Pattern Recognition

Guillaume Lemaître

22 mai 2008

Plan

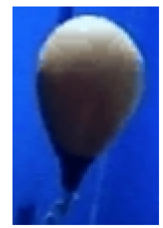
Plan

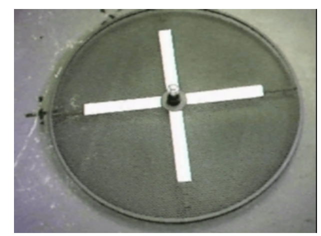
1. Introduction
2. Methods
3. Performance
4. Plan

Aim

Create a system which detect different objects :

- Mid water target
- Bottom target
- Tyre
- Cone





Aim

Constraints :

- « Real-Time » system
- Use minimum ressources (CPU, memory, ...)

Technologie

Use technologie :

- Acquisition with analog camera and card MPEG 4
- Intel OpenCV (Computer Vision) librairy
- Programmation in C++

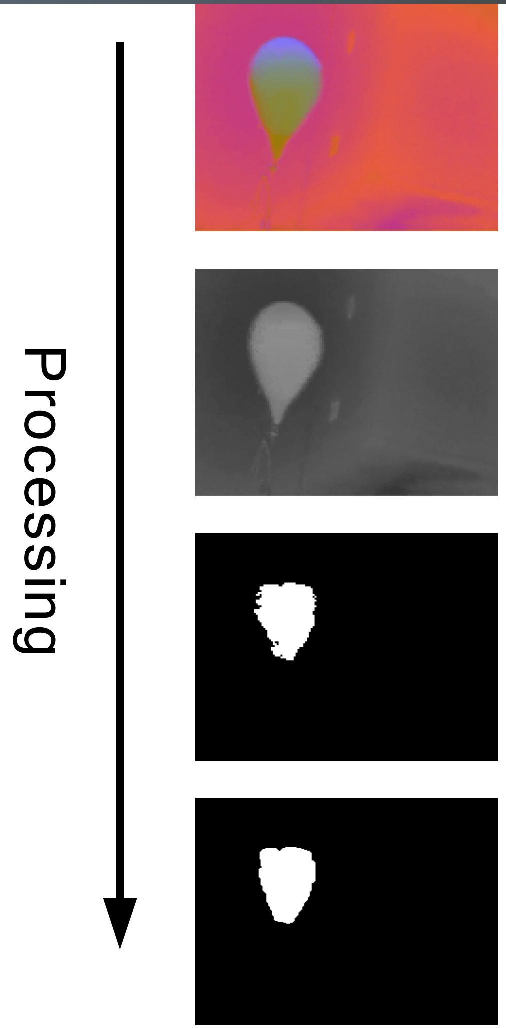
Plan

1. Introduction
2. Methods
3. Performance
4. Plan

Color detection :

Processing :

- Convert BGR Image to YCrCb
- Take only channel Cr
- Threshold to keep the equivalence of orange color
- Morphology operation : open



Color detection :

Processing for mid water target :

- Search contours
- Calculate perimeter and area
- Calculate circularity $circularity = \frac{(4 \times \Pi \times area)}{(perimeter^2)} = 1 \; if \; perfect \; circle$

Decision :

- If circularity superior to a specific threshold

Form detection :

Lign Hough Transform :

- Each points admit an infinity of straight lines.
- The general equation of lines which pass by one point is :

$$y_i = a.x_i + b \qquad \textit{with point coordinates}\,(x_0, y_0)$$

- But we use the polar representation which is :

$$\rho(\Theta) = x_0.\cos(\Theta) + y_0 \sin(\Theta) \qquad \textit{with point coordinates}\,(x_0, y_0)$$

Form detection :

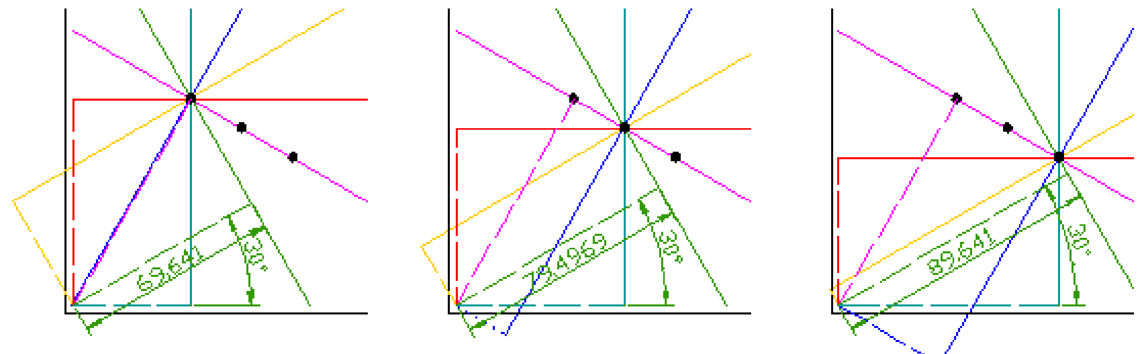
Lign Hough Transform :

- Each lines is characteristic of two parameters Θ and ρ

- Plan of Hough :

$$\rho(\Theta) = x_0 . \cos(\Theta) + y_0 \sin(\Theta)$$

$$\textit{with point coordinates}\,(x_0, y_0)$$



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5

Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

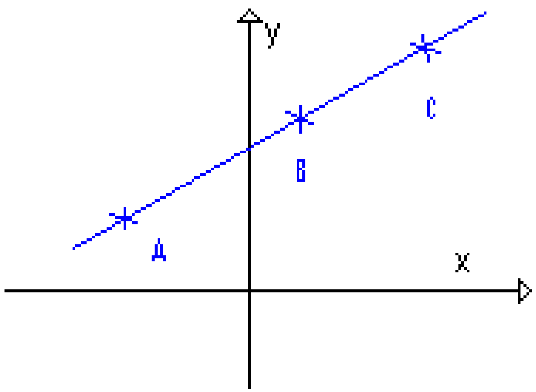
Form detection :

Lign Hough Transform :

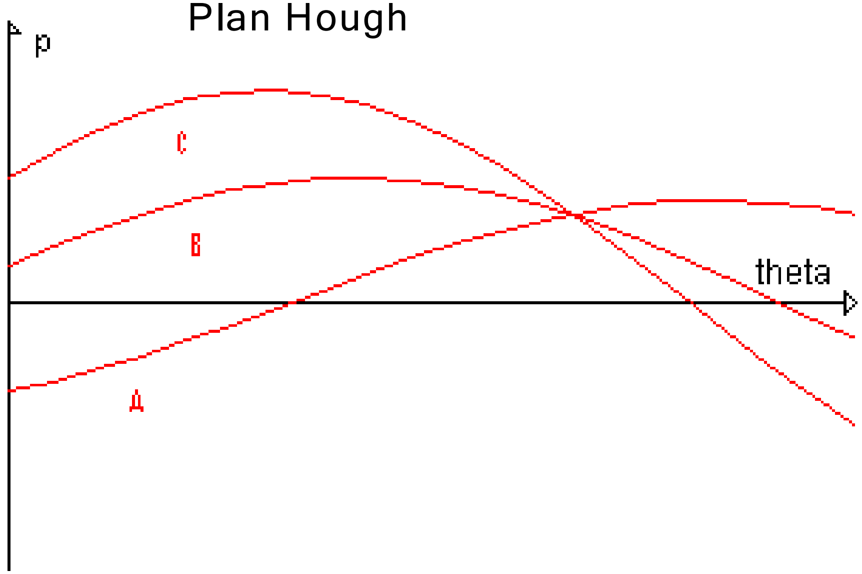
- Each point has a representation in space of Hough

➜ **Intersection in space of Hough represents a straight line in space cartesien**

Plan cartesien



Plan Hough



Form detection :

Lign Hough Transform :

- Implementation of Line Hough Transform in OpenCV :
  - *CvSeq* cvHoughLines2(CvArr* image, void* line_storage, int method, double rho, double theta, int threshold, double param1=0, double param2=0)*

    - Rho : Distance resolution in pixel
    - Theta : Angle resolution in pixel
    - Threshold : Number minimum of points

Form detection :

Circle Hough Transform :

- The general equation of circle is :

$$(x_0 - a)^2 + (y_0 - b)^2 = r^2 \qquad \text{with point coordinates}\,(x_0, y_0)$$

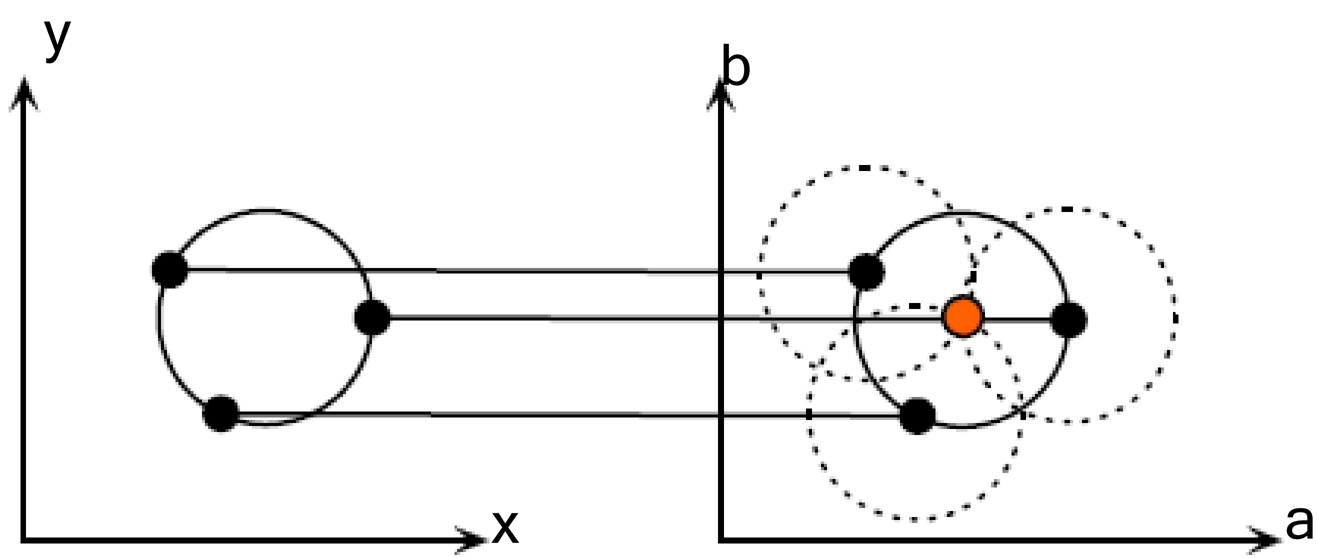
- The parametrics equations are :

$$x = a + R.cos(\Theta) \quad <=> a = x - R.cos(\Theta)$$

$$y = b + R.sin(\Theta) \quad <=> b = y - R.sin(\Theta)$$

Form detection :

Circle Hough Transform :



Each point in geometric space (left) generates a circle in parameter space (right). The circles in parameter space intersect at the (a, b) that is the center in geometric space.

Form detection :

Circle Hough Transform :

- Implementation of Circles Hough Transform in OpenCV :
  - *CvSeq* cvHoughCircles(CvArr* image, void* circle_storage, int method, double dp, double min_dist, double param1=100, double param2=100)*
    - min-dst : minimum distance between centers
    - param1 : threshold Canny
    - param2 : Number minimum of points on circles

Tracking Object :

Mean-shift Algorithm :

- Aim : Search a model in image

- Two stages :
  - Initialisation : definition model
  - Processing : search model in image

Tracking Object :

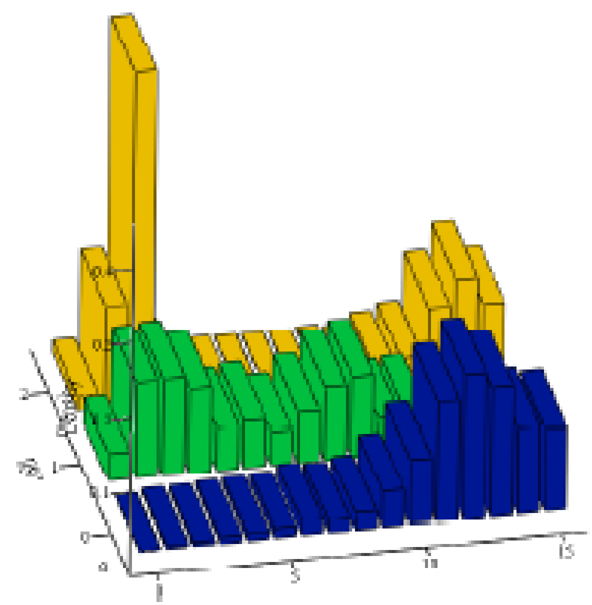
Mean-shift Algorithm :

- Initialisation : definition model :

➜ Create a histogram with discretisation of the representation choosen (hue, saturation ...)

➜ Calculate density gradient estimation of the representation choosen

Cible



$$\hat{q}_u = C \cdot \sum_{i=1}^{n} k(\|\mathbf{x}_i^*\|^2) \cdot \delta(c(\mathbf{x}_i^*), u)$$

$$C = \frac{1}{\sum_{i=1}^{n} k(\|\mathbf{x}_i^*\|^2)}$$

Tracking Object :

Mean-shift Algorithm :

- Initialisation : definition model :

- Initialisation of the current position in $y_0 <- y(t)$

Tracking Object :

Mean-shift Algorithm :

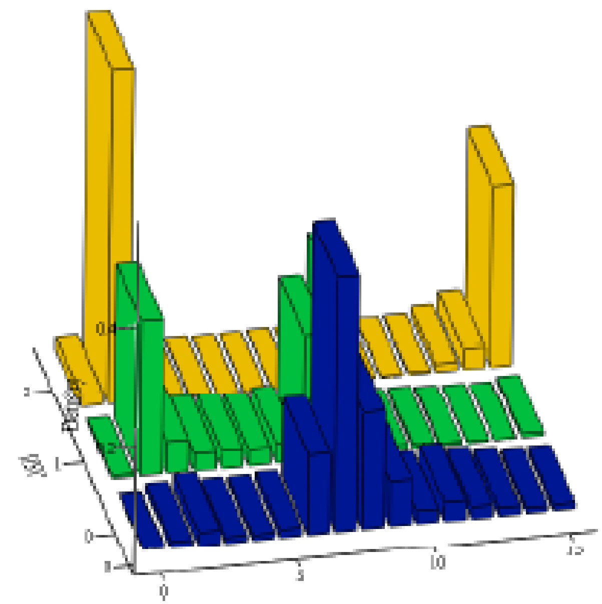
- Iteration : search model in current image :
  - Calculate density gradient estimation of current candidate in :

$$p(y_0)=(p_u(y_0))_{u=1\ldots m}$$

  - Calculate the Bhattacharya distance

$$\rho(\mathbf{y}_0)=\rho(\mathbf{p}(\mathbf{y}_0),\mathbf{q})=\sum_{u=1}^{m}\sqrt{p_u(\mathbf{y}_0)\cdot q_u}$$

Candidat centré en y



$$\hat{p}_u(\mathbf{y})=C_h\cdot\sum_{i=1}^{n_h}k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|^2\right)\cdot\delta(c(\mathbf{x}_i),u)$$

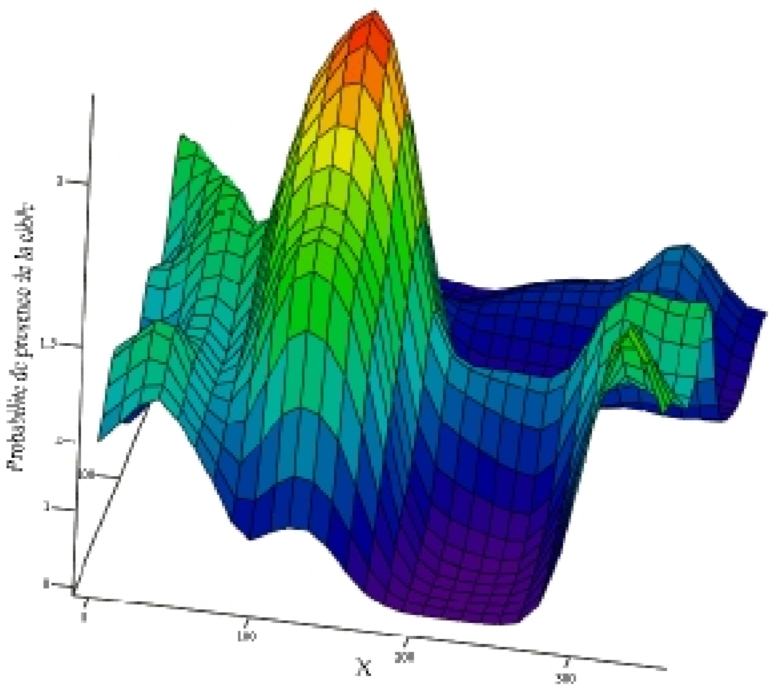
$$C_h=\frac{1}{\sum_{i=1}^{n_h}k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|^2\right)}$$

Tracking Object :

Mean-shift Algorithm :

- Iteration : search model in current image :

➜ The Bhattacharya distance measures the similarity between two discrete probability which are here the density gradient estimation of model and current candidate



$$\hat{\rho}(y) = \hat{\rho}(\hat{p}(y), \hat{q}) = \sum_{u=1}^{m} \sqrt{\hat{p}_u(y) \cdot \hat{q}_u}$$

Tracking Object :

Mean-shift Algorithm :

- Iteration : search model in current image :

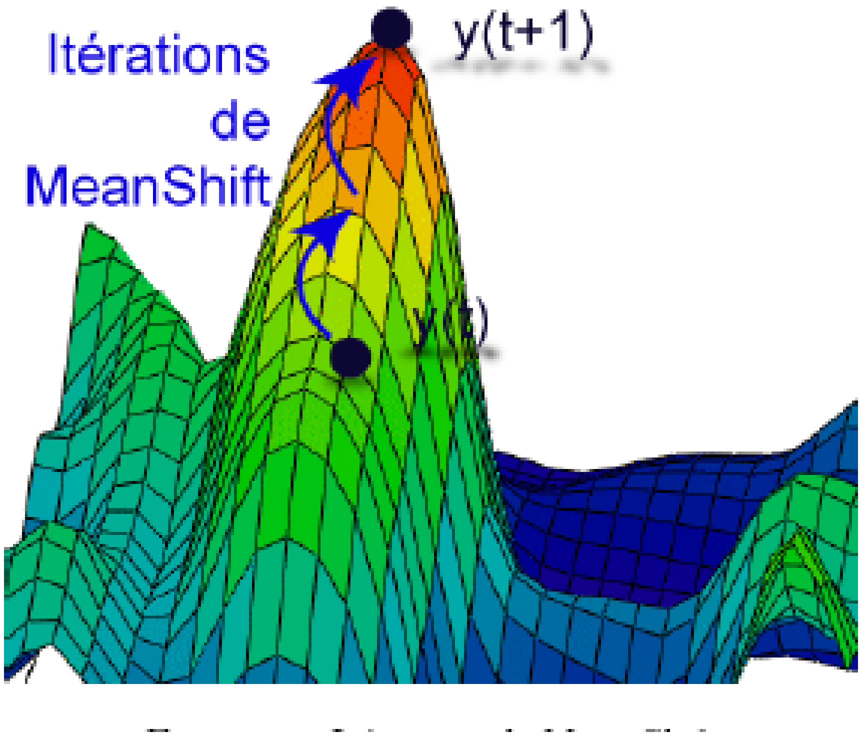
  - Calculate weigth vector

$$\mathbf{w} = \{w_i\}_{i=1 \ldots n_h}$$

$$w_i = \sum_{u=1}^{m} \delta(c(\mathbf{x}_i), u) \cdot \sqrt{q_u / p_u(\mathbf{y}_0)}$$

  - Calculate position of next candidate

$$\mathbf{y}_1 = \frac{\sum_{i=1}^{n_h} x_i \cdot w_i \cdot g(\|\frac{y_0 - x_i}{h}\|^2)}{\sum_{i=1}^{n_h} w_i \cdot g(\|\frac{y_0 - x_i}{h}\|^2)}$$



Tracking Object :

Mean-shift Algorithm :

- Implementation of Mean-Shift in OpenCV :
  - *int cvCamShift(const CvArr* prob_image, CvRect window, CvTermCriteria criteria, CvConnectedComp* comp, CvBox2D* box=NULL);*
  - *void cvCalcBackProject(IplImage** image, CvArr* back_project, const CvHistogram* hist);*

Plan

Comparison

Methods	Advantage	Disadvantage
Color detection	• Detection less fast than other presented • Only for orange object • Don't working far of object • Use 40% of CPU • High dependance of thresholds	• Working very nice near of object
Hough Transformation	• Many false alarm • Use 25% of CPU • Don't working far of object • High dependance of parameters	• More fast than previous method
Mean-Shift algorithm	• High dependance of histogram of start • Sensitive at noise	• Method very fast • Working far of object • Use 1% CPU

Plan

Plan

- Application of methods to detect different objects