Université Pierre et Marie Curie

Master IMA
PIMA

Project report

Student : Mario Viti
mario.viti@etu.upmc.fr
Teacher : Nicolas Bredeche

30 mai 2017

0.1 Project presentation

0.1.1 Presentation

Realization of an embedded image processing/vision system solution for Raspberry [1] powered Thymio robots. Thymio-2 were originally designed at EPFL [2] and mostly used for educational experiments. They're easy to program, not expensive and have been extended with a Raspberry PI 3 equipped on-board camera and battery to create a standalone robot1. To design and run robotics experiment an ad hoc framework has been developed.

0.1.2 ThymioPYPI framework

ThymioPYPI [3] is a framework for programming a Thymio2 with a Raspberry PI in Python 2.X developed at ISIR Institut des Systèmes Intelligents et de Robotique [4]. Experiments constitute the specific behaviours desired for the robots. ThymioPYPI provides a network infrastructure for shipping software to the Thymios and monitoring the distributed system.

0.1.3 Requirements

The main product of the PIMA project has been an extension module program to support experiments in which image processing and or computer vision tasks are involved, such as recognition of obstacles and other Thymios. This slave application must run transparently and at a minimum frequency of service fos of $10hz$. This constraint must be assured in order to avoid making action inconsistent with the sorrounding environment. In order to achieve this goal, implemntation choices need to take into account the threadoff between performance and precision. The requests for the system where to estimate distance and roation of the robots relatively to the camera point of view. Rotation estimation wasn't sufficiently satisfying so it will not be discussed and left as a next step for improvement. Each Thymio robot will be associated with a tag as shown in Figure 1.

0.1.4 Hardware and software

Raspberry Pi camera uses CSI port connection which can be accessed via custom API in python 2.X. This API provides efficient stream caputure via a VGA video port. Raspberry Pi avaliable at ISIR comes with a built from source 2.4 Opencv[7] library which is essetial for image processing tasks.

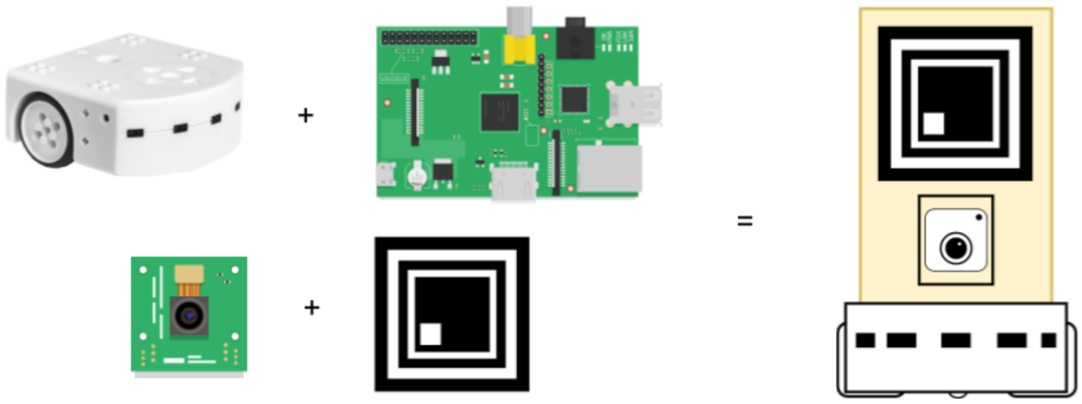


FIGURE 1 – Thymio robot with Raspberry Pi, Camera and Tag.

1. https://www.raspberrypi.org/documentation/
2. https://www.Thymio.org/fr:Thymio
3. https://github.com/nekonaute/ThymioPYPI
4. http://www.isir.upmc.fr/

0.2 Tag detection in real time

In this section a possible formalization of a realtime image processing application or detection task is proposed. A realtime image processing or detection task processes a stream of frames coming from a camera sensor and it can be defined as a service, slave or server application. As such, it must comply to provide the specified correct output within the other sole constraint of maximum time of service *tos* or minimum frequency of service *fos* imposed by the master or client. Thus it can be defined by the set *{outputs,tos}*.

In the specific case of a *tag detction* task, the correct outputs are : the contours of the tags present in the current frame, the identificators of each tag and the estimated distance from the camera.

When *tos* constraints in time are narrow, to minimize *tos* and maximize *fos* a realtime processing task can be divided in 2 phases : a *fine analysis* and a *coarse analysis*.

0.2.0.1 Fine analysis.

A fine analysis consists of the best effort analysis of the current frame, this step is usually more computationally intensive than a coarse analysis. For the proposed implmentation in tag detection **Algorithm 1** coincide with a fine analysis.

0.2.0.2 Coarse analysis.

The coarse analysis makes use of data acquired from the fine analysis, impling that fine analysis takes place before coarse analysis, for example as an hypothesis we assume that ideally little amount of change occours from frame to frame, therefore previous informations are still valid and coarse analysis uses fine analysis results to confirm or refuse the previously acquired beliefs and also tries to estimate current results as precisely as possible. For the proposed implmentation in tag detection **Algorithm 2** coincide with the coarse analysis.

```
Data: currentFrame
Result: tagIDs, tagCountours, tagDistances
edgeImage = Canny(currentFrame);
tagCountours = chekAreaRatio(edgeImage);
tagImages = rectifyImage(currentFrame,tagCoutours);
tagIDs = readID(tagImages);
tagDistances = estimateDistances(tagCoutours);
```

Algorithm 1: tagDetection

```
Data: previousTagContours,previousFrame,currentFrame
Result: tagCountours, tagDistances
tagContours = LKpyramid(previousFrame,currentFrame,previousTagContours);
tagDistances = estimateDistances(tagCoutours);
```

Algorithm 2: motionDetecion

```
frameCounter = 0;
frameBufferingLenght = 2;
Global running;
while running do
    currentFrame = readFrameFromCamera();
    if frameCounter%frameBufferLenght == 0 then
        tagIDs, tagCountours, tagDistances = tagDetection(currentFrame);
    else
        tagCountours, tagDistances = motionDetecion(tagCountours,currentFrame,previousFrame);
    end
    frameCounter = (frameCounter+1)%frameBufferLenght;
    previousFrame = CurrentFrame;
end
```

Algorithm 3: postProcessing

Algorithm 3 combines the two phases analysis and merges the results. The resulting *tos* will be $tos_{avg} = \frac{tos_{fine}+tos_{coarse}}{2}$ with $tos_{coarse} < tos_{fine}$ we'll achieve better performances on average.

0.2.1 Edge detection

For edge detection it has been adopted the popular solution of the Canny[1] edge detection algorithm implementation in Opencv [5]. This gradient based algorithm is suited to the task as it provides a luminance invariant simplification of the scene, meeting the requirement of tag detection in different lighting settings. For this realtime application it has been used a varinat to the threshold values proposed by Canny in his papaer for the *histeresis filter* (Figure 2) [6] : min-max values are calculated starting from the mean $\hat{x} \pm \epsilon$ luminance value sampled from the image. Thus sampling the whole image is costly, sample mean is computed by sampling different sub regions of the image in a *Montecarlo* fashion. Connectivity is vital to the tag detection as tags are defined by enclosed sets of contours (more on the definition of the tag later in this document),this solution has prooved to be more effective than canonical values in detecting tags in different light settings.

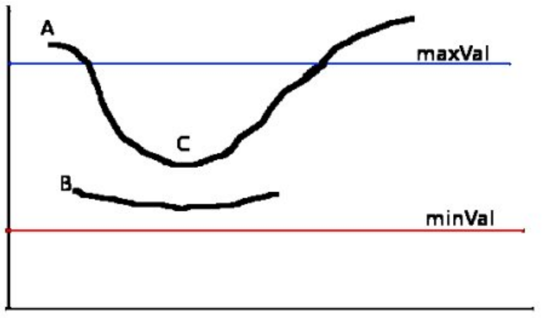


FIGURE 2 – Illustrating an example of a non connected edge thresholded by histeresis.

0.2.2 Motion detection

For motion detection another popular solution has been adopted : Lucas Kanade optical flow in its pyramidal variant also implemnted in OpenCV [7]. *"A matching process that must accommodate large displacements can be very expensive to compute. Simple intuition suggests that if large displacements can be computed using low resolution image information great savings in computation will be achieved. Higher resolution information can then be used to improve the accuracy of displacement estimation by incrementally estimating small displacemen"*[2]. Although a hierarchical or multi resolution or pyramidal implementation of a motion field detector is independent of the field model, a pyramidal implementation of the LK optical flow alghotrithm can detect large quantities of motion as the constancy hypothesis of small amount of motion due to the linear approximation to the first term of Taylor series expansion still holds when images are subsampled and the *brightness constancy constraint equation* is applyed to a small 5x5 pixels window. Furthermore as the purpose was to detect the shift of cornerlike features, this algorithm was best suited for the task.

0.2.2.1 Treadoff

As each level of the pyramid adds accuracy but also computations, the number of levels must be tuned accordingly in order to comply with the time of service constraints. For the presented solution a pyramid with 4 levels has been used with good results.

0.3 Software development

For the implementation of an image processing tasks a *Python* package *camera_ tools* has been developed to integrate the *ThymioPYPI* framework.

0.3.1 Design

The software is designed as a **metaDetector**.

The metaDetector design abstraction is defined at multiple levels : it is a set of functions { *preProcessing,postProcessing* } with the output of the *preProcessing* is piped to the *postProcessing* function as shown in **Algorithm 4**. Furthermore it is an **Interface** for image processing implemented as **Class**, tag detection is realized by *implementing* a *metaDetector*. The nature of this division, the details of the

5. `http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html?highlight=canny`
6. Image source OpenCV documentation
7. `http://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html`

preProcessing and postProcessing will be discussed further in the next section.

```
Data: frame
Result: postProcessingOutput
preProcessingOutput = preProcessing(frame);
postProcessingOutput = postProcessing(preProcessingOutput);
```

Algorithm 4: metaDetector

For a more detailed description on the usage of the software it has been released the *ThymioPYPI manual.*[8]. The most important modules of the package are :

camera_tools package :

— CameraController : handles RaspberryPi camera module PiCamera.

— ImageProcessor : holds the CameraController and handles the buffer in which pixels are stored by applying preProcessing and postProcessing to it.

Detector : exposes the ImageProcessor capabilities by defining the metaDetector Interface.

— TagDetector : implements the metaDetector interface.

— image_utils : collection of functions for image processing.

— tag_recognition : collection of functions for tag detection.

8. `https://github.com/nekonaute/ThymioPYPI/blob/master/Robocologie_manual.odt`

0.3.1.1 Realtime image processing

The *Raspberry 3 b* is equipped with a *1.2 GHz 64-bit quad-core ARM Cortex-A53* which allows different threads to execute on different cores. A realtime application can exploit multithreading by definig a set of independet tasks. We'll define 3 types of independent tasks in our application.

1. **capture** : this task is mostly camera dependent and it evolves capturing the image from the camera and converting to readable format.
2. **preProcessing** : in the previous section this was described as one part of the meta algorithm for image processing, it refers to any task that involves "non heavy" (linear) computation on images i.e. RGB to HSV conversion .
3. **postProcessing** : in the previous section this was described as one part of the meta algorithm for image processing, it refers to any task that involves "heavy" (non linear) computation on images i.e. EdgeDetection involving Convolution.

This set of tasks defines a general realtime application running on a on board solution with or wihtout GPU support to maximize throughput.

Figure 3, shows how the 3 tasks need to be secured by locking critical session on shared buffer among threads. *PreProcessing* function computional requirement is a suggestion as it's applyed to a shared buffer, sharing buffers limits memory usage but serializes computation. As an hypothesis keeping linear computation to the *PreProcessing* step will ensure a capture request returning after *PreProcessing* ended and a copy of the buffer is kept for *postProcessing*.

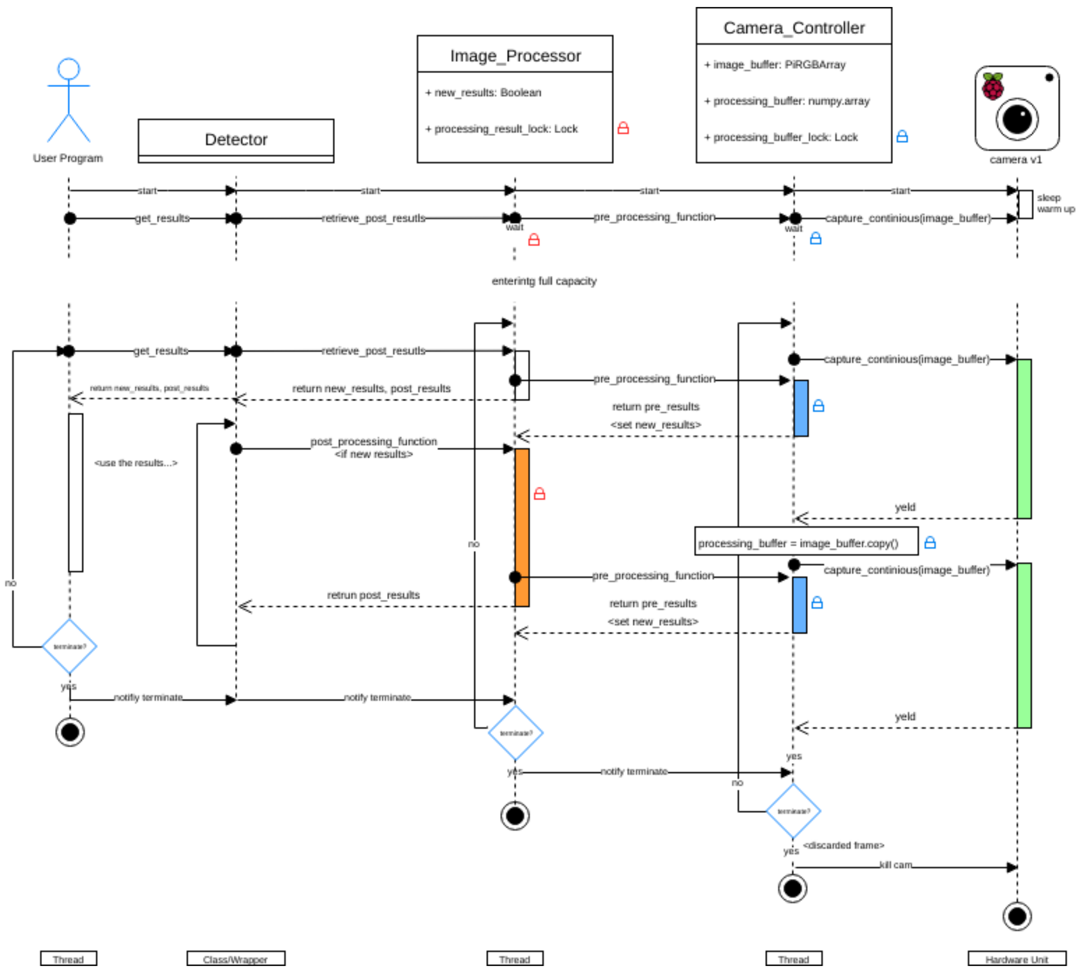


FIGURE 3 – lifecycle of a image processing realtime application.

0.4 Tags

0.4.1 Encoding

An encoded tag is modeled as a discreete bi-dimensional serie $e(x, y) = \sum_{i,j}^{n,m} a_{i,j} h(x - iS_b, y - jS_b), n, m \in \mathbb{N}$ with $h(x, y) = Arect_{S_b}(x - S_b/2, y - S_b/2)$ and T_b the *space per bit* and $a_{i,j} \in \{-1, 1\}$, the wave shape $h(x, y)$ and the simbol $a_{i,j}$ will encode the information in binary format. For better precision $m - ary$ encoding are better avoided because of the high noise in the communication channel.

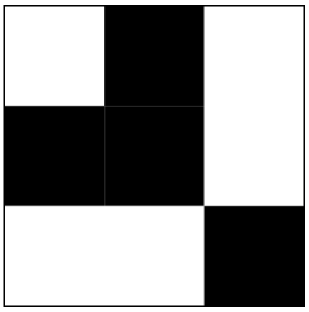


FIGURE 4 – an example of a 3x3 bits encoded tag.

The noise in the communication channel is modeled as an additive white gaussian noise or **AWGN** with constant dsp equal to N_0. In a one dimensional encoding while in presence of this noise applied to a signal just described the signal to noise ratio **SNR** of an SLI filter or receptor $g_r(t)$ is maximixed by applying the *Cauchy-Schwarz* theorem.

$$SNR = \frac{\int_{-\infty}^{+\infty} g_r(t)h(t_0 - t)dt}{\sqrt{N_0 \int_{-\infty}^{+\infty} g_r(t)^2 dt}} \leq \frac{\sqrt{\int_{-\infty}^{+\infty} h(t_0 - t)^2 dt}}{\sqrt{N_0}}$$

From which it is derived the analitical form of the optimal adaptive filter $g_r(t) \propto h(t_0 - t)$, this can be applied to the 2D case without loss of generality $gr(x, y) \propto h(x, y)$.

0.4.2 Reading

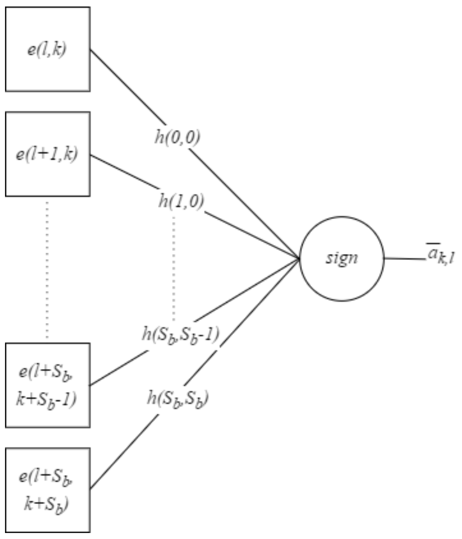


FIGURE 5 – perceptron reading one bit.

The sequence of symbols received $\hat{a}_{k,l}$ is the result of the sampling of the signal processed by the adaptive filter which consists of a meaningful term, or simply signal and a noise term. By means of the adaptive filter the signal term will be greater than the noise term, ideally.

$$\hat{a}_{l,k} = sign((h * h)(S_b/2, S_b/2)a_{l,k} + (n * h)(l, k))$$

with $n(x, y)$ being the **AWGN** random signal and $(h * h)(T_b/2, T_b/2)a_{l,k} > (n * h)(l, k)$ respectively the meaningful and noise term.

The discreete version of $(h * h)(T_b/2, T_b/2)a_{l,k}$ is a dot product $\sum_{i,j}^{S_b,S_b} Arect_{S_b}(i - S_b/2, j - S_b/2)e(l + i, k + j)$.

The computation involved in this "decisional process" is exactly the same that occours when using a *perceptron.* In this case however the *weitghts* for the linear classification ar not learned but computed anallitically, as in theory this is the optimal filter, a perceptron will tend to this solution.

0.4.3 Perspective invariant feature detection

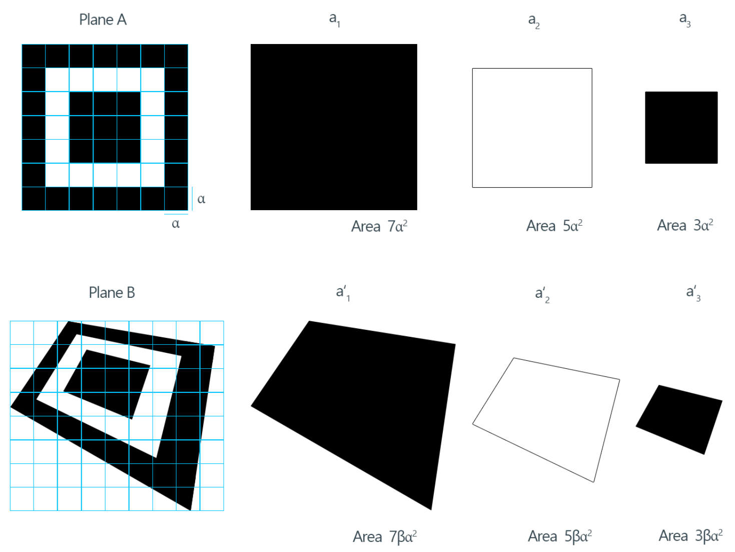


FIGURE 6 – area ratios for example tag with perspective transformation.

The first step in tag detection is to extract perspective covariant descriptors that minimize the complexity of the scene from which compute perspective invariant descriptors. From a pair of two closed edges present in the image we can compute area ratios, closed edges represent the covariant descriptos from which a one dimensional feature is then extracted. This procedure can be used to identify a tag with a known area ratio within closed edges present in the tag itself, this technique has been first used in a very popular techonlogy : the QR code[3].

The next phase involves the apriori knowledge of the shape of the tag. Once the tag countour is identified in the frame, the warp or *perspective transform matrix* can be estimated and inverted, by applying the inverse prespective transformation to the pixelse within the detected contour we obtain a rectified image that can be read like as an encoded image as described previously.

0.4.4 Tag description

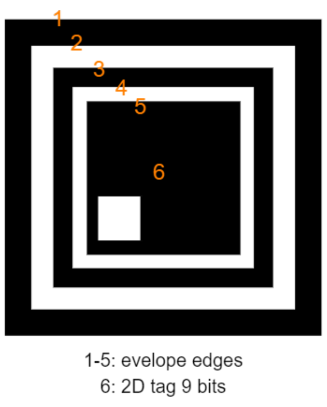


FIGURE 7 – custom Thymio Tags.

A custom tag has been developed for the Thymio robots as seen in Figure 1. It is contained into several layers of edges to ensures no false positives. A custom tag has been developed for the Thymio robots and has been installed on the robots as seen in Figure 1. In Figure 7 it is shown the hierarchical structure of edges from the outermost to the innermost there are 3 generation of edges. For each edge i a predecessor is the edge numbered with a number j and $i < j$, between a father and a child the area ratio is know and it is used for a more roboust detection. The innermost square contains the information or tag id encoded as described previously. The ids are encoded into 9 bits yelding 512 possible configuration allthough two among these are better avoided (0 and 511).

0.5 Results

The tag detection system met the requirements of a minimum $fos \geq 10hz$ and reaching a $12hz$ this performences however are achieved in optimal settings, problem arose wile testing and will be discussed in this section and the next.

0.5.1 Tracking evaluation

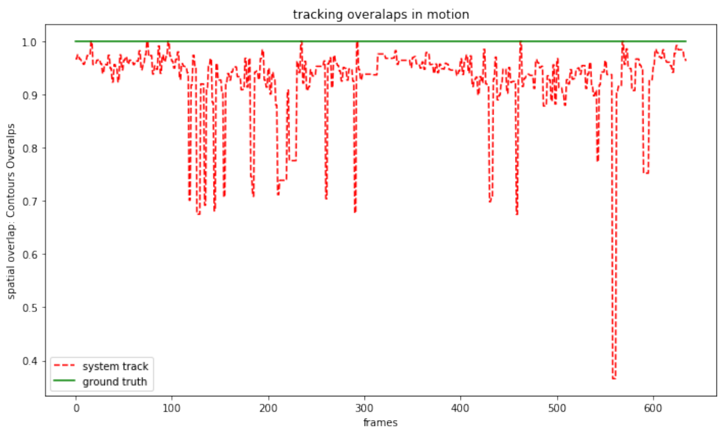


FIGURE 8 – different experimental settings.

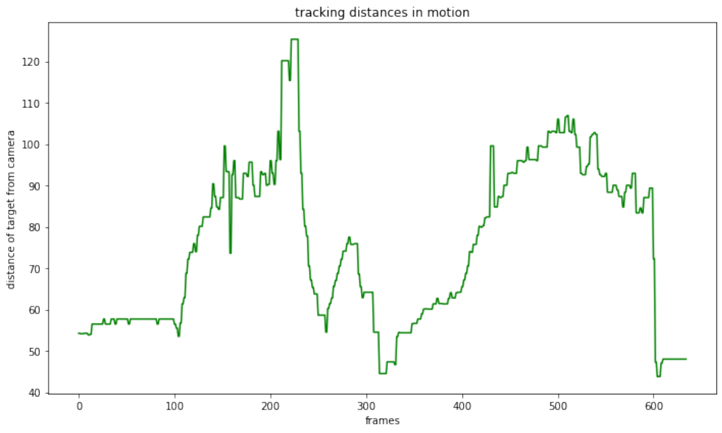


FIGURE 9 – different experimental settings.

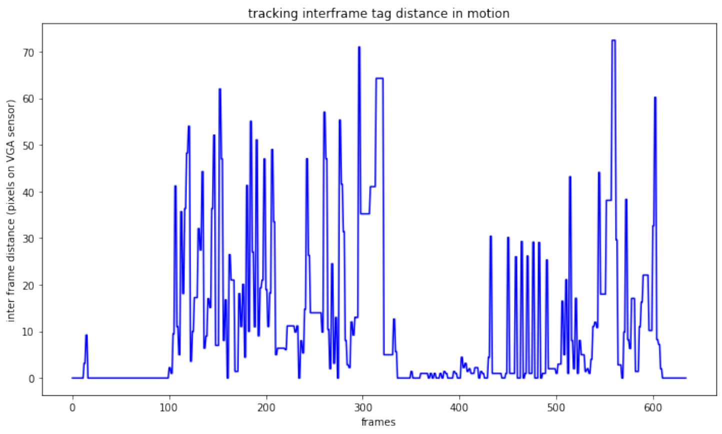


FIGURE 10 – different experimental settings.

This tag detection application can be evaluated as a **tracking system** as desrcibed in[4]. We will use [4] as a guideline for metrics and performance estimation practices as these has been used on other popular tracking systems. As described in [4] *"First we will define the concepts of spatial and temporal overlap between tracks, which are required to quantify the level of matching between Ground Truth (GT) tracks and System (ST) tracks, both in space and time. The spatial overlap is defined as the overlapping level $A(GTi, STi)$ between GTi and STi tracks in a specific frame k :"*

$$A(GTik, STik) = \frac{Area(GTik \cap STik}{Area(GTik \cup STik}$$

In order to evaulate the performances an experiment has been set up in the arena with Thymio robots. A robot moved in front of the camera randomly, the tracking is executed once by using the *fine analysis* on each frame (GT), subsequently the tracking is done by intervealing *fine analysis* and *coarse analysis*(ST) based on motion estimation, with the cotours overlaps are computed for each frame (Figure 8). For each frame, distance from the camera (Figure 9) and interframe distance (Figure 10) are registered in order to correlate how distance from the camera and velocity affects the tracking system.

False alarm track (FAT) of False Positive (FP) :

[4] *"Although it is easy for human operators to realise what is a false alarm track (event) even in complex situation, it is hard for an automated system to do so. Here, we give a practical definition of false alarm track".*

$$FAT = \frac{\sum_{k=0}^{n} A(GTik, STik)}{n} < thr$$

, n = number of frames, $thr = 0.6$.

Track matching error (TME) :

[4] *"This metric measures the positional error of system tracks. TME is*

the average distance error between a system track and the GT track. The smaller the TME, the better the accuracy of the system track will be." In this application the distance is normalized with the maximal distance possible as the diagonal of the VGA window :

$$TME = \frac{\sum_{k=0}^{n} Dist(GTik, STjk)}{n(VGA_d)}$$

[4] *"where Dist() is the Euclidean distance between the centroids of GT and the system track : TMED is the standard deviation of distance errors"* In Figure 11 two tracks are overalapped as they appear on screen, specifically only the center of the contour is registered.

— FAT : 0.005
— TME : 0.02
— TMED : 0.01

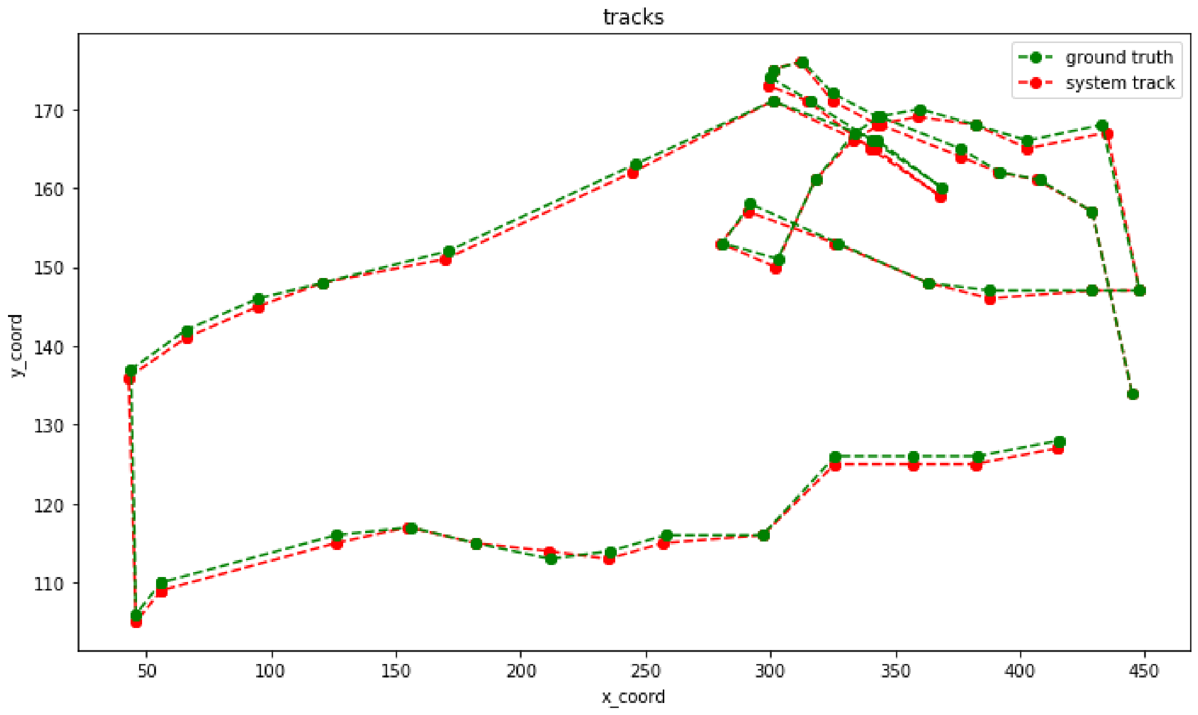


FIGURE 11 – different experimental settings.

0.5.2 Experiments

0.5.2.1 Validation

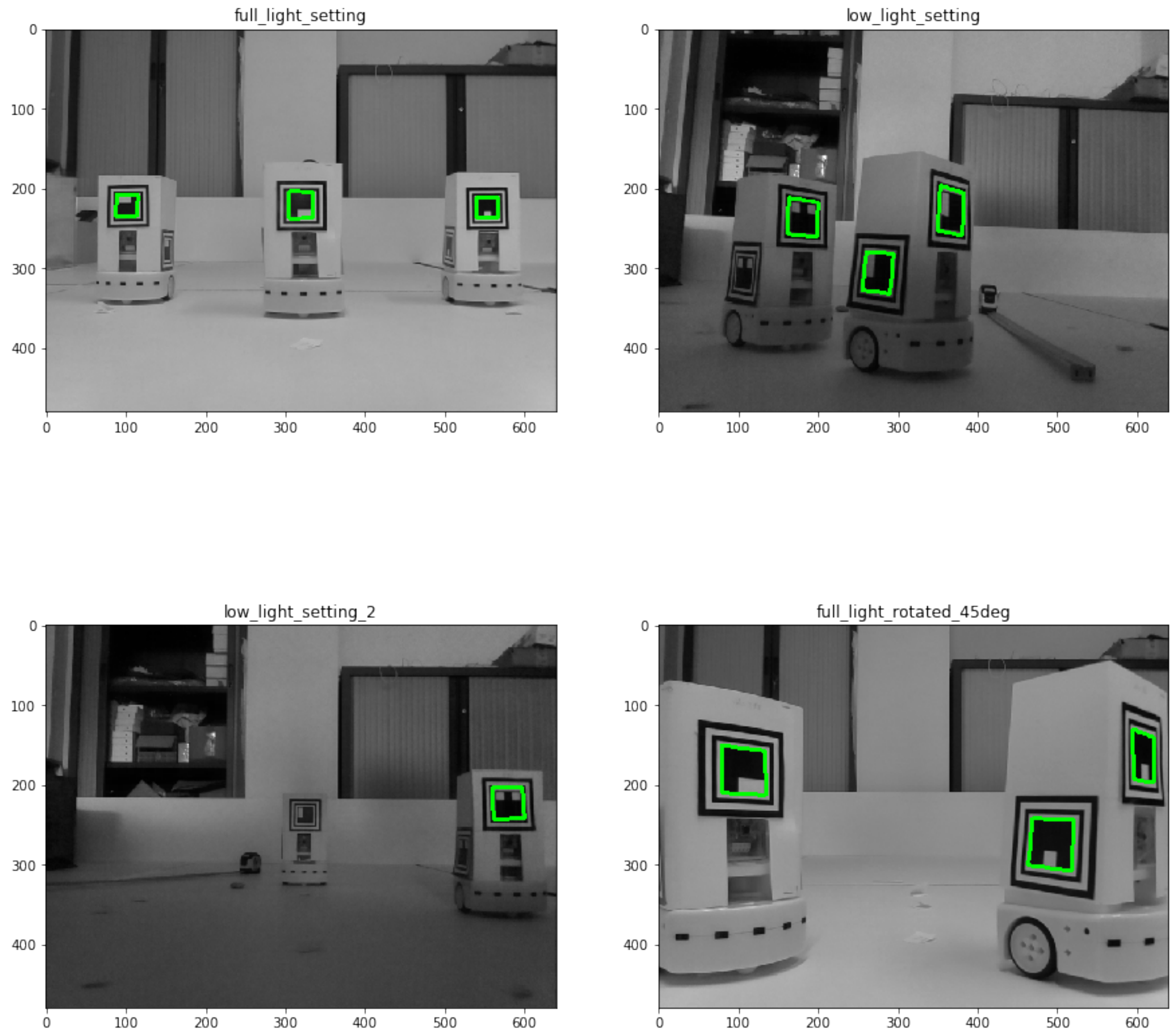


FIGURE 12 – different experimental settings.

In order to evaulate the performances an experiment has been set up in the arena with Thymio robots. Earch robot was positioned in a precise location and it was shown to the camera. This experiment is repeated in two different settings, with a full light setting with homogeneus light coming from artificial lamps and a low light settings wiht a noisy background, non homogeneus light and a gradient like soft shadow overlaying the scene. It is one last time repeated by changing the rotation with respecto ro the camera of 45°.

In order to measure the performance it has been defined a per second *frequency of detecion* as $\frac{hit_t}{max_t(hit_t)}$ where a *hit* occours when tag t is detected a number hit_t of times per second and $max_t(hit_t) = fos$.

Caveat These results are contingent to the setting : the size of the tag and the definition of the sensor influence the results as performance are in funtion of the area in pixels of the tag on the sensor which is a function of the tag real dimensions and the camera definition.

Settings :
- tag innermost side size : 4cm
- sensor definition : VGA (640,480)

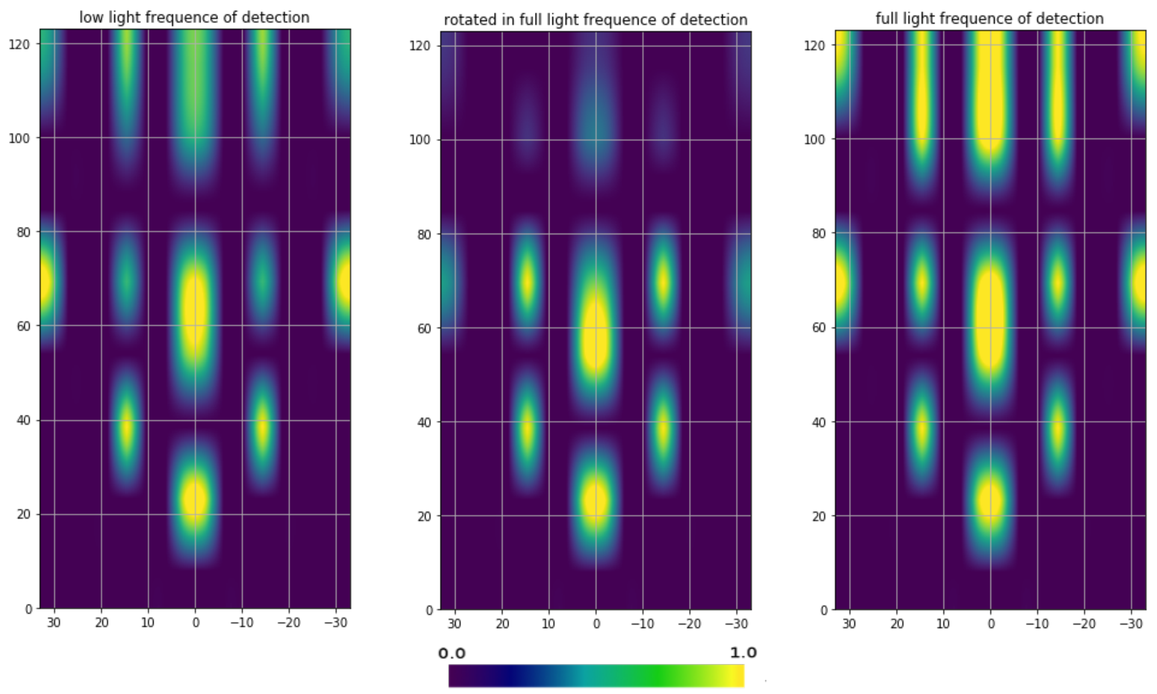


FIGURE 13 – frequencies of detecion against positions (in cm).

Distance estimation has been evaluated agaist real distance from the camera. In presence of low light and noise it's noticeable an higher variance in the error. Also it seems that a systematic bias is present in distance estimation when distance augments this is probably due to the sampling of a continuos shape on a discrete grid which makes the apparent size different from the real size.

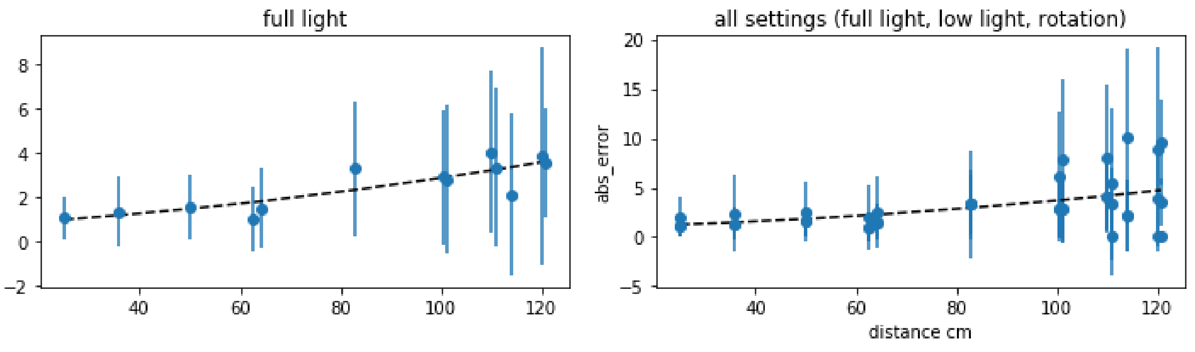


FIGURE 14 – absolute error of real and estimated distance againts real distance, bars indicates variance in the absolute error.

0.5.2.2 Follow tag

After validation a simple experiment has been designed to test the tag detection software. These experiment were designed to proof how the tag detection improves the behaviour of the robot and its interaction with the sorroundings. The experiment were incrementally difficult for the robots.

Experiment 1. There are 2 Thymio robots acting in this experiment (Figure 15 left) each robot moves foreward and follow the tag when in view. *Thymio1* tries to reach a target, *Thymio2* just follows the first.

Experiment 2. (Figure 15 center) This experiment is opposite to experiment 1, there's no increase in difficulty.

Experiment 3. (Figure 15 right) this experiment increases difficulty for the Thymio robots. There are still 2 actors in this setting but the behaviour changes : each robot must avoid each other and try to reach the target (which is identified by a different id).

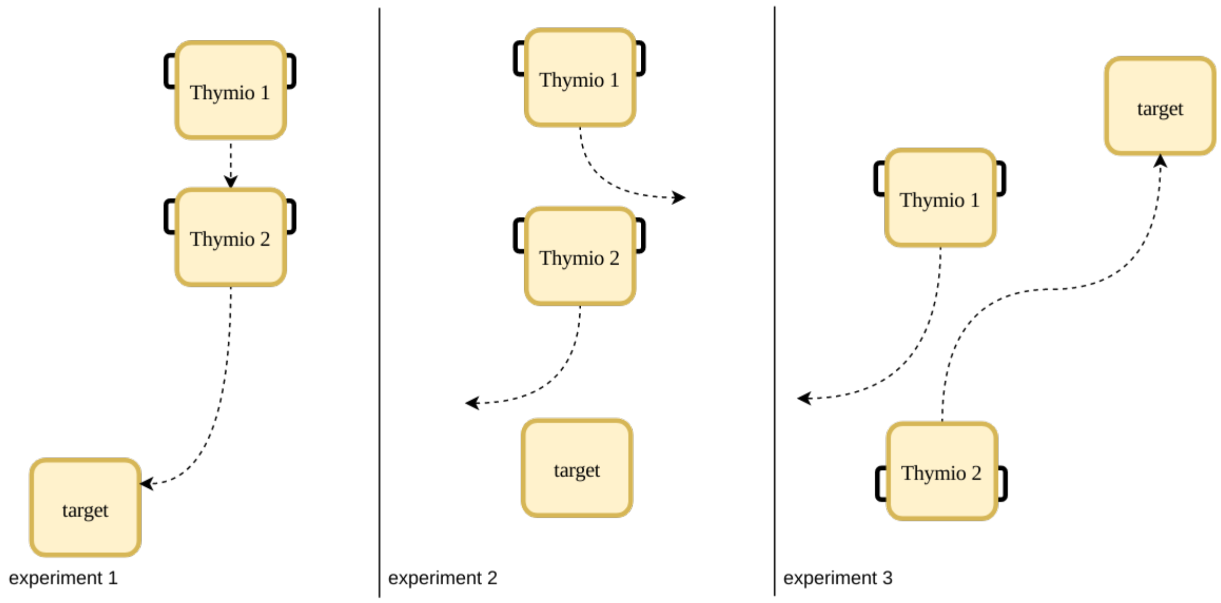
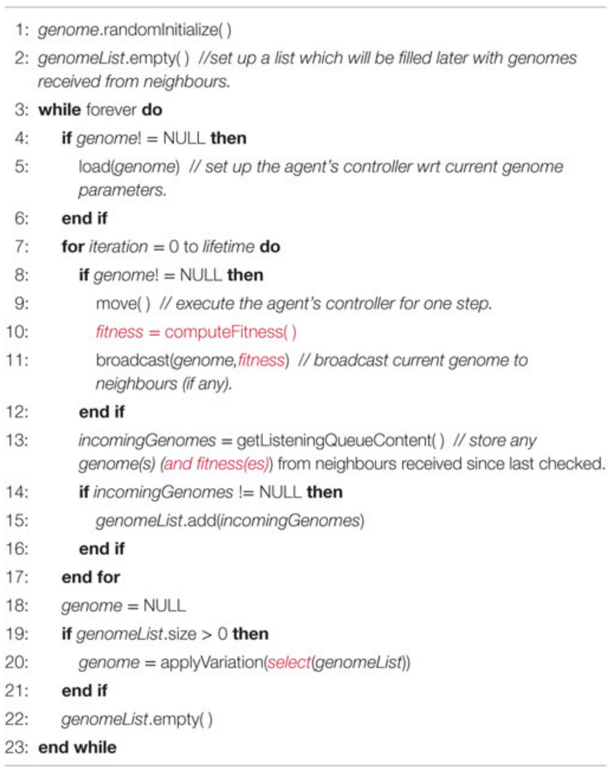


FIGURE 15 – Experiment follow tag setting 1 and 2.

0.5.2.3 Evolutionary behaviours



```
genome.randomInitialize( )
genomeList.empty( ) //set up a list which will be filled later with genomes
  received from neighbours.
while forever do
    if genome! = NULL then
        load(genome) // set up the agent's controller wrt current genome
        parameters.
    end if
    for iteration = 0 to lifetime do
        if genome! = NULL then
            move( ) // execute the agent's controller for one step.
            fitness = computeFitness( )
            broadcast(genome,fitness) // broadcast current genome to
            neighbours (if any).
        end if
        incomingGenomes = getListeningQueueContent( ) // store any
        genome(s) (and fitness(es)) from neighbours received since last checked.
        if incomingGenomes != NULL then
            genomeList.add(incomingGenomes)
        end if
    end for
    genome = NULL
    if genomeList.size > 0 then
        genome = applyVariation(select(genomeList))
    end if
    genomeList.empty( )
end while
```

FIGURE 16 – Basic "vanilla" version of embodied evolutionary robotics algorithm[5].

As a final experiment I've been proposed to team up with other two students in M1 Androide Parhams Shams and Tanguy Soto. As a part of a team I could test how could the developed software support an advanced robotics experiment. Embodied evolutionary robotics describes a large area of evolutionary genetics algorythm applyed to robotics. Parhams Shams and Tanguy Soto implemented in their PANDROIDE an evolutionary algorithm for obstacle avoidance descripted in [5] and [6], first in a non distribuited fashion, so that the best behaviour was selected among the exemplar behaviour or fitting by a central system maximizing the fitness and broadcasting the genetic information, and a second distribuited version to be executed on the Thymio robots in an areana. The fitness in this case is a series of parameters like the distance or the number of times hitting an obstacle, genetic infomations ore genome are a series of parameters for motors and sensors tuning the behaviour of the robots, the behaviour associated to an higher fitness is rewarded and carryed on as good genetic material, bad genetic material is discarded (natural selection), there are many variants for fitting functions and the selection of parameters and the ones implemented were proposed in [6].

In Figure 16[5] a basic "vanilla" version of a embodied evuolutionary robotics algorithm is described in pseudocode. In the distributed version the genome is passed among neighbours, to experience its whereabouts a robot must be able to recognize other robots so that the instruction *getListenintQueue-Countet()* and *broadcastGenomes(genome,fitness)* can be implemented correctly.

The tag detection application served to sense neighbours and by binding tags to an ip address the robots could communicate theredfore share genetic information based on their proximity.

0.5.3 Limits

Motion blur affects largely tag detection, blurred edges lead to poor edge localization and this affects the detection capacity of the Canny algorithm. This is particularly noticeable when point of view is moving, a problem that has been overlooked when testing the system.

0.6 Discussion

Motion blur is currently the biggest obstacle that this implementation is facing. A complicated solution could be to deconvolve the motion blur with a known a priori point spread function PSF, but it is an expensive procedure and it may not work for all tags as they may be affected by different "amounts" of blur, so a single PSF is not meaningfull enough, a simpler solution could be binarizationm thresholding, this method will eventually find edges even in presence of poor localization, the downside of such method is that finding the optimal threshold is not a simple task. It has been envisioned a solution which involves getting the threshold by applying k-means for luminance quantization on the current image histogram and using the quatization values as thresholds similarly to the *Otsu* method but with less parameters and with multiple thresholds. Something similar has been tried in early stages of the implementation but subsequently abandoned because of poor performances. Another possibility is to use a multiresolution system, motion blur affects the localization of edges but when definition is reduced and the image sub-sampled with nearest interpolation, delocalization of edges decrease, the motion detection analysis is also easier at it would need less levels. A qualitative test in this direction has been run upon suggestion of the supervisor Nicolas Bredeche and it looked promising. Still another possibility is to use machine learning to fill in when detection is interrupted because of motion blur, another interesting solution would be to use "active tags" or led signals, this kind of tags would be easily detectable but this simplification comes with an increase in the cost of the robots.

Bibliographie

[1] J. Canny, "A Computational Approach to Edge Detection," vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986.

[2] JJ.R. Bergen, P. Anandan, K. J. Hanna, R. Hingorani, "Hierarchical model-based motion estimation", Computer Vision - ECCV'92 pp. 237-252, May 1992.

[3] M. Hara, M. Watabe, T. Nojiri, T. Nagaya, Y. Uchiyama, "Optically readable twodimensional code and method and apparatus using the same" US 5 726 435, 10 mar 1998.

[4] F. Yin, D. Makris, S. Velastin and Digital Imaging Research Centre, Kingston University London, UK "Performance Evaluation of Object Tracking Algorithms" Computer Vision – ACCV 2006, pp. 151–161, Jan. 2006.

[5] N. Bredeche, E. Haasdijk, A. E. Eiben, On-Line, "On-Board Evolution of Robot Controllers", in Artifical Evolution : 9th International Conference, Evolution Artificielle, EA, 2009, Strasbourg, France, pp. 110-121, Oct 2009.

[6] J.M. Montanier, S. Carrignon, N. Bredeche "Behavioral specialization in embodied evolutionary robotics : Why so Difficult ?" Front. Robot. AI vol.3 art 38, Jul 2016.

[7] OpenCV Computer Vision Library. [Online]. Accessible : http ://opencv.org/ [Accessed :23-May-2017].