HS-GreenFist: Beach Cleaner Robot
Team Description Paper - LARC 2013. IEEE Open Category

Rel Guzman Apaza [*1], Edwin Gutierrez Linares [*2], Enrique A. Soto Mendoza [*3], Elvis D. Supo Colquehuanca [#4]

[*] *Systems Engineering, National University of St. Agustin*
Arequipa, Peru
[1] r.guzmanap@gmail.com
[2] edwin641@gmail.com
[3] e.sotomendoza@gmail.com

[#] *Electronic Engineering, National University of St. Agustin*
Arequipa, Peru
[4] elvis.supo@gmail.com

***Abstract*—This paper describes the development of the robot we made to compete in the Open category of LARC (Latin American Robotics Competition) 2013. The architecture and programming of the robot is designed to counter the environmental pollution problem; in this case the robot must be able to navigate on the beach, collect cans found in this area and transport them to a deposit.**

***Resumo*—Este artigo descreve o desenvolvimento do robô que fizemos para competir na categoria Open do LARC (Competição Latino Americana de Robótica) 2013. A arquitetura e programação do robô é projetado para combater o problema da poluição do meio ambiente, neste Caso o robô deve ser capaz de navegar na praia, coletar latas encontradas nesta área e transportá-los para um depósito.**

***Index Terms*—LARC, robotics, beach cleaner, computer vision, kinect, arduino**

I. Introduction

In the current context of environment protection and care, waste collectors robots loom large, because they locate, collect and dispose garbage in a controlled, autonomous and fast way. In this specific case it was implemented an autonomous robot capable to navigate in sand, collecting cans and transporting them to a particular deposit. Developing waste collectors robots is currently a research and investment matter, so this document presents an efficient solution to the problem of accumulation of garbage on the beaches. The name of the robot is HS-GreenFist developed by the group of students named HS-AQP.

This manuscript is organized as follows: in Section II is described the mechanical parts of the robot. Section III presents the computer vision system and the Kinect device used to perform obstacle avoidance and the cans detection. Section IV depicts the digital images processing and segmentation techniques .Section V shows how soda cans are classified and recognized. Section VI describes the electronic configuration with the arduino boards. Finally, the major conclusions of the work are drawn in Section VII.

II. Mechanical Structure

We decided to design a robot adapted to a low budget. The main structure is shown in Figure 1 on which all the components are mounted is made of steel and the body is formed of various materials among like MDF (Medium Density Fiberboard) and acrylic, while the arm is mainly made of acrylic to reduce it's weight. The design includes an arm that allows the robot to pick up the cans in any position, while a fixed "ramp" is responsible for depositing the cans aided by its own weight.

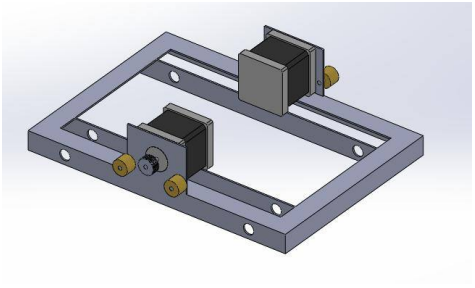


Figure 1: Main Structure

A. Locomotion

The robot uses four wide tires allowing it to move under the the same principle on which the caterpillar track perform its movement. It was initially thought of using caterpillar track, but the low pressure this could make, coupled with construction problems that might arise finally made us decide to use wide tires. The movements that are expected to accomplish are three:

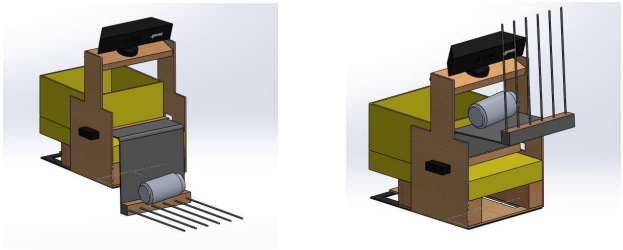
1) Takes the robot back and forth.
2) Rotate the robot on its axis both clockwise and anti-clockwise movement.
3) Correct the robot motion towards the left or right.

The movements of the motors are intended to be performed in two speeds to which we will simply call fast and slow.

B. Excavator Arm

The excavator arm is shown in 2 and consists of a single body with a claw on the end that allows the robot to collect

the cans through the sand, it will also have a small scanning system as an actuator fixed to the claw. The movement of the excavator arm is defined by two movements, one to raise or lower the arm itself, and another to ensure that the can into the claw. Movements are commanded by servomotors. The body of the excavator arm consists mainly of acrylic.



(a) Arm in "down" state. (b) Arm in "up" state

Figure 2: Excavator Arm

C. Ramp

The ramp shown in Figure 3 will deposit the cans in the container, and it's movement is controlled by a servomotor. It's made of cardboard for it's consistency and lightness, and then it allows the servo motor to use just a little torque.

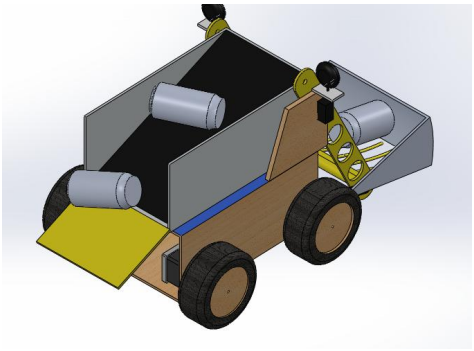


Figure 3: Ramp

III. Vision System

For autonomous systems, being aware of the surrounding environment is important for decision making. The idea of using two viewpoints of the scene with cameras to get depth information was originally developed by Charles Wheatstone (1832), with the stereoscope [1]. The Kinect was made according to this idea but with different hardware components described in the following subsection.

One of the most complex and crucial tasks in the proposed robot is the proper detection of cans and obstacles, and this is made using some algorithms from computer vision, described in the following sections.

A. Kinect

Device developed by Microsoft for the Xbox 360, it allows programmers to recognize gestures and voice commands controlling it [2]. The kinect components are shown in Figure 4.

Table I: Kinect Specifications, extracted from [3]

Feature	Specification
Field of view	58° horizontal, 43° vertical, 70° diagonal
Elevation Angle	+/-27 degrees
Frame rate (depth/color)	30 frames/second
Camera Resolution	640x512 px

We use it's depth sensing (IR Projector and IR Camera), RGB camera, and the motor. The Kinect is limited to it's hardware specifications showed in Table I.

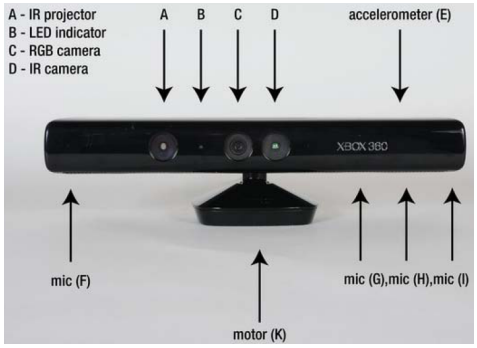


Figure 4: Kinect external component identification

We use this device because it already calculate a depth map, that is similar to the disparity map that is obtained with a stereo vision system, We found the operational range for recognizing depth by experimentation and it goes from $0.48m$ a $3.5m$,

B. Robot Vision

In the robot, the Kinect is positioned in the top allowing the RGB camera to see very near objects. We decided to put the Kinect inclined 45° to the front, and according to the specifications showed in Table I and the operational range found, we found useful to take advantage of the motor, and we define two states:

- The state M1 when the motor is moved 27° and the Kinect is inclined 72° below the horizontal.
- The state M2 when the motor is moved $-27°$ and the Kinect is inclined 18° below the horizontal.

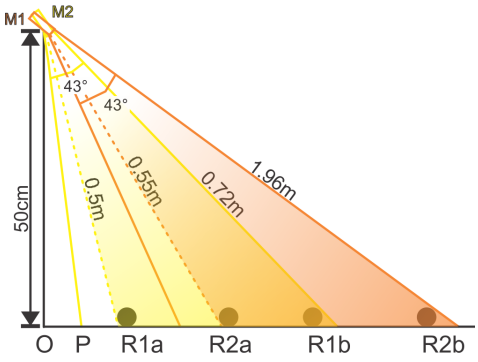


Figure 5: Representation of the Kinect Vision (Horizontal View)

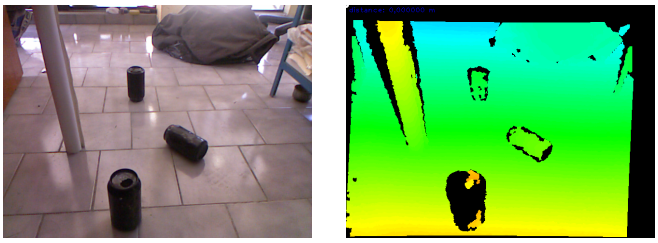
A representation of the Kinect Vision is shown in Figure 5, there is a the yellow region ($R1a$ to $R1b$) and an orange region ($R2a$ to $R2b$), the robot can determine depths being in states M1 and M2 correspondingly. The region between O and P where the robot don't see any object is very small, and in the region between P and $R1a$ the robot must use just the RGB camera. We measured all distances and the robot can recognize depth from $R1a = 36cm$ to $R2b = 1.86m$, then it recognize objects in a radio of $1.86m$.

C. Kinect Depth Sensing Process

The process of how to make the robot have an accurate position in the real world and to avoid obstacles can be determined in two steps and do the process we used OpenNI that has a middle-ware for the Kinect as well as a basic driver, and OpenCV for the video processing.

1) **Calibration:** Align the RGB and depth data. According to [4] OpenNI has it's predefined parameters for calibration and it's not necessary to do the process.
2) **Re-projection:** Get distances from depth map by triangulation. We use the predefined functions from [5] we get the following:
   - `CV_CAP_OPENNI_BGR_IMAGE` for 8 bit BGR Image (converted from the RGB signal of the camera)
   - `CV_CAP_OPENNI_DISPARITY_MAP}` for 16 bit disparity map from calibrated camera.
   - `CV_CAP_OPENNI_POINT_CLOUD_MAP` for 3D Reprojection Matrix

We transform the 16 bit disparity map from the Kinect, to 8 bit for visualization, both BGR Image and disparity map are shown in Figure 6.



(a) BGR Image (b) Colorized depth map

Figure 6: Inputs from the Kinect

IV. Image Processing

With the aim to get the most accuracy target, we need to process the incoming images first with pre-processing and segmentation techniques.

A. Pre-Processing

We need pre-process the video frames in such environment conditions where the illumination is variable, to do this we use some techniques in this order:

1) **Image Enhancement using Single Scale Retinex (SSR)**: Using the Theory of Retinex proposed by [6], in [7] is proposed an algorithm to accomplish the process of enhancement using neighborhood operations. Retinex takes an input digital image I and produces an output image R on a pixel by pixel basis as in equation 1, getting the image in Figure 7.

$$R_i(x,y) = \log(I_i(x,y)) - \log[F_i(x,y) * I_i(x,y)] \quad (1)$$

where i is the color channel, because it's applied to each channel in the BGR image, and $*$ is the convolution operation. In [8], the matrix F is the Gaussian Kernel 2, and we use this approach.

$$F(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

2) **Gaussian Blur with $7x7$ kernel**: We use the same kernel to remove the noise produced by the camera and the previous enhancement process.

Figure 7: Image enhanced with Retinex and Gaussian blur

B. Segmentation

As all technology people know, commonly images are represented in the RGB color space, unfortunately when we talk computer vision RGB values vary too much depending of light conditions. In contrast HSV color space is much better in this way; Hue depicts the color; saturation is the quantity color or its purity and finally the value represents the light intensity. To get the regions of interest where the cans are located, we first use a color segmentation limiting the image color values in HSV color space, $0 \leq H \leq 180$, $0 \leq S \leq 50$, $0 \leq V \leq 70$, and we can get the resulting image in Figure 8.

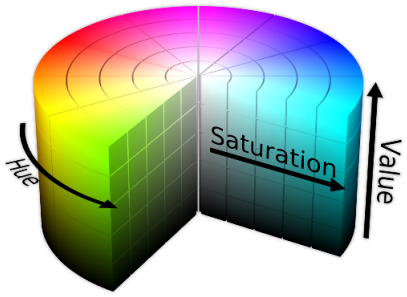


Figure 8: HSV colorspace

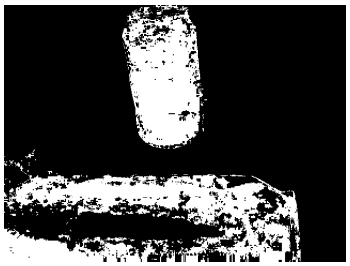


Figure 9: Segmentation using HSV

C. Structural Analysis and Shape Descriptors

As we have seen in Figure 9 , we doesn't get a regular binary image. So we need to join neighborhood points, then we must apply dilation algorithm getting the image in Figure 10 and then for each blob we get a convex hull and a bounding rectangle like in Figure 11.

Figure 10: Color segmentation using HSV

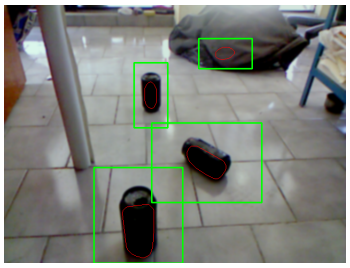


Figure 11: Segmentation with Convex Hull and Bounding Rectangle

After that we discriminate them by shape and area using the bounding rectangle, we get the image in Figure 12.

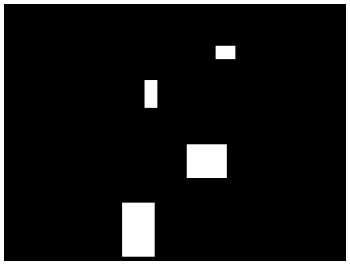


Figure 12: Discrimination by shape and area

V. Can recognition

A. Feature Detection

Color is not sufficient for can recognition because there are other dark objects like shadows and the obstacles.

1) Normalized Image Histogram values from 0 to 255
2) Speeded-Up Robust Features(SURF) features, SURF is an improvement of SIFT algorithm to detect keypoints from an image, but in contrast SURF is much faster. We use the SURF implementation from OpenCV [9].

B. Support Vector Machine (SVM)

It's a supervised learning algorithm that solves the problem of classification between cans and other objects, it was initially proposed by [10].

C. SVM for Linearly non-separable data

Unfortunately there can be data on incorrect regions like in Figure 13.

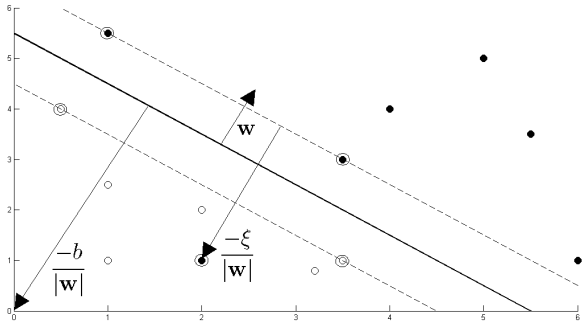


Figure 13: Hyper-plane between two non-separable regions

Because of the limitations of the original SVM, we have to use a kernel function K that defines a function denoted by ϕ, this function ϕ converts each point x_i to a space where the points are linearly separable [11], like in Figure 14.

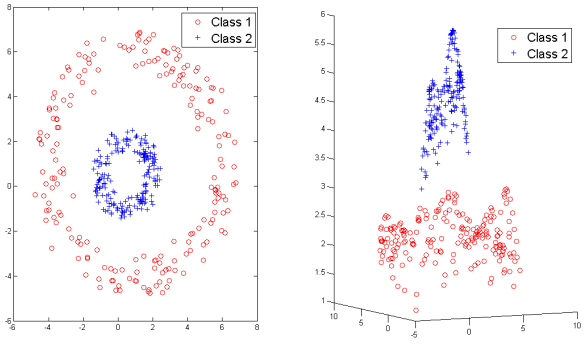


Figure 14: Hyper-plane separating data in other space

Then, we apply the following equation to label each data.

Table II: Example of signal's codes

SIGNAL	ORDER
0001	Turn Left (on it's axis)
0010	Turn Right (on it's axis)
0011	Turn Left (curve/ position correction)
0100	Turn Right (curve/ position correction)

$$\min_{w,b,\xi} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{L}\xi_i \qquad (3)$$
$$y_i(w^T\phi(x_i)+b) \geq 1-\xi_i,$$
$$\xi_i \geq 0, i = 1,\ldots,L$$

Where ξ_i is the tolerance, the parameter C is used to control this tolerance, and ϕ is a function that maps each point in order to change the data to be linearly separable. Each new point is classified using Equation 4.

$$f(x) = sign(w\cdot\phi(x)+b) \qquad (4)$$

The kernel function we use is the Sigmoidal Function: $K(x_i,x_j) = \tanh(\gamma(x_i\cdot x_j)+a)$, where the parameters a,b,γ are defined by user. After we discriminate the blobs by shape and area, we classify each one getting just the ones that contain cans, getting the image in Figure 15.

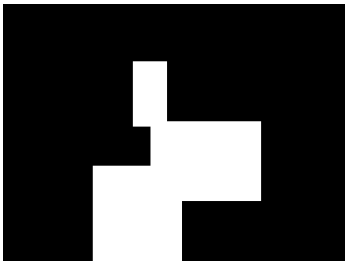


Figure 15: Regions that contain cans, classified with SVM

VI. Electronics

We decided to use Arduino because it's the most commonly used hardware for electronic development specially for robots, also because our team members have experience with it.

Arduino can interpret information from environment through its input pins from a wide range of sensors and can manifest back its environment through its lights, engines, servos, and any others actuators [12]. The Arduino's board microcontroller is programmed by of its own programming language called Arduino. In the presented robot we used Arduino Uno to communicate between the Computer and Pic 16F87, this communication is done by the computer's serial port; receiving and sending 4 bytes signals shown in Table II to the pic and executing appropriate orders.

VII. Conclusions and Future Work

As a project's general conclusion, we highlight the input towards the polution's reduction in our environment, It's clear that our project is a prototype, but we could say that we are founding bases to create something bigger.

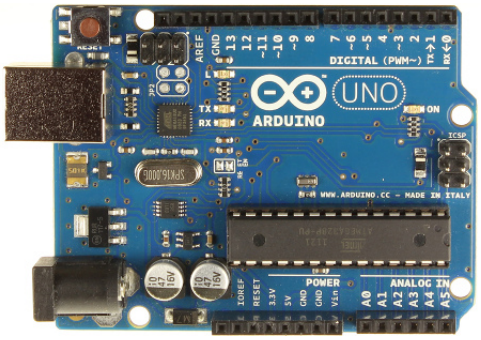


Figure 16: Arduino Board

On the other side, in the specific conclusions we detach the junction of basic scients: the mechanic part, and the use of its physics' basic principles; the electric part on the same way and the usage of computer vision to reach the desired goal of creating an autonomous robot.

Acknowledgment

The group thanks to everyone who helped and supported us during the research and development of the project mainly to Eveling Castro Gutierrez (School of Systems Engineering, UNSA) who supported us with the kinect device.

References

[1] P. Corke, *Robotics Vision and Control: Fundamental Algorithms in Matlab*, 2011.
[2] J. Webb and J. Ashley, *Beginning Kinect Programming with the Microsoft Kinect SDK*, 2012.
[3] J. S. Jean, *Kinect Hacks*, 2013.
[4] N. Burrus, "Introduction to surf," http://labs.manctl.com/rgbdemo/.
[5] OpenCV, "Openni with opencv," http://docs.opencv.org/doc/user_guide/ug_highgui.html.
[6] E. H. Land, "An alternative technique for the computation of the designator in the retinex theory of color vision." 1986.
[7] D. J. Jobson, Z. Rahman, and G. A. Woodell, "Properties and Performance of a Center/Surround Retinex," 1997.
[8] A. Hurlbert, "Formal connections between lightness algorithms," 1986.
[9] OpenCV, "Introduction to surf," http://docs.opencv.org/master/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html.
[10] C. Cortes and V. Vapnik, "Support Vector Networks," 1995.
[11] C. Chang and C. Lin, "LIBSVM : a Library for Support Vector Machines," 2003.
[12] S. Monk, *30 Arduino Projects for the Evil Genius*, 2010.