БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ • BULGARIAN ACADEMY OF SCIENCES

ПРОБЛЕМИ НА ТЕХНИЧЕСКАТА КИБЕРНЕТИКА И РОБОТИКАТА, **68**
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS, **68**

София • 2017 • Sofia

Vision system for recognizing objects using Open Source Computer Vision (OpenCV) and Robot Operating System (ROS)

Denis Chikurtev

Institute of Information and Communication Technologies – BAS

Email: denis@iinf.bas.bg

Abstract: *this article represents the development, structure and properties of a vision system for service robots. To develop the system, we use OpenCV and ROS. Main goal of the vision system is to recognize basic objects of the home environment like caps, plates, cutlery, medicines, and others. To recognize a particular object vision system determines its shape and color. To determine the shape of the objects we use edge detection and to recognize the color we use color filtering. The experimental results show that the system successfully recognizes objects.*

Keywords: *computer vision, OpenCV, ROS, service robot, object recognition*

1. Introduction

Service robots, which are robots serving human needs beyond the factory floor, are playing an increasingly important role in many aspects of human life. These robots come in many designs and perform tasks that range from aerial surveillance, bomb disposal, farming, and warehouse logistics to teaching children and assisting the elderly [1], [2], [3]. They use machine vision and image processing components, subsystems, and technologies to

image, store, and interpret data about the world around them, and perform actions based on the data [4], [5].

Some of the primary vision technologies used in service robots are: structured light systems, two-camera stereo systems, time-of-flight sensors, lidar, and single-lens camera systems. Other sensing/locating technologies may be combined with the vision components to provide even more information to the robots, including GPS navigation, radar, sonar, and inertial guidance. For more sophisticated robots, simultaneous localization and mapping (SLAM) is critical to build maps of unknown environments or to update maps within known environments, while at the same time keeping track of the current location of the robot [6] [7].

Our vision system uses two cameras. The first camera is the main robot camera, which is placed on the robot platform. That camera is Microsoft Kinect sensor. We use the depth sensor of the Kinect for navigation and the RGB camera of the Kinect for object recognition. The second camera is placed on the gripper of the articulated robotic arm. That camera is stereo camera and we use it to navigate the robotic arm to grasp objects. In this paper we represent the process of object recognition and object location. The Kinect recognizes object, after that determines the distance from the robot to the object. When the robot goes near enough to the object, we activate the second camera to guide the robotic arm to grasp the object [8].

2. OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library [9]. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

2.1. Color spaces in OpenCV

Gray: A color space that effectively eliminates color information translating to shades of gray: this color space is extremely useful for intermediate processing; such as face detection.

BGR: The blue-green-red color space, in which each pixel is a three-element array, each value representing the blue, green, and red colors: web developers would be familiar with a similar definition of colors, except the order of colors is RGB

HSV: hue is a color tone, saturation is the intensity of a color, and value represents its darkness (or brightness at the opposite end of the spectrum)

2.2. The Fourier Transform

All waveforms are just the sum of simple sinusoids of different frequencies. In other words, the waveforms you observe all around you are the sum of other waveforms.

This concept is incredibly useful when manipulating images, because it allows us to identify regions in images where a signal (such as image pixels) changes a lot, and regions where the change is less dramatic.

The concept of Fourier Transform is the basis of many algorithms used for common image processing operations, such as edge detection or line and shape detection.

2.3. High pass filter

A high pass filter (HPF) is a filter that examines a region of an image and boosts the intensity of certain pixels based on the difference in the intensity with the surrounding pixels.

2.4. Low pass filter

If an HPF boosts the intensity of a pixel, given its difference with its neighbors, a low pass filter (LPF) will smoothen the pixel if the difference with the surrounding pixels is lower than a certain threshold. This is used in denoising and blurring. eg., one of the most popular blurring/smoothening filters, the Gaussian blur, is a low pass filter that attenuates the intensity of high frequency signals.

2.5. Edge detection

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in It is a multi-stage algorithm and we will go through each stages.

Noise Reduction

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter.

Finding Intensity Gradient of the Image

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2};$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right);$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions. OpenCV puts all the above in single function, cv2.Canny() [10].

Hysteresis: The final step. Canny does use two thresholds (upper and lower):

a. If a pixel gradient is higher than the *upper* threshold, the pixel is accepted as an edge

b. If a pixel gradient value is below the *lower* threshold, then it is rejected.

c. If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the *upper* threshold.

Canny recommended an ***upper: lower*** ratio between 2:1 and 3:1.

3. ROS with OpenCV

Our robot is controlled by the ROS. There is vision_opencv stack, that provides packaging of the OpenCV library for ROS. There are two main packages for using OpenCV in ROS:

- **cv_bridge**: Bridge between ROS messages and OpenCV;
- **image_geometry**: Collection of methods for dealing with image and pixel geometry.

ROS passes around images in its own **sensor_msgs/Image** message format, but many users will want to use images in conjunction with OpenCV. CvBridge is a ROS library that provides an interface between ROS and OpenCV. CvBridge can be found in the **cv_bridge** package in the **vision_opencv** stack [11].

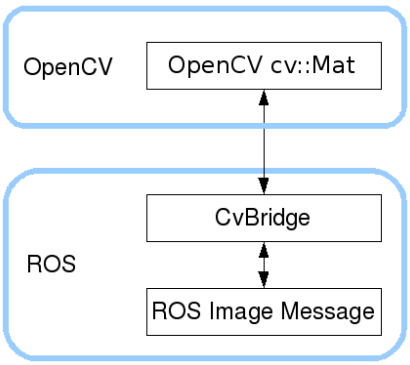

Figure 1. Communication between OpenCV and ROS

image_geometry contains Python and C++ libraries that simplifies interpreting images geometrically using the parameters from **sensor_msgs/CameraInfo**. Although CameraInfo contains all the information required to rectify a raw image and project points onto it, it is highly recommended that you use this library, since performing these operations correctly over the space of all camera options can be non-trivial.

The camera parameters in CameraInfo are for a full-resolution image; region-of-interest alone significantly complicates the creation of rectification maps and requires adjusting the projection matrix. Adding options such as subsampling (binning) to CameraInfo would further complicate the correct interpretation of the corresponding Images. Using image_geometry simplifies and future-proofs imaging code [12].

The image_geometry classes are written to be used in an Image/CameraInfo message callback similar to **cv_bridge**.

In order to maintain invariance, the CameraModel classes offer read-only access to specific parameters and matrices. Setting a CameraModel can only be performed with full information using the fromCameraInfo() functions.

These classes use the OpenCV camera model. Accessors to camera matrices in the format expected by OpenCV are provided for easy integration.

4. Vision system description

As we said previously our vision system include Kinect and stereo camera (fig. 2). The Kinect has RGB camera which we use for recognition and depth sensor to measure the distance to the object. VideoCapture can retrieve the following data:

a. data given from depth generator:

- CV_CAP_OPENNI_DEPTH_MAP - depth values in mm (CV_16UC1);

- CV_CAP_OPENNI_POINT_CLOUD_MAP - XYZ in meters (CV_32FC3);
- CV_CAP_OPENNI_DISPARITY_MAP - disparity in pixels (CV_8UC1);
- CV_CAP_OPENNI_DISPARITY_MAP_32F - disparity in pixels (CV_32FC1);
- CV_CAP_OPENNI_VALID_DEPTH_MASK - mask of valid pixels (not ocluded, not shaded etc.) (CV_8UC1).

b. data given from BGR image generator:

- CV_CAP_OPENNI_BGR_IMAGE - color image (CV_8UC3);
- CV_CAP_OPENNI_GRAY_IMAGE - gray image (CV_8UC1).

In order to get depth map from depth sensor we use VideoCapture::operator. For getting several data maps we use VideoCapture::grab and VideoCapture::retrieve.
For setting and getting some property of sensor` data generators we use VideoCapture::set and VideoCapture::get methods respectively.

Since two types of sensor's data generators are supported (image generator and depth generator), there are two flags that should be used to set/get property of the needed generator:

- CV_CAP_OPENNI_IMAGE_GENERATOR – A flag for access to the image generator properties.
- CV_CAP_OPENNI_DEPTH_GENERATOR – A flag for access to the depth generator properties. This flag value is assumed by default if neither of the two possible values of the property is not set.

Figura na robota

Both Kinect and stereo camera are connected to the USB ports of the robot's computer. While the Kinect is looking for an object the stereo camera is in standby. When the object is recognized and the robot moves next to it by specific distance, the computer turn on the stereo camera and starts to control the robotic manipulator.

To recognize the objects, we get the edges of the scene. After that the vision searching for specific shapes [13]. If there are similar shapes, then the system checks for the shape of the same object by its color [14]. So for the

edge recognition we use Canny edge detection with two thresholds. For the color conversion we transform the RGB video signal to HSL image and extract only one channel, L will be processed. The advantages of using one channel instead of three channel are the reduction of the processing time and complexity. The *L* value contains lightness value of the input image where *L* is calculated as shown:

$$\mathrm{L} = \frac{\max(R,G,B)+\min(R,G,B)}{2};$$

where, *L* is the lightness value, *R* is the red channel of the input image, *G* is the green channel of the input image and *B* is the blue channel of the input image.

5. Results

As results we provide some figures of the edge detection (fig. 2), color recognition (fig. 3) and shape and color recognition (fig. 4). In figure 2 are shown three types of edge detection. In figure 3 we recognize red color and the result image is showing only that color. In figure 4 is shown the recognition of shapes and the colors of the objects. Depending on the intensity if the light we can correct manualy or automaticly the levels of the lights of the camera.

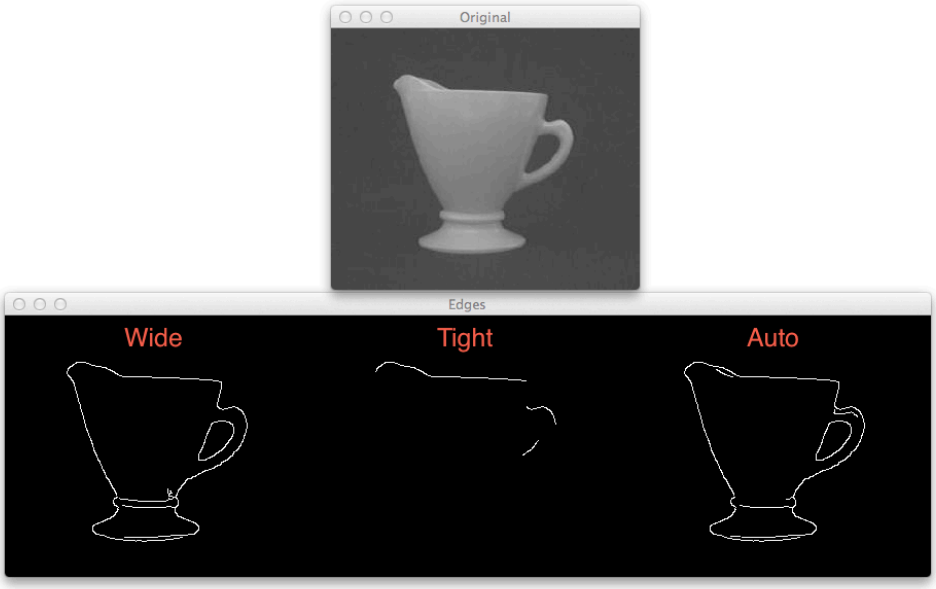

Figure 2. Edge detection and recognition of the shape of a cap.

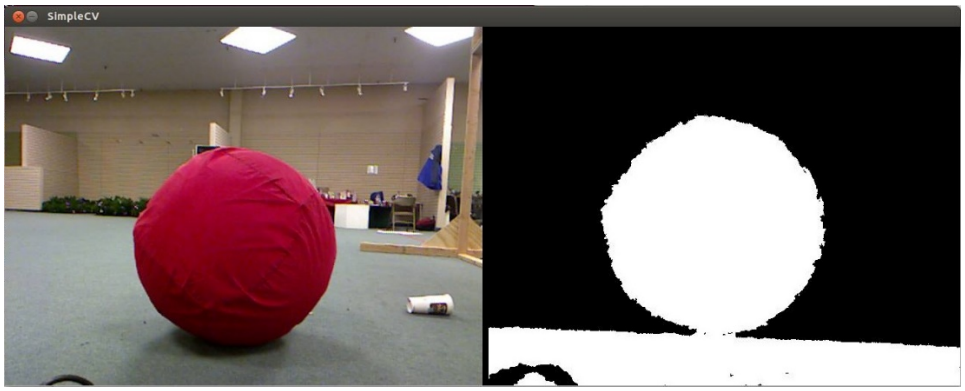

Figure 3. Colour recognition.

Figure 4. Shape and colour recognition.

6. Conclusion

To perform daily tasks service robots, have to recognize the objects around. Our vision system successfully recognizes shapes and colors of the objects. That vision system will provide to the robots the abilities to adapt and interacrt with the objects form the environment. On that way the service robots will realize their main task – to help the people.

Acknowledgments

The research work reported in the paper is partly supported by the projects funded by the Young Scientists Grants, reg. No 102/2016.

References

1. Chivarov, N.; Shivarov, S.; Yovchev, K.; Chikurtev, D.; Shivarov, N. - Intelligent Modular Service Mobile Robot ROBCO 12 for Elderly and Disabled Persons Care; Robotics in Alpe-Adria-Danube Region (RAAD), 2014 23rd International Conference on, Smolenice, Slovakia, 3-5 Sept. 2014, Print ISBN: 978-1-4799-6797-1. p. 343-348.

2. Nayden Chivarov, **Denis Chikurtev**, Kaloyan Yovchev, Stefan Shivarov - Cost-Oriented Mobile Robot Assistant for Disabled Care; TECIS 2015, 16th IFAC Conference on Technology, Culture and International Stability, Sozopol, Bulgaria, 24-27 September 2015

3. Ulrich Reiser, Christian Connette, Jan Fischer, Jens Kubacki, Alexander Bubeck, Florian Weisshardt, Theo Jacobs, Christopher Parlitz, Martin H¨agele, Alexander Verl, “Care-O-bot 3 - Creating a product vision for service robot applications by integrating design and technology”, The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009 St. Louis, USA

4. Zheng Li and Yi Ruan, “Autonomous Inspection Robot for Power Transmission Lines Maintenance While Operating on the Overhead Ground Wires”, International Journal of Advanced Robotic Systems, Vol. 7, No. 4 (2010), ISSN 1729-8806, pp. 111-116

5.http://web.archive.org/web/20040829092603/http://www.dcs.qmul.ac.uk:80/research/vision/publications/papers/bmvc97/node1.html

6. https://en.wikipedia.org/wiki/David_H._Hubel

7. https://en.wikipedia.org/wiki/Torsten_Wiesel

8. Kanade, T. Picture processing system by computer complex and recognition of human faces. PhD thesis, Kyoto University, November 1973

9. Brunelli, R., Poggio, T. Face Recognition through Geometrical Features. European Conference on Computer Vision (ECCV) 1992, S. 792–800.

10. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1. IEEE, 2001, pp. 1-511.

11. Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

12. Ahonen, T., Hadid, A., and Pietikainen, M. Face Recognition with Local Binary Patterns. Computer Vision - ECCV 2004 (2004), 469–481.

Система зрения для распознавания объектов с использованием Open Source Computer Vision (OpenCV) и операционной системы робота (ROS)

Денис Чикуртев

Институт информационных и коммуникационных технологией - БАН

Email: denis@iinf.bas.bg

Аннотация: *Эта статья представляет собой разработку, структуру и свойства системы зрения для сервисных роботов. Для разработки системы мы используем OpenCV и ROS. Основная цель системы зрения - распознать основные объекты домашней среды, такие как колпачки, тарелки, столовые приборы, лекарства и другие. Чтобы распознать конкретную систему объектного зрения, она определяет ее форму и цвет. Чтобы определить форму объектов, мы используем обнаружение кромок, а для распознавания цвета используется цветовая фильтрация. Экспериментальные результаты показывают, что система успешно распознает объекты.*

Ключевые слова: *Компьютерное зрение, OpenCV, ROS, сервисный робот, распознавание объекто*