Vision 2D: Lab

Sounkalo DEMBÉLÉ, Mayra Yucely BEB CAAL
Université de Franche-Comté

Contents

1 Introduction **2**

2 Training **3**
2.1 Experimental setup 3
2.2 Analysis of imaging system 4
2.2.1 Reading of video stream from camera 4
2.2.2 Objective of 25 mm focal length 4
2.2.3 Objective of 25 mm focal length with extension tube 5
2.3 Reading, processing and analysis of single image 5
2.4 Reading, processing and analysis of video stream from file 7

3 Testing **8**

Chapter 1

Introduction

The educational objective of the lab is initiation into computer vision through OpenCV with Python by applying it to the metrology of colored targets. OpenCV is the reference library in computer vision, it is open source and used extensively by both academia and business.

Chapter 2

Training

The purpose of this part is learning of OpenCV. No report will be requested or mark be attributed.

2.1 Experimental setup

Used vision system includes :

- a backlight,
- a frontlight,
- a USB 2D camera embedded with a standard objective along with, or not, an extension tube,
- a computer running Python (from Anaconda) with OpenCV version 3.4[1] .

The camera is an uEyeSE from IDS. Its specifications include:

- CMOS color sensor of 1/2" (diagonal: 8 mm),
- format full frame of 2048 x 1536 pixels,
- depth of 8 bits,
- acquisition using rolling shutter method,
- full frame frequency of 11 Hz,
- USB2.0 output.

The specifications of objective include:

- C-mount,
- focal lenght of 25 mm,
- aperture from 1.6 to 16 mm,
- minimal working distance of 250 mm,

[1]The management of the versions of OPENCV is special: several versions continue to be updated in parallel, 2, 3 and 4. For compatibility with the lab, it is recommended, if you wish to make a personal installation, to use version 3, regardless of the release.

- maximal working distance of infinity.

The specifications of extension tube include:

- C-mount,
- lenght from 5 to 40 mm.

Working distance can be modified for focusing purpose, assuming height of backlight is about 13.5 cm.

2.2 Analysis of imaging system

An OpenCV program will be used for the characterization of the used optical imagery. It should be remembered that focusing in such imagery includes the adjustments of, working distance, imaging distance and aperture of diaphragm.

2.2.1 Reading of video stream from camera

Copy, analyze and test the following code for reading a video stream until `SPACE` is clicked to stop:

```
"""
VISION 2D:
Training code
"""

import cv2

"""
Reading a video stream
"""
# Configure video stream source: 0 is the defaut one
cam = cv2.VideoCapture(0)
if (not cam.isOpened):
    print ('Impossible to read the camera !')

# Display stream until clic on SPACE when mouse pointer in video display
while (True):
    ret, frame = cam.read()
    cv2.imshow('video', frame)
    if cv2.waitKey(2)>=27:
        break
# Deallocate memory
cam.release()
cv2.destroyAllWindows()
```

Listing 2.1: Python Code for reading a video stream

2.2.2 Objective of 25 mm focal length

Use the image of a graduated ruler to estimate the maximum and minimum working distances, and the minimum field of view.
Therefore, compute the depth of field, the maximum magnification and the minimum resolution.

2.2.3 Objective of 25 mm focal length with extension tube

Recalculate parameters: maximum and minimum working distances, minimum field of view, depth of field, maximum magnification and minimum resolution.

Do the results conform to theory?

An application of such imaging system is analysis of watch parts.

2.3 Reading, processing and analysis of single image

Consider the following code, that read, processes and analyzes the image `paperAppr.png` (figure 2.1).

Analyze the code then run it: during image display with `imshow`, the value of pixel under cursor is displayed botton left.

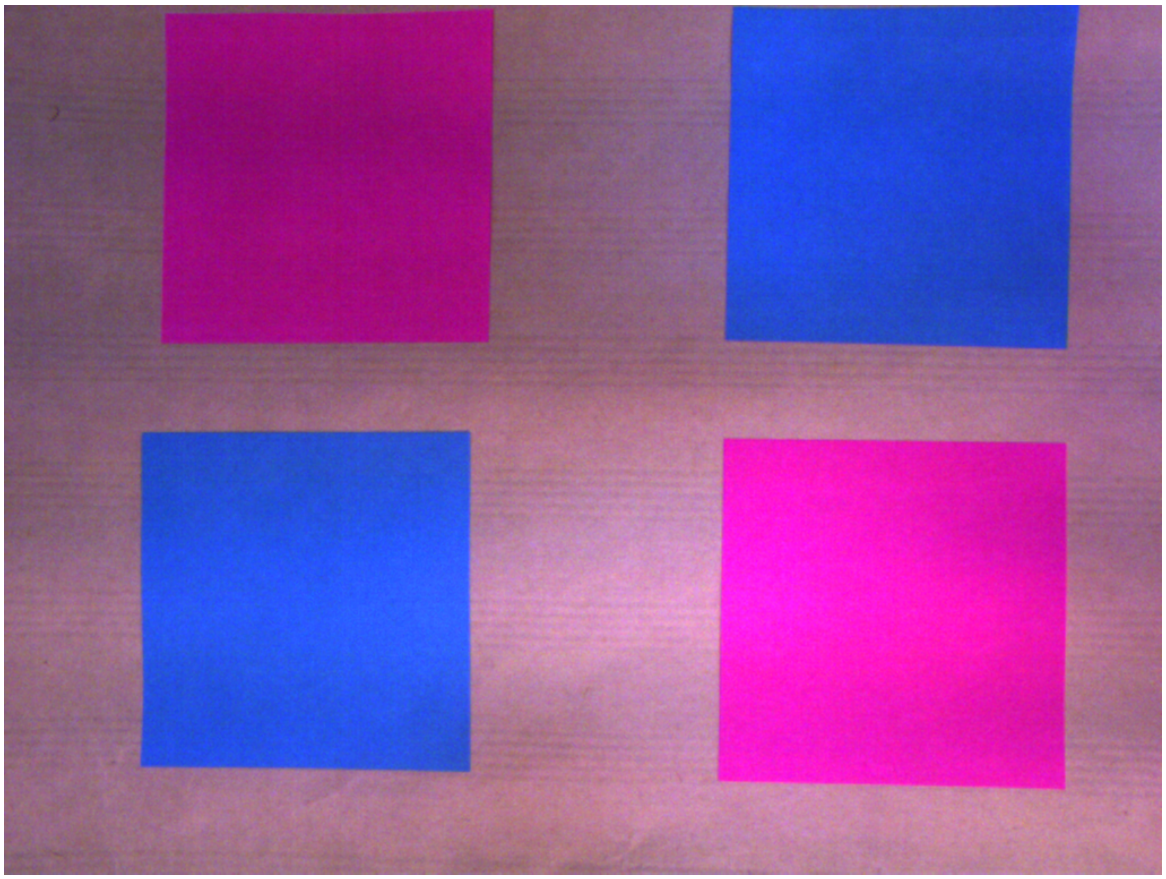

Figure 2.1: Color image for testing.

```
"""
VISION 2D:
Training code
"""

import cv2
```

```
"""
Reading a single image
"""
# Read and display BGR image
bgr = cv2.imread ( './paperAppr.png', cv2.IMREAD_UNCHANGED )
if (not bgr.data):
    print ('Impossible to read image !')
cv2.imshow("RGB image", bgr)
cv2.waitKey(0)

# Convert BGR into HSV and display
hsv = cv2.cvtColor(bgr, cv2.COLOR_BGR2HSV)
cv2.imshow("HSV image", hsv)
cv2.waitKey(0)

# Extract hue and display
h,s,v = cv2.split(hsv)
cv2.imshow("Hue image", h)
cv2.waitKey(0)

# Convert BGR into GRAYSCALE and display
gray = cv2.cvtColor(bgr, cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray image", gray)
cv2.waitKey(0)

# Display grayscale image with a color map
clr = cv2.applyColorMap( gray, cv2.COLORMAP_JET )
cv2.imshow("Color map image", clr)
cv2.waitKey(0)

# Global thresholding of GRAYSCALE and display
rv, bry = cv2.threshold(gray, 0,255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
cv2.imshow("Binary image", bry)
cv2.waitKey(0)

# Local thresholding of GRAYSCALE and display
bry2 = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
           cv2.THRESH_BINARY, 11, 2)
cv2.imshow("Binary image 2", bry2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Listing 2.2: Python Code for reading a video stream

Complete the code to perform the following tasks:

1. erosion, dilatation, opening and closing of images, grayscale as well as global-thresholded binary images (functions `erode`, `dilate` et `morphologyEx`, respectively),
2. calculating the regions of binary image assuming a closed edge corresponds to a region (function `findContours`),
3. calculating and displaying bounding rectangles of regions (function `boundingRect`),
4. deblurring of grayscale image with rectangle kernel (function `blur`) then with gaussian kernel (function `GaussianBlur`),

5. enhancement of grayscale image with morphological gradient (`morphologyEx`), Sobel gradient (function `Sobel`) then with Canny gradient (function `Canny`).

2.4 Reading, processing and analysis of video stream from file

It is similar to reading a video stream from a camera except the source number is replaced with the file name including extension, inside single or double quotes.

Chapter 3

Testing

The purpose of this part is testing of your skills in computer vision through the use of OpenCV. It is requested to supply a report describing the solution and the corresponding code. They will be used for mark.

Some colored papers such that each paper has a unique uniform color are considered: pink, purple, blue, green, orange, yellow and white (figure 3.1).

Figure 3.1: Reference image for color and sector calibration.

The papers move in a space structured in sectors numbered from top to bottom: 0 to 3.

The problem to solve consists in recognizing the color and position of the papers from their images recorded in a video (paperEval.avi).

The image `paperEval.png` is provided for manual calibration of the sectors and colors, assuming it is acquired in the same conditions as the video.

More precisely, the application should deliver per frame the following informations and save them in a file `resultats.txt`: identification of any object by its appearance number in the image, its color, the sector in which it (its center of inertia) is located (figure 3.2).

Figure 3.2: A frame illustrating application output.

It is advised to organize the code as follows

- a function to manually calibrate the seven colors of the problem, i.e. to determine minimal and maximal hues of every color;
- a function to manually calibrate the three sectors of the problem, i.e. to determine minimal and maximal locations of every sector;
- a function to identify every paper by its order of appearance and calculate its color and sectors;
- a function for display information: number, color, sector.