University of Bologna

Blob Analysis

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

Introduction (1)

- **Once foreground/background segmentation has been accomplished, in most applications the next task deals with analysis of the individual foreground objects to achieve some kind of high-level knowledge on the observed scene.**

- **Such knowledge may pertain detecting what types of objects are contained in the scene, measuring their position and orientation, assessing whether objects show or not manufacturing defects or inaccuracies ….**

- **Accordingly, individual objects, usually referred to as *blobs* (Binary Large Objects) or *regions*, first need to be isolated within the overall foreground region. As already pointed out, this step is known as *connected components labeling*.**

- **Then, individual objects can be processed to extract specific *features* related to the required kind of high-level knowledge. For example, several features may be computed to determine the *shape* of objects, so as to detect whether one or more specific types of objects to be picked-up by a robot are present in the observed scene. For the robot to pick detected objects, other *features* related to their *position* and *orientation* need typically to be computed.**

Introduction (2)

- **The process whereby features are extracted from blobs is often referred to as *Blob Analysis*. Features may be computed either from all the pixels belonging to the blob (*region features*) or from boundary pixels only (*boundary* or *contour features*). A plethora of diverse features has been proposed in literature: we will introduce here just a few among those most widely used.**

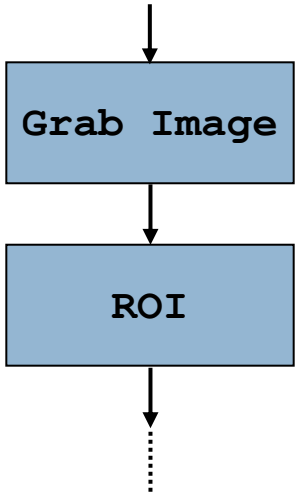
- ***Shape*-related features need to exhibit invariance to the transformations the image of the object may undergo in the addressed settings. Mostly, shape-features are required to fulfil invariance to a similarity transformation (translation, rotation and scale change), i.e. need not to change if the object appears at a different position or rotated or shows a different size in the image.**

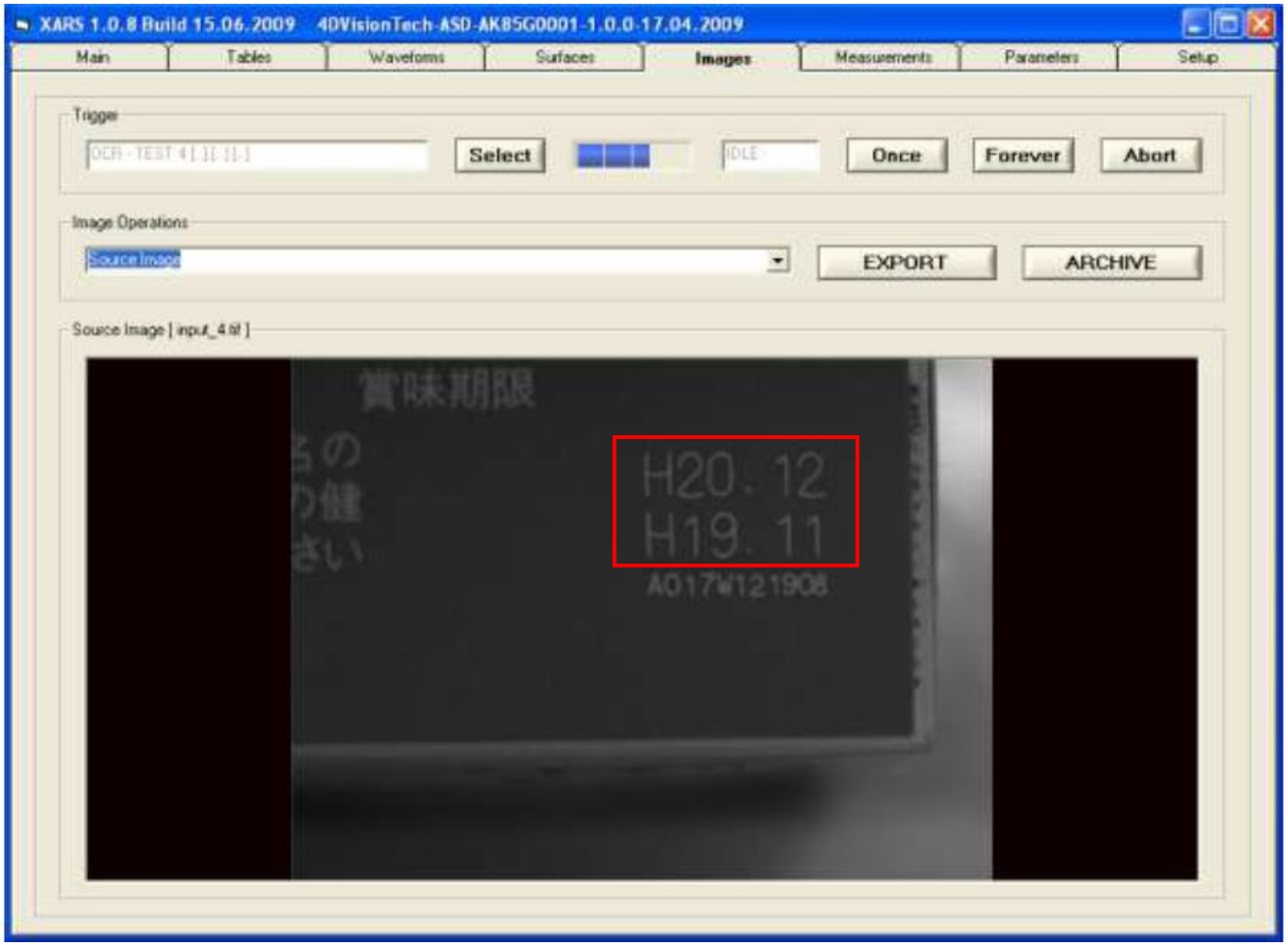
- **Object detection based on blob features can be accomplished either by handcrafted rules or based machine learning techniques. With the latter approach, a classification function to map a feature vector into a set of object classes is learned from training samples.**

Example: Text Analysis (Quality Control and OCR)

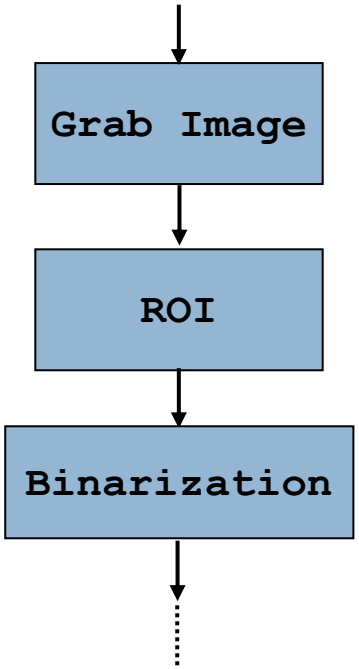
High-level Task:

- Quality Control (H,W, Skew, Spacing)
- OCR

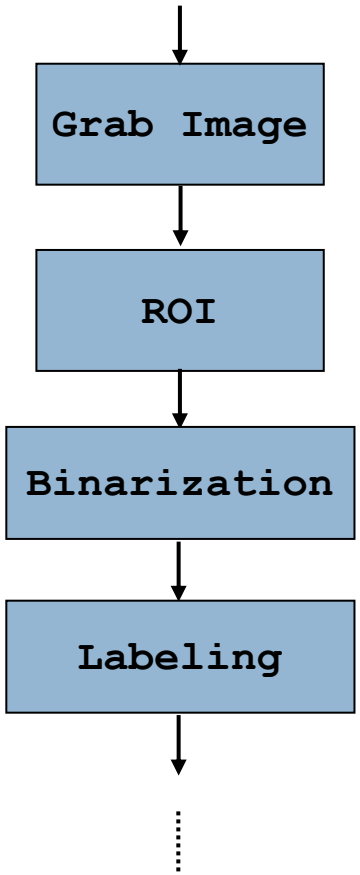


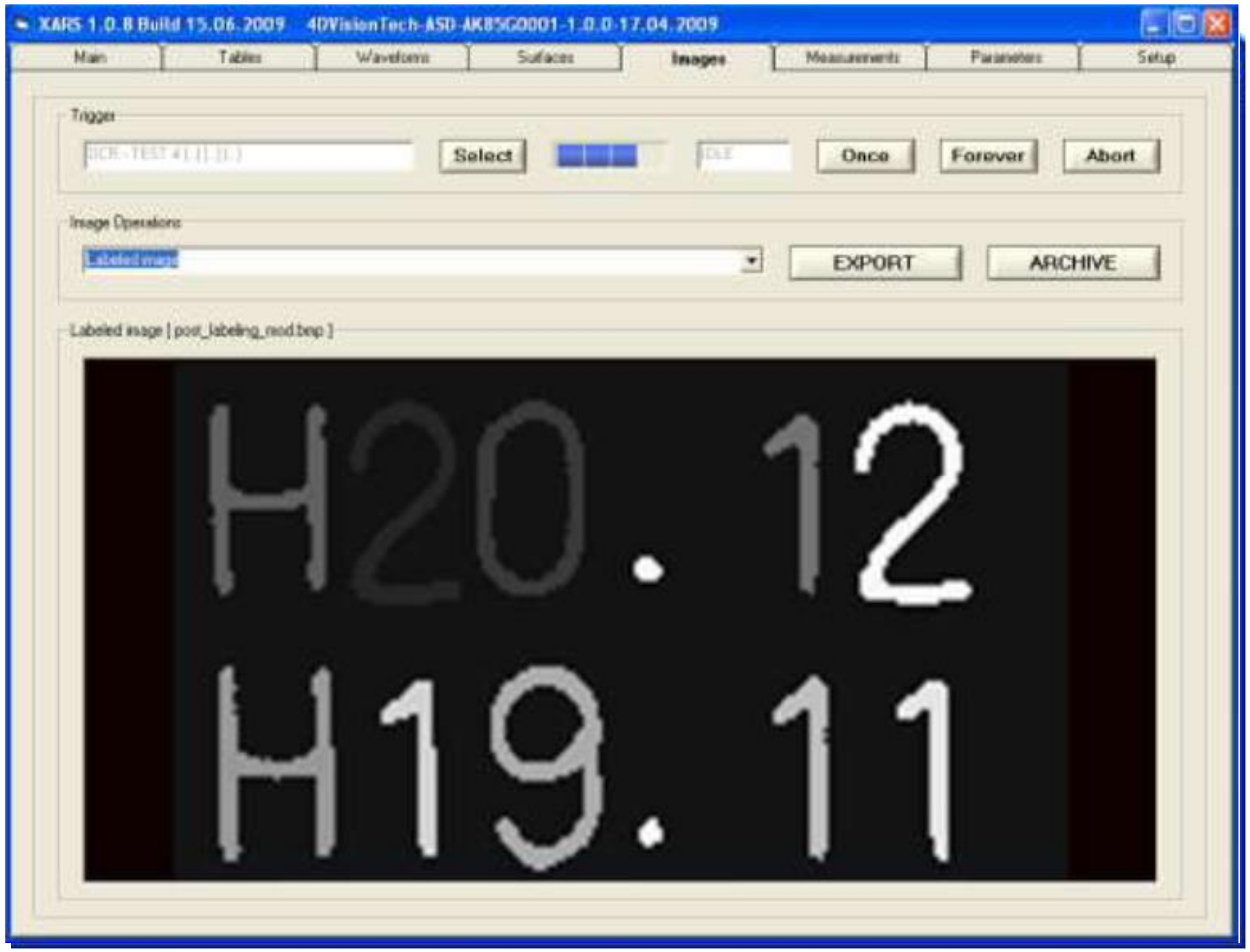


Binary Image……

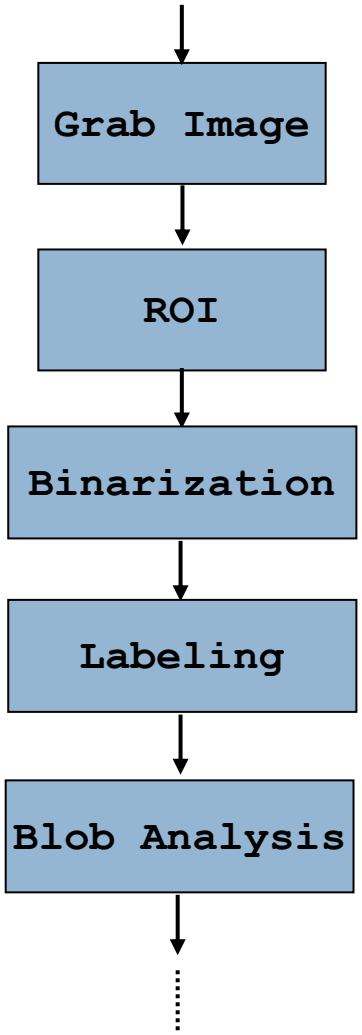


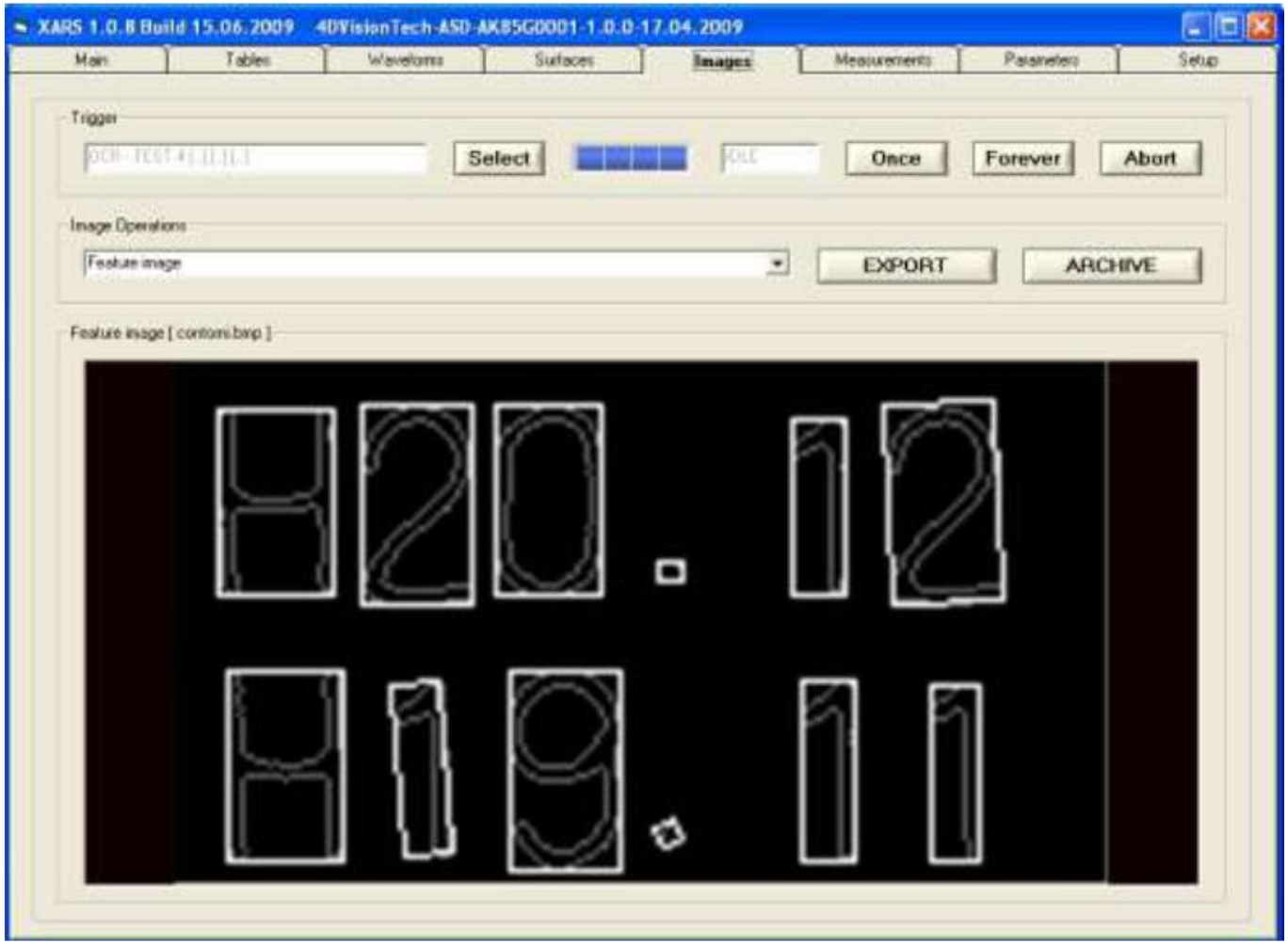
……Labeled Image





Quality Control by Blob Analysis





Distances and Connectivity (1)

- **To discuss algorithms aimed at labeling the connected components of a binary image, first we need to introduce the notion of connectivity, which in turn is related to that of *distance* on the discrete plane, E^2.**
  - Given $p_1(i_1, j_1), p_2(i_2, j_2), p_3(i_3, j_3) \in E^2$, D is a distance iff:

$$\begin{aligned} D(p_1, p_2) &\geq 0,\ D(p_1, p_2) = 0 \Leftrightarrow p_1 = p_2 \\ D(p_1, p_2) &= D(p_2, p_1) \\ D(p_1, p_3) &\leq D(p_1, p_2) + D(p_2, p_3) \end{aligned}$$

  The two main discrete distances defined in E^2 are defined in the following.

  - The city-block distance, D_4, between p_1 e p_2 is given by:

$$D_4(p_1, p_2) = |i_1 - i_2| + |j_1 - j_2|$$

  According to the above definition, a pixel can be reached from another through horizontal or vertical shifts only.

Distances and Connectivity (2)

The set of points having distance $\leq r$ from a given one is a rhombus with diagonals of length $2r+1$, e.g with $r=2$:

$$\begin{array}{ccccc} & & 2 & & \\ & 2 & 1 & 2 & \\ 2 & 1 & 0 & 1 & 2 \\ & 2 & 1 & 2 & \\ & & 2 & & \end{array}$$

Defined as *neighbours of p* the set of points having $D=1$ from p, it follows that, using $D=D_4$, such a set is given by:

$$\begin{array}{ccc} & n & \\ n & p & n \\ & n & \end{array}$$

and is usually called 4-neighbourhood of p (hereinafter also denoted as $n_4(p)$).

Distances and Connectivity (3)

- The chessboard distance, D_8, between p_1 and p_2 is given by:

$$D_8(p_1, p_2) = max\left(|i_1 - i_2|, |j_1 - j_2|\right)$$

According to this distance, horizontal or vertical and diagonal shifts have the same weight.

The set of points having distance $\leq r$ from a given one is a square with side of length $2r + 1$, e.g with $r = 2$:

$$\begin{matrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{matrix}$$

The set of neighbours of p such as $D_8 = 1$ is called 8-neighbourhood of p (hereinafter also denoted as $n_8\left(p\right)$):

$$\begin{matrix} n & n & n \\ n & p & n \\ n & n & n \end{matrix}$$

Connected Components of a Binary Image

- A path of length n from pixel p to pixel q is a sequence of pixels $p = p_1, p_2, \ldots p_n = q$ such as p_i and p_{i+1} are neighbours according to the chosen distance. Both $D_4(p, q)$ and $D_8(p, q)$ have the same length as the smallest path between p and q.

- A set of pixels, R, is said to be a *connected* region if for any two pixels p, q in R there exists a path contained in R between p and q. Again, depending on the chosen distance being either D_4 or D_8, R is said to be a *4-connected* or *8-connected* region respectively.

- A set of pixels is said to be a *connected* foreground (background) region if it is a connected region and includes foreground (background) pixels only.

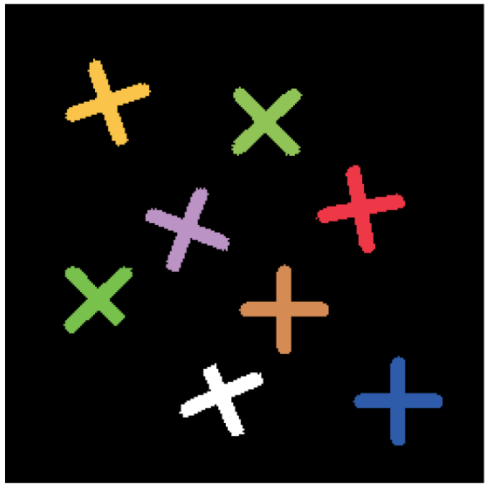
A *connected component* of a binary image is a <u>maximal</u> connected foreground region.

Connected Components Labeling

- **Computation whereby pixels belonging to different connected components are given different labels, while background pixels are left unaffected (e.g. they may keep the previous label or, equivalently, a new one)**

Labeling output displayed as a gray-scale image.

Labeling output displayed by (pseudo) colours



The classical 2-scans algorithm

- **By the first scan foreground pixels take temporary labels based on those given to already visited neighbours, which depend on both the chosen distance, e.g. D_4, and the scan order, e.g. *left-right, top-down*.**

- **Upon the first scan, different blobs have certainly been given different labels, though, depending on shape, this may be the case for connected parts of a single blob too.**

- **Hence, the second scan allows a unique final label to be assigned to those parts belonging to the same blob that had been given different temporary labels by the first scan.**

- **Purposely, *equivalent* temporary labels need to be found between the two scans, so as to assign a unique final label to each of the *equivalence classes* among temporary labels.**

First scan (1)

4-connectivity !

$l_q = l_p = B \rightarrow l_x = Newlabel$

$l_q \neq B,\ l_p = B \rightarrow l_x = l_q$

$l_q = B,\ l_p \neq B \rightarrow l_x = l_p$

$l_q = l_p \neq B \rightarrow l_x = l_q \text{ or } l_x = l_p$

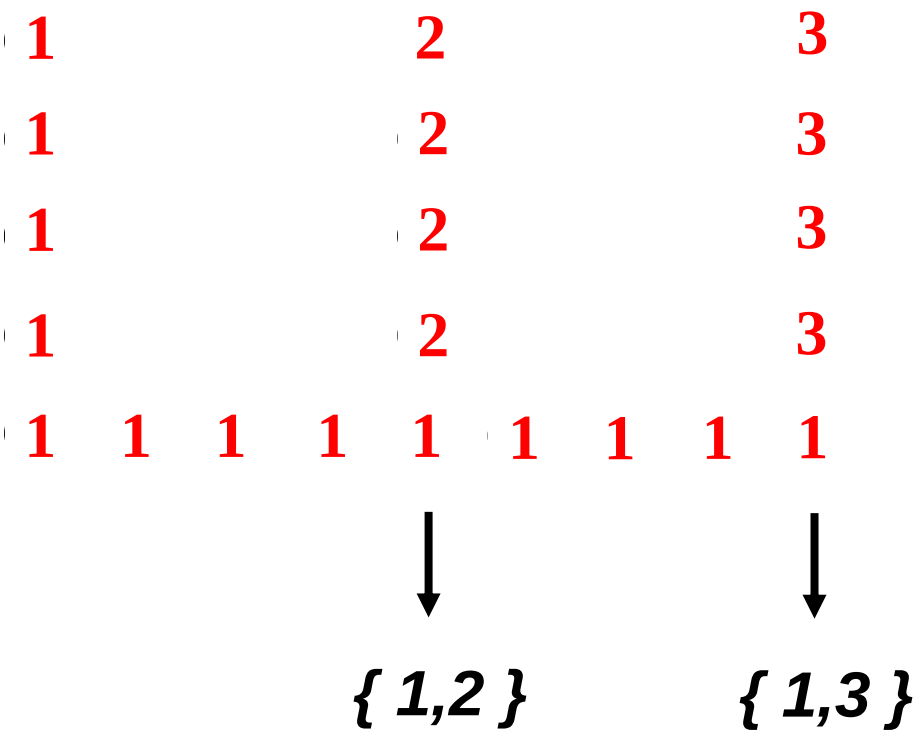
$l_q \neq B,\ l_p \neq B,\ l_q \neq l_p \rightarrow l_x = l_q \text{ or } l_x = l_p,\ \{ l_q, l_p \}$

lq and *lp* are equivalent labels !

First scan (2)

```
// lp,lq,lx: labels assigned to p,q,x
// B:background, F:foreground.
// FIRST SCAN:
for(i=1; i<NROWS-1; i++)
for(j=1; j<NCOLS-1; j++) {
if (I[i,j]==F) {
lp = I[i-1,j];
lq = I[i,j-1];
if(lp == B && lq == B) {
NewLabel++;
lx = NewLabel;}
else if((lp != lq)&&(lp != B)&&(lq != B)){
// REGISTER EQUIVALENCE (lp,lq)
lx = lq;}
else if(lq != B) lx = lq;
else if(lp != B) lx = lp;
I[i,j] = lx;} }
// FIND EQUIVALENCE CLASSES
// SECOND SCAN
```

Example



After the first scan: $\{1,2\}, \{1,3\} \longrightarrow \{1,2,3\} \longrightarrow \{1,2,3\}$

Handling equivalences (1)

1. **During the first scan the equivalences found between temporary label pairs are recorded into an $N \times N$ matrix (N: number of temporary labels):**

$$B = \begin{array}{c} \\ 1 \\ i \\ N \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & \cdots\cdots & j & \cdots\cdots & N \end{array} \\ \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \end{array} \Rightarrow B[i,j] = \begin{cases} 1, \; if \; \{i,j\} \\ 0, \; otherwise \end{cases}$$

Because label equivalence is an equivalence relation:

$$i = j \rightarrow B[i, j] = 1$$

$$B[i, j] = B[j, i] \;\; \forall \, i, j = 1 \ldots N \quad \Rightarrow \quad \textbf{B is symmetric}$$

Handling equivalences (2)

2. **After the first scan, B is processed so as to elicit the equivalences among temporary labels implied by virtue of the transitive property:**

$$\{i, j\} \rightarrow \{i, k\} \ \forall \{k, j\}$$

$$B[i,k] = B[i,k] \ OR \ B[j,k] \ \forall \ k = 1...N$$

Thus, through a scan by columns, all the equivalences implied by the transitive property can be recorded into B :

```
for (j=1,....N)
 for (i=1....N)
  if (B[i,j]=1)AND(i≠j)
   for(k=1....N)
     B[i,k]=B[i,k] OR B[j,k];
```

Handling equivalences (3)

For the sake of example, let us assume the following equivalences to have been found during the first scan:

$$\{1,2\}, \{4,5\}, \{2,6\}$$

	1	2	3	4	5	6
1	1	1				
2	1	1				1
3			1			
4				1	1	
5				1	1	
6		1				1

	1	2	3	4	5	6
1	1	1				1
2	1	1				1
3			1			
4				1	1	
5				1	1	
6	1	1				1

After the computation, B contains the information related to equivalence classes, a row recording a "1" in each column associated to a label belonging to the same equivalence class as the row.

Handling equivalences (4)

3. **Having found equivalence classes, it is now necessary to assign a unique final label to each of them. To this purpose, a simple table initialized by the identity function may be conveniently deployed:**

$$\mathbf{LUT} = \begin{matrix} 1 \\ \vdots \\ i \\ \vdots \\ N \end{matrix} \begin{bmatrix} 1 \\ \vdots \\ i \\ \vdots \\ N \end{bmatrix}$$

```
for (i=1,....N)
 for (j=1....N)
  if (B[i,j]=1)AND(i≠j){
       LUT[j]=i;
       for(k=1....N)
       B[j,k]=0;}
```

4. **Accordingly, the second scan boils down to an image look-up operation by the table:**

```
for (i=1....NROWS-1)
 for (j=1....NCOLS-1)
      I[i,j]=LUT[I[i,j]];
```

Example

Back to the previous example:

B

	1	2	3	4	5	6
1	1	1				1
2	1	1				1
3			1			
4				1	1	
5				1	1	
6	1	1				1

LUT

1	1
2	2
3	3
4	4
5	5
6	6

Example

Back to the previous example:

B

	1	2	3	4	5	6
1	1	1				1
2						
3			1			
4				1	1	
5				1	1	
6	1	1				1

LUT

1	1
2	1
3	3
4	4
5	5
6	6

Example

Back to the previous example:

B

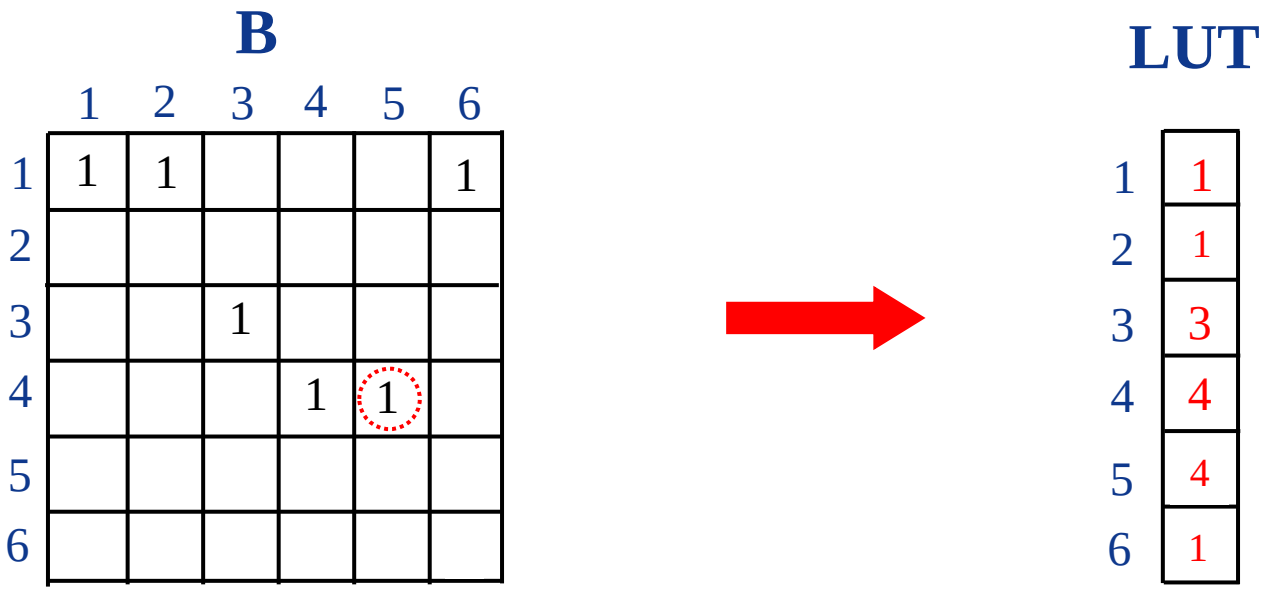
	1	2	3	4	5	6
1	1	1				1
2						
3			1			
4				1	1	
5				1	1	
6						

LUT

1	1
2	1
3	3
4	4
5	5
6	1

Example

Back to the previous example:



Further straightforward manipulation of the LUT before the second scan would allow final labels to be consecutive numbers (i.e. 1,2,3…)

Handling equivalences during the first scan

- **A simple and efficient variation of the classical two-scan algorithm allows handling equivalences directly during the first scan (*Di Stefano, Bulgarelli "A Simple and Efficient Connected Components Labeling Algorithm", ICIAP 1999*).**

- **Efficiency comes from equivalence classes being always up-to-date during the first scan, so that newly detected equivalences already implied by virtue of the transitive property need not be handled. Indeed, checking for equivalences occurs in the equivalence rather than label domain: one thus needs to do nothing whenever two conflicting labels are already known to belong to the same equivalence class.**

- **Handling of equivalences relies on a simple 1D array (referred to as *Class Array*), which maps each label onto its equivalence class, and is then used as a LUT to carry out the second scan.**

Key Idea

```
// lp,lq,lx: labels assigned to p,q,x
// B:background, F:foreground.
// FIRST SCAN:
for(i=1; i<NROWS-1; i++)
for(j=1; j<NCOLS-1; j++) {
if (I[i,j]==F) {
lp = I[i-1,j];
lq = I[i,j-1];
if(lp == B && lq == B) {
NewLabel++;
lx = NewLabel;}
else if((lp != lq)&&(lp != B)&&(lq != B)){
// REGISTER EQUIVALENCE (lp,lq)
lx = lq;}
else if(lq != B) lx = lq;
else if(lp != B) lx = lp;
I[i,j] = lx;} }
// FIND EQUIVALENCE CLASSES
// SECOND SCAN
```

Handling the equivalence: if the two *labels* are equivalent their classes are merged

$$\mathbf{C} = \begin{matrix} 1 \\ \vdots \\ i \\ \vdots \\ N \end{matrix} \begin{bmatrix} 1 \\ \vdots \\ i \\ \vdots \\ N \end{bmatrix}$$

$$\mathbf{C} = \begin{matrix} \\ \vdots \\ lp \\ \vdots \\ \\ \end{matrix} \begin{bmatrix} \\ \vdots \\ lq \\ \vdots \\ \\ \end{bmatrix}$$

Implementation

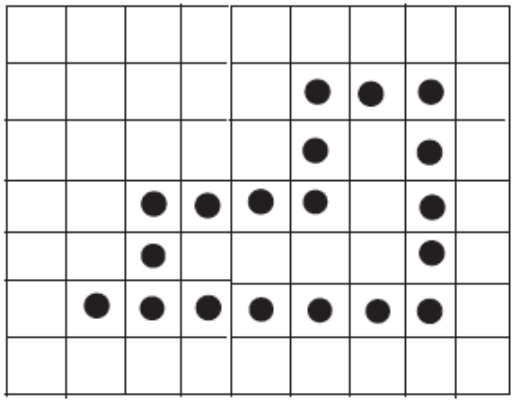
- **Class merging: if *lp* is equivalent to some other labels, all such labels should be marked as equivalent to *lq* (transitive property).**

```
for (k=1;k<=NewLabel;k++)
if (C[k]==lp) C[k]=lq;
```

- **Checking whether two labels are equivalent takes place within the class rather than label domain, so as to avoid wasting computation to handle equivalences already implied by previous ones due to the transitive property.**

```
if((C[lp]!=C[lq])&&(lp!=B)&&(lq!=B))
{C_lp = C[lp];
for(k=1; k<=Newlabel; k++)
if (C[k] == C_lp) C[k]=C[lq];
}
```

Example



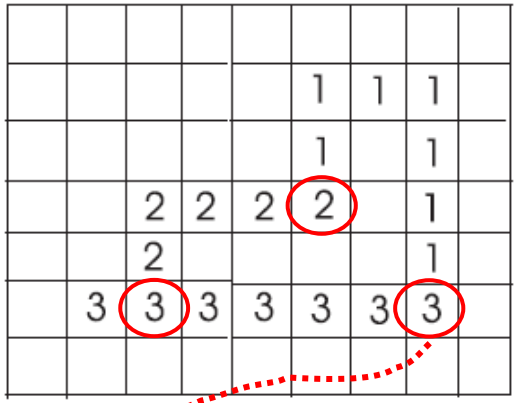


Image	Classical Algorithm	Proposed Algorithm
cel4	281	78
fun1	2527	1318
hnd1	742	473
hse4	262	124
mon1	6404	224
pcb1	1025	138
pcb2	2449	260
rods	1158	88
chocs	2390	103
son3	1317	425
tls1	7222	224
txt2	380	198
txt3	168	138
wrm1	1483	221

Table 1. Number of conflicts handled by the labeling process.

Not handled because it is already implied by previous equivalences !

$$\mathbf{C} = \begin{matrix} 1 \\ 2 \\ 3 \end{matrix}\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \Rightarrow \mathbf{C} = \begin{matrix} 1 \\ 2 \\ 3 \end{matrix}\begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix} \Rightarrow \mathbf{C} = \begin{matrix} 1 \\ 2 \\ 3 \end{matrix}\begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}$$

Area and Barycentre

• **As already pointed out, once *regions* have been identified, usually the next step consists in computing *features* associated with each region. In the following we will introduce several feature types, starting from very simple ones, such as area (A) and barycentre (B):**

$$A = \sum_{p \in R} 1$$

The area just amounts to the number of pixels belonging to the region.

$$p = \begin{bmatrix} i \\ j \end{bmatrix} \rightarrow B = \begin{bmatrix} i_b \\ j_b \end{bmatrix} : \; i_b = \frac{1}{A} \sum_{p \in R} i; \; j_b = \frac{1}{A} \sum_{p \in R} j$$

The barycentre is the centre of mass of the region, which is seen as a set of particles of unitary mass.

Perimeter (1)

- **The perimeter is defined as the length of the contour of the region. To measure such a length, we need to define first which pixels of the region belong to its contour.**

- **A pixel, *p*, belonging to a region is said to belong to the contour of the region if there exists at least one background pixel, *q*, between its neighbours. Accordingly, we end up with defining a different contour (C_4 or C_8) based on the adopted neighbourhood (*4- or 8-neighbourhood*):**

$$\exists q \in n_4(p) \rightarrow p \in C_4$$
$$\exists q \in n_8(p) \rightarrow p \in C_8$$

It can be easily observed that C_4 is a 8-connected curve while C_8 is 4-connected.

- **The perimeter can then be estimated as follows:**

$$P_8 = \sum_{p \in C_4} 1$$

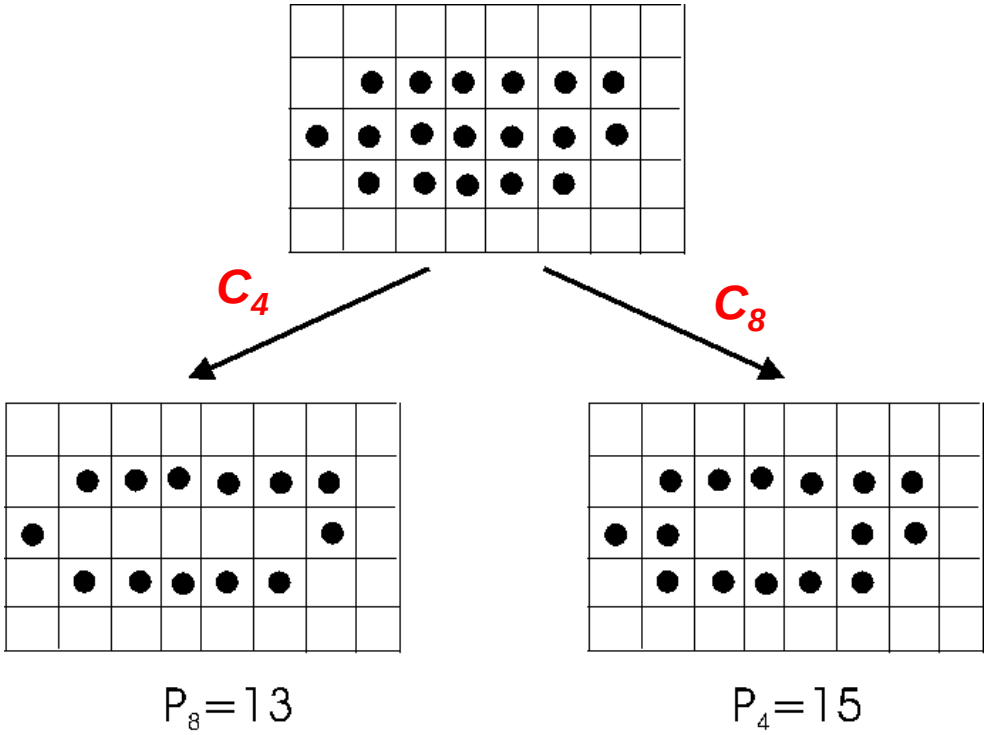
Length of the 8-connected contour C_4

$$P_4 = \sum_{p \in C_8} 1$$

Length of the 4-connected contour C_8

Perimeter (2)

- P_8 tends to underestimate the "true" length, P_4 to overestimate it:

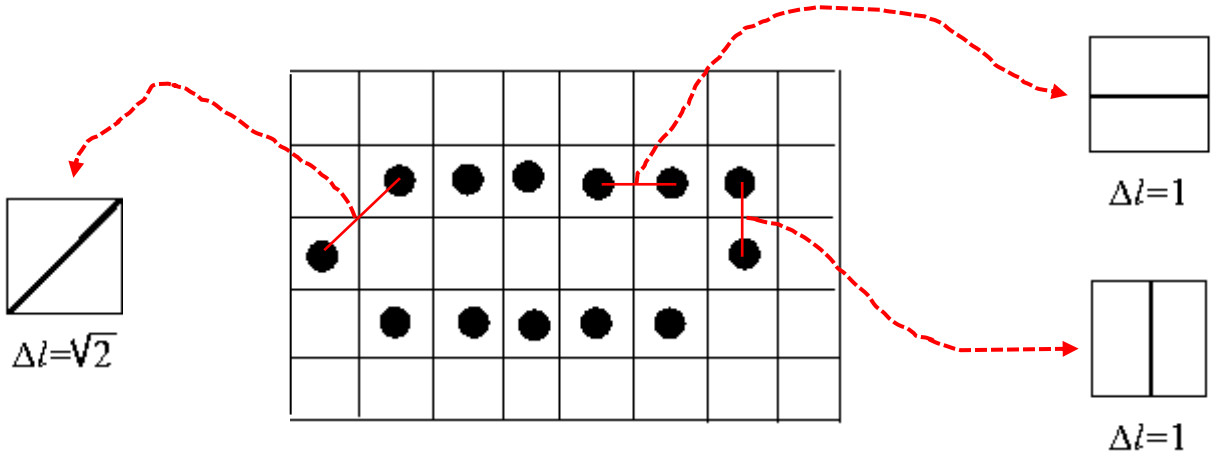


So that one might wish to average the two measurements when aiming at a more accurate approximation:

$$\tilde{P} = \frac{P_4 + P_8}{2}$$

Perimeter (3)

• **Considering C_4 (i.e. the *8-connected* contour), a more accurate approximation of the perimeter can be obtained by taking into account whether the "ideal" curve would better join two nearby pixels through a horizontal/vertical segment or a diagonal one:**



Accordingly, once a scan order around the contour has been defined:

$$C_4 = \{p_1, p_2 \dots p_m\}$$

$$\tilde{P}_8 = \sum_{p_k : p_{k+1} \in n_4(p_k)} 1 + \sum_{p_k : p_{k+1} \in (n_8(p_k) - n_4(p_k))} \sqrt{2}$$

Compactness (aka *Form Factor*)

- **Simple shape-feature defined as follows:**

$$C = \frac{P^2}{A}$$

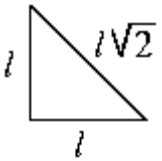
Squared perimeter divided by area. In case of continuous 2D shapes it takes the minimum value (i.e. 4 π) for a circle.

It is a dimensionless quantity, and, as such, a *scale-invariant* feature. Two further examples:

Square with side of length l:

$$C = \frac{(4l)^2}{l^2} = \frac{16l^2}{l^2} = 16$$

Isosceles right triangle:



$$C = \frac{\left(l\left(2+\sqrt{2}\right)\right)^2}{\frac{l^2}{2}} = \frac{l^2\left(2+\sqrt{2}\right)^2}{\frac{l^2}{2}} = 2\left(2+\sqrt{2}\right)^2 \cong 23.3$$

Haralick's Circularity

• **In the discrete domain, compactness takes its smallest value not for a circle but for a octagon o diamond (depending on whether the 8-connectivity or 4-connectivity is employed to calculate the perimeter). Therefore, Haralick has proposed the following circularity feature:**

$$C = \{p_1, p_2 \dots p_m\}, \quad p_k = \begin{bmatrix} i_k \\ j_k \end{bmatrix}, \quad B = \begin{bmatrix} i_b \\ j_b \end{bmatrix}, \quad d_k = \sqrt{(i_k - i_b)^2 + (j_k - j_b)^2}$$

Distance from a contour point to the barycentre (in a circle all such distances are equal to the radius).

$$\mu_R = \frac{1}{m}\sum_{k=1}^{m} d_k, \qquad \sigma_R^2 = \frac{1}{m}\sum_{k=1}^{m} (d_k - \mu_R)^2$$

$$\tilde{C} = \frac{\mu_R}{\sigma_R}$$

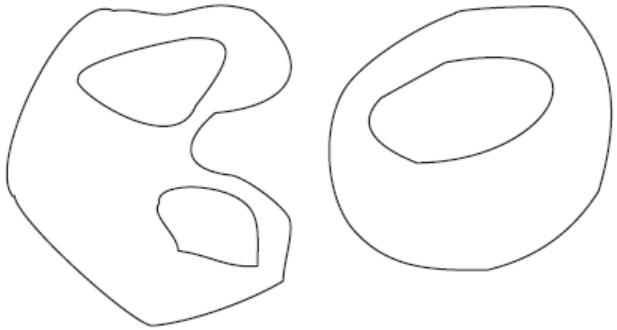
Again, a scale invariant feature due to the ratio between the mean distance and the standard deviation of distances being dimensionless. The smaller the standard deviation (i.e. distances are more similar one to another) the higher the circularity.

Euler Number

- **The Euler number of a binary image is defined as:**

$$E = C - H$$

where *C* is the number of connected components and *H* the number of holes.



- **In practise, *E* is computed for each connected component, so as to determine the number of its holes (e.g., *E=0* means that the blob has 1 hole).**

- ***E* is a topological feature, i.e. it is invariant to the so called *rubber sheet* transformations.**

Moments

- **The moment of order (*m,n*) of a region is defined as :**

$$M_{m,n} = \sum_{p \in R} i^m j^n$$

with *m* ≥ *0*, *n* ≥ *0* .

- **The Area is the moment of order (*0,0*):**

$$M_{0,0} = \sum_{p \in R} i^0 j^0 = \sum_{p \in R} 1 = A$$

- **Other relevant ones are the second moments (*m+n=2*):**

$$M_{0,2} = \sum_{p \in R} i^0 j^2 = \sum_{p \in R} j^2$$

Moment of inertia wrt the i-axis

$$M_{2,0} = \sum_{p \in R} i^2 j^0 = \sum_{p \in R} i^2$$

Moment of inertia wrt the j-axis

$$M_{1,1} = \sum_{p \in R} i^1 j^1 = \sum_{p \in R} ij$$

Deviation moment of inertia

Invariance to translation and scaling

- **The previously defined moments change according to the position of the region in the image. Invariance to translation can be achieved by simply calculating the moments relative to the barycentre:**

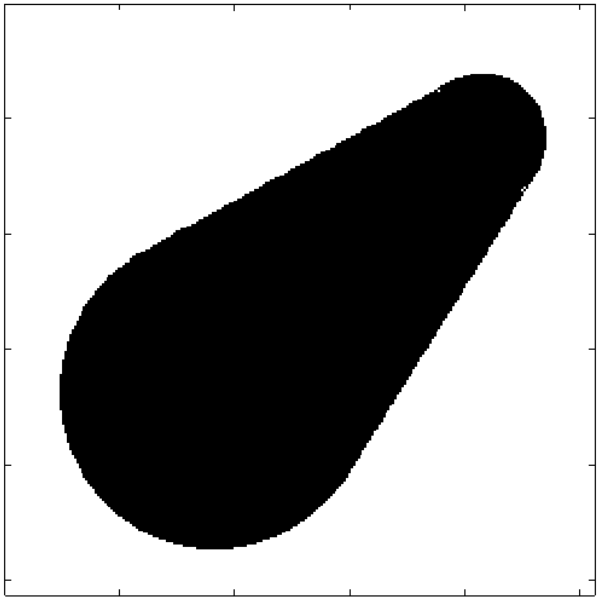
$$M'_{m,n} = \sum_{p \in R} (i - i_b)^m (j - j_b)^n$$

with such moments being usually referred to as *central moments.*

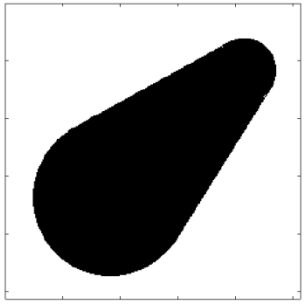
- **To achieve also invariance to scaling, *central moments* need to be *normalized*:**

$$V_{m,n} = \frac{M'_{m,n}}{A^{\alpha}}, \quad \alpha = \frac{m+n}{2} + 1$$

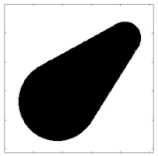
Example



256 x 256



128 x 128



64 x 64

$M'_{m,n}$

	256x256	128x128	64x64
M'_{20}	$6.1067 \cdot 10^7$	$3.8184 \cdot 10^6$	$2.3967 \cdot 10^5$
M'_{02}	$6.4680 \cdot 10^7$	$4.0366 \cdot 10^6$	$2.5315 \cdot 10^5$
M'_{11}	$-3.9126 \cdot 10^7$	$-2.4467 \cdot 10^6$	$-1.5327 \cdot 10^5$
M'_{30}	$-1.4325 \cdot 10^8$	$-4.4165 \cdot 10^6$	$-1.3700 \cdot 10^5$
M'_{03}	$3.0780 \cdot 10^8$	$9.7300 \cdot 10^6$	$3.1312 \cdot 10^5$
M'_{21}	$8.0244 \cdot 10^8$	$2.4995 \cdot 10^7$	$7.9233 \cdot 10^5$
M'_{12}	$-8.9164 \cdot 10^8$	$-2.7851 \cdot 10^7$	$-8.8653 \cdot 10^5$

$V_{m,n}$

	256x256	128x128	64x64
V_{20}	0.1035	0.1037	0.1037
V_{02}	0.1096	0.1096	0.1096
V_{11}	−0.0663	−0.0664	−0.0663
V_{30}	−0.0016	−0.0015	−0.0015
V_{03}	0.0033	0.0034	0.0035
V_{21}	0.0087	0.0087	0.0088
V_{12}	−0.0097	−0.0097	−0.0098

Hu's moments

- **Hu has shown that shape features invariant to rotation, translation and scaling can be defined based on normalized central moments. Such invariant features, usually referred to as Hu's moments, are shown on the right side of the slide:**

- **Hu proves invariance for continuous shapes. Nonetheless, when computed on *blobs* the above quantities turn out reasonably stable across similarities.**

$$h_1 = V_{20} + V_{02}$$

$$h_2 = (V_{20} - V_{02})^2 + 4V_{11}^2$$

$$h_3 = (V_{30} - 3V_{12})^2 + (V_{03} - 3V_{21})^2$$

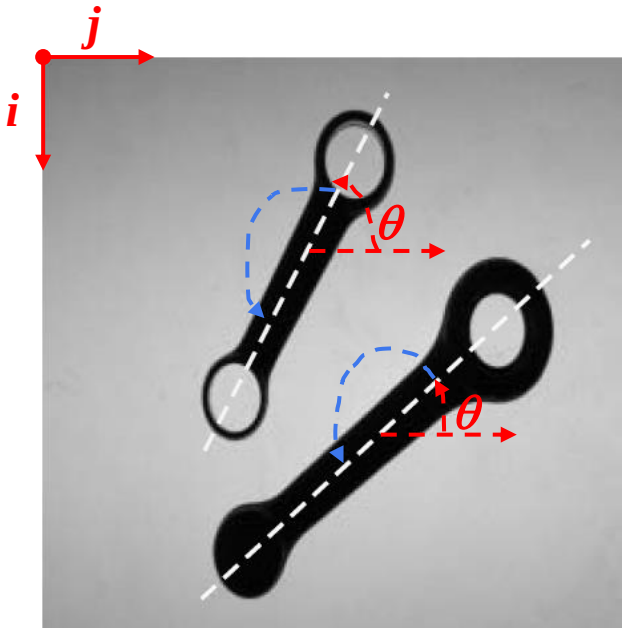
$$h_4 = (V_{30} + V_{12})^2 + (V_{03} + V_{21})^2$$

$$\begin{aligned} h_5 &= (V_{30} - 3V_{12})(V_{30} + V_{12})\left((V_{30} + V_{12})^2 - 3(V_{03} + V_{21})^2\right) \\ &+ (3V_{21} - V_{03})(V_{03} + V_{21})\left(3(V_{30} + V_{12})^2 - (V_{03} + V_{21})^2\right) \end{aligned}$$

$$\begin{aligned} h_6 &= (V_{20} - V_{02})\left((V_{30} + V_{12})^2 - (V_{03} + V_{21})^2\right) \\ &+ 4V_{11}(V_{30} + V_{12})(V_{03} + V_{21}) \end{aligned}$$

$$\begin{aligned} h_7 &= (3V_{21} - V_{03})(V_{30} + V_{12})\left((V_{30} + V_{12})^2 - 3(V_{03} + V_{21})^2\right) \\ &+ (3V_{12} - V_{30})(V_{03} + V_{21})\left(3(V_{30} + V_{12})^2 - (V_{03} + V_{21})^2\right) \end{aligned}$$

Orientation



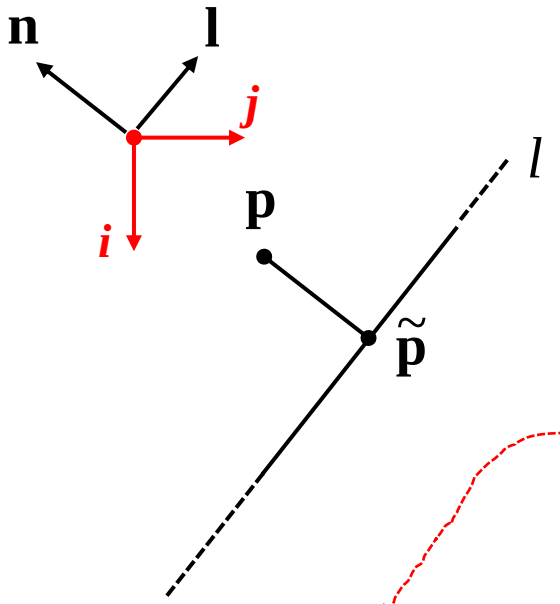
- **Many applications, such as e.g. robot guidance, require to determine the orientation of the objects appearing in the image.**
- **If an object is somewhat elongated, its orientation can be defined according to the direction of the *axis of least inertia*, i.e. the line through the barycentre of least moment of inertia, which is often referred to in the computer vision literature as *major axis:***

$$major\ axis = \arg\min_{l}\left(\sum_{p\in R} d_l^2(p)\right)$$

- **Accordingly, the orientation is typically determined as the angle, θ, between the major axis and the horizontal axis (i.e. the *j* axis with the adopted notation).**
- **As the major axis is a line, the thus defined object orientation can be determined *modulo* π, i.e. both θ and $\theta+\pi$ represent the same orientation.**

Distance from a point to a line

- **Given the line** $l : aj + bi + c = 0$ **, the squared distance from a point,** $p = \begin{bmatrix} i \\ j \end{bmatrix}$ **, to the line can be expressed as:** $d_l^2(p) = \dfrac{(aj + bi + c)^2}{a^2 + b^2}$



$$\mathbf{p} = \begin{bmatrix} i \\ j \end{bmatrix}, \quad \tilde{\mathbf{p}} = \begin{bmatrix} \tilde{i} \\ \tilde{j} \end{bmatrix}, \quad \mathbf{l} = \begin{bmatrix} a \\ -b \end{bmatrix}, \quad \mathbf{n} = \begin{bmatrix} b \\ a \end{bmatrix},$$

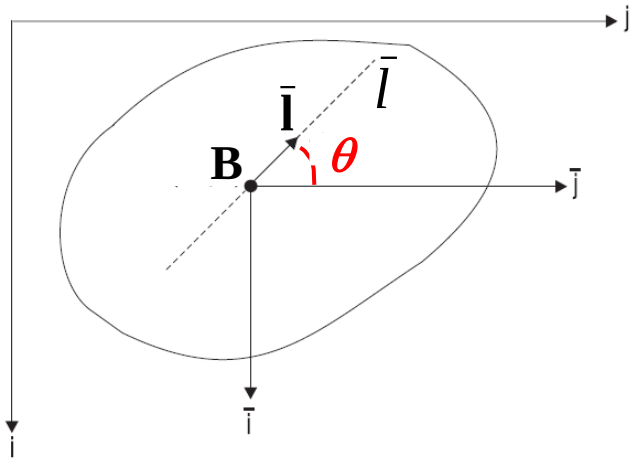
$$\mathbf{n} \cdot (\mathbf{p} - \tilde{\mathbf{p}}) = |\mathbf{n}||\mathbf{p} - \tilde{\mathbf{p}}| \cos\omega, \ \cos\omega = \pm 1$$

$$d_l(p) = \frac{\mathbf{n} \cdot (\mathbf{p} - \tilde{\mathbf{p}})}{|\mathbf{n}| \cos\omega} = \frac{b(i - \tilde{i}) + a(j - \tilde{j})}{\sqrt{a^2 + b^2} \cos\omega}$$

$$d_l(p) = \frac{aj + bi + c}{\sqrt{a^2 + b^2} \cos\omega} \rightarrow d_l^2(p) = \frac{(aj + bi + c)^2}{a^2 + b^2}$$

Signed "distance", the sign depending on p lying on either of the two sides wrt the line

Considering a line through the barycentre of the object….



$$\mathbf{p}=\begin{bmatrix} i \\ j \end{bmatrix},\ \mathbf{B}=\begin{bmatrix} i_b \\ j_b \end{bmatrix},\ \bar{\mathbf{p}}=\begin{bmatrix} \bar{i} \\ \bar{j} \end{bmatrix}=\begin{bmatrix} i-i_b \\ j-j_b \end{bmatrix},\ \bar{\mathbf{l}}=\begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}=\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

$$\bar{l}:\bar{\mathbf{p}}=\lambda\bar{\mathbf{l}} \rightarrow \begin{cases} \bar{i}=\lambda\alpha \\ \bar{j}=\lambda\beta \end{cases} \rightarrow \frac{\bar{j}}{\bar{i}}=\frac{\beta}{\alpha} \rightarrow \alpha\bar{j}-\beta\bar{i}=0$$

$$d_l^2(p)=\frac{\left(a\bar{j}+b\bar{i}+c\right)^2}{a^2+b^2}: \quad \begin{cases} a=\alpha \\ b=-\beta \\ c=0 \\ a^2+b^2=\alpha^2+\beta^2=\sin^2\theta+\cos^2\theta=1 \end{cases} \rightarrow d_l^2(p)=\left(\alpha\bar{j}-\beta\bar{i}\right)^2$$

- **The moment of inertia with respect to the above line through the barycentre can thus be expressed as:**

$$M\left(\bar{\mathbf{l}}\right)=\sum_{p\in R} d_l^2(p)=\sum_{p\in R}\left(\alpha\bar{j}-\beta\bar{i}\right)^2$$

Finding the major axis (1)

$$M(\bar{\mathbf{l}}) = \sum_{p \in R} (\alpha \bar{j} - \beta \bar{i})^2$$

$$M(\bar{\mathbf{l}}) = \alpha^2 \sum_{p \in R} \bar{j}^2 - 2\alpha\beta \sum_{p \in R} \bar{i}\bar{j} + \beta^2 \sum_{p \in R} \bar{i}^2$$

$$\sum_{p \in R} (j - j_b)^2 = M'_{0,2} \qquad \sum_{p \in R} (i - i_b)(j - j_b) = M'_{1,1} \qquad \sum_{p \in R} (i - i_b)^2 = M'_{2,0}$$

$$M(\bar{\mathbf{l}}) = \alpha^2 M'_{0,2} - 2\alpha\beta M'_{1,1} + \beta^2 M'_{2,0}$$

$$\begin{cases} \alpha = -\sin\theta \\ \beta = \cos\theta \end{cases} \rightarrow \quad M(\theta) = \sin^2\theta\, M'_{0,2} + 2\sin\theta\cos\theta\, M'_{1,1} + \cos^2\theta\, M'_{2,0}$$

Therefore, finding the major axis boils down to determining the minimum of the above function of θ.

Finding the major axis (2)

A necessary condition to determine such a minimum consists in the first derivative being zero:

$$M(\theta) = \sin^2\theta\, M'_{0,2} + 2\sin\theta\cos\theta\, M'_{1,1} + \cos^2\theta\, M'_{2,0}$$

$$\frac{dM(\theta)}{d\theta} = 2\sin\theta\cos\theta\, M'_{0,2} + 2\left(\cos^2\theta - \sin^2\theta\right)M'_{1,1} - 2\sin\theta\cos\theta\, M'_{2,0}$$

$\sin 2\theta$ $\quad$ $\cos 2\theta$ $\quad$ $\sin 2\theta$

$$\frac{dM(\theta)}{d\theta} = \sin 2\theta\left(M'_{0,2} - M'_{2,0}\right) + 2\cos 2\theta\, M'_{1,1}$$

$$\frac{dM(\theta)}{d\theta} = 0 \rightarrow \tan 2\theta = -\frac{2M'_{1,1}}{\left(M'_{0,2} - M'_{2,0}\right)} \rightarrow 2\theta = \arctan\left(-\frac{2M'_{1,1}}{\left(M'_{0,2} - M'_{2,0}\right)}\right) = -\arctan\left(\frac{2M'_{1,1}}{\left(M'_{0,2} - M'_{2,0}\right)}\right)$$

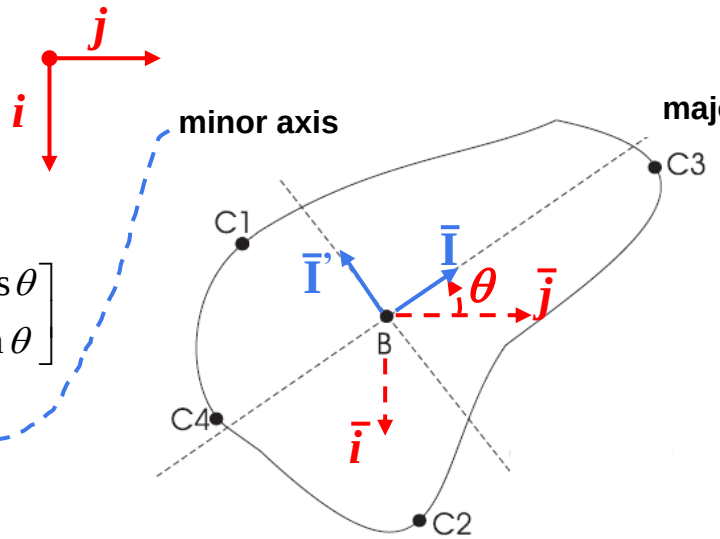
$$\theta = -\frac{1}{2}\arctan\left(\frac{2M'_{1,1}}{\left(M'_{0,2} - M'_{2,0}\right)}\right),\ 2\theta = -\arctan\left(\frac{2M'_{1,1}}{\left(M'_{0,2} - M'_{2,0}\right)}\right) + \pi \rightarrow \theta = -\frac{1}{2}\arctan\left(\frac{2M'_{1,1}}{\left(M'_{0,2} - M'_{2,0}\right)}\right) + \frac{\pi}{2}$$

Analysis of the second derivative (positive) would show the solution to provide a minimum → major axis

Analysis of the second derivative (negative) would show the solution to provide a maximum → minor axis.

Major and minor axes

The major and minor axes can be expressed conveniently in the image reference frame:



$$\bar{\mathbf{l}}' = \begin{bmatrix} \beta \\ -\alpha \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

$$\bar{\mathbf{l}} = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

$$\alpha\bar{j} - \beta\bar{i} = 0$$

$$\beta\bar{j} + \alpha\bar{i} = 0$$

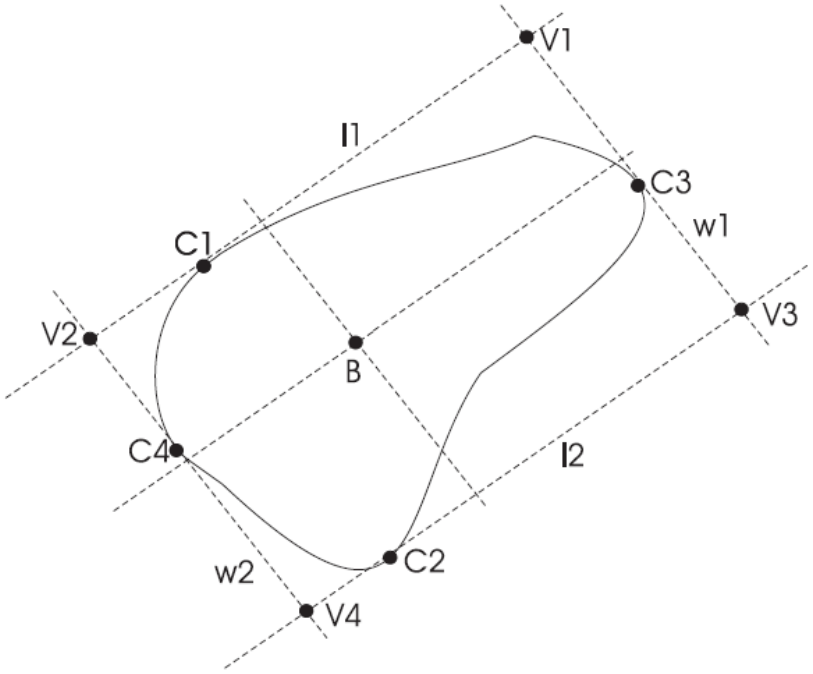
$$\begin{bmatrix} \bar{i} \\ \bar{j} \end{bmatrix} = \begin{bmatrix} i - i_b \\ j - j_b \end{bmatrix}$$

$$\alpha(j - j_b) - \beta(i - i_b) = 0 \rightarrow \alpha j - \beta i + \beta i_b - \alpha j_b = 0 \rightarrow \begin{cases} a = \alpha \\ b = -\beta \\ c = \beta i_b - \alpha j_b \end{cases} \rightarrow aj + bi + c = 0$$

$$\beta(j - j_b) + \alpha(i - i_b) = 0 \rightarrow \beta j + \alpha i - \beta j_b - \alpha i_b = 0 \rightarrow \begin{cases} a' = \beta \\ b' = \alpha \\ c' = -\beta j_b - \alpha i_b \end{cases} \rightarrow a'j + b'i + c' = 0$$

Oriented Enclosing Rectangle (1)

Given the two axes, we might wish to draw a bounding box aligned to the object, which is usually referred to as oriented MER (Minimum Enclosing Rectangle). Purposely, the four points laying at maximum distance on opposite sides of the two axes need to be determined.

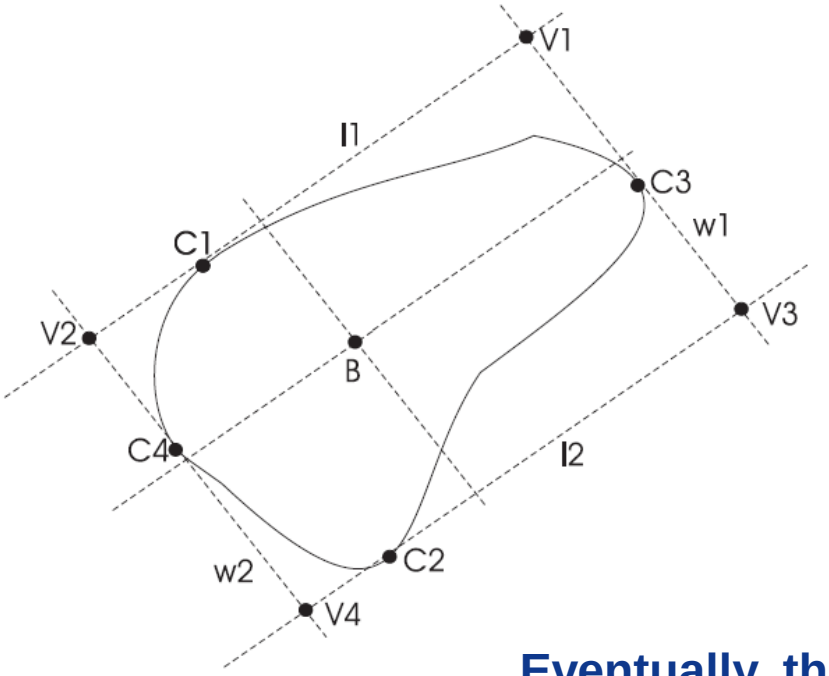



```
% major axis: aj+bi+c=0, minor axis: a'j*b'i+c'=0,
% edges is a binary image representing the contour of the object,
% image size is (nr,nc).

dMAmin=100000; dMAmax=-100000; dMImin=100000; dMImax=-100000;
normMA=sqrt(a*a + b*b); normMI=sqrt(a'*a' + b'*b');
for i=1:nr
for j=1:nc
if (edges(i,j)==1)
dMA=(a*j+b*i+c)/normMA;
dMI=(a'*j + b'*i+c')/normMI;
if (dMA<dMAmin)
dMAmin=dMA; i1=i; j1=j; end
if (dMA>dMAmax)
dMAmax=dMA; i2=i; j2=j; end
if (dMI<dMImin)
dMImin=dMI; i3=i; j3=j; end
if (dMI>dMImax)
dMImax=dMI; i4=i; j4=j; end
end
end
end
% C1: (i1, j1), C2: (i2, j2), C3: (i3, j3) C4: (i4, j4)
```

The algorithm relies on the signed "distance" to a line turning out positive or negative depending on the point lying on either of the two sides wrt the line.

Oriented Enclosing Rectangle (2)



Then, we can then find the lines through C1, C2 parallel to the major axis and those through C3,C4 parallel to the minor axis:

$$
\begin{aligned}
l_1: &\quad aj + bi + c_{l_1} &\Rightarrow &\quad c_{l_1} = -(aj_1 + bi_1)\\
l_2: &\quad aj + bi + c_{l_2} &\Rightarrow &\quad c_{l_2} = -(aj_2 + bi_2)\\
w_1: &\quad a'j + b'i + c_{w_1} &\Rightarrow &\quad c_{w_1} = -(a'j_3 + b'i_3)\\
w_2: &\quad a'j + b'i + c_{w_2} &\Rightarrow &\quad c_{w_2} = -(a'j_4 + b'i_4)
\end{aligned}
$$

Eventually, the vertexes of the oriented MER are given by:

$$
\begin{aligned}
j_{V_1} &= (bc_{w_1} - b'c_{l_1}) / (ab' - ba'), \quad i_{V_1} = (a'c_{l_1} - ac_{w_1}) / (ab' - ba')\\
j_{V_2} &= (bc_{w_2} - b'c_{l_1}) / (ab' - ba'), \quad i_{V_2} = (a'c_{l_1} - ac_{w_2}) / (ab' - ba')\\
j_{V_3} &= (bc_{w_1} - b'c_{l_2}) / (ab' - ba'), \quad i_{V_3} = (a'c_{l_2} - ac_{w_1}) / (ab' - ba')\\
j_{V_4} &= (bc_{w_2} - b'c_{l_2}) / (ab' - ba'), \quad i_{V_4} = (a'c_{l_2} - ac_{w_2}) / (ab' - ba')
\end{aligned}
$$

Features related to the MER

- **Length (*L*) and width (*W*):**

L: extent of the object along the major axis

$$L = d_{V_1V_2} = d_{V_3V_4} = \sqrt{(i_{V_1} - i_{V_2})^2 + (j_{V_1} - j_{V_2})^2} = \sqrt{(i_{V_3} - i_{V_4})^2 + (j_{V_3} - j_{V_4})^2}$$

W: extent of the object along the minor axis

$$W = d_{V_1V_3} = d_{V_2V_4} = \sqrt{(i_{V_1} - i_{V_3})^2 + (j_{V_1} - j_{V_3})^2} = \sqrt{(i_{V_2} - i_{V_4})^2 + (j_{V_2} - j_{V_4})^2}$$

- **Elongatedness:** $E = \dfrac{L}{W}$

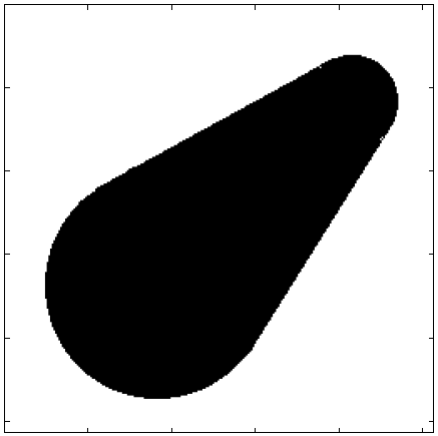
- **Rectangularity:** $R = \dfrac{A}{LW}$ A: object's area, LW: area of the oriented MER

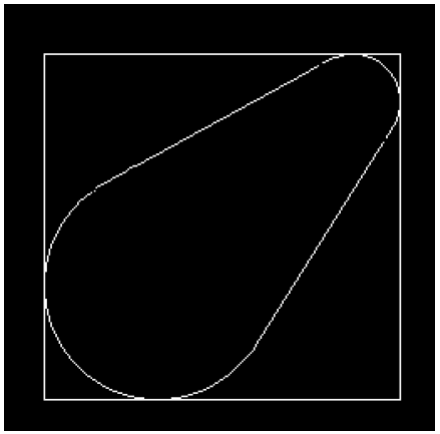
- **Ellipticity:** $E = \dfrac{A}{A_{LW}}, \qquad A_{LW} = \dfrac{\pi}{4}LW$

A: object's area, A_{LW}: area of the ellipse oriented exactly as the object and having major and minor axis lengths equal to L and W respectively.

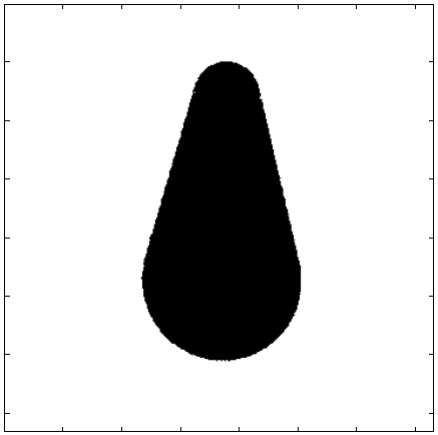
Examples (1)

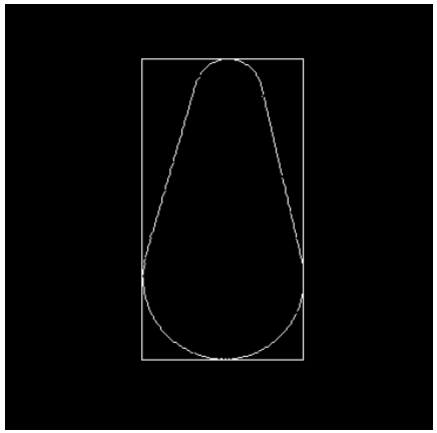
Computation of *L*, *W* and *R* based on an image-aligned MER does not provide invariance to rotation:





$$W = 207,$$
$$L = 212,$$
$$R = 0.5535$$

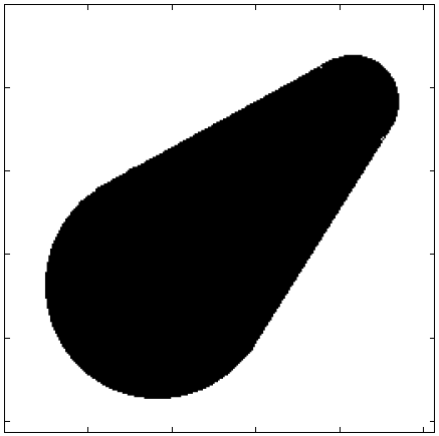


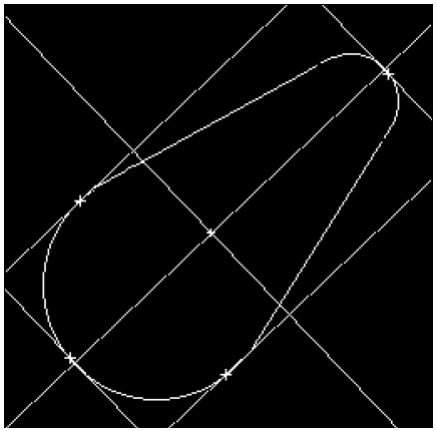


$$W = 257,$$
$$L = 137,$$
$$R = 0.6899$$

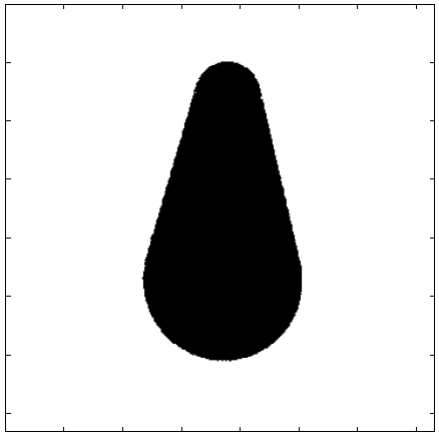
Examples (2)

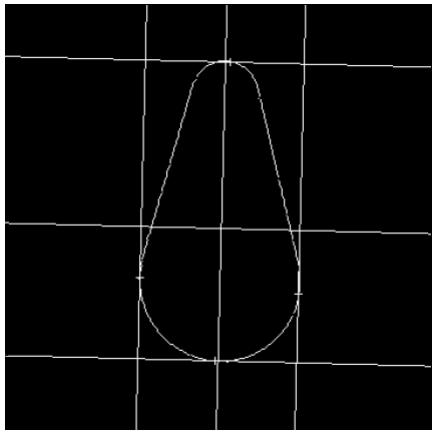
Invariance to rotation is instead achieved as long as the features (*L, W, R)* are computed according to the oriented MER:





$$W = 135.3212,$$
$$L = 254.8374,$$
$$R = 0.7043$$





$$W = 135.3099,$$
$$L = 255.2466,$$
$$R = 0.7033$$

Features from the covariance matrix

• **Orientation, size and elongatedness of a region can also be estimated based on a different interpretation of the data. In particular we assume the image coordinates of the pixels belonging to the region to be samples of a 2D random variable (i.e. a bivariate random variable), Accordingly, we compute the mean (i.e. the barycentre of the object) and the covariance matrix:**

$$\mathbf{p} = \begin{bmatrix} i \\ j \end{bmatrix}, \ \mathbf{p} \in R, \quad \mathrm{A} = \sum_{\mathbf{p} \in R} 1$$

$$\mathbf{B} = \begin{bmatrix} i_b \\ j_b \end{bmatrix} = \frac{1}{A} \sum_{\mathbf{p} \in R} \mathbf{p} = \frac{1}{A} \begin{bmatrix} \sum_{\mathbf{p} \in R} i \\ \sum_{\mathbf{p} \in R} j \end{bmatrix}, \qquad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2_{ii} & \sigma^2_{ij} \\ \sigma^2_{ij} & \sigma^2_{jj} \end{pmatrix}$$

$$\sigma^2_{ii} = \frac{1}{A} \sum_{\mathbf{p} \in R} (i - i_b)^2, \ \sigma^2_{jj} = \frac{1}{A} \sum_{\mathbf{p} \in R} (j - j_b)^2, \ \sigma^2_{ij} = \frac{1}{A} \sum_{\mathbf{p} \in R} (i - i_b)(j - j_b)$$

$$\frac{M'_{2,0}}{A} \qquad \frac{M'_{0,2}}{A} \qquad \frac{M'_{1,1}}{A}$$

Diagonalization of the covariance matrix

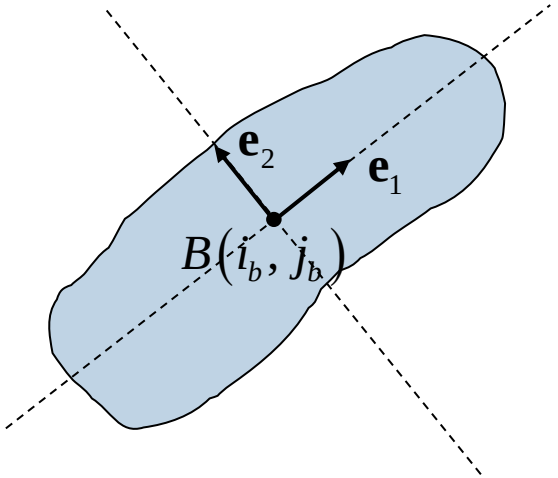
- **As already pointed out, as Σ is real and symmetric:**

$$\Sigma = \mathbf{R}\mathbf{D}\mathbf{R}^T: \quad \mathbf{R} = \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

$$\begin{cases} \lambda_1 : \text{largest eigenvalue}, \rightarrow \mathbf{e}_1 \\ \lambda_2 : \text{smallest eigenvalue}, \rightarrow \mathbf{e}_2 \end{cases}$$

direction of maximal dispersion of the random variable (major axis)

direction of minimal dispersion of the random variable (minor axis)

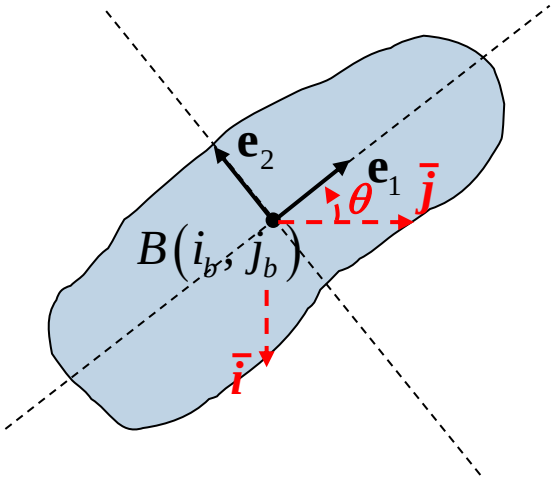


Determining the orientation (1)

$$\Sigma \mathbf{e_1} = \lambda_1 \mathbf{e_1} \rightarrow (\Sigma - \lambda_1 \mathbf{I})\mathbf{e_1} = 0$$

$$\begin{pmatrix} \sigma^2{}_{ii} - \lambda_1 & \sigma^2{}_{ij} \\ \sigma^2{}_{ij} & \sigma^2{}_{jj} - \lambda_1 \end{pmatrix} \begin{pmatrix} e_{1i} \\ e_{1j} \end{pmatrix} = 0$$

$$(\sigma^2{}_{ii} - \lambda_1) e_{1i} + \sigma^2{}_{ij} e_{1j} = 0$$
$$\sigma^2{}_{ij} e_{1i} + (\sigma^2{}_{jj} - \lambda_1) e_{1j} = 0$$



Let's then consider the unit eigenvector: $\mathbf{e_1} = \begin{bmatrix} e_{1i} \\ e_{1j} \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$

$$-(\sigma^2{}_{ii} - \lambda_1)\sin\theta + \sigma^2{}_{ij}\cos\theta = 0$$
$$-\sigma^2{}_{ij}\sin\theta + (\sigma^2{}_{jj} - \lambda_1)\cos\theta = 0$$

$$\lambda_1 = \sigma^2{}_{jj} - \sigma^2{}_{ij}\tan\theta$$

$$-\sigma^2{}_{ii}\sin\theta + (\sigma^2{}_{jj} - \sigma^2{}_{ij}\tan\theta)\sin\theta + \sigma^2{}_{ij}\cos\theta = 0$$

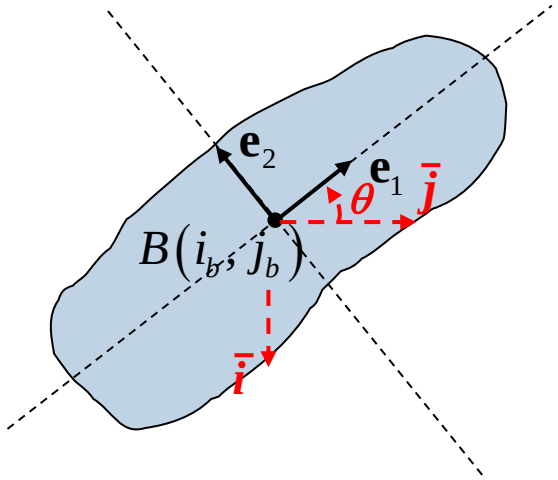
Determining the orientation (2)

$$-\sigma^2_{ii}\sin\theta + \left(\sigma^2_{jj} - \sigma^2_{ij}\tan\theta\right)\sin\theta + \sigma^2_{ij}\cos\theta = 0$$

$$-\sigma_{ii}{}^2\tan\theta + \left(\sigma_{jj}{}^2 - \sigma_{ij}{}^2\tan\theta\right)\tan\theta + \sigma_{ij}{}^2 = 0$$

$$-\sigma_{ii}{}^2\tan\theta + \sigma_{jj}{}^2\tan\theta - \sigma_{ij}{}^2(\tan\theta)^2 + \sigma_{ij}{}^2 = 0$$

$$\sigma_{ij}{}^2(1-(\tan\theta)^2) = \left(\sigma_{ii}{}^2 - \sigma_{jj}{}^2\right)\tan\theta$$



$$\frac{\tan\theta}{1-(\tan\theta)^2} = \frac{\sigma_{ij}{}^2}{\sigma_{ii}{}^2 - \sigma_{jj}{}^2}$$

$$= \frac{1}{2}\tan 2\theta$$

$$\tan 2\theta = \frac{2\sigma_{ij}{}^2}{\sigma_{ii}{}^2 - \sigma_{jj}{}^2}$$

$$\theta = \frac{1}{2}\tan^{-1}\frac{2\sigma_{ij}{}^2}{\sigma_{ii}{}^2 - \sigma_{jj}{}^2}$$

$$\theta = -\frac{1}{2}\tan^{-1}\frac{2\sigma_{ij}{}^2}{\sigma_{jj}{}^2 - \sigma_{ii}{}^2} = -\frac{1}{2}\tan^{-1}\frac{2M_{1,1}{}'}{M_{0,2}{}' - M_{2,0}{}'}$$

Computing the eigenvalues

$$\boldsymbol{\Sigma}\mathbf{e} = \lambda\mathbf{e} \rightarrow \left(\boldsymbol{\Sigma} - \lambda\mathbf{I}\right)\mathbf{e} = 0 \rightarrow \det\left(\boldsymbol{\Sigma} - \lambda\mathbf{I}\right) = 0$$

$$\det\begin{pmatrix} \sigma_{ii}^2 - \lambda & \sigma_{ij}^2 \\ \sigma_{ij}^2 & \sigma_{jj}^2 - \lambda \end{pmatrix} = 0 \rightarrow \left(\sigma_{ii}^2 - \lambda\right)\left(\sigma_{jj}^2 - \lambda\right) - \left(\sigma_{ij}^2\right)^2 = 0$$

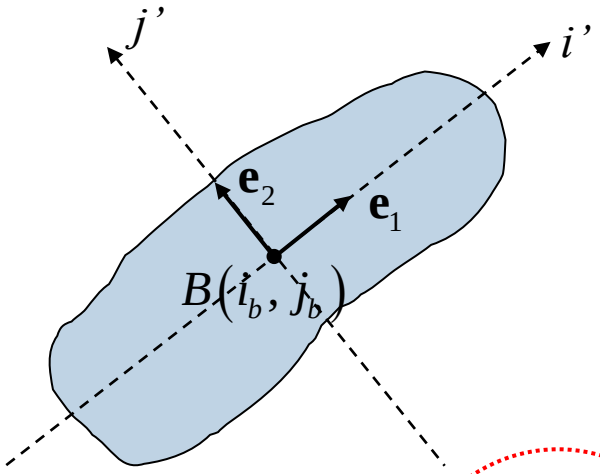
Characteristic Polynomial

$$\lambda^2 - \lambda\left(\sigma_{ii}^2 + \sigma_{jj}^2\right) + \sigma_{ii}^2\sigma_{jj}^2 - \left(\sigma_{ij}^2\right)^2 = 0 \Rightarrow \lambda_1, \lambda_2$$

This. solving the quadratic equation yields the eigenvalues:

$$\lambda_1 = \frac{\sigma_{ii}^2 + \sigma_{jj}^2 + \sqrt{\left(\sigma_{ii}^2 - \sigma_{jj}^2\right)^2 + 4\left(\sigma_{ij}^2\right)^2}}{2}, \qquad \lambda_2 = \frac{\sigma_{ii}^2 + \sigma_{jj}^2 - \sqrt{\left(\sigma_{ii}^2 - \sigma_{jj}^2\right)^2 + 4\left(\sigma_{ij}^2\right)^2}}{2}$$

Size and elongatedness



$$\mathbf{p}' = \begin{bmatrix} i' \\ j' \end{bmatrix}: \quad \mathbf{p}' = \mathbf{R}^T(\mathbf{p} - \mathbf{B}) \quad \forall \mathbf{p} \in R$$

Hotelling Transform: brings pixel coordinates into a central reference frame with axes aligned to the eigenvectors

$$L \approx 2\sqrt{\lambda_1}$$
$$W \approx 2\sqrt{\lambda_2}$$
$$E = \sqrt{\frac{\lambda_1}{\lambda_2}}$$

Faster approximations based on computing the eigenvalues only (i.e. without applying the Hotelling Transform).

$$L = i'_{\max} - i'_{\min}$$
$$i'_{\max} = \max(i'), \mathbf{p} \in R$$
$$i'_{\min} = \min(i'), \mathbf{p} \in R$$

$$W = j'_{\max} - j'_{\min}$$
$$j'_{\max} = \max(j'), \mathbf{p} \in R$$
$$j'_{\min} = \min(j'), \mathbf{p} \in R$$

The square roots of the eigenvalues are the semi-lenghts of the major and minor axes of the ellipse: $\mathbf{p}^T \Sigma^{-1} \mathbf{p} = 1$

which is centered at the barycentre and has the same orientation as the object (fitting ellipse) .

Main References

1) R. Gonzales, R. Woods, “Digital Image Processing – Third Edition”, Pearson Prentice-Hall, 2008. ok
2) R. Fisher, S. Perkins, A. Walker, E. Wolfart, “Hypermedia Image Processing Reference”, Wiley, 1996 (http://homepages.inf.ed.ac.uk/rbf/HIPR2/)
3) R. Haralick, L. Shapiro “Computer and Robot Vision – Vol. I and II, Addison-Wesley Publishing Company, 1993. ok
4) M. Sonka, V. Hlavac, R. Boyle, “Image Processing, Analysis and Machine Vision”, Chapman & Hall, 1993.
5) L. Di Stefano, A. Bulgarelli, “A simple and efficient connected components labeling algorithm”, Int. Conf. on Image Analysis and Processiong (ICIAP), 1999.
6) C. Grana, D. Borghesani, R. Cucchiara “Optimized Block-based Connected Components Labeling with Decision Trees”. IEEE Transactions on Image Processing, 2010.
7) B. Horn, “Robot Vision”, The MIT Press, 1986.
8) C. Steger, M. Ulrich, C. Wiedemann, “Machine Vision Algorithms and Applications”, Wiley-VCH, Weinheim, 2008.