Computer and Machine Vision Systems: Lab #5

David Rappaport and Jeff Venicx

April 5, 2015

1 Project Proposal

For our extended lab, we propose developing a robot which will use computer vision to autonomously detect and follow a specified object. Leveraging the computer vision techniques acquired this semester, as well as the algorithms provided in the OpenCV library, our goal is to create a powerful and robust system which can perform reliably in a variety of uncontrolled settings. Since this project is broad in scope, we have divided our desired features into three functional tiers, as detailed below: minimum, target, and optimal.

1.1 Minimum Functionality

We begin by defining the core functionality of the robot—what we would view as the minimum criteria for the project to be considered a success. At this level, we expect the robot to be able to detect and follow a predetermined, brightly-colored moving object in a well-lit, uncluttered environment. In order for this to occur, we must implement a series of features, as shown in the block diagram below:

Figure 1: Minimum functionality

We begin by simply capturing an image. Currently, we plan to use a single Logitech webcam attached to a BeagleBone Black single-board computer running Ubuntu Linux.

Next, we perform a series of steps designed to facilitate blob detection. Per Kaehler and Bradski (Learning OpenCV), it is typically easiest to perform blob detection after

converting the image to a color space whose axis is aligned with brightness, such as HSV or YUV. We convert the image to the HSV (hue-saturation-value) color space, which indeed makes our task much easier, since the color of our object is now captured by a single variable (hue), while the amount and brightness of that color is determined by the saturation and value variables, respectively. From this point, we can easily apply a thresholding filter based on each parameter, yielding a binary map showing only objects in a desired range of HSV values.

We can now apply a blob detection algorithm to the binary map to identify the centroid of our object. In a tightly controlled environment, the binary map will likely contain very little besides our uniquely colored object, making the blob detection process fairly straightforward (i.e. we may be able to detect blobs solely based on area criteria due to low noise in the binary map).

Finally, based on the estimated coordinates of our object, we can drive towards the object, using proportional control to keep it in the center of our field of view. For mobility, we currently plan to use off-the-shelf motor drivers attached to a prefabricated chassis such as the SparkFun Electronics Magician Chassis.

In order to validate the feasibility of this project, we have implemented a program (attached) which performs most of the functionality described above. Since we do not currently have a functional robot chassis, our proof-of-concept instead emulates the robot's movements via an on-screen indicator and output on the command line. This behavior is demonstrated in an attached screen recording.

1.2 Target Functionality

We now explore our target functionality. At this stage, we add features which build upon those described in the previous section, enhancing the robot's overall performance and versatility. These new features are indicated in yellow in the block diagram below:

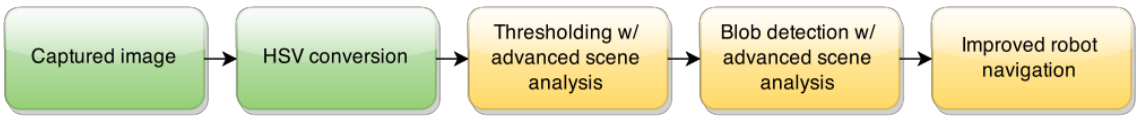

Figure 2: Target functionality

Our overarching goal in this stage is to improve the robot's detection abilities, allowing it to perform in more challenging environments. While we still plan on using a known, brightly-colored object as our target, we may test the robot in a cluttered environment with other objects of similar hues or shapes, or in a darker room where color is less easily deter-

mined. In order to cope with these environmental challenges, we will need to improve the thresholding and blob detection features as defined in the previous section. For the former, we may employ histogram analysis or other techniques to determine the ideal threshold values. For the latter, we may employ other parameters (e.g. circularity, convexity, etc.) to allow the robot to track the object even when the color cannot be confirmed. Our goal is to develop logic so that the robot can autonomously determine the best set of "rules" to use in real time, allowing it to track the object through various lighting conditions, etc.

Additionally, we plan to integrate new features which will improve the robot's navigational abilities compared to the simple "chase" functionality described in the previous section. For example, by adding hysteresis to the blob detection process, we can continue to drive toward the object's estimated position even when the object has been temporarily obscured. Additionally, we will attempt to estimate the object's proximity based on its perceived size, allowing us to drive more quickly initially and then slow to a stop as we approach the object.

1.3 Optimal Functionality

Finally, we define our optimal functionality. This stage represents our "stretch" goals—goals which we believe we can implement, but which may represent a greater challenge than the previous stage. These new features are indicated in red in the block diagram below:

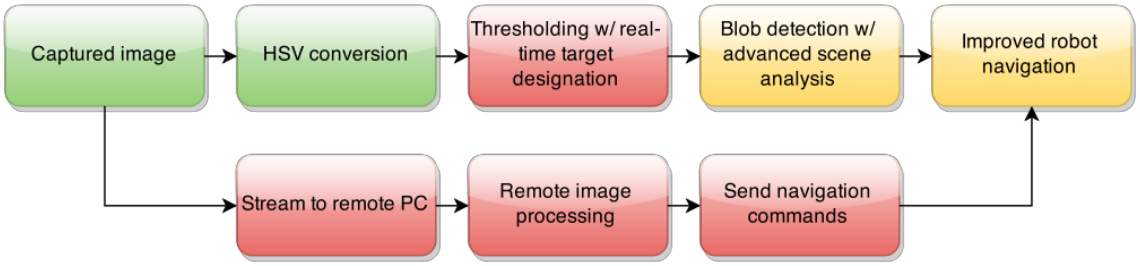

Figure 3: Optimal functionality

Our first new feature expands the thresholding stage to incorporate real-time target designation. Through a given process (e.g. shining a laser pointer at a target object, holding a target object in front of the robot, etc.), we will be able to instruct the robot to switch between following targets of different colors on-the-fly.

Our second addition is a set of features designed to allow for remote processing of the captured video data. Although we expect the BeagleBone to be able to handle the computational demands of this project, there are certainly applications where it is necessary or more practical to offload processing to a remote computing resource—in fact,

"Explore Video Analytics In The Cloud", written by Professor Siewert, specifically discusses cloud-based processing of video which cannot be directly processed on an embedded device. Therefore, we will seek to design a system which will allow us to capture video on the BeagleBone, transmit it to a PC via WiFi, perform the requisite processing on the more powerful computer, and then transmit navigational instructions back to the robot. Our goal is for this system to at least be able to perform with the same degree of competence as the "minimum functionality" specification. In order to achieve this, we will need to investigate various compression techniques and resolutions to find the optimal balance between image quality and latency.