Facial Detection and Text Recognition Project

By: Rob Christenson, Michael Tingey

Project Description

Using an embedded Linux device, software was developed that allows the device to interface with a webcam to discern the following:

- Is a face on screen?
- Is text on screen?
- If there is text, what does it say?

Project Goals (Overall)

The overall goal for our project was to be able to present a document or a face to a webcam connected to a Linux device and, upon program execution, have the device determine if a face is present or not, and if text is present, it should be displayed.

Implementation

The project was implemented using C++ to interface both with the OpenCV API (Open Source Computer Vision) and the Tesseract OCR (Optical Character Recognition) engine. Both Tesseract and OpenCV provided relatively straight forward solutions to the needs of our project. By employing Tesseract to deal with any text and OpenCV to deal with faces, our finished project had powerful algorithms working behind the scenes, as both software packages have been around for a long time and have big sponsors, such as Google, backing them. While Tesseract focuses on the single task of character recognition, OpenCV has many, many modules to choose from, and in many cases, there are multiple ways to perform the same task. In the end, we utilized functionality from the Image Processing module (for facial detection) and the High-level GUI and Media I/O module (for interacting with the webcam). On the hardware side of things, our final project simply consists of a BeagleBone embedded Linux device and a Logitech c920 HD Pro webcam.

Project Components

Our project is comprised of three major software functions, which perform image capture, facial detection, and optical character recognition.

Image Capture

The image capture function of the software is responsible for interfacing with the physical webcam and, upon program execution, capturing image data from the camera. Thanks to the powerful functionality of OpenCV, our program is able to perform thresholding on the captured image before it is saved, which helps improves the probability of a successful execution.

Facial Detection

The facial detection component of the project begins by converting the captured image from the RGB color space to grayscale, and then equalizes the image's histogram (which helps to improve facial detection). Then, the program utilizes a local binary pattern (LBP) cascade designed for faces to detect possible faces within the image. Following that, if a face is detected, the program outputs the result. Otherwise, our program continues on to look for text.

Optical Character Recognition

The OCR works by setting the color spectrum of the image to grey scale and then uses a pixel brightness threshold algorithm to force each pixel to become either completely white or black in order to detect text. By running the image through the Tesseract engine, any text is extracted and reported to the user.

Project Challenges

The design of the project was a relatively smooth experience, with the easy and challenging aspects summarized below:

Easy

The image capture turned out to be a simple matter that just required the use of simple OpenCV commands to capture, format, and save the requisite image file. The use of the OCR engine was also a simple matter, however preprocessing of the image to guarantee greater accuracy turned out to be a challenge.

Difficult

One of the most challenging aspects of our project was in preprocessing the image before the OCR algorithm could be run on it. The image needed to be set to greyscale and a brightness threshold algorithm applied in order ensure the text appeared in stark contrast to the background. The greyscale and threshold algorithm had to be determined on our own through the use of photo editing software in order to find the right combination of processes.

Another problem that we ran into that we were not expecting, was how difficult it was to incorporate our separate contributions to the project into one program that could run on the Raspberry Pi 2 (which was our original choice of embedded Linux device). The problem arose

because we wrote our project using OpenCV version 2.4, without realizing that Raspbian Jessie requires the use of OpenCV version 3.0. Luckily, Rob had OpenCV 2.4 installed on his BeagleBone, so we decided our final implementation of the project would be done on that, which still keeps the project tied to an embedded Linux device.

Group Members

Rob Christenson

Contributions: Image capture functionality, optical character recognition, BeagleBone hardware

Michael Tingey

Contributions: Facial detection functionality