7 Computer Vision

Cameras and other sensors that sense the environment, such as scanning sonars for underwater applications or infrared cameras to sense heat distributions, are an important source of sensory information. The main challenge with such data is how to interpret them and how to recognize objects from the data steam. Vision is a major sensory source for humans and our brain is specialized in interpreting signals from our eyes. In this chapter we will use a webcam and discuss some basic methods from computer vision to be used in later experiment.

Machine learning has contributed considerably to recent progress in computer computer vision including tracking and object recognition. However, we will not enter into this discuss here.

7.1 Basic camera processing with OpenCV

OpenCV is a popular collection of methods for computer vision. While it is mainly a C++ library, some methods are accessible through a python interface. Most of the examples here use the cv2 library, which is part of the python(x,y) distribution. Documentation for cv2 is available at `http://docs.opencv.org`.

Table **??** shows a basic program to read and image from a file and to display it using two cv2 functions. The `waitKey()` is included so that the program waits for a key press before terminating since the termination of the program will also terminate the camera window.

To read continuously from a webcam we have to set up a video capture. Table **??** shows a basic program to read from a webcam. We first set up a video capture and then enter an infinite loop to read from this video capture. To terminate this loop we included a break when the `ESC` key is pressed.

Let us start manipulating a singe image. As a first example lets us created a new image I^{mean} by averaging over a 10 by 10 region in the image. For this we define an array $k(u, v)$ with indices u and v running between 0 and 10 where all elements are one, $k(u, v) = 1$. The new average image is then defines as

Table 7.1 Reading from a webcam and manipulating it.

```
import cv2
im = cv2.imread('picture3.jpg')
cv2.imshow('camera',gray)
cv2.waitKey()
```

Table 7.2 Reading from a webcam and manipulating it.

```
import cv2

cap = cv2.VideoCapture(1)

while True:
    ret,im = cap.read()
    cv2.imshow('camera',im)
    if cv2.waitKey(10) == 27:
        break
```

$$I^{\text{mean}}(x,y) = \sum_{u,v} I(x-u, y-v)k(u,v). \tag{7.1}$$

The new image is a bit smaller than the original, but there are also ways to adjust for this like using periodic boundaries. The matrix k is called a kernel, and the operation described in eq.**??**convolution is called a **convolution**. The resulting image is a smoothened version of the original image. This operation has blurring effect does not seem very useful in image processing. However, it can actually be useful when downsampling images or to reduce noise in the image. Also, rather than using a constant kernel it is more common to use a Gaussian kernel

$$k(u,v) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-u^2-v^2}{\sigma^2}}. \tag{7.2}$$

Smoothing with Gaussian kernels is a common technique in computer vision.

Exercise

1. Write a program to calculate the average image
2. Downsample the image and compared the down sampled image of the original version and the down sampled version based on a smoothened image with a constant kernel and a Gaussian kernel.

7.2 Finding a color blob

An easy way to localize some environmental object is by tagging it with some unique colour and trying to detect this in the image. This will be used later for some exercises in localization and planing. for the following exercise take some coloured electrical tape of some other coloured material and attach it to the robot arm. We can first test it statically, but we will later use it to detect the location of the arm when the arm is moving.

To detect a certain colour in an image we need to process the colour channels. We can write a little application that takes an image and in which we could point to a

location in the image to return the values. This program is shown in Table **??**. (explain program)

Once we have RGB value for the target colour we can use them to locate the colour in a video stream. For this it is useful to take the some of the absolute differences between a video screen colour values and the target values. Small values indicate pixels close to the target colour. Since the target area corresponds to a cluster of such pixels, we could use an averaging such as with Gaussian smoothing followed by finding the minimum to locate the centre of the target area.

An alternative to the colour method for finding the position of the robot arm it motion segmentation. Segmentation of an image is an important step in building scene representations, and the following sections talks about some methods that commonly build the basis of segmentation for still images. The beauty of video streams is that there is more information in it that we can use for segmentation. In the example with the robot arm we assume that only the robot arm is moving. We can therefore use differences of video captures in consecutive frames to determine the moving object.

Finally we want to translate the tracking of the robot arm to a number representing the degrees of rotation of the upper motor of the robot arm. For this we will use machine learning techniques. The first is to use linear regression on the motion segmented robot arm. The other is to use the support vector regression to map the (x, y) coordinates to rotation angles. Note that both cases correspond to supervised learning that require measurements that we will use as teacher signals.

Exercise

- Write a program to locate a colour blob in a video stream and indicate this target location with a circle. Similarly, use as an alternative motion segmentation and compare the location estimation in form of a pixel coordinate between the two methods.
- Write a program that translates a pixel coordinate to the estimation of the rotation angle of the motor and compare the location estimation of the two segmentation methods with the coordinates returned by the motors.