CMSC 143: Object-Oriented Programming with Robots

Lab 5: Motion Detection

Due October 9, 2014

In this lab we will practice some of the methods that we have learned for analyzing images. In particular, we will create a program that acts as a motion detector that could be used for security purposes. We will use a technique called background subtraction to detect changes in a scene. Using gray scale images (`takePicture('gray'); getGray(px); setGray(px, value)`) simplifies the task slightly. Complete the following tasks:

- `frameDifference(background, image)`: Create a function that "subtracts" `background` from `image`. Your function should iterate through every pixel in `image` and subtract the value of the corresponding pixel in the `background`. This function should return a new image where each pixel is the absolute value of the difference between pixels of the two input images. Include examples images in your report.

background image | second image | result

- `detectChanges(background, image, threshold)`: Create a function that is similar to `frameDifference` but instead of the resulting image being the difference, we are going to only show the pixels in `image` that have changed substantially. Specifically, those pixels that have changed by a value of `threshold`. Compare the results of three different threshold values, which value works best for detecting changes?

background image | second image | result

- `securityGuard(time)`: This function first takes a 'background' image of what the scene initially looks like (for instance, when you set the motion alarm). Then for `time` seconds, continually takes pictures and passes them along with the background image to `detectChanges()`. If there are substantial changes in the image, save a picture or an animated gif file of the intruder.

Learning Objectives

- Analyze images
- Implement a background subtraction algorithm
- Empirically evaluate an algorithm

Deliverables

Submit an electronic copy of your lab using moodle. Your submission should be a zip file that expands to a folder with two files:

```
cmsc143_lab5_LASTNAME/
      lab5.py
      lab5.pdf
```

EXTRA: ALPHA BLENDING

Along with "subtracting" images, as we did by frame differencing, we can also add them. We can combine the pixels of two images by a weighted average. We weigh one image by a factor called α that ranges from $0 - 1$ and the other image by $1 - \alpha$. This results in a transparency effect and is often called **alpha blending**. Create a new function called `alphaBlend(image1, image2, alpha)` that iterates through the pixels of `image1` and the corresponding pixels from `image2`. Then the red, green, and blue values of the corresponding pixel of the new image is the weighted sum of those from `image1` and `image2`.

$$R_{new} = \alpha \times R_1 + (1 - \alpha) \times R_2 \tag{1}$$

$$G_{new} = \alpha \times G_1 + (1 - \alpha) \times G_2 \tag{2}$$

$$B_{new} = \alpha \times B_1 + (1 - \alpha) \times B_2 \tag{3}$$