Adding Vision To Salamander/Snake-Robot

Benoit RAT

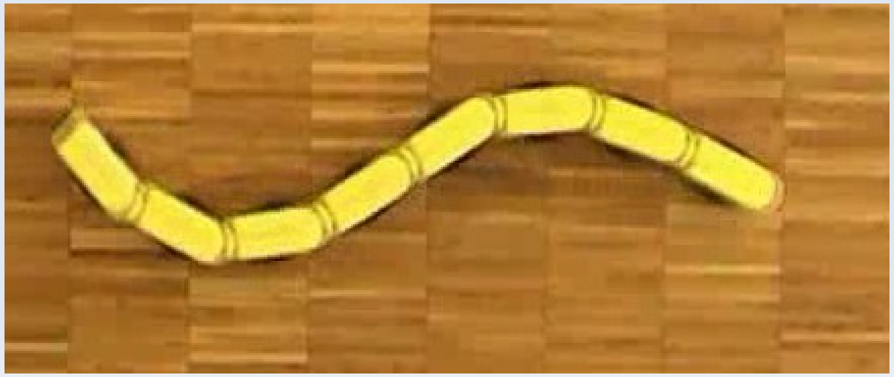
Supervised by
Dr. François Fleuret - CVLAB

In Collaboration with BIRG

Snake/Salamander Robots

The salamander and the snake robots are developed by the BIRG:

- Bio-inspired robots.
- The salamander can walk.
- The snake can crawl.
- Both can swim.
- They like pink color.



A Bio-Inspired Model

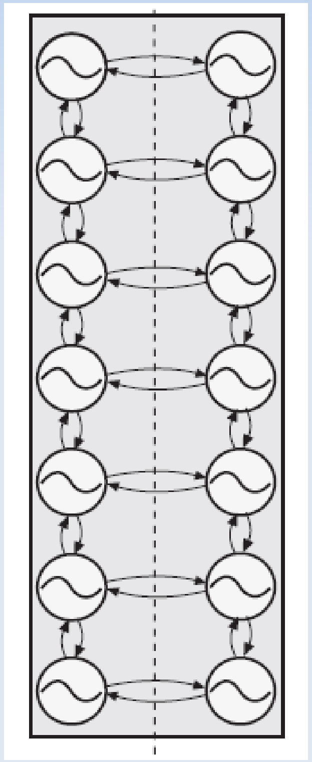
To obtain realistic gaits we need :

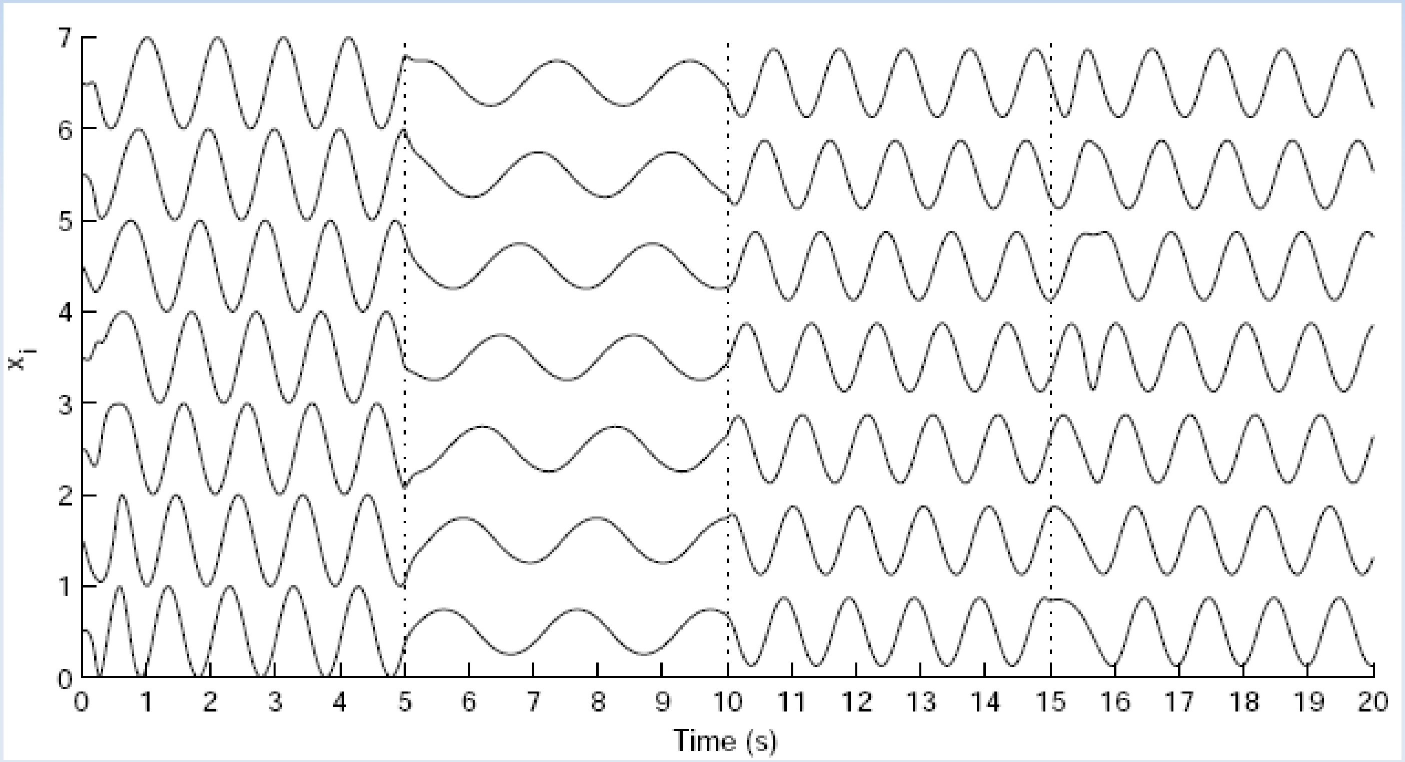
- Redundancies in movement
- synchronization of the different degrees of freedom.
- Handle brutal stimulus changes with smooth transitions.

Consequently, the bio-inspired model that we use generates gaits by giving:

Specific parameters to coupled non-linear oscillators.

The Central Pattern Generator





Specific parameters to coupled non-linear oscillators.

AV2SR

A radio-camera was set on the robot in order to generate different stimuli.

The project has been divided in 2 parts:

- 1 - Color tracking.
- 2 - Matching patch tracking

(Find geometrical transformation between 2 frames)

Color Tracking

Track a uniform color ball using color density model.

We first start to simulate our goal:

1st capture

Use the ball position and size as stimuli for the salamander/snake robot.

Color Tracking :
Color Space

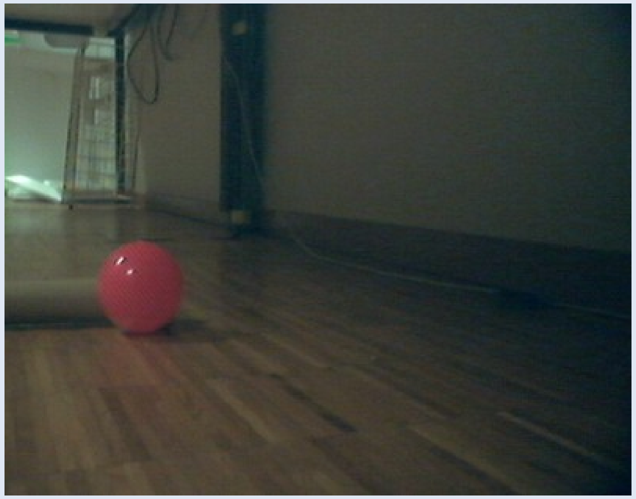
We use YCrCb Space:

- Bio-inspired color space with luma (rod cells) and chromacities (cone cells) separated
- Mainly used in video in europe.
- Linear transformation from RGB.

Color Tracking:
Illuminant adaptation

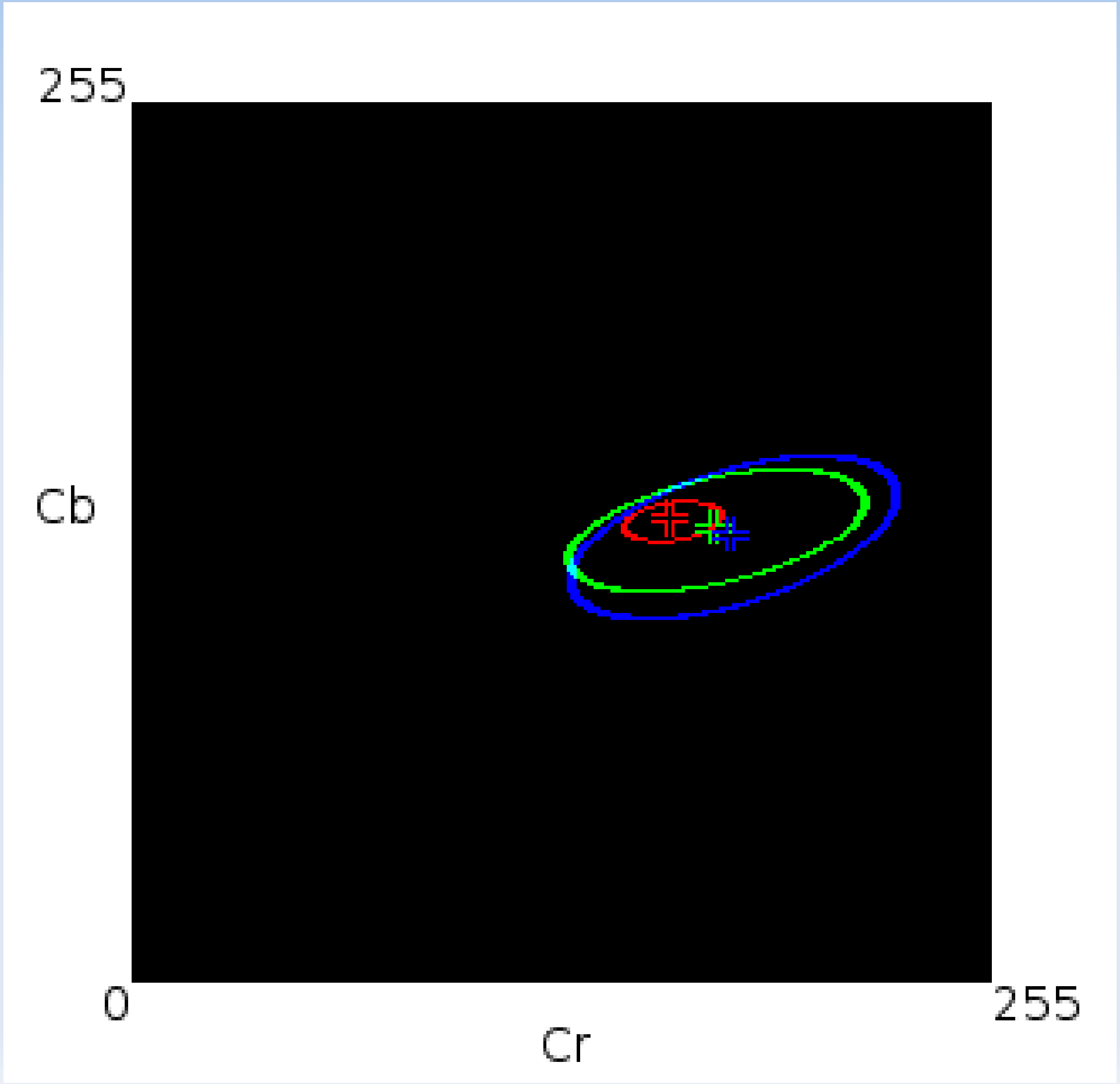
We use a general illuminant adaptation:

- We model the color (Cr,Cb) of the pink ball under different type of illumination.
- Then, to select one of the three models we look at the mean Y (luma) of the whole image.

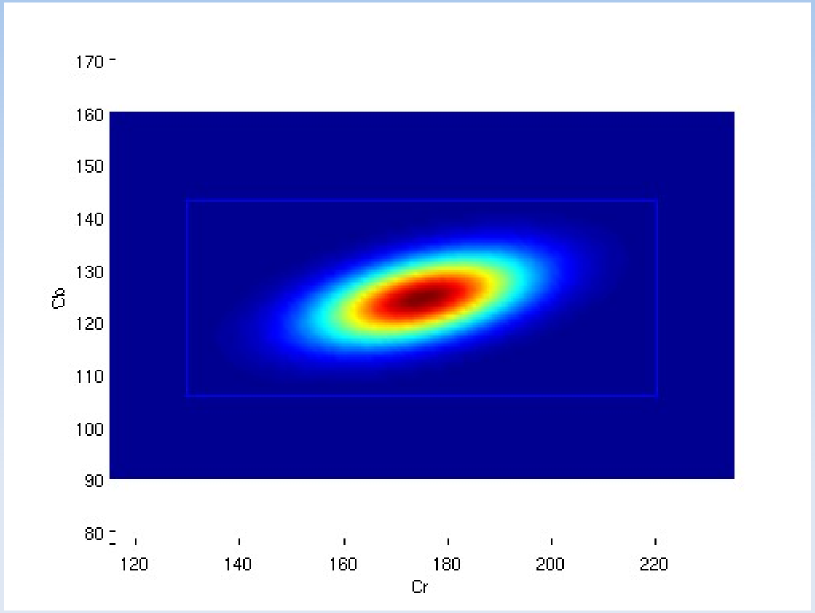


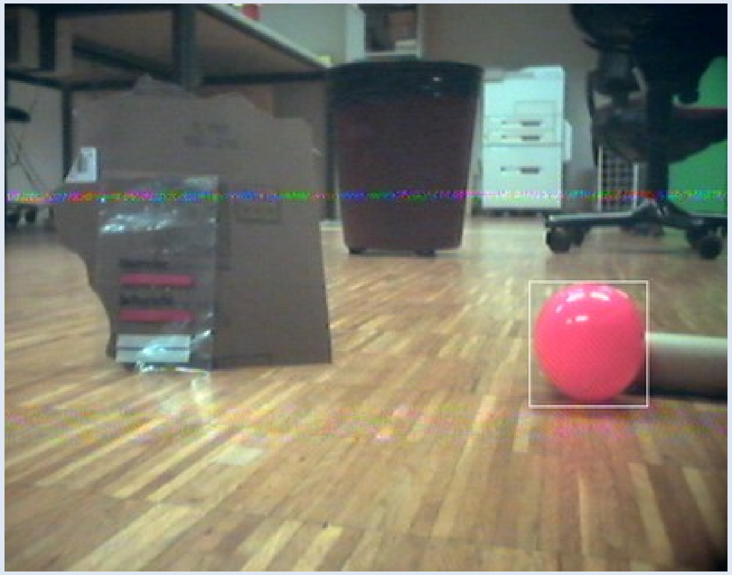
Color Tracking:
Illuminant adaptation

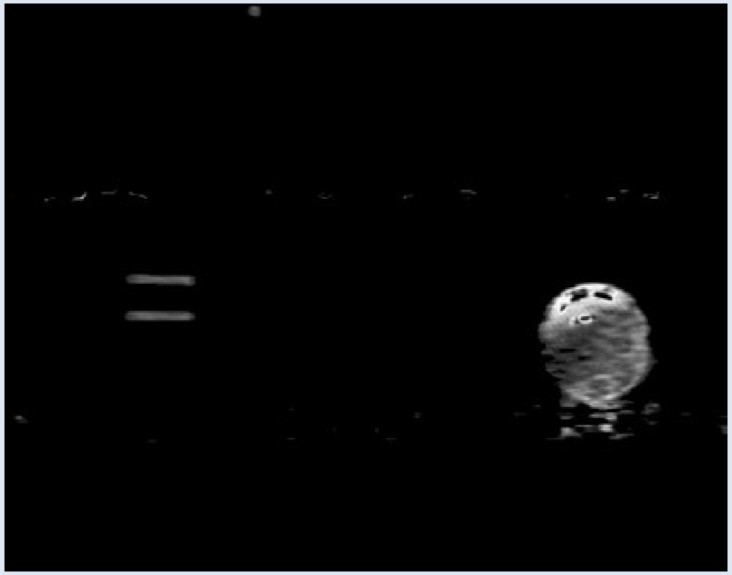
- Red: Low
- Green: Medium
- Blue: High



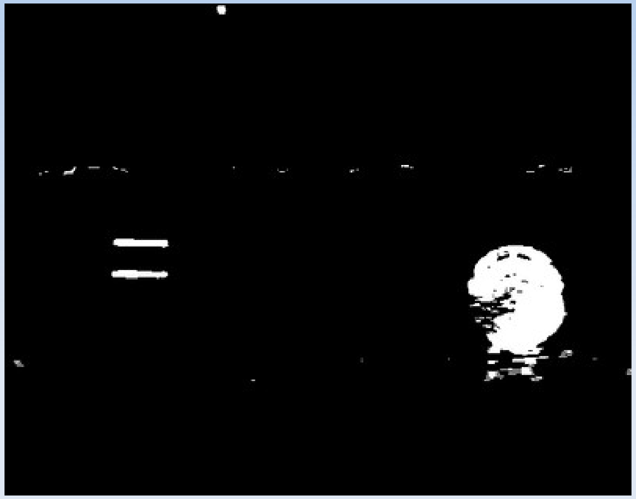
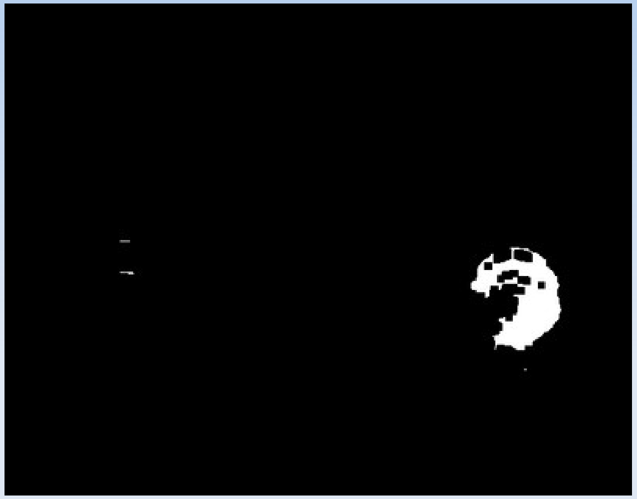
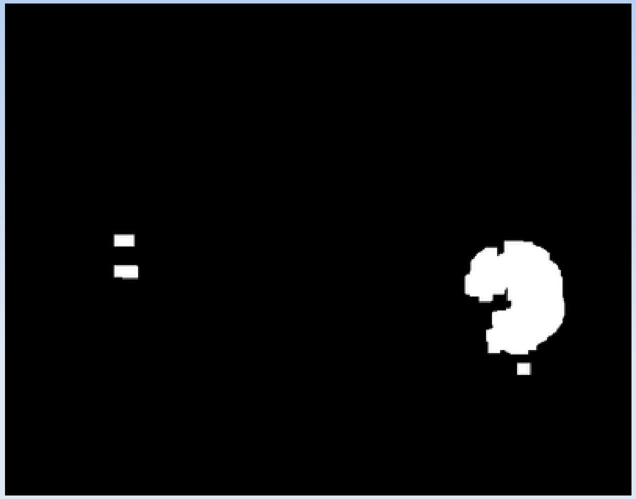
Color Tracking:
Gaussian color distribution







Color Tracking:
Morphological operations

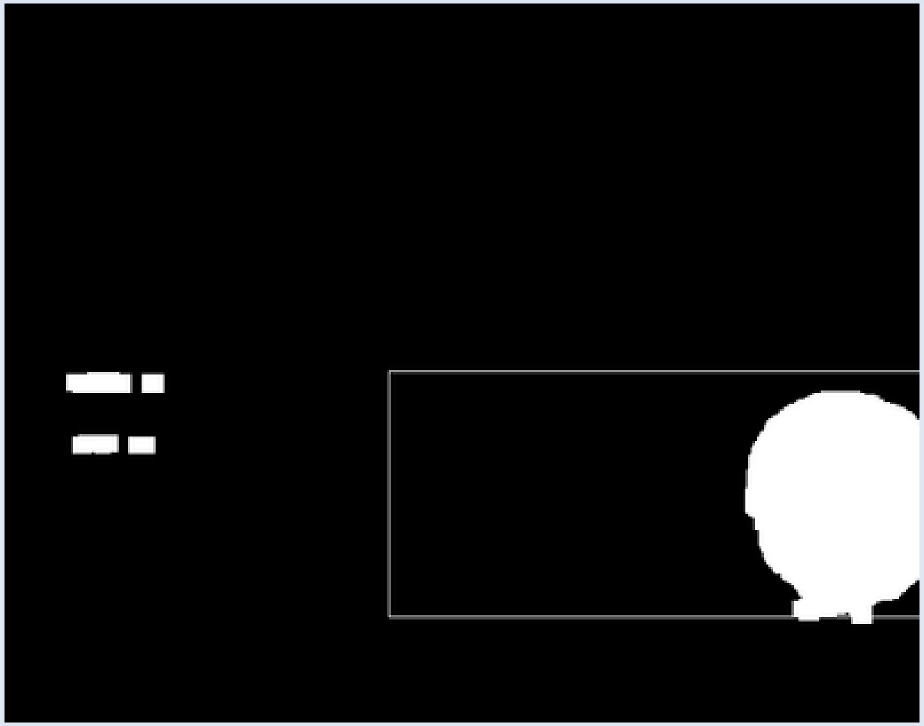




Threshold, Erode, Dilate

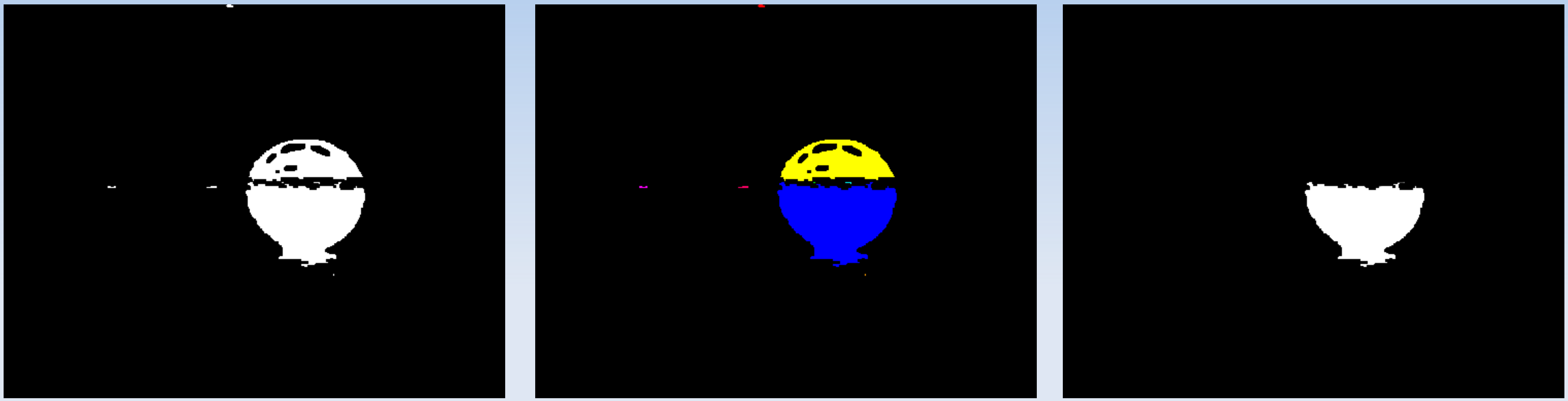
(Remove noises)

Color Tracking:
Findind mask I

Searching mask position by only computing mean and variance of the white pixel positions.



Color Tracking:
Finding mask II - Blob algorithm



Middle: Label all contiguous pixels.
Right: Find the largest connex component.

Cons:
- Do not handle separated masks.
- Relatively too slow for real time purpose.

Color Tracking:
Finding mask III - Best square

- Finding a number of pixel in a square that is proportional to biggest disk inside it.

Exhaustive search to find the maximum score for all possible square side length d and position x,y:

$$score(x, y, d) = \sum_{|x'-x| \leqslant d, |y'-y| \leqslant d} (mask(x', y')) - \kappa d^2$$

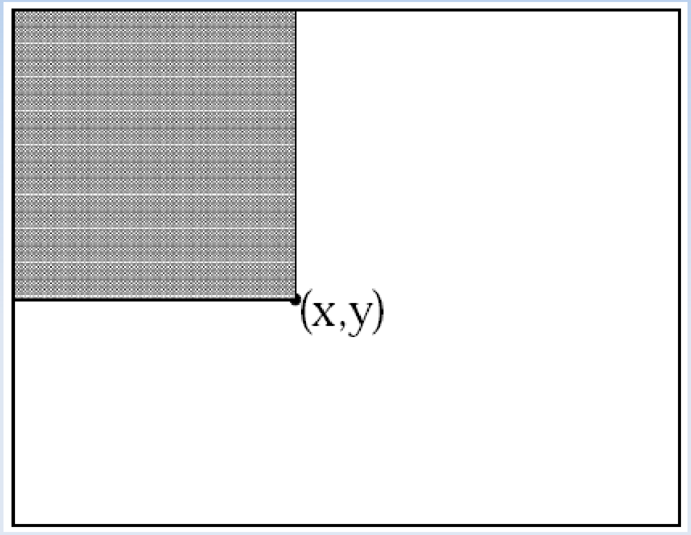
With the following relation between square and circle areas:

$$\kappa = 1 - \frac{\pi r^2}{d^2} = 1 - \frac{\pi}{4} = 0.2$$

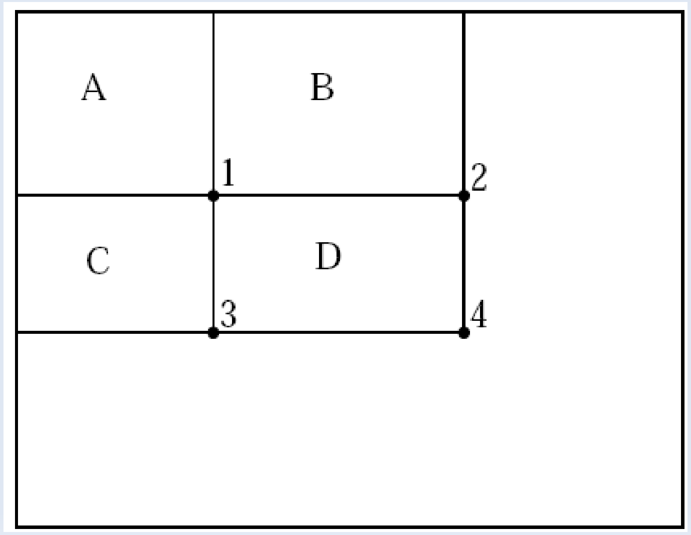
Color Tracking:
Acceleration by using Integral Image

Compute quickly the sum of pixels in all possible rectangles in the images.

$$iimg(x, y) = \sum_{y'=0}^{H} \sum_{x'=0}^{W} img(x', y')$$



- The sum within D can be computed as : AREA(D) = (4+1) - (2+3)
- ((A+B +C +D) +A)–((A+C) + (A+B)) = AREA(D)



Discovered for computer graphic and texture mapping by Crow 84, Reused by Simard 99 and Viola 01.

Color Tracking:
Results

We have now an algorithm that can find in real time the position and size of the ball:

Offline 1 (low illuminant)

Offline 2 (normal illuminant)

On-line Color Tracking:

Two stimuli are sent to the robot:

- Drive of the robot (can be considered as the speed).
- Turn of the robot.

They respectively come from the size and the position of the target square.

Heuristic approach for drive:

It is modulated by looking at its mean over 5 (quick changes) and 50 frames (stables changes).

Snake View - Online

Out View 1 - Out View 2

On-line Color Tracking

Conclusion:

Adding artificial vision in bio-robotics is interesting since several goals are stimulated by vision.

Avoiding obstacle, following objects, exploration...

- Embedded vision also simulates the feedback of a neural network.

  The robot modulates its gait considering the changes of its vision field.

Matching patch tracking

Goal: Know the relative position of the robot and its head movement.

Find the geometrical transformation (similitude) between two frames.

This transformation is computed by matching small patches (20x20 pixels) looking at their grey-level histogram.

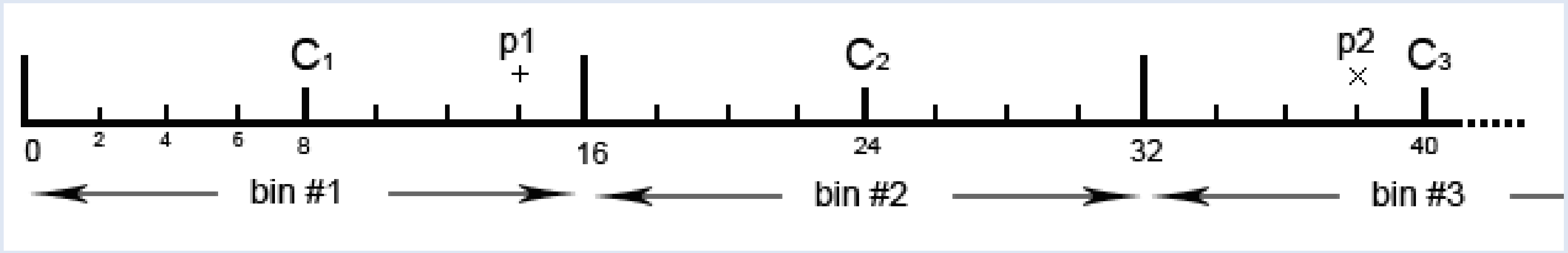
MPT: Histograms Building

Histogram of a patch on the grey level is not robust:

(small variation of light –> the histogram changes)

We use votes in two bins to handles this effect:

p1 votes: 0.6 in bin#1, 0.4 in bin#2



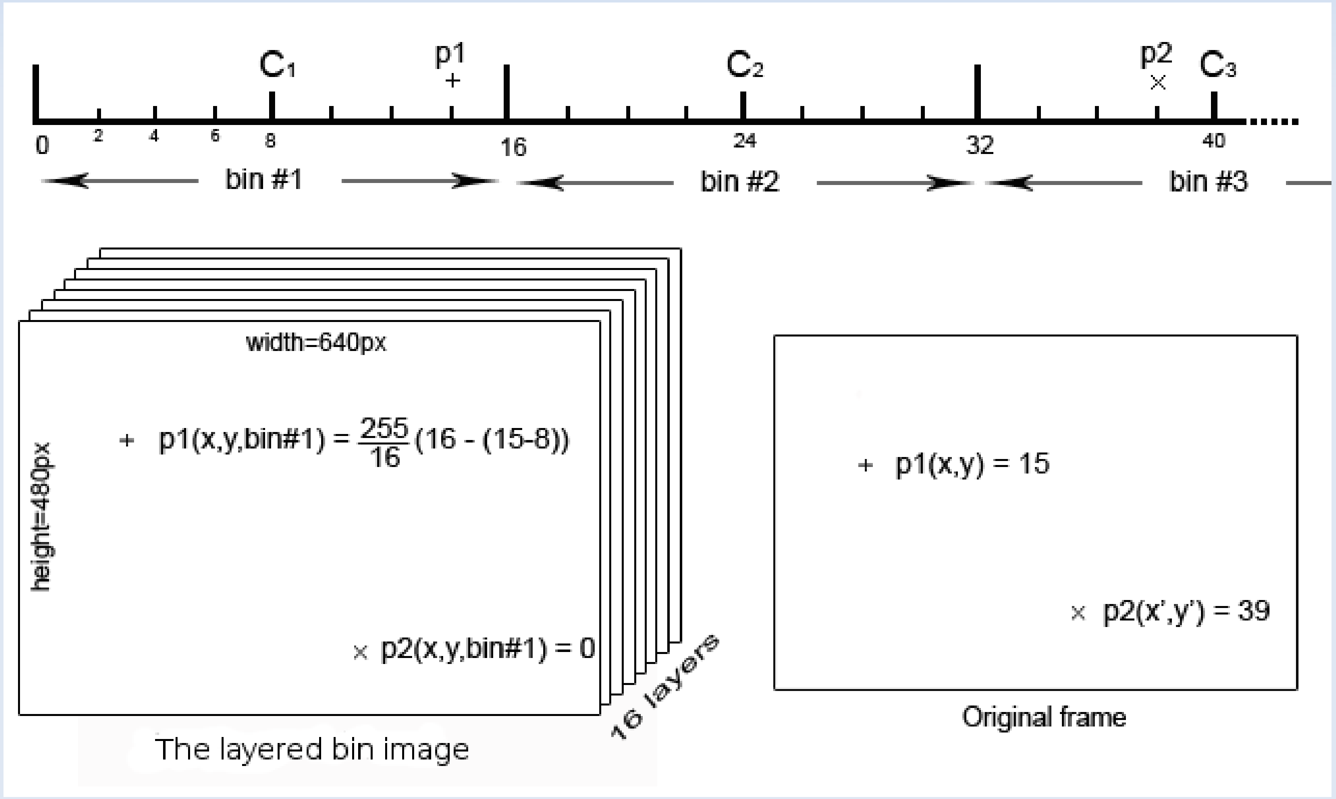
Computing issue:

To find the match of a patch on the next frame we need to look at all possible patch histogram on the next frame.

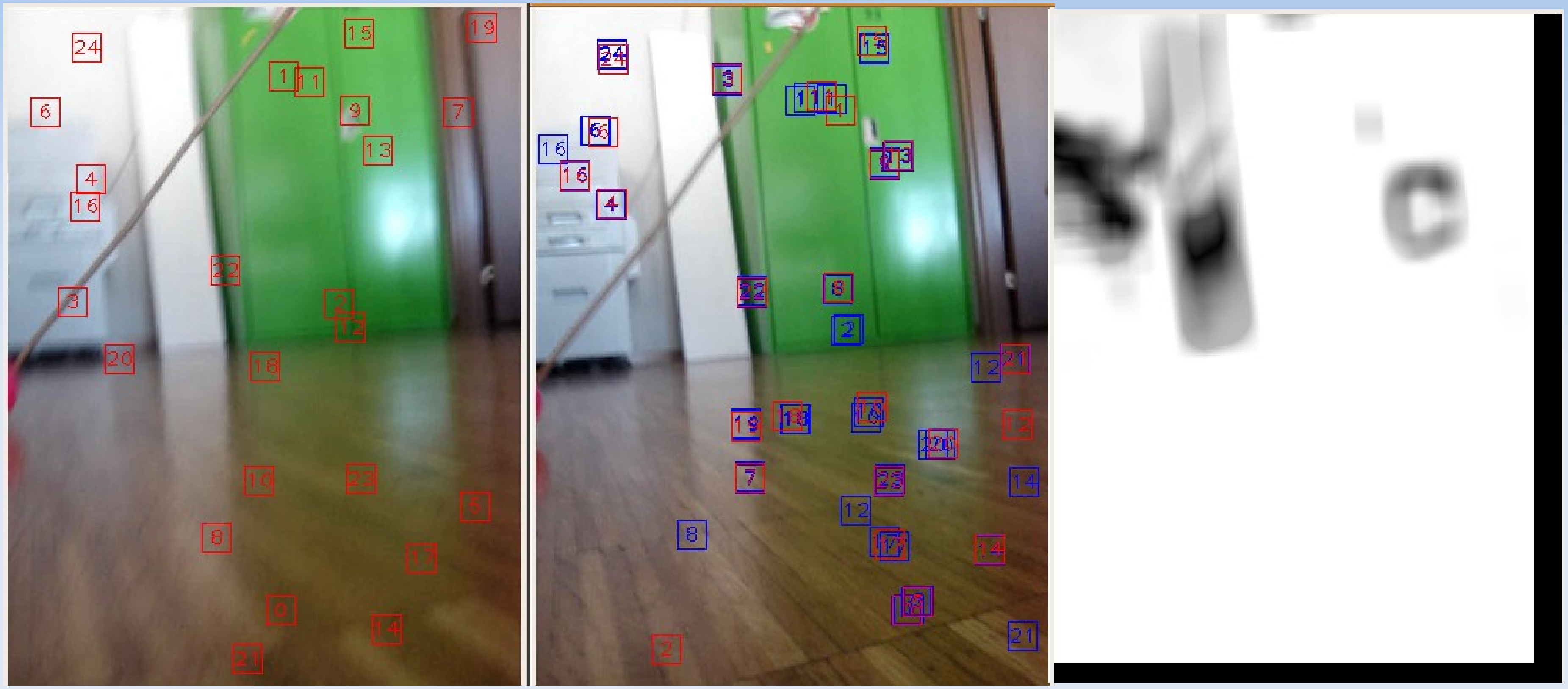
MPT : Layered bin image

Each pixel votes in 2 layers of our layered bin image

Exhaustive search of matching: Integral Image



MPT: Best matching patches

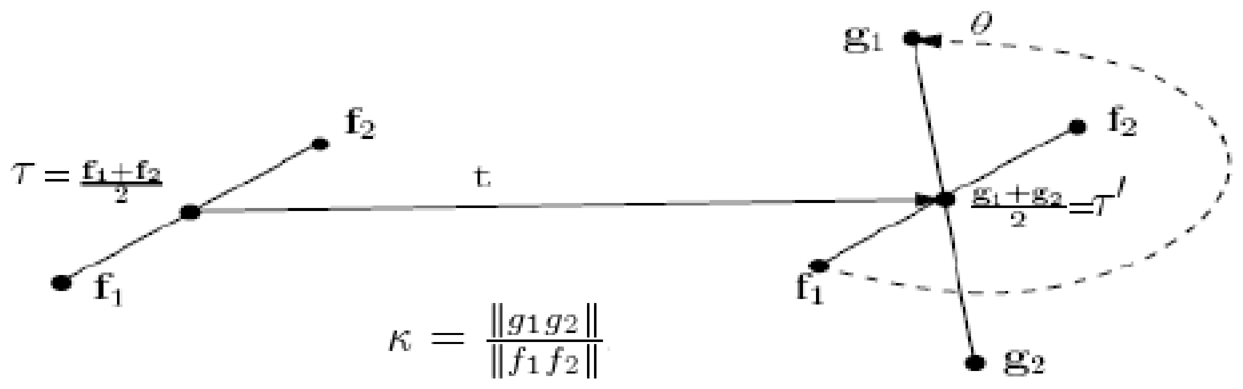


Left : 50 randomly selected patches.
Middle: Corresponding 3 best matching patches.
Right: Matching error of patch 4.

MPT : Finding transformation

Giving two pairs of points we are able to compute a similitude transformation.

$$T = \begin{vmatrix} I & \tau \\ 0^t & 1 \end{vmatrix} \begin{vmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} \kappa & 0 & 0 \\ 0 & \kappa & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} I & \tau' \\ 0^t & 1 \end{vmatrix}$$


g_1
θ
f_2
f_2
$\tau = \frac{f_1+f_2}{2}$
t
$\frac{g_1+g_2}{2} = \tau'$
f_1
f_1
$\kappa = \frac{\|g_1 g_2\|}{\|f_1 f_2\|}$
g_2

Find the transformation that fit to the maximum number of matching: RANSAC

MPT: A round of RANSAC

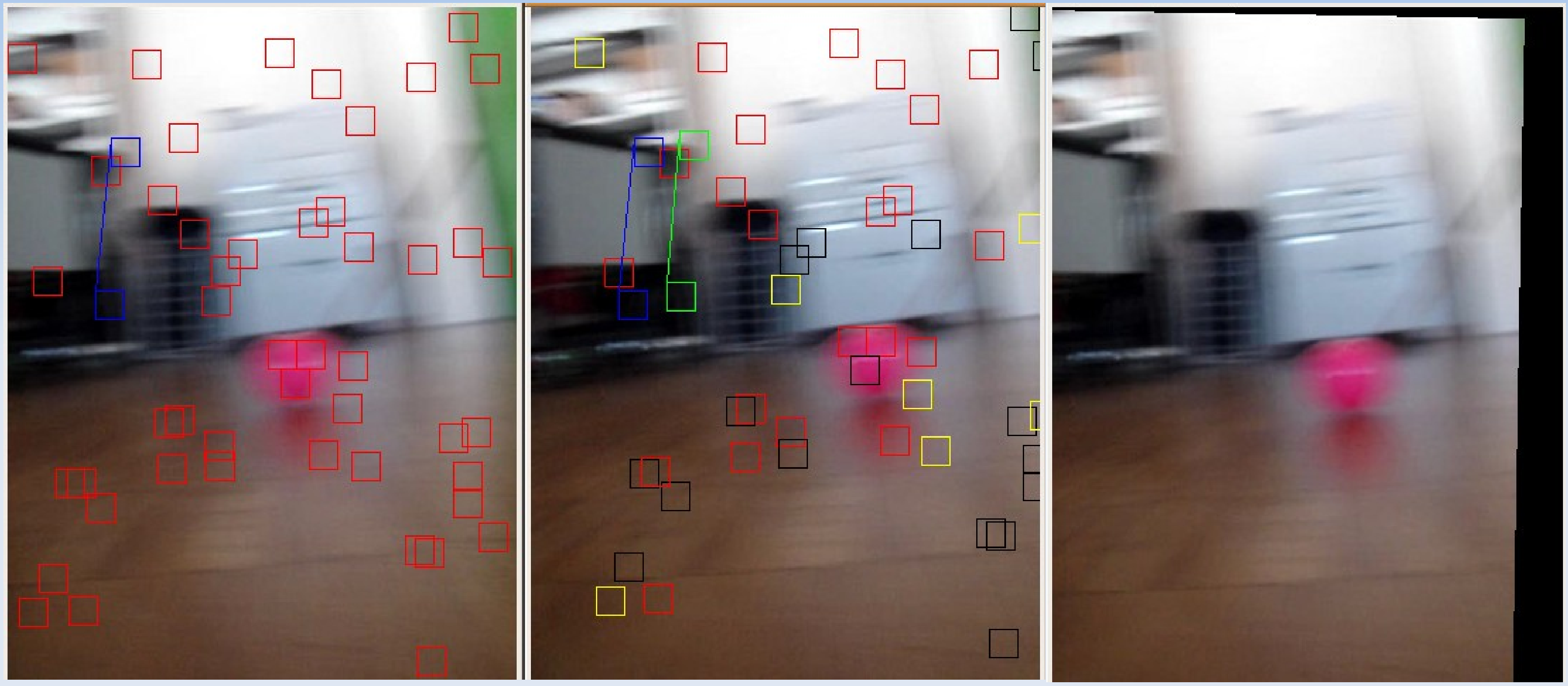
- Take two random source patches.
- Compute for each corresponding top best matching a transformation ***T***.
- Computing the quality of the transformation ***T*** with the other patches.

In our specific version:

- Use robust distance error.

- Handle point that are sent outside the image field.

MPT: Results



Left: Frame(n) and the source patches
Middle: Frame(n+1) and the T(source patch)
(Red:Good, Yellow:Medium, Black: Bad)
Right: Warp image corresponding to T(Frame (n+1))

MPT: Conclusion

Computation cost too heavy.

Computing a sequence of transformation can help to:

- Find the relative position of the robot.
- Rectify video stream for tele-operator