**

Basic Principle (3)

- Considering then ***n* collinear image points**, we can notice that their corresponding transforms (i.e. parameter space lines) will intersect at a **single parameter space point** representing the image line along which such *n* points lay:



- According to the HT principle, thus, rather than seeking to find an extended shape into the image, we try to detect a specific point-feature (intersection of lines) in the parameter space.

Basic Principle (4)

- **Therefore, given a sought analytic shape represented by a set of parameters, the HT consists in mapping image points (i.e. usually edge points) so as to create curves into the parameter space of the shape.**

- **Intersections of parameter space curves indicate the presence of image points explained by <u>a certain instance of the shape</u>, the more the intersecting curves the more are such image points and thus the higher is the evidence on the presence of that instance in the image.**

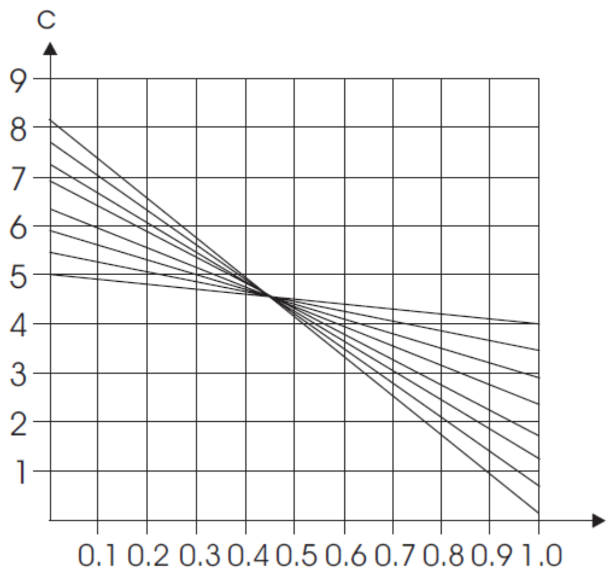
- **Detecting objects through the HT consists in finding parameter space points through which many curves do intersect, indeed a *local* rather than *global* detection problem.**

- **To make it work in practice, the parameter space needs to be quantized and allocated as a memory array, which is typically refereed to as *Accumulator Array (AA)*.**

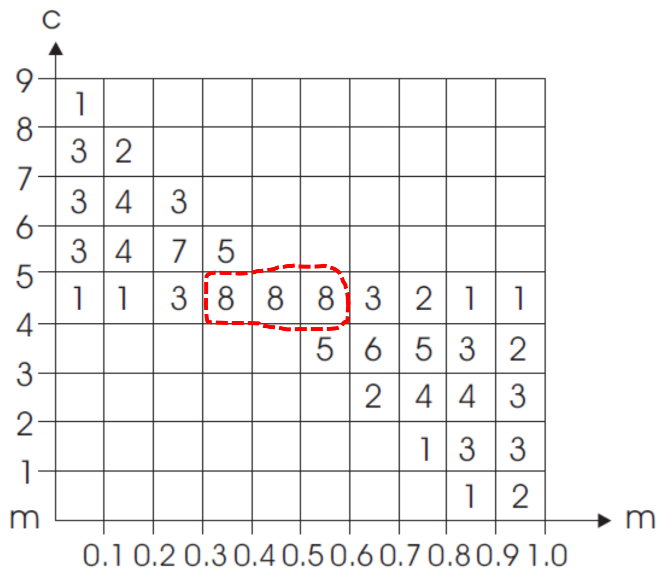
- **Then, curves are "drawn" into the *AA* by a so called *voting* process: the transform equation is repeatedly computed to increment the bins satisfying the equation. Accordingly, a high number of intersecting curves at a point of the parameter space will provide a high number of votes into a bin of the *AA*. Finding parameter space points through which many curves do intersect is thus implemented in practise by finding *peaks* of the *AA,* i.e. local maxima showing a high number of votes (again, an NMS process).**

Voting into the AA for lines

Continuous Parameter Space



AA (Quantized Parameter Space)



1									
3	2								
3	4	3							
3	4	7	5						
1	1	3	8	8	8	3	2	1	1
					5	6	5	3	2
						2	4	4	3
							1	3	3
								1	2

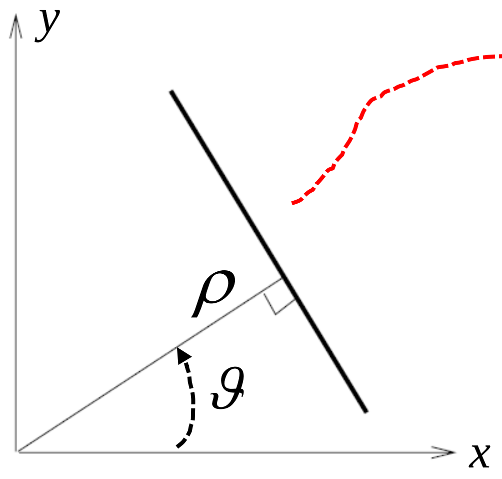
The AA highlights the presence of a line with $m \in [0.3, 0.6],\ c \in [4, 5]$

To detect the line more accurately, the AA should be quantized more finely.

The HT is robust to noise because spurious votes due to noise unlikely accumulate as coherently into a bin as to trigger a false detection. A partially occluded object can still be detected provided that the threshold on the minimum number of votes requited to declare a detection is lowered according to the degree of occlusion to be handled.

HT for Line Detection

- The usual line parametrization considered so far (i.e. *y-mx-c=0*) is impractical due to *m* spanning an infinite range. The so called *normal parametrization* is therefore adopted in the HT for lines:



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

with mage points $(\hat{x}, \hat{y})$ being mapped into sinusoidal curves of the (ϑ, ρ) parameter space:

$$\rho = \hat{x}\cos\vartheta + \hat{y}\sin\vartheta$$

- With the normal parametrization: $\vartheta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \quad \rho \in [-\rho_{\max}, \rho_{\max}]$

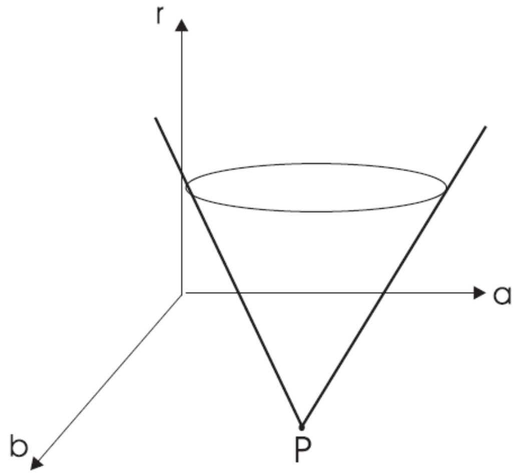
while ρ_{max} is usually taken as large as the image diagonal : $N \times N \text{ pixels} \rightarrow \rho_{\max} = N \cdot \sqrt{2}$

OpenCV: cv2.HoughLines()
OpenCV: Hough Line Transform

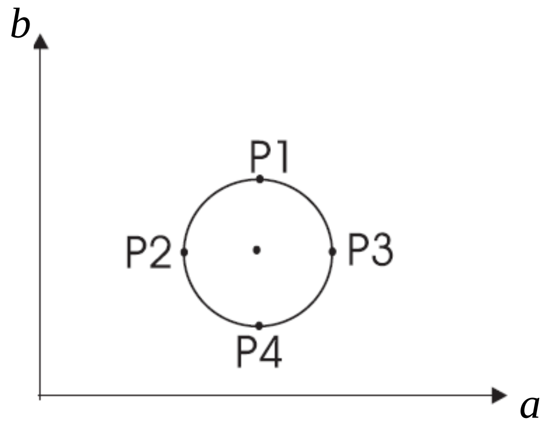
HT for Circle Detection (1)

- **In the case of circular objects, the equation** $(x-a)^2+(y-b)^2=r^2$ **is interpreted as a mapping from the image space to the 3-dimensional parameter space** (a, b, r) **.**

- **Accordingly, equation** $(\hat{x}-a)^2+(\hat{y}-b)^2=r^2$ **provides all the parameter triplets representing circles through image point** $(\hat{x},\hat{y})$**.**



$HT\left(P\left(\hat{x},\hat{y}\right)\right)$: cone with vertex at P



Intersection with plane $r=\bar{r}$

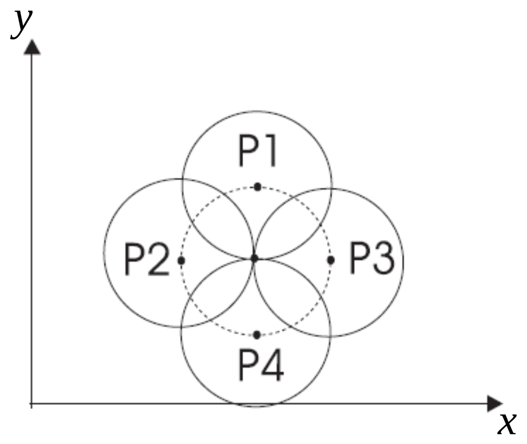


Image circles definied by $\left(P_i,\bar{r}\right)$

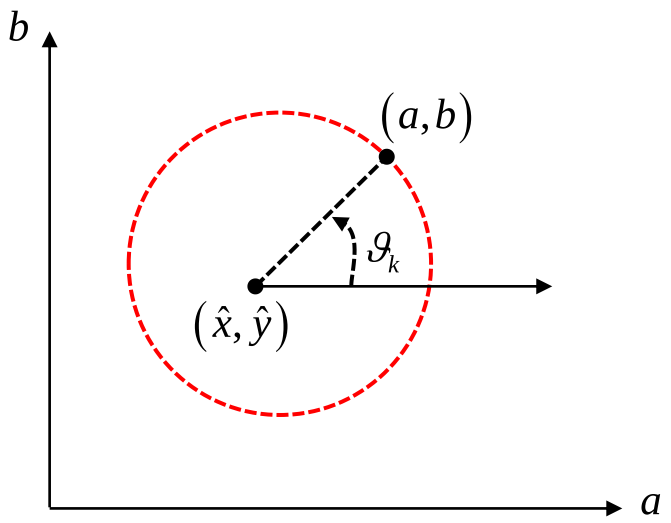
HT for Circle Detection (2)

- **In a practical implementation, a stack of images may be adopted as 3D Accumulator Array, each layer of the stack being associated with a radius value. To cast votes, circles may be successively "drawn" in each layer:**

$$\forall \bar{r} \in \left[r_{\min}, r_{\max} \right]: \quad \begin{cases} a = \hat{x} + \bar{r} \cdot \cos \vartheta_k \\ b = \hat{y} + \bar{r} \cdot \sin \vartheta_k \end{cases} \qquad \vartheta_k \in [0, 2\pi]$$

$$\downarrow$$

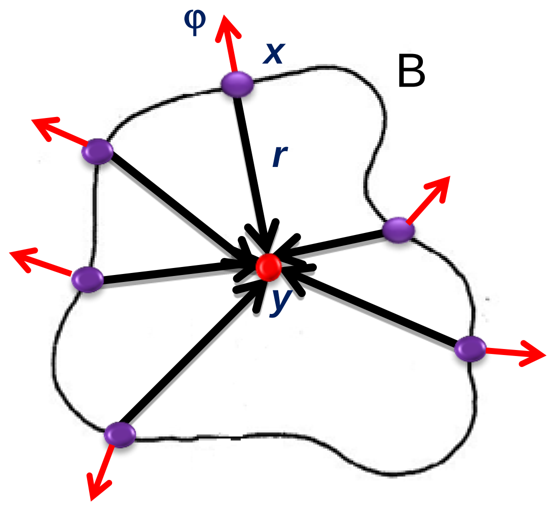
$$\begin{cases} \Delta \vartheta = \dfrac{2\pi}{M} \\ \vartheta_k = k \Delta \vartheta, \; k = 0 \ldots M \end{cases}$$



OpenCV: cv2.HoughCircles().
OpenCV: Hough Circle Transform

GHT: Generalized Hough Transform (Off-line)

- **The HT has been extended to detect arbitrary (i.e. non analytical) shapes:**

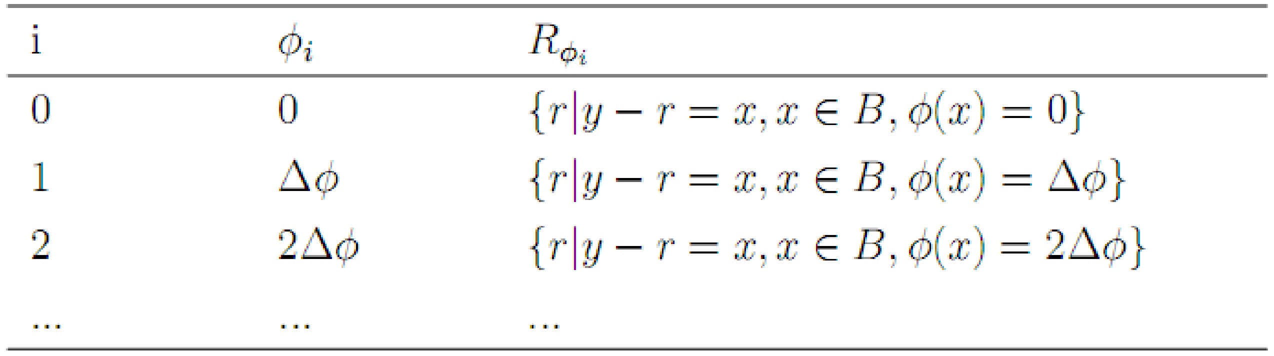


Off-line Phase (to build the object's model)

1. Detect edges (B) in the model image.

2. Choose a reference point *y* (e.g. barycentre of B).

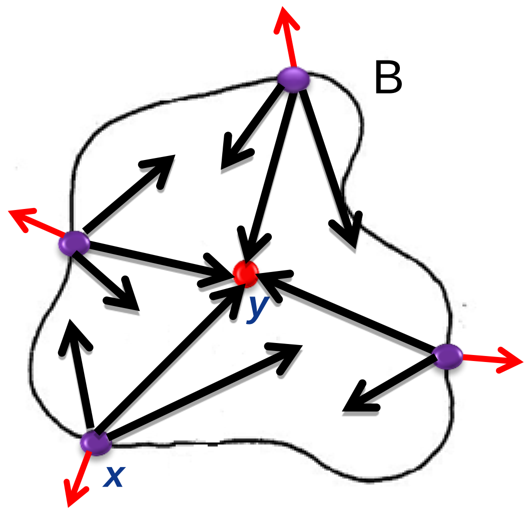
3. For each point *x* belonging to object's border *B*:
- **3.1 Compute vector *r* from *y* to *x* (*i.e. r = y-x*).**
- **3.2 Compute gradient's direction *φ(x)***
- **3.3 Store *r* in a table (R-Table) indexed by a chosen gradient's direction quantization step (Δφ).**



i	ϕ_i	R_{ϕ_i}
0	0	$\{r \mid y - r = x, x \in B, \phi(x) = 0\}$
1	$\Delta\phi$	$\{r \mid y - r = x, x \in B, \phi(x) = \Delta\phi\}$
2	$2\Delta\phi$	$\{r \mid y - r = x, x \in B, \phi(x) = 2\Delta\phi\}$
...	...	...

An entry in the R-Table can contain several r vectors.

GHT: Generalized Hough Transform (On-line)



On-line Phase (object detection)

1. Detect edges in the target image and initialize an other image as accumulator array, *A[y].*

   For each edge pixel *x* of the target image :

   2. Compute gradient direction φ
   3. Quantize φ to index the R-Table. For each r_i vector stored into the indexed row:

      3.1 Compute the position the reference point *y*:

      $y = x + r_i$

      3.2 Cast a vote into the accumulator array:

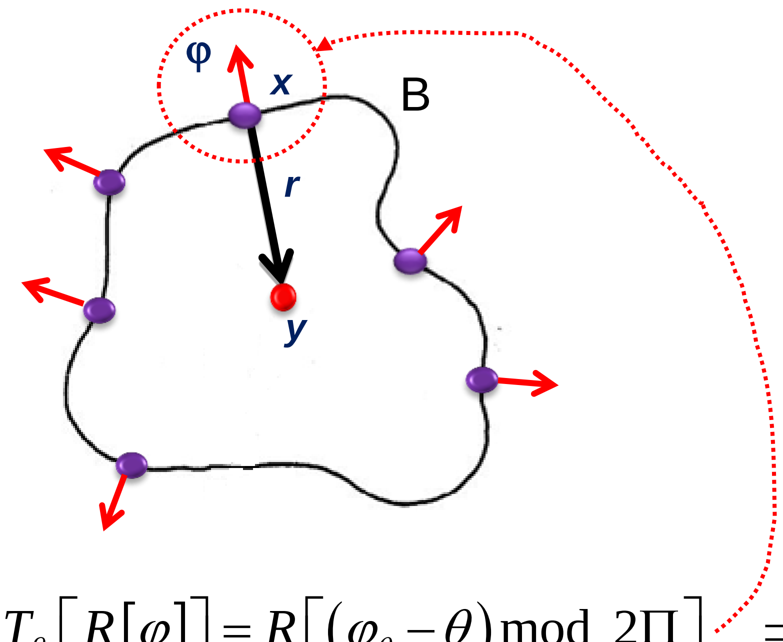
      $A[y]{+}{+}$

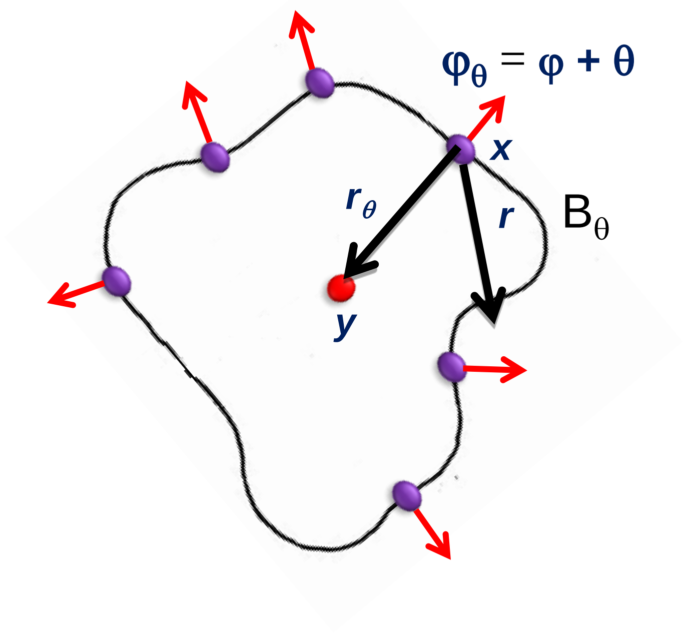
4. As usual with the HT, instances of the sought object are detected by finding peaks of the accumulator array.

i	ϕ_i	R_{ϕ_i}
0	0	$\{r \mid y - r = x, x \in B, \phi(x) = 0\}$
1	$\Delta\phi$	$\boxed{r1,r2,r3}$
2	$2\Delta\phi$	$\{r \mid y - r = x, x \in B, \phi(x) = 2\Delta\phi\}$
...	...	...

GHT - Invariance to Rotation and Scale

- **Handling rotation**





As we do not know θ, we need to quantize the rotation range and try all possible discretized rotations (<u>brute force approach</u>)

$$T_\vartheta\left[R[\varphi]\right] = R\left[(\varphi_\theta - \theta)\bmod 2\Pi\right] \Rightarrow \forall r : y = x + r_\theta = x + ROT(r,\theta),\ \ A[y,\theta]{+}{+}$$

- **Handling scale**

$$\forall r : y = x + r_s = x + s \cdot r,\ \ A[y,s]{+}{+}$$

- **Handling rotation and scale (*<u>similarity</u>*)**

$$\forall r : y = x + s \cdot r_\theta,\ \ A[y,\theta,s]{+}{+}$$

The scale range needs to be quantized, alike. <u>Brute force approach</u>, hardly feasible in case of similarity.

GHT with Local Invariant Features (*e.g.* SIFT) aka *Star Model* – Off-line

1. Detect keypoints (*e.g.* DoG) and compute descriptors (*e.g.* SIFT) in the model image:

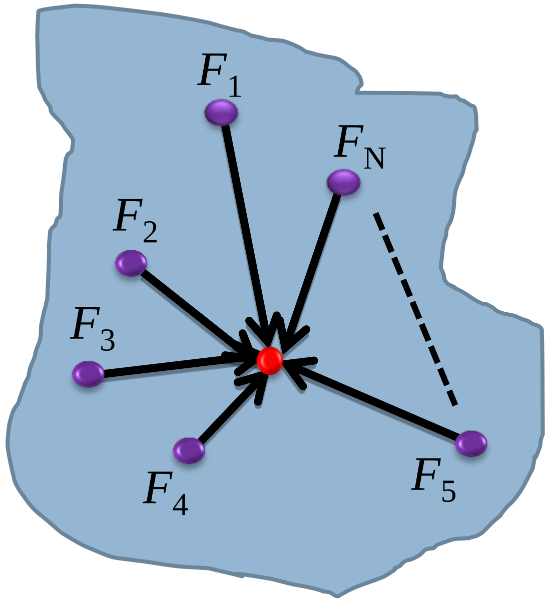
$$F = \{F_1, F_2, \dots F_N\}, \qquad F_i = (\mathbf{P}_i, \mathbf{D}_i, \varphi_i, S_i)$$

2. Choose a reference point (e.g. barycentre of $\mathbf{P}_i$).

$$\mathbf{P}_C = \frac{1}{N}\sum_{i=1}^{N}\mathbf{P}_i$$

3. Compute the joining vectors between the features and the reference point:

$$\forall F_i \in F \quad \rightarrow \mathbf{V}_i = \mathbf{P}_C - \mathbf{P}_i$$



The *Star Model* consist of the features and their joining vectors (no need of the R-Table):

$$\downarrow$$

$$F_i = (\mathbf{P}_i, \mathbf{D}_i, \varphi_i, S_i, \mathbf{V}_i)$$

GHT with Local Invariant Features - On-line

1. Detect keypoints and compute descriptors in the target image and initialize an accumulator array, A[P_c]:

$$\tilde{F} = \{\widetilde{F_1}, \widetilde{F_2}, \dots \widetilde{F_M}\},\ \widetilde{F_j} = (\widetilde{\mathbf{P}_j}, \widetilde{\mathbf{D}_j}, \widetilde{\varphi_j}, \widetilde{S_j})$$

2. Match descriptors between target and model features:

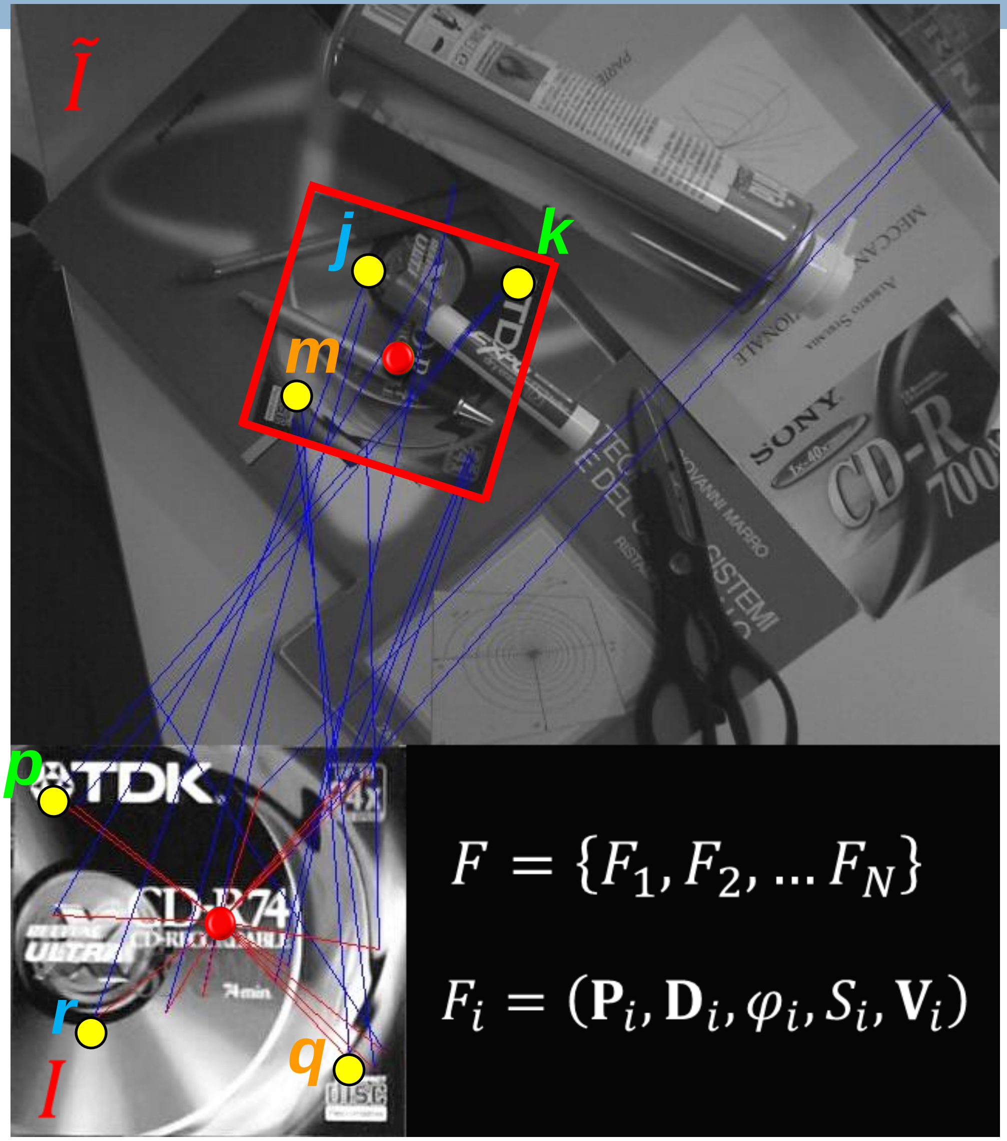
$$\forall \widetilde{F_j} \in \tilde{F}:\ \widetilde{D_j} \Rightarrow D_i$$

3. For each target feature that matches a model feature, compute the position of the reference point (3.1) and cast a vote into the accumulator array (3.2):

$$\forall\ (\widetilde{F_j}, F_i): \widetilde{D_j} \Rightarrow D_i$$

$$\widetilde{\mathbf{P}_{C_j}} = \widetilde{\mathbf{P}_j} + \mathbf{V}_i \quad (3.1)$$

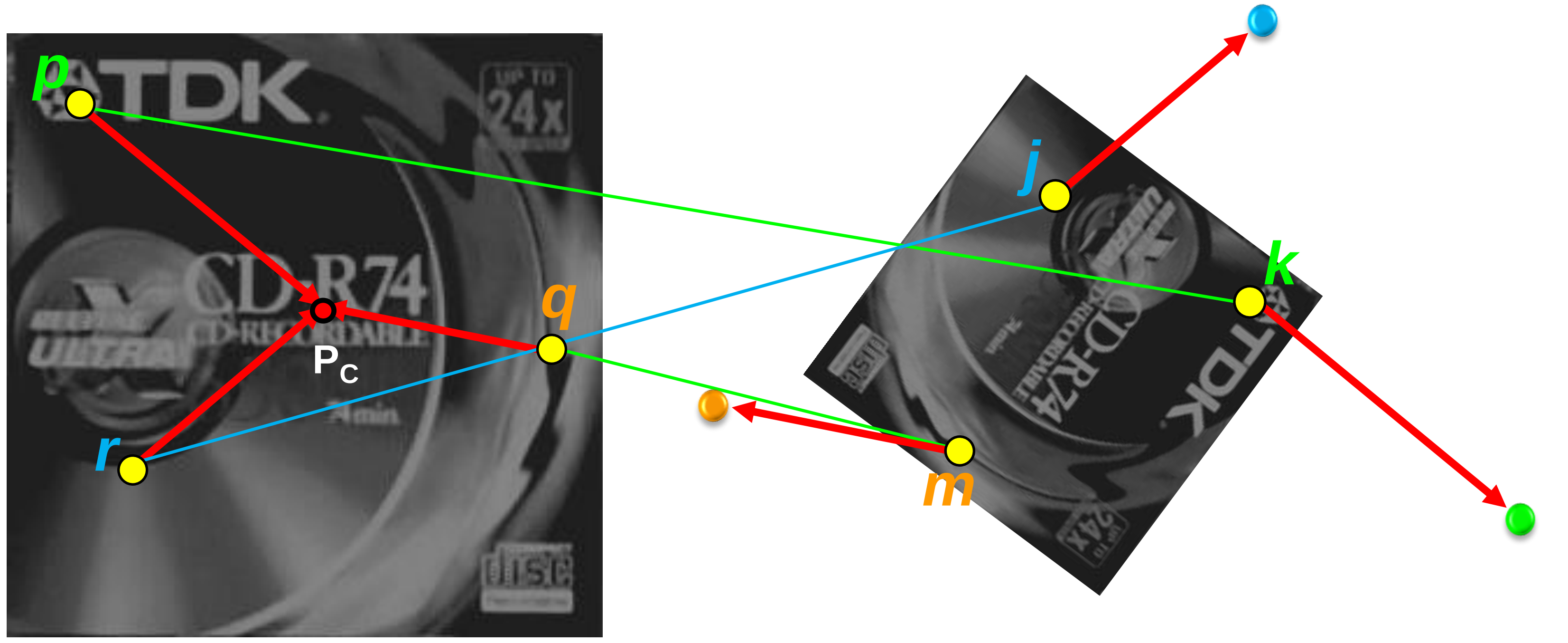
$$\mathbf{A}\left[\widetilde{\mathbf{P}_{C_j}}\right]++ \quad (3.2)$$



$$F = \{F_1, F_2, \dots F_N\}$$

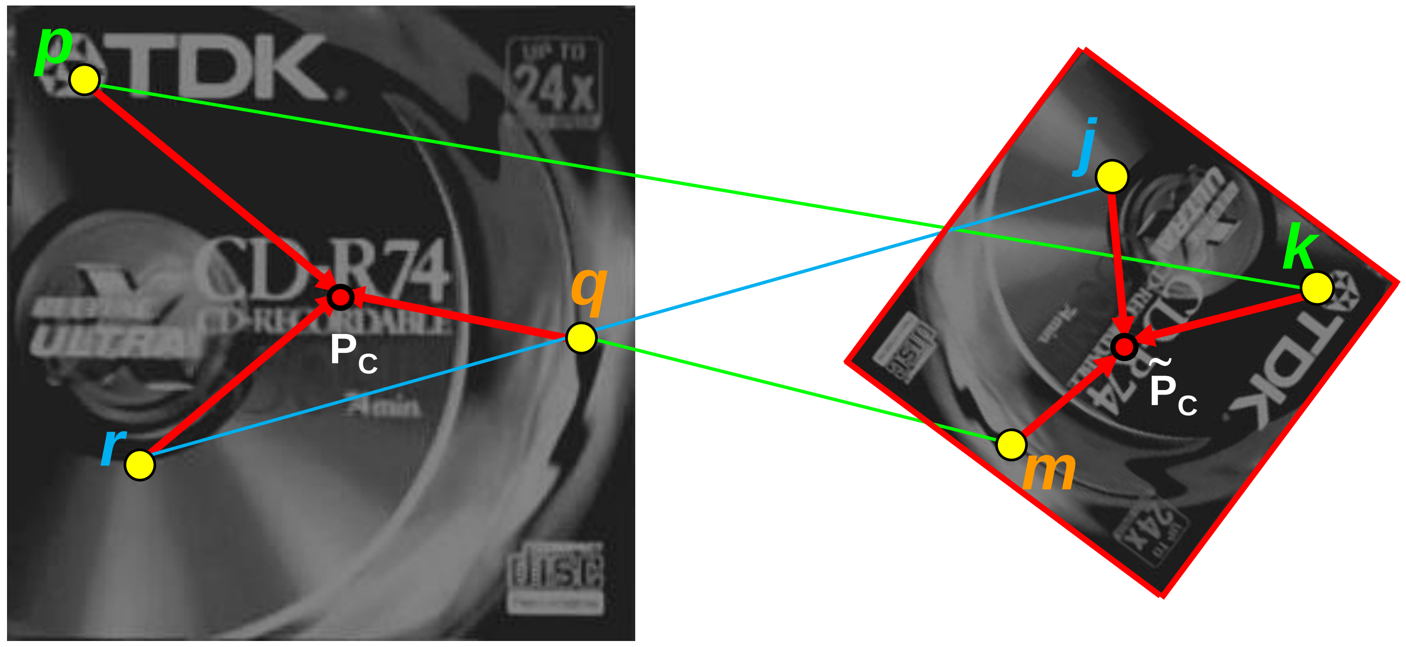
$$F_i = (\mathbf{P}_i, \mathbf{D}_i, \varphi_i, S_i, \mathbf{V}_i)$$

Similarity Invariant Voting (1)



$$\forall \left(\tilde{F}_j = (\tilde{\mathbf{P}}_j, \widetilde{\mathbf{D}_j}, \widetilde{\varphi_j}, \tilde{S}_j), F_i = (\mathbf{P}_i, \mathbf{D}_i, \varphi_i, S_i, \mathbf{V}_i)\right) : \widetilde{D_j} \Rightarrow D_i \longrightarrow \widetilde{\mathbf{P}_{C_j}} = \tilde{\mathbf{P}}_j + \mathbf{V}_i$$

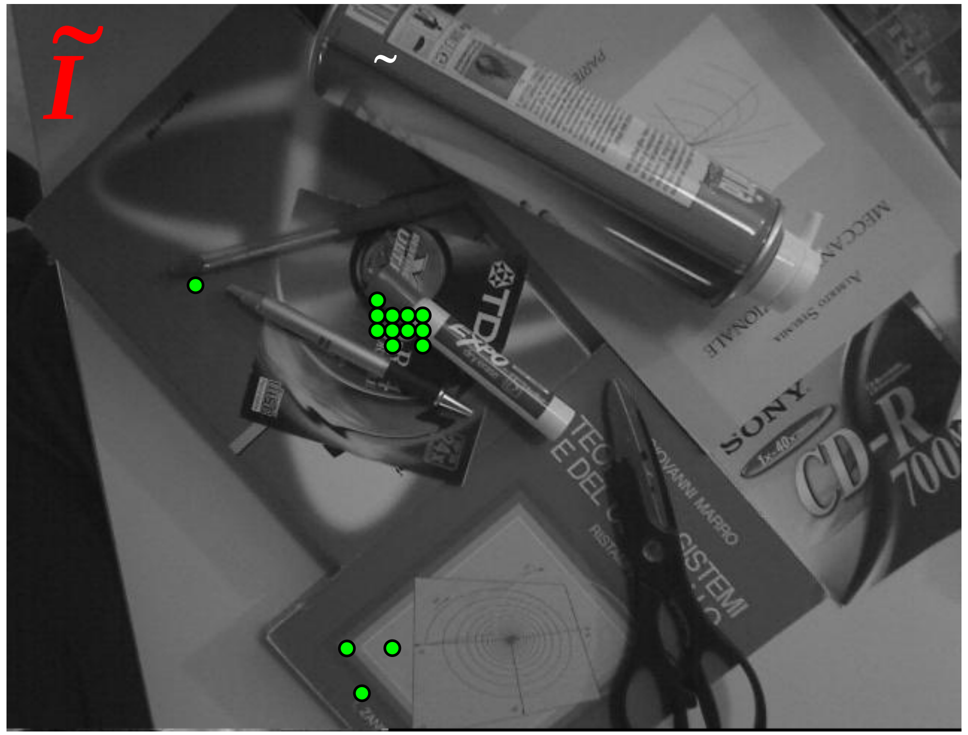
Similarity Invariant Voting (2)

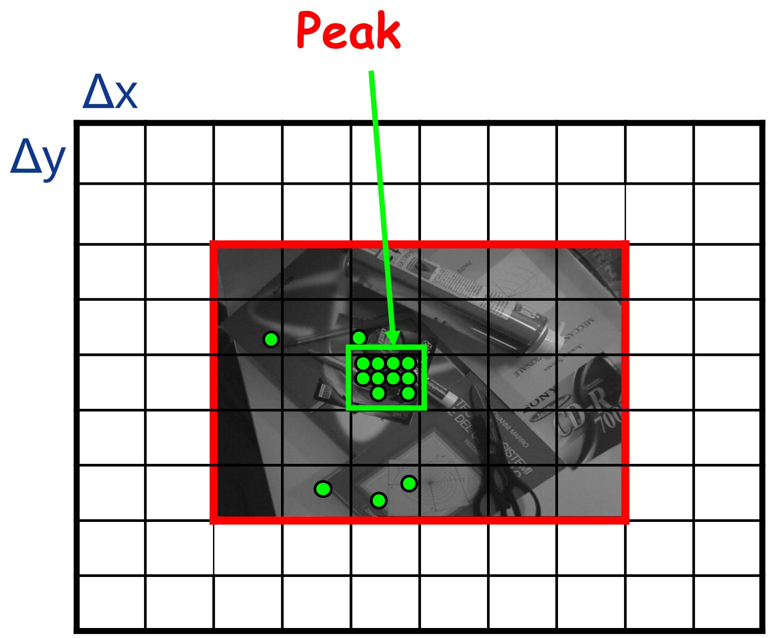


$$\forall\left(\widetilde{F_j}=\left(\widetilde{\mathbf{P}_j},\widetilde{\mathbf{D}_j},\widetilde{\varphi_j},\widetilde{S_j}\right),F_i=\left(\mathbf{P}_i,\mathbf{D}_i,\varphi_i,S_i,\mathbf{V}_i\right)\right):\widetilde{D_j}\Rightarrow D_i \longrightarrow \widetilde{\mathbf{P}_{C_j}}=\widetilde{\mathbf{P}_j}+\mathbf{V}_i$$

$$\Delta\varphi_j=\widetilde{\varphi_j}-\varphi_i,\ \Delta S_j=\frac{\widetilde{S_j}}{S_i} \longrightarrow \widetilde{\mathbf{P}_{C_j}}=\widetilde{\mathbf{P}_j}+\Delta S_j\cdot R\left(\Delta\varphi_j\right)\mathbf{V}_i$$

2D Accumulator Array (Quantized Pose Space)



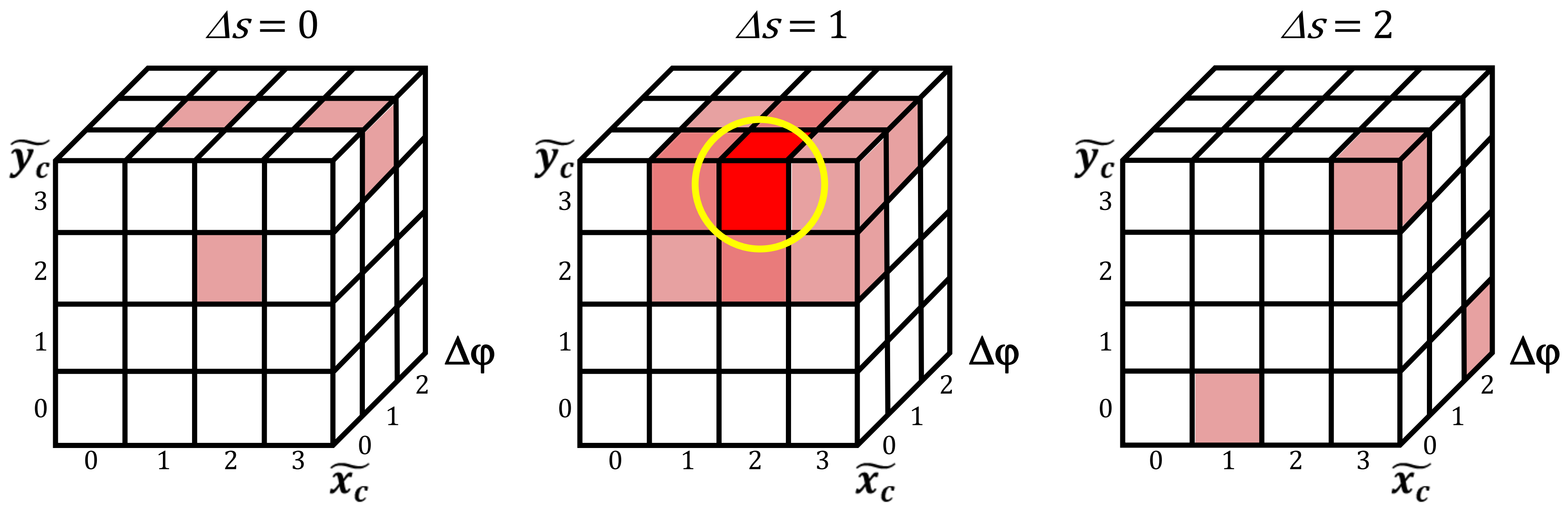


Quantized Translation: $\widetilde{P_c} = (\widetilde{x_c}, \widetilde{y_c})$

A more accurate pose (e.g. similarity or homography) can then be estimated based on the **correspondences** associated with the votes cast in the bin(s) showing up as peak(s). However, with a 2D AA, votes coming from different rotation ($\Delta\varphi$) and scale (ΔS) hypotheses may accumulate into the same bin. This may yield false positives and requires robust (RANSAC) estimation of the pose (e.g. similarity or homography) as the set of correspondences associated with the peak(s) may include wrong ones (outliers).

4D Accumulator Array – Quantized Similarity

- **Each bin does correspond to a different quantized translation, $(\widetilde{x_c}, \widetilde{y_c})$, rotation, $\Delta\varphi$, and scale, ΔS.**

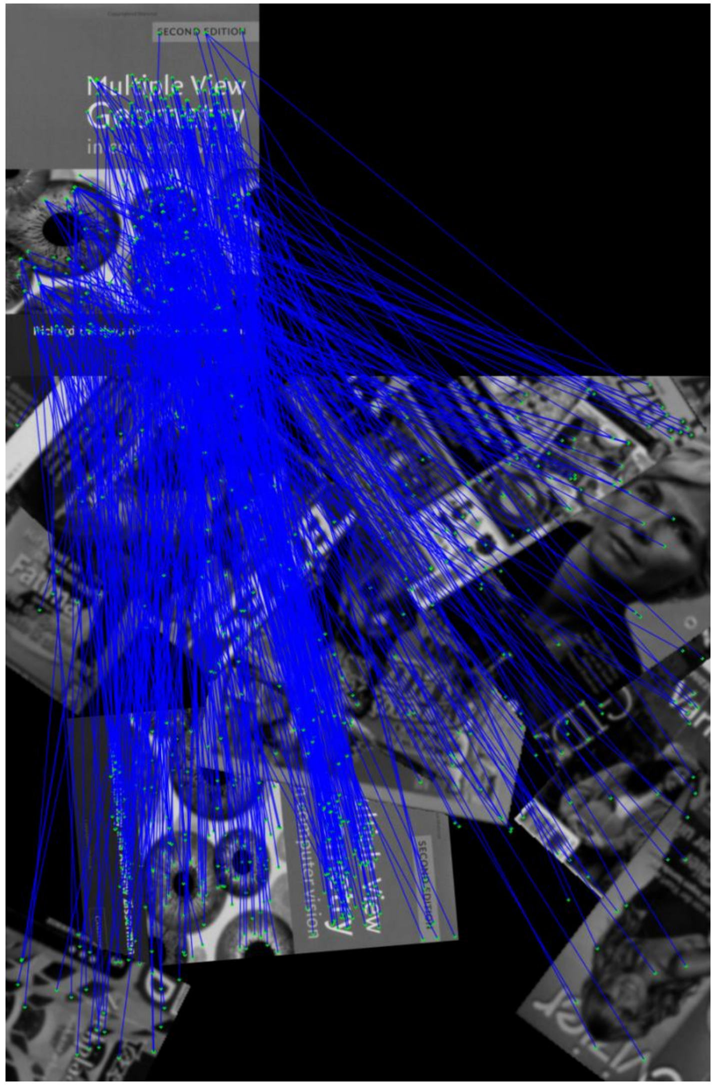


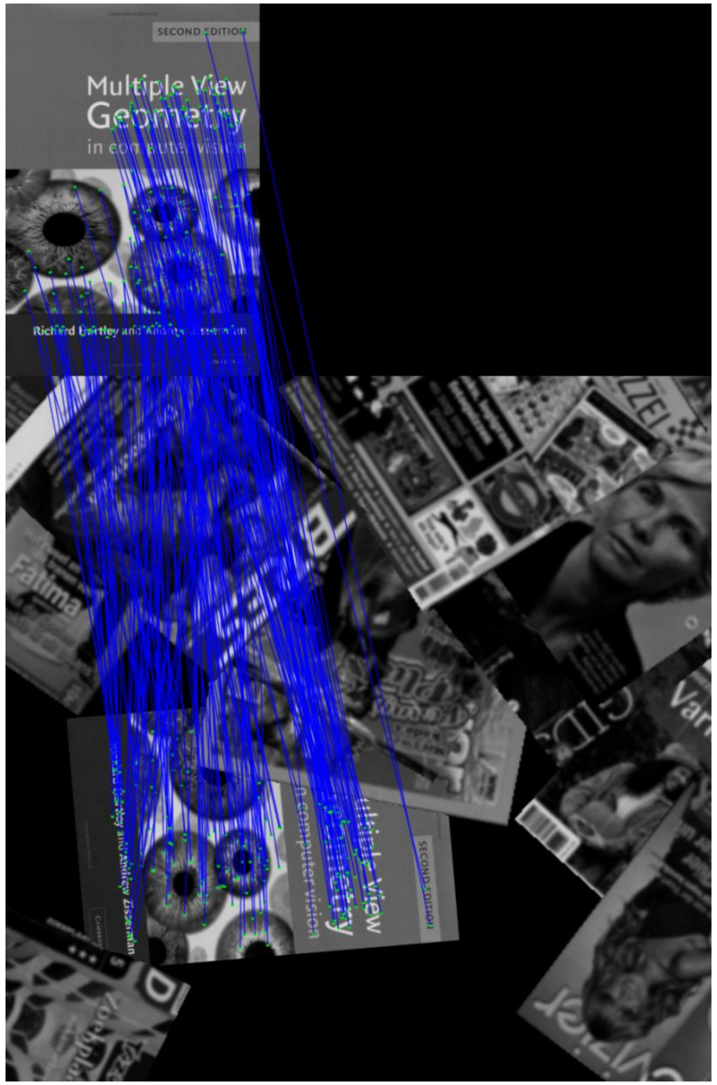
In this example the pose (similarity) space in quantized into a 4D AA with 4×4 bins for the translation, and 3 bins per both rotation and scale.

- **Each correspondence provides a quantized pose (translation, rotation, scale) hypothesis that casts a vote into a unique bin within the 4D AA. Coherent correspondences will cast votes into the very same bin.**
- **Thus, peaks in the 4D AA highlight the most likely pose hypotheses concerning the sought objects. Peaks are then thresholded to decide whether each such hypothesis should be accepted or rejected.**

Example of wrong correspondences filtered out by the GHT

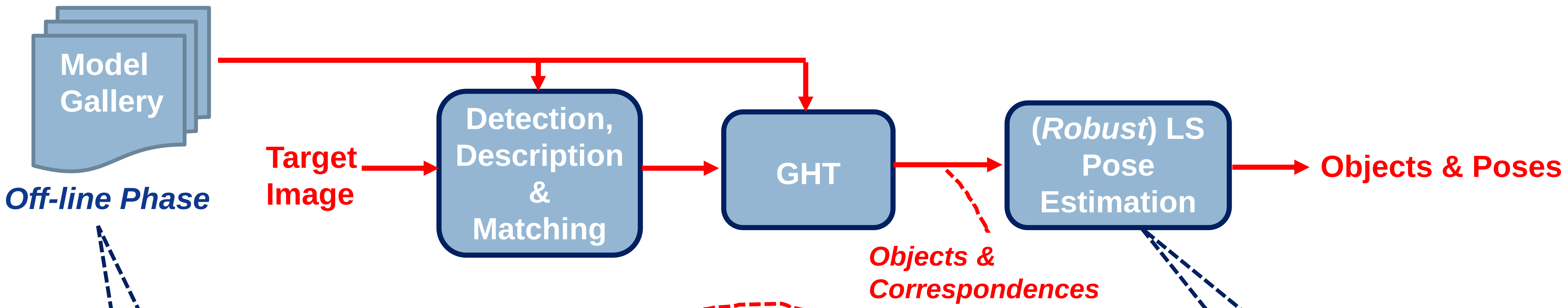
All correspondences, many of which being wrong.

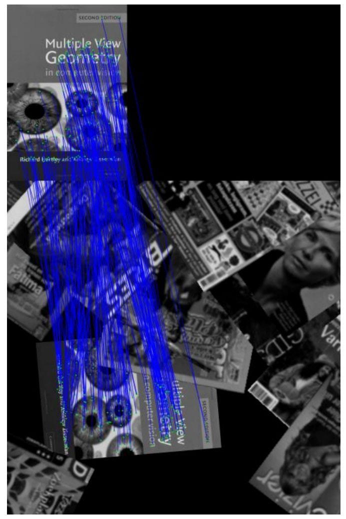




Filtered correspondences: only those associated with votes cast into the main peak of the AA are kept.

Whole instance detection pipeline based on local invariant features and GHT


Model Gallery
Off-line Phase
Target Image
Detection, Description & Matching
GHT
(*Robust*) LS Pose Estimation
Objects & Poses
Objects & Correspondences

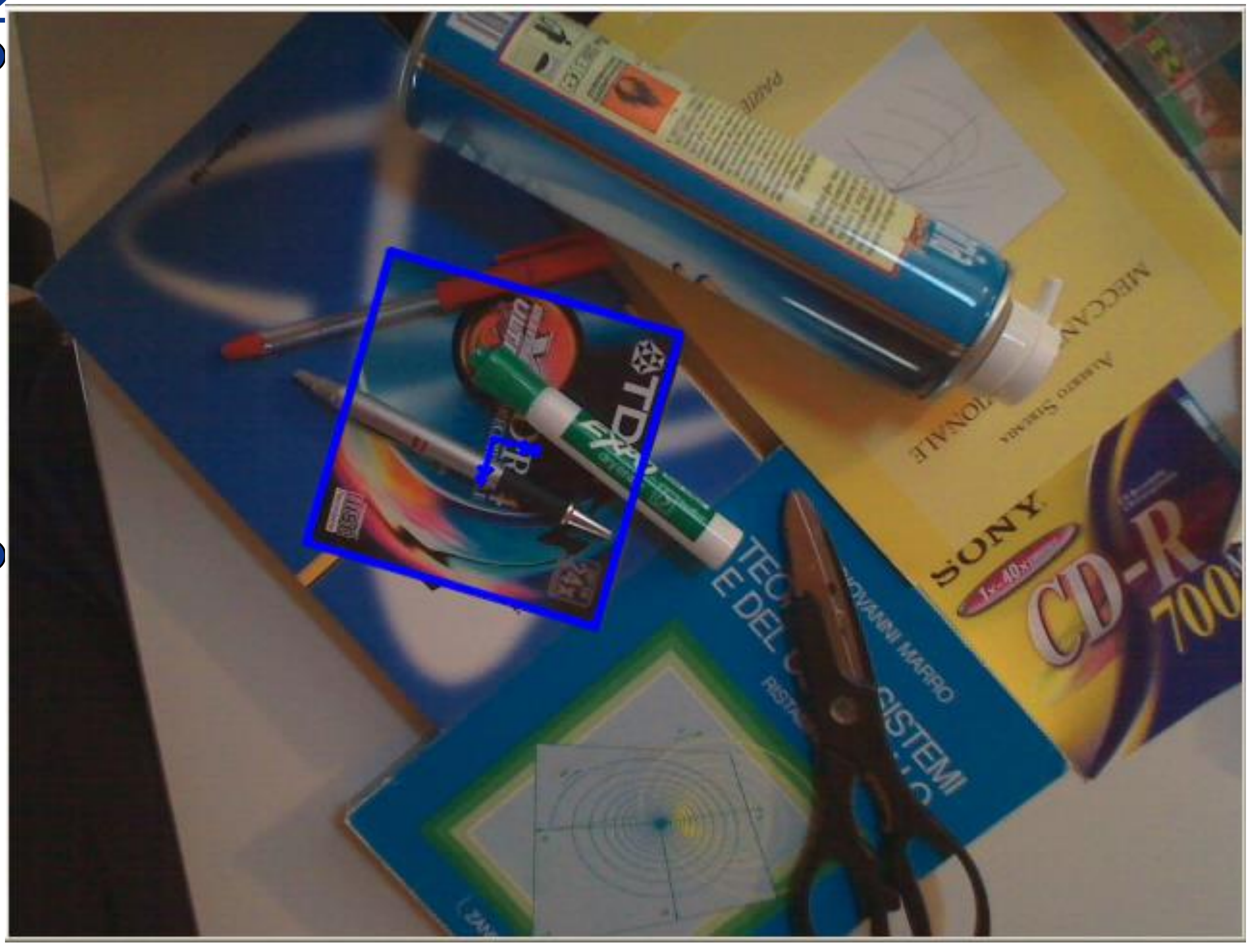


Best efficienly with a single (e.g. BBF) index for all descriptors)

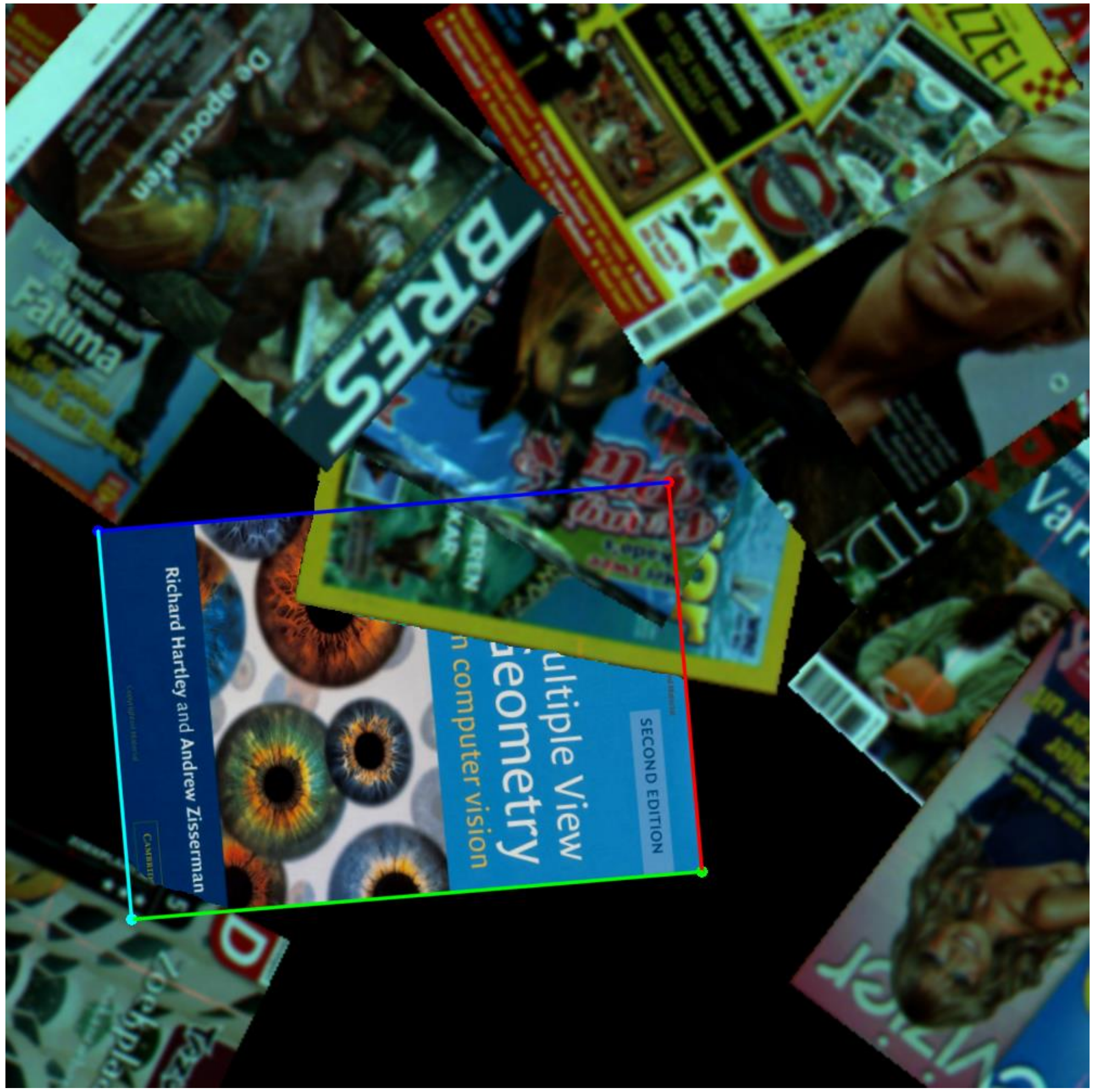
- *Homography*
- ***Similarity***
- *Affine…..*

Appendix 3 ***addresses Least Squares (LS) estimation of a similarity.***

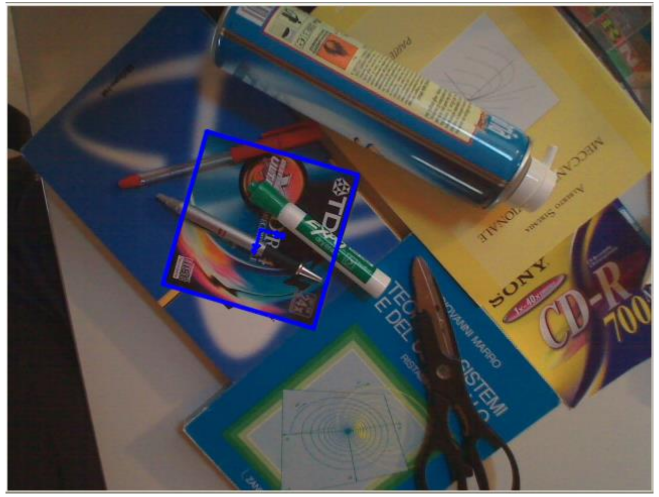
Exemplar Results (1)

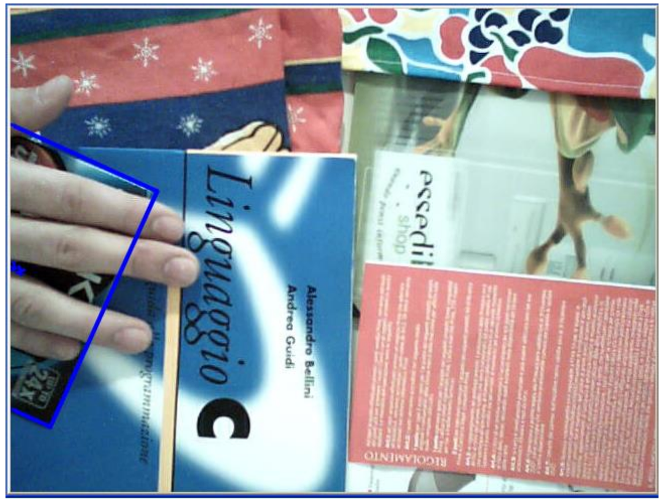


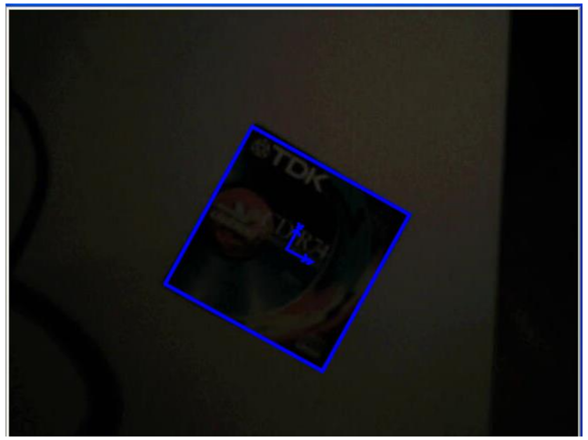
Exemplar Results (2)

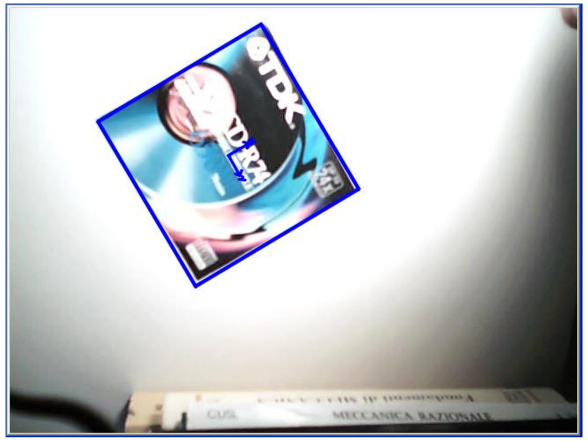


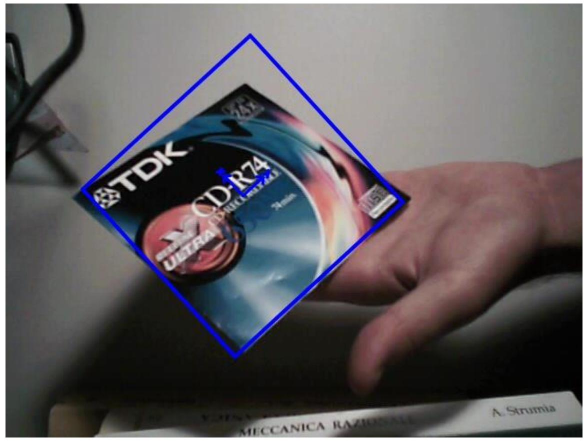
Exemplar Results (3)



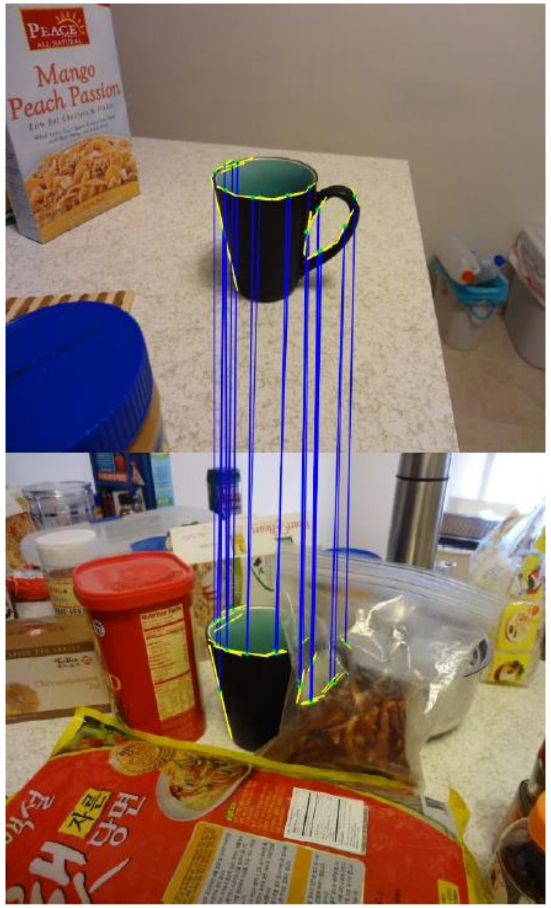


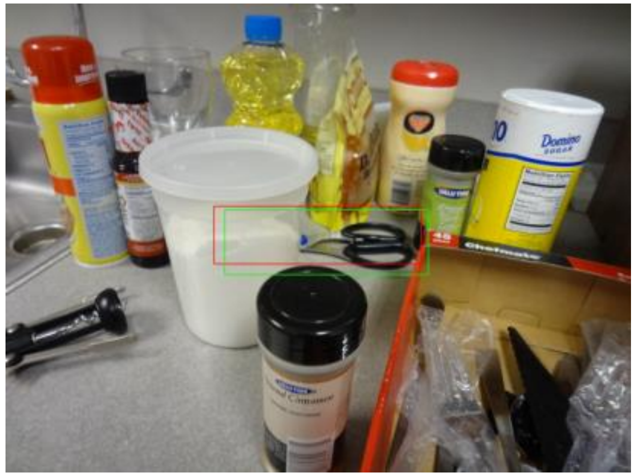






Exemplar Results (BOLD)





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

Appendix 1- A taxonomy of fast template matching approaches

- ***Approximate Methods*: the result is not guaranteed to be the same as would be provided by a full-search:**
  - ***Image Pyramid***

    Full-search at top level, then local refinement throughout successive levels down to the original full-size image.
  - ***Sub-Templates***

    A relatively small sub-template (e.g. a salient region or a set of salient points) is sought first in order to highlight a set of candidate positions, then the full-size comparison is carried out at these positions only.
- ***Exact* (aka *Exhaustive*) *Methods*: the result is guaranteed to be the same as would be provided by a full-search:**
  - ***Call-out* with dissimilarity functions (e.g. SAD-SSD): computation of the function at the current position is terminated as soon as its value gets higher than the current minimum (or a threshold).**
  - **Matching in the Fourier domain (via efficient FFT algorithms).**
  - **Deployment of efficiently computable *bounds* of the (dis)similarity function to skip candidates which cannot improve the current degree of matching.**

Appendix 1 - Template Matching via FFT (1)

- **Due to the Convolution Theorem, template matching can be carried out in the Fourier domain. Indeed, in case of the NCC function:**

Box-filtering

$$NCC(i,j) = \frac{1}{K(i,j)}\left(T(i,j) \circ I(i,j)\right), \quad K(i,j) = \left\|\tilde{I}(i,j)\right\| \cdot \left\|T\right\|$$

Computed off-line once and for all

$$T(i,j) \circ I(i,j) = I(i,j) * T(-i,-j)$$

$$I(i,j) * T(-i,-j) \Leftrightarrow \Im\left(I(i,j)\right) \cdot \Im\left(T(-i,-j)\right) = \Im\left(I(i,j)\right) \cdot \bar{\Im}\left(T(i,j)\right) = I(u,v) \cdot \bar{T}(u,v)$$

IFFT

FFT

$$NCC(i,j) = \frac{1}{K(i,j)} \Im^{-1}\left(I(u,v) \cdot \bar{T}(u,v)\right)$$

FFT

Appendix 1 - Template Matching via FFT (2)

- **As far as the SSD is concerned:**

$$SSD(i,j)=\underbrace{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\big(I(i+m,j+n)-T(m,n)\big)^2}$$

$$\underbrace{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}I(i+m,j+n)^2}_{\left\|\tilde{I}(i,j)\right\|^2}+\underbrace{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}T(m,n)^2}_{+\left\|T\right\|^2}-2\underbrace{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}I(i+m,j+n)\cdot T(m,n)}_{-2\left(T(i,j)\circ I(i,j)\right)}$$

$$-2\big(\underbrace{T(i,j)\circ I(i,j)}\big)$$

$$\overset{\text{IFFT}}{\mathfrak{I}^{-1}}\Big(\overset{\text{FFT}}{I(u,v)}\cdot\underset{\text{FFT}}{\bar{T}(u,v)}\Big)$$

Appendix 1 - Template Matching via FFT (3)

- **Computational complexity of Template Matching in the signal domain:**

$$O(M \cdot N \cdot W \cdot H)$$

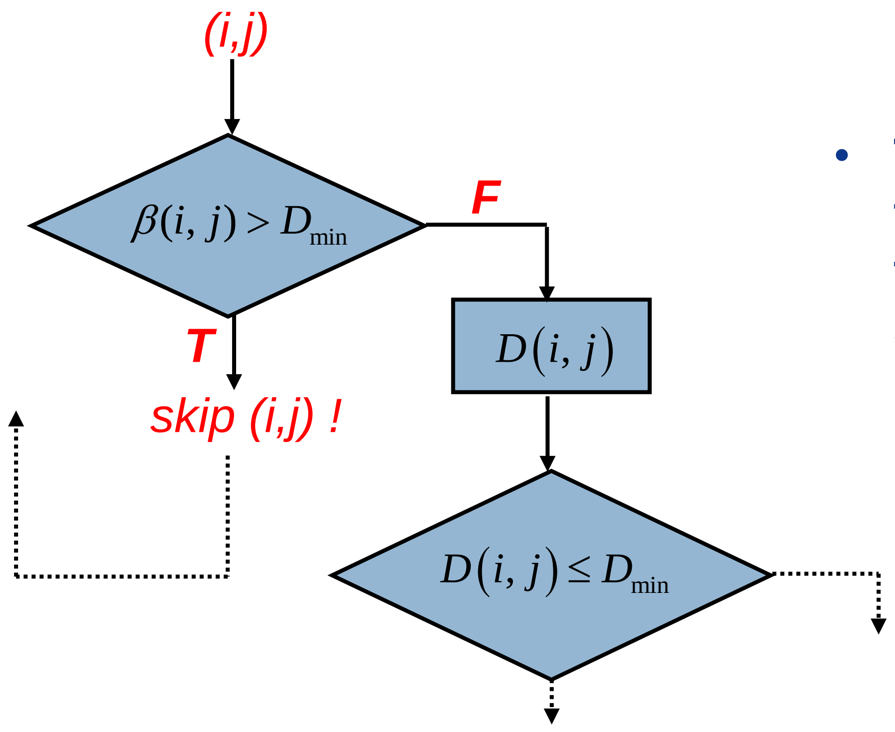
- **Computational complexity of Template Matching in the Fourier domain:**

$$O\Big(\underbrace{W \cdot H \cdot \log_2 (W \cdot H)}_{\text{Independent of template size!}}\Big)$$

- **Usually, Template Matching turns out faster in the Fourier domain when template size is large. Nonetheless, it is worth pointing out that template matching in the Fourier domain requires *floating-point* arithmetic operations which are often exceedingly slow in embedded architectures without FPU (*Floating Point Unit*).**

Appendix 1 - *Bound*-based methods

- Let *D(i,j)* be the *dissimilarity* function to be minimized to detect *T* and let us assume that there exists a *lower bound* of this function: *β(i,j) ≤ D(i,j).*



- The final outcome is the same as it would have been provided by a full-search !

- The approach is faster than a full-search provided that *β (i,j)* can be calculated much more efficiently than *D(i,j)* (*efficient bound*) and it turns out enough accurate to skip the computation of *D(i,j)* at a large number of image positions (*effective bound*).

- In case of template matching based on similarity functions, *β (i,j)* must be an *upper-bound: β(i,j) ≥ S(i,j).*

Efficient and effective *bounds* have been proposed for SAD, SSD, NCC and ZNCC.

Appendix 1 - Successive Elimination Algorithm

$$\mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \Rightarrow \left\|\mathbf{X}\right\|_p = \left(\sum_{i=1}^{N} \left|x_i\right|^p \right)^{1/p} \longrightarrow \left| \left\|\mathbf{X}\right\|_p - \left\|\mathbf{Y}\right\|_p \right| \le \left\|\mathbf{X} - \mathbf{Y}\right\|_p$$

Triangle Inequality:

$$p = 1, \;\; \mathbf{X} = \tilde{I}(i, j), \;\; \mathbf{Y} = T \longrightarrow \left| \left\|\tilde{\mathbf{I}}(i, j)\right\| - \left\|\mathbf{T}\right\| \right| \le \left\|\tilde{\mathbf{I}}(i, j) - \mathbf{T}\right\|$$

$$\underbrace{\left| \overbrace{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \tilde{I}(i+m, j+n)}^{\textit{Box-Filtering}} - \overbrace{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} T(m,n)}^{\textit{Off-line}} \right|} \le \underbrace{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \left| I(i+m, j+n) - T(m,n) \right|}_{\textit{SAD}}$$

$$\beta(i, j) \le SAD(i, j)$$

SEA was originally proposed (Lee & Salari,TIP, 1995) for the *block matching* step carried out to estimate motion in video compression, then it has been used widely also for template matching. An advanced and more effective approach was been proposed later (Tombari, Mattoccia & Di Stefano, PAMI 2009).

Appendix 2 – More info on Shape-based Matching (1)

- **In the original Shape-based matching formulation (Steger, 2002), object’s pose is given by a similiarity (roto-translation plus scale change). The method has then be extended so as to allow estimation of a homography (Hofhauser et.al., 2008).**

- **The search process follows a pyramidal approach, with the number of levels, l_{max}, chosen by the user. In the off-line training phase, the model image (*T*), is repeatedly scaled according to the chosen pyramidal structure.**

- **Given the rotation ($\phi_{min}\ldots\phi_{max}$) and scale variation ($\sigma_{min}\ldots\sigma_{max}$) ranges, all possible rotated and scaled version of the model image are pre-computed for each pyramid level. Steger suggests to rotate and scale the model image and then extract the edges rather than rotate and scale the edges extracted from the model image.**

- **The rotation and scale quantization steps can be either determined automatically (Borgefors, PAMI, 1988) or chosen by the user. At each next pyramid level, the rotation step is doubled.**

Appendix 2 – More info on Shape-based Matching (2)

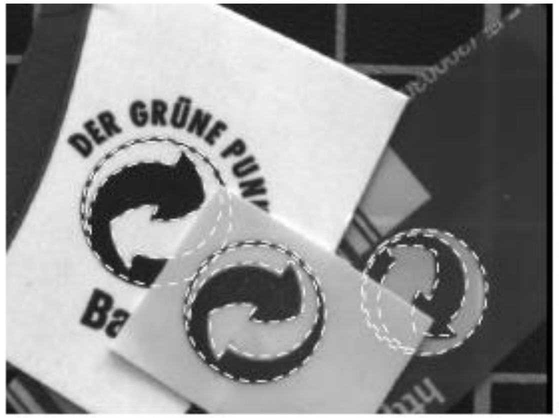
- In the on-line search phase, all the pre-computed scaled and rotated versions of *T* are sought for at the top level of the pyramid by the usual sliding window approach.

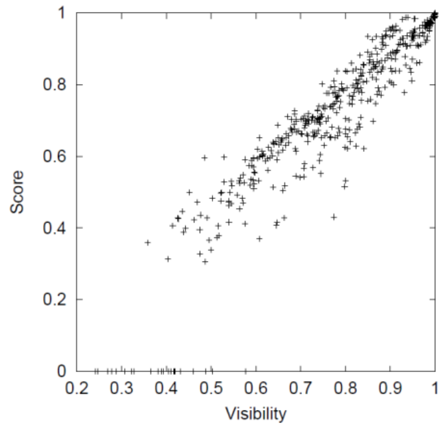
- Candidate matches are obtained by *non-maxima suppression* followed by thresholding (S_{min}) of the similarity function, *S(i,j),* at the found extrema. The previously introduced *call-out* condition is deployed to speed-up the search.

- Candidate matches are then propagated through pyramid levels. At each, a small neighbourhood of the pose estimated at the previous level is explored, so as to either reject the match or validate and propagate it to the next one.

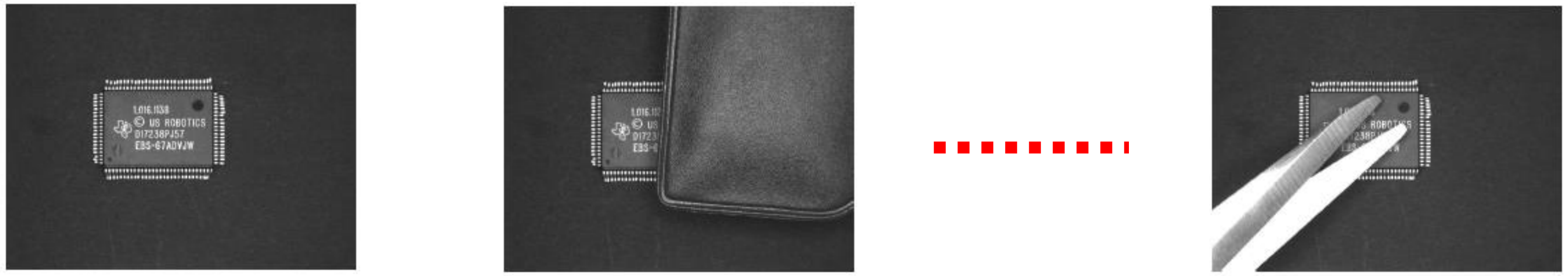
- Once a match is established by a local search in the original image, *I,* the pose estimation resolution is increased by an extrapolation process relying on fitting a second degree polynomial (Steger, PAMI, 1998). This allows translation to be estimated with sub-pixel resolution and scale and rotation with a finer resolution than the chosen quantization steps. Optionally, accuracy may be improved by a final least-squares pose refinement procedure (Steger, 2002).

Appendix 2- Some Results by Shape-based Matching





(a) Score vs. visibility



500 images of an IC characterized by different occlusion levels

Smin=0.3 →: RR (*Recogniton Rate*): 478/500 (95.6%).
If the images featuring an occlusion level higher than > 0.7 are not considered (15 out of 22 *misdetections*), the RR increases to 98.6%.

Appendix 3 – Similarity Estimation (1)

- **Given the *positions* of the feature correspondences validated by the GHT:**

$$\tilde{\mathbf{P}}_i \Leftrightarrow \mathbf{P}_i \;\; i = 1 \ldots n, \quad \tilde{\mathbf{P}}_i = \begin{pmatrix} \tilde{u}_i \\ \tilde{v}_i \end{pmatrix}, \;\; \mathbf{P}_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix}$$

We seek to estimate the similarity (pose) which brings the model image into the target image:

$$\tilde{\mathbf{P}}_i = s\mathbf{R}\mathbf{P}_i + \mathbf{t} \quad \Rightarrow \quad \begin{pmatrix} \tilde{u}_i \\ \tilde{v}_i \end{pmatrix} = \begin{pmatrix} s\cos\vartheta & -s\sin\vartheta \\ s\sin\vartheta & s\cos\vartheta \end{pmatrix} \begin{pmatrix} u_i \\ v_i \end{pmatrix} + \begin{pmatrix} t_u \\ t_v \end{pmatrix}$$

$$m = s\cos\vartheta, \; n = s\sin\vartheta \quad \Rightarrow \quad \begin{pmatrix} \tilde{u}_i \\ \tilde{v}_i \end{pmatrix} = \begin{pmatrix} m & -n \\ n & m \end{pmatrix} \begin{pmatrix} u_i \\ v_i \end{pmatrix} + \begin{pmatrix} t_u \\ t_v \end{pmatrix}$$

Appendix 3 – Similarity Estimation (2)

Let us rewrite the equation by grouping the unknown similarity parameters into a vector

$$\begin{pmatrix} u_i & -v_i & 1 & 0 \\ v_i & u_i & 0 & 1 \end{pmatrix} \begin{pmatrix} m \\ n \\ t_u \\ t_v \end{pmatrix} = \begin{pmatrix} \tilde{u}_i \\ \tilde{v}_i \end{pmatrix}$$

and then set-up an overdetermined (2*n* equations in 4 unknowns) linear system by considering all the available correspondences.

$$\begin{pmatrix} u_1 & -v_1 & 1 & 0 \\ v_1 & u_1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ u_n & -v_n & 1 & 0 \\ v_n & u_n & 0 & 1 \end{pmatrix} \begin{pmatrix} m \\ n \\ t_u \\ t_v \end{pmatrix} = \begin{pmatrix} \tilde{u}_1 \\ \tilde{v}_1 \\ \vdots \\ \tilde{u}_n \\ \tilde{v}_n \end{pmatrix} \Rightarrow \mathbf{Ax} = \mathbf{b}$$

Appendix 3 – Similarity Estimation (3)

The previous system admits a least-square solution, which can be computed either by the *normal equations* or the *pseudo-inverse*:

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

- **Once (*m*,*n*) are found, rotation (ϑ) and scale (*s*) may be easily disentangled :**

$$\hat{\vartheta} = \tan^{-1}\left(\frac{n}{m}\right), \ \hat{\vartheta} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

$$\hat{\vartheta} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \Rightarrow \vartheta \in [0, 2\pi] \ :$$

$$\hat{\vartheta} \in \left[0, \frac{\pi}{2}\right] \Rightarrow \vartheta = \hat{\vartheta} + sign(m)\cdot\pi, \quad \hat{\vartheta} \in \left[-\frac{\pi}{2}, 0\right[\Rightarrow \vartheta = \hat{\vartheta} + 2\pi - sign(m)\cdot\pi$$

$$\text{with } sign(m) = \begin{cases} 1, & \text{if } m < 0 \\ 0, & \text{otherwise} \end{cases}$$

$$s = \frac{m}{\cos\vartheta} = \frac{n}{\sin\vartheta}$$

Main References (1)

1) Martin, J. and Crowley, J. (1995). *Experimental comparison of correlation techniques*. In Proc. Int. Conf. On Intelligent Autonomous Systems, volume 4, pages 86–93.
2) W. Li and E. Salari, *Successive elimination algorithm for motion estimation*, IEEE Trans. on Image Processing, vol. 4, no. 1, pp. 105–107, 1995
3) F. Tombari, S. Mattoccia, L. Di Stefano, *“Full search-equivalent pattern matching with Incremental Dissimilarity Approximations”,* IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 31, N.1, pp. 129-141, Jan. 2009
4) Steger, C, *Occlusion, clutter, and illumination invariant object recognition*. In Kalliany, R., Leberl, F., eds.: International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences. Volume XXXIV, part 3A., Graz (2002).
5) A. Hofhauser, C. Steger, N. Navab, *Edge-based Template Matching and Tracking for Perspectively Distorted Planar Objects*, Proc. 4th International Symposium on Advances in Visual Computing ISVC '08, 2008.
6) Borgefors, G., *Hierarchical chamfer matching: A parametric edge matching algorithm.* IEEE Trans on Pattern Analysis and Machine Intelligence 10(6), pp. 849–865, 1988.
7) Steger, C., *An unbiased detector of curvilinear structures*. IEEE Trans. on Pattern Analysis and Machine Intelligence 20(2), pp. 113–125, 1998.
8) D.H. Ballard, *Generalizing the Hough Transform to Detect Arbitrary Shapes*, Pattern Recognition, Vol.13, No.2, p.111-122, 1981
9) GHT on Wikipedia: https://en.wikipedia.org/wiki/Generalised_Hough_transform

Main References (2)

10) M. Fischler, R. C. Bolles. *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*" Comm. ACM. 24 (6): 381–395, 1981.
11) RANSAC on Wikipedia https://en.wikipedia.org/wiki/Random_sample_consensus
12) D. A. Forsyth, Jean Ponce *Computer Vision, a modern approach*. Prentice Hall. ISBN 978-0-13-085198-7, 2003
13) C. Steger, M. Ulrich, C. Wiedemann, "Machine Vision Algorithms and Applications – 2nd Edition", Wiley-VCH, Weinheim, 2018.
14) R. Szeliski *Computer Vision: Algorithms and Applications*", 2nd Edition, Springer, 2021. https://szeliski.org/Book/