# Computer vision for Eurobot 2013

Ondřej Staněk

2013-06-06

**Abstract**

Computer vision plays important role in mobile robotics. Robot equipped with a camera can retrieve various information about its surroundings. It can recognize objects and perform operations that would be difficult or even impossible without the ability of "robotic sight". Web camera is a cheap computer gadget, yet very advanced and universal sensor for a mobile robot.

However, digital camera produces loads of image data and its processing is demanding both on the computational performance of hardware and complexity of the software algorithms. Fortunately, an open-source computer vision library; OpenCV is available. The library makes a very nice framework for computer vision applications and includes the key image-processing algorithms, all of them ready to work out of the box.

In the past, it was a hassle to make the OpenCV library work on arbitrary hardware platform. Compiling the library from scratch was quite deterring for many potential users. Happily, an OpenCV port for Android devices was introduced. Installing the OpenCV library is then a matter of two taps in the Android Market. Every smart phone is equipped with a camera and it has reasonable processing power to run image processing algorithms. In light of these facts, a smart phone is ideal candidate for a "brain" of a small mobile robot.

This paper describes the use of Android smart phone for robotic machine vision and mobile robot control. A simple mobile robot with 8-bit microcontroller was transformed to advanced camera-enabled robot, just by mounting a smart phone to it. This camera upgrade greatly extended the range of tasks the robot can perform in the Eurobot 2013 competition.

Apart from solving a concrete computer vision problem, we also describe theoretical background of detecting circles in image, using Hough transform in particular.

# Part I
# Analysis

Eurobot is a popular robotic contest that calls for use of computer vision techniques to handle complex tasks, such as object detection and manipulation. In this part, one of the Eurobot 2013 contest activities is described and analyzed from the computer vision standpoint.

## 1 The Eurobot contest

Robotic competitions introduce every year new challenges for teams of students and individuals. One of these competitions is Eurobot, an international amateur robotics contest, organized since 1998. It is aimed mainly to graduate students in technical fields. Every year, new rules are published and new game elements are presented. This year (2013), robots are supposed to celebrate their birthday. Robots must unwrap gifts in order to reveal their content, they should serve drinks to guests, put cherries on the cake and last but not least, blow out as many candles on the cake as possible. In this paper, we focus only on the task of blowing out the candles.

### 1.1 Blowing out the "candles"

Goal of the project is to implement an algorithm for identification and classification of specific game objects of the Eurobot 2013 competition. The objects of interests are called "candles". A "candle" is a game element that consists of a green tennis ball and supporting cylinder that holds the tennis ball. The color of the supporting cylinder is either white, red or blue. There are many "candles" distributed evenly on a half-circular cake, the position of the candles is specified in rules and does not vary. However, the colors of the supporting cylinders has to be identified by the robot. The robot will go around the candles, identify them and classify them according to the color of the base cylinder. Based on the recognized color of the candle, robot may perform an action of blowing out the candle. It will use its manipulator to push the tennis ball inside the hollow cylinder, until the ball dispersals completely. To achieve this, robot has to recognize the position of the candle precisely, so that it can perform the action.

## 2 Camera Feedback for Robot Positioning

The robot has to move to a position so that the candle of interest is exactly in front of the robot and in a specified distance. Only then, the robot can trigger its manipulator to push the ball inside the cylinder.

A feedback loop mechanism will be used for positioning the robot. The robot will acquire and process image from camera, and it will calculate the position

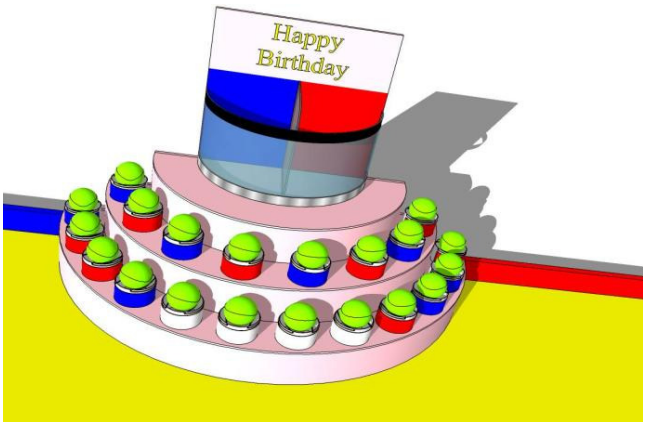


Figure 1: Eurobot 2013 - the cake with candles

and size of the candle in the image. Then, it will drive its motors so that it gets the candle to the center of the image and it will move towards the candle until it is close enough to perform the action.

# 3 Candle Recognition

The main feature of each candle is the yellow-green tennis ball which represents candle flame. Whereas the color of the base cylinder varies, the color of the tennis ball is always the same, so it makes sense to focus on the tennis ball recognition in the first place. However, because the tennis ball is partially hidden in the cylinder, it does not form a complete circle in the captured image. The tennis ball detection has to be designed so that it overcomes this limitation.

# 4 Hough Transform for Circle Detection

Hough transform is a general method of finding basic geometrical shapes in digital images. In this paper we focus on particular application, how the general concept of Hough transform is used to identify imperfect circles in the image acquired from camera.

Circle of radius $r$ and center $(a, b)$ in Cartesian coordinates is a set of points $\{(x, y)\}$ that satisfy the analytical equation

$$(x - a)^2 + (y - b)^2 = r^2 \tag{1}$$

When we are given radius $r$ and circle center $(a, b)$, we can use the equation to construct a raster image of that circle. That's a basic task of computer graphics. However, in image analysis, we face an inverse problem. Given a binary raster

image (set of points $I = \{(x, y)\}$), we want to find circle parameters $a$, $b$ and $r$ so that the equation (1) holds for *almost*[1] all points $(x, y)$ in set[2] $I$.

Strictly mathematically speaking, Hough Circle Transform $\mathcal{H}$ is a mapping from the image space $(x, y)$ to the parameter space $(r, a, b)$; $\mathcal{H} : \mathbb{R}^2 \to \mathbb{R}^3$. How does the mapping work? The best way is to think of it as an inverse operation to "Circle drawing". "Circle drawing" $\mathcal{C}$ is a mapping from parameter space $(r, a, b)$ to image space $(x, y)$; $\mathcal{C} : \mathbb{R}^3 \to \mathbb{R}^2$.

The mapping $\mathcal{C}$ maps any arbitrary set of parameters $P \subseteq \mathbb{R}^3$ to an image $I \subseteq \mathbb{R}^2$:

$$\mathcal{C}(P) = \{(x, y) | (x - a)^2 + (y - b)^2 = r^2 \wedge (r, a, b) \in P\}$$

Then, we simply define Hough Circle Transform $\mathcal{H}$ as inverse of the "Circle drawing" transform $\mathcal{C}$:

$$\mathcal{H}(I) = P \iff \mathcal{C}(P) = I$$

Please note that transformation $\mathcal{H}$, due to its definition, only transforms geometrically perfect images of circles. If image $I$ is not a perfect circle image, then $\mathcal{H}(I)$ is not defined for such $I$. Also keep in mind the images considered are not actually finite bitmaps as we know them from computer graphic, but rather an infinite sets in vector space $\mathbb{R}^2$.

To address the circle detection in finite bitmaps, we need to define the Hough transform in a different way. To be precise, we will first define the terms we are working with:

- binary bitmap image $I$ is any (finite) set $I \subset \mathbb{N}^2$. We say that image $I$ has spacial dimension $M \times N$ if $M, N \in \mathbb{N}$ are the lowest integers for which this formula holds:

$$(x, y) \in I \Rightarrow x < M \wedge y < N$$

- Observation: The space $\mathfrak{I}$ of all binary bitmap images is the power set of $\mathbb{N}^2$:

$$\mathfrak{I} = \mathcal{P}(\mathbb{N}^2) = 2^{\mathbb{N}^2}$$

- The circle $p$ in parametric form is a triplet $p = (r, a, b)$, $r \in \mathbb{N}$, $a, b \in \mathbb{Z}$. Note that we consider only discrete values of parameters.

- The circle parameter space $\mathfrak{P}$ accommodates the triplets $(r, a, b)$.

$$\mathfrak{P} = \mathbb{N} \times \mathbb{Z}^2$$

- The parameter accumulator $\mathcal{A}$ is a function that for every parameter triplet $(r, a, b) \in \mathfrak{P}$ assigns the frequency of occurrence $c \in \mathbb{N}$.

$$\mathcal{A} : \mathfrak{P} \to \mathbb{N}$$

[1] not all points, due to expected imperfection in the source image
[2] assuming the set is sufficiently big to represent the circle

- The set of all parameter accumulators $\mathcal{A}$ is the parameter accumulator space $\boldsymbol{A}$.
- Hough transform $\underline{\mathcal{H}}$ for binary bitmap images is a mapping from the binary bitmap image space $\mathfrak{I}$ to a parameter accumulator space $\boldsymbol{A}$

$$\underline{\mathcal{H}} : \mathfrak{I} \to \boldsymbol{A}$$

For every bitmap image $I$, the Hough transform finds an accumulator $\underline{\mathcal{H}}(I) = \mathcal{A}$ which is defined as follows:

$$\mathcal{A}((r,a,b)) = |\{\mathcal{U}_{x,y} | (r,a,b) \in \mathcal{U}_{x,y}, (x,y) \in I\}|$$

The set of parameters $\mathcal{U}_{x,y} \subseteq \mathfrak{P}$ is constructed for for every point $(x,y)$ in the bitmap $I$:

$$\mathcal{U}_{x,y} = \left\{ (\text{Round}(r), \text{Round}(a), \text{Round}(b)) \,|(x-a)^2 + (y-b)^2 = r^2, r > 0, a, b, r \in \mathbb{R} \right\}$$

The Hough transform $\underline{\mathcal{H}} : \mathfrak{I} \to \boldsymbol{A}$ for binary bitmap images actually does not give the circle parameters directly, as the general Hough transform $\mathcal{H} : \mathbb{R}^2 \to \mathbb{R}^3$ would. Instead, it gives us the parameter accumulator $\mathcal{A} \in \boldsymbol{A}$. This allows to define the Hough transform $\underline{\mathcal{H}}$ for any input bitmap. The circles (from parameter space $\mathfrak{P}$) have to be determined by investigating the returned parameter accumulator $\mathcal{A}$.

So, to get the parametric expression for all circles represented in bitmap $I$, we need to investigate the accumulator function $\mathcal{A} = \mathcal{H}(I)$. For every possible circle $(r,a,b)$ the accumulator function gives the occurrence number $c = \mathcal{A}((r,a,b))$. The number $c$ is the number of pixels in bitmap $I$ that represent the considered circle $(r,a,b)$. The higher is the number $c = \mathcal{A}((r,a,b))$, the higher is the likelihood that the circle $(r,a,b)$ is in represented in the image bitmap $I$.

To get the circle $p_{\max} = \{(r,a,b)\}$ with the most likelihood, we just need to find the maximum of the parameter accumulator $\mathcal{A}$:

$$p_{\max} = \arg \max_{(r,a,b) \in \mathfrak{P}} \mathcal{A}((r,a,b))$$

If we need to find more circles in the image, we are searching for local maxima of the accumulator function $\mathcal{A}$.

It is obvious that searching for a circle parameters is a matter of interpretation of the accumulator $\mathcal{A}$. We might claim further conditions on the local maxima to suppress detection of false circles. Especially, it is necessary to define a lower limit for the accumulator maxima, to ensure that no false circles are detected in image that actually does not contain any circles.

This was the complete description of the original Hough transform for circle detection in bitmaps. The original method, however is not efficient, as calculating the complete three-dimensional parameter space takes a lot of of processing time. Therefore, in OpenCV library, a more effective method is implemented. We will describe the effective method for finding circles in the next part.

# Part II
# Implementation

## 5 OpenCV library (for Android)

OpenCV library has to be installed as a stand-alone Android application. It is available in the Android Market, so the installation to any Android device is just a matter of two taps. No settings or adjustments to the library is necessary, the library works out the of box. Once installed, it provides an OpenCV environment for other Java Android applications.

## 6 Hough Circle Transform in OpenCV

OpenCV library implements Hough transform for circles. As pointed out in the Analysis part, classical Hough Circle Transform calculates a 3-dimensional parameter accumulator space. To recall, the three dimensions are the center coordinates $x$, $y$ and circle radius $r$. Calculating the accumulator volume would need a lot of memory and time - the original method is ineffective.

In OpenCV, the *Hough gradient method* is implemented for finding the circles. This method uses only 2-dimensional accumulator and thus it is more effecient. The Hough gradient method takes a bitmap image as an argument. First, it performs an edge detection[3] on the source image. That yields the outlines of all shapes in the image. Then it calculates the Sobel derivatives[4]. These derivatives are used for an approximation of the gradient at every nonzero point in the bitmap. Assuming that every nonzero pint is a part of some circle, the calculated gradient at that point is orthogonal to the tangent of the possible circle. The gradient points in the direction towards[5] to the circle center.

For every pixel, we draw that line into the 2-dimensional $x$, $y$ accumulator plane. These lines will intersect, and the accumulator value will be higher at points where lines intersects the most. The local maxima of the accumulator plane are the possible circle centers.

However, this method has some shortcomings. First, it has a problem with detection of concentric circles. Second, the radius of the circle has to be determined additionally, we just described how to get the circle centers. The implementation of Hough gradient method in OpenCV calculates the radius as well, but the documentation warns that the value might not be very accurate and if better accuracy is requested, the programmer should implement another method for detecting the circle radius.

There is one improvement to this method that allows to specify the maximum and minimum radius of circles we want to detect. The trick is to draw just

[3] using the cvCanny() function
[4] using cvSobel()
[5] or outwards, we cannot tell, but for sure the slope is approximately the same

line segments to the accumulator, not the whole lines. The bounds of the line segments are determined by the minimum and maximum radius to detect. OpenCV implements this feature. This is highly beneficial, not only it simplifies filtering the output, but it also reduces computation time.

# 7 Tennis ball recognition

Figure 2: Original image (recognized circle marked)

## 7.1 Masking out the tennis ball pixels

The tennis ball has a very specific yellow-green color, that is unique in the picture. To mask out all the pixels that belong to the tennis ball, we convert the input image to HSV[6] color space and then we select just the pixels that are in the specified HSV range. That gives us a binary bitmap mask, shown in figure 3.

As shown on the sample image, not all pixels of the tennis ball were masked. That's because the tennis ball has white lines and black lettering. For the later Hough transform, we need to get rid of such artifacts in the masked image.

## 7.2 Morphological operations

We use two morphological operations to suppress the artifacts in the masked image [1]. Image *opening* and *closing* operations are the way to go. The *closing*

[6] Hue, Saturation, Value

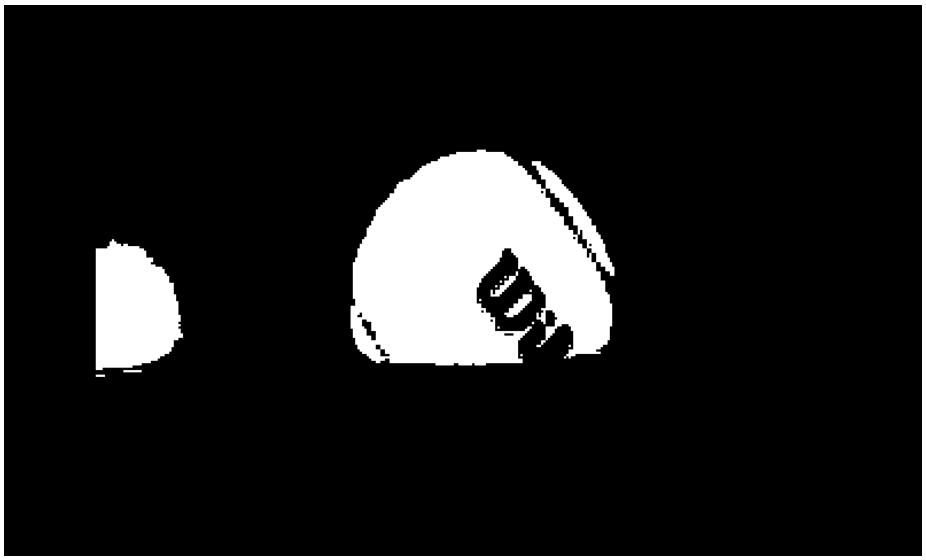

Figure 3: Masked image

morphological operation removes the small black parts inside the ball by growing the foreground. Of course, this has the side effect of growing the ball outline. Therefore, after closing the image, we apply opening operation, that contracts the foreground and scales it back to original shape.

### 7.3 Hough transform

Then, we apply a Gaussian blur to reduce noise and avoid false circle detection, as suggested in the OpenCV documentation [2].

Finally, we pass the blurred gray-scale image to the *HoughCircles()* OpenCV function. Some additional parameters have to be specified and fine tuned, for instance:

- the minimum distance of the circle centers
- the interval for circle radii
- accumulator resolution

Tuning these parameters was done experimentally, observing the results in real environment. The *HoughCircles()* function returns an array of circles that were found in the image (pixel coordinates of the center $(a, b)$ and radius $r$).

## 8 Camera feedback for robot positioning

Our goal is to position the robot so that the ball is directly in front of robot's manipulator. Once we know the precise position of a ball in the captured image,

Figure 4: Mask after morphological operations and Gaussain blur

we can drive the motors of the robot so that the tennis ball moves in the image towards the center. That means the circle in the image has to be center aligned and of a specified radius. First, the robot moves so that it gets the ball in the center of the camera image. Then, if the radius of the ball is too small, robot moves towards the ball until the radius is within a specified range. Only then it can perform an action with its manipulator.

This feedback algorithm was tested on real robot. The testing conditions were simplified - there was just one ball present on a test table and the light conditions were stable. In this settings, the robot performed quite well. However, when testing on a real Eurobot playground, this algorithm was not very robust. Due to varying light conditions and multiple balls, not all balls were detected correctly. The robot had a problem to fix to one ball, and when detection of the desired ball failed in one sample, it decided to move towards another ball on the image. That resulted in hesitant behavior of the robot. Moreover, the motors of the robot are undersized to the overall robot's weight, and together with the poor quality of wheel encoders, the movement of robot is very non-uniform. That was a big source of trouble for the feedback mechanism.

# 9 Simplification of the problem

With respect to all the problems mentioned in previous section, we decided to decrease degrees of freedom of the feedback system. The mechanical construction of the robot was improved, we added a frontal spacer element with wheels. This fixed the distance of the robot from the "cake" with candles (tennis balls)

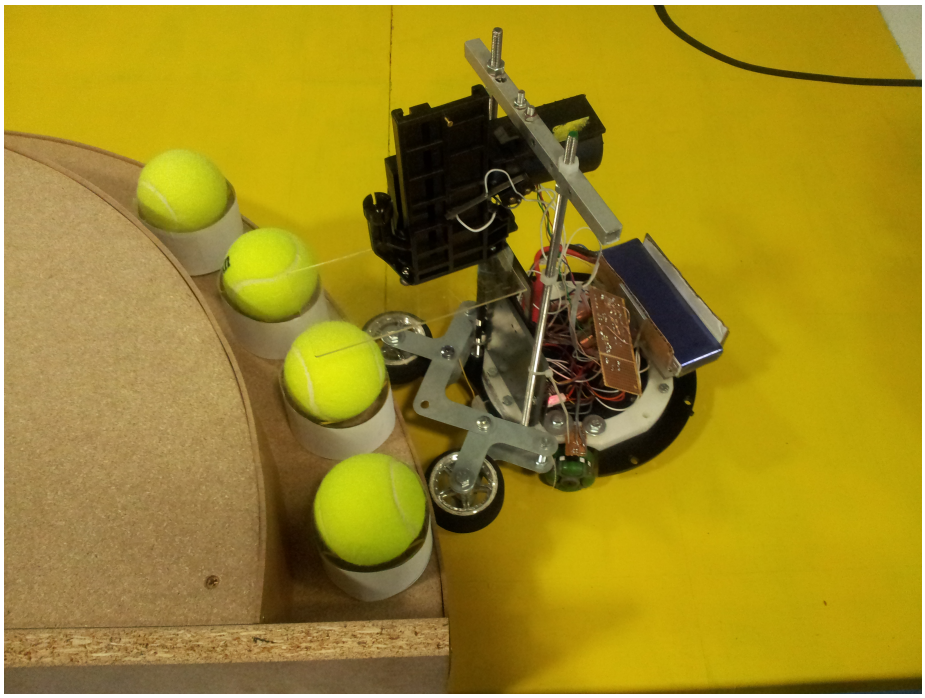

Figure 5: Hanuman robot - frontal spacing element with wheels

and eliminated two degrees of freedom. This design decision simplified the image processing and the feedback mechanism a lot. The robot is moving around the "cake" in a fixed distance and it's camera always looks towards the center of the cake. This has the nice benefit that in the processed image, balls does not overlap, the circle has fixed radius and the background is always the same.

The robot moves around the cake until it detects a tennis ball in the center of the screen. When this situation occurs, it stops and activate its manipulator to blow out the "candle". After finishing the action, it continues moving around the cake.

# 10 Robotic platform

## 10.1 Mechanics

The Hanuman robot has an omni-directional wheelframe. The three wheels allow movement in any arbitrary direction, or rotation in place. That makes the platform very flexible for solving different tasks. In front of the robot, there is a manipulator that pushes the tennis balls inside their supporting cylinders. Robot is equipped with a holder for a smart phone. The smart phone camera looks straight forward underneath the ball manipulator.

### 10.2 Electronics

The robot is controlled with a simple embedded system. On board, there is an ATmega128 8-bit microcontroller and supporting electronic circuity for controlling the motors. The robot is equipped with a standard Bluetooth serial RFCOMM adapter. This adapter plays the key role in communication with the smart phone, which runs the image processing application.

### 10.3 Firmware

The ATmega128 microcontroller is programmed in C language. The firmware implements drivers for motors, encoders, ball manipulator and other sensor. It also provides a useful service menu that can be accessed from any computer or smart phone via Bluetooth. Finally, it supports binary protocol for communication with the Android phone.

# Conclusion

A computer vision algorithm for Eurobot 2013 competition was developed and successfully tested. Android platform was proved to be well suited for image processing and robot control. Although the main focus of this project is the computer vision algorithm, a complete robotic system was developed, including mechanics, hardware and firmware. Testing the CV algorithm on the real robot proves the applicability of the developed software. During development, we faced problems with varying ambient light. Eventually, these problems were solved by enabling the integrated camera lightening. All parts of the complex robotic system work and cooperate together to accomplish the given task.

## References


[1] http://wiki.elphel.com/index.php?title=OpenCV_Tennis_balls_recognizing_tutorial

[2] http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html

[3] http://opencv.org/platforms/android.html

[4] Gary Bradski, Adrian Kaehler: Learning OpenCV, O'Reilly, September 2008: First Edition, ISBN: 978-0-596-51613-0