BeBould: The Rock Route Finder

Eric Wedemire

Michael Kaye

Sean Casey

**Introduction**

Prior to the COVID-19 pandemic and resultant lockdowns, common pastimes of the authors of this paper included rock climbing and bouldering. In these activities, one is required to navigate to the top of a wall using predefined routes that range along a standardized scale of difficulty and consist of colour-coded hand- and footholds. The difficulty of a route is determined principally by the number of holds present, their placement, and their physical form, which can vary widely.

The inspiration for BeBould came when the authors (as well as two of their friends) participated in the HackED 2020 hackathon hosted by the University of Alberta Computer Engineering Club in January 2020. The goal of the project was initially to gain experience with computer vision software using the authors’ shared passion of bouldering as a basis by developing software to recognize the holds in images of bouldering walls.

The scope of this project was then expanded to incorporate machine learning tools with the goal of identifying the holds in a photo. As such, the problem being addressed is one of object detection and classification, and therefore is a supervised learning problem. Initially, the goal was to identify the holds and classify them into the routes to which they belonged (the routes being distinguishable by the colour of the holds). However, given the relatively short time frame of the project, this was simplified to merely detecting the presence of a hold on a given section of the climbing wall in the images. As such the, the problem can be considered a binary classification task with “hold” or “not hold” as the two possible outcomes for each section of the wall.

The data used in this project consists of photos of the climbing walls at a local bouldering gym, Rock Jungle. As the problem is a binary classification task, there are only two possible labels for each case: positive (contains a hold) and negative (does not contain a hold).

The major challenge in this task is for the algorithm to be able to detect the holds while accounting for the inherent noise present in the images. A bouldering wall often contains holes for the holds to be screwed into, labels indicating the starting and finishing positions of each route, aesthetic decorations, and the various logos of the gym and manufacturers of the holds. As well, the holds themselves can often be partially obscured by chalk (used widely by climbers to help maintain their grip) which can sometimes make the holds difficult to distinguish in photos. Therefore, our algorithm must be able to overcome this noise in the data when attempting to detect holds.

To tackle this problem, the approach that was employed consisted of an implementation of the sliding window algorithm to sequentially scan each photo, and the use of an SVM-based classifier to analyze each subsection produced by the sliding window algorithm.

**Methods**

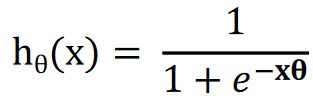
The data for this project consisted of photos taken of the climbing walls in a local Edmonton bouldering gym, Rock Jungle. The data was divided into two sets: 80% of the data was used for training, and the remaining 20% of the data was used for testing the trained algorithm. For the training data, subsections from each photo were extracted that contained either a hold, or a non-hold (such as tape, a coloured section of wall, route markers, and other examples of non-hold noise). As well, for both positive and negative training data, three different sizes of image subsections were used, 32 by 32, 64 by 64, and 129 by 128 pixels. For the testing data, no preprocessing was necessary as we employed the sliding window technique to scan the entire unaltered photo.

We paired the sliding window algorithm with two different classification algorithms. The first algorithm we attempted was a Support Vector Machine (SVM) classifier that made use of the Histogram of Oriented Gradient (HOG) method for obtaining image features. The idea behind HOG is that an image is divided into smaller connected regions called cells, and for the pixels of each cell, a histogram of gradient directions is computed, allowing for edge detection within each cell. These gradients are then used as the features of each image in our data sets.

[SMV explanation goes here]

We implemented our SVM classifier in Python 3 using the Scikit-Learn machine learning library. We obtained a basic skeleton code from a source on the internet [MIKE, TALK ABOUT WHERE YOU GOT THE CODE FROM]. From here, we tweaked the hyperparameters of the SVM and HOG algorithms until we were able to develop a model that could identify bouldering holds with an acceptable precision and recall. These hyperparameters included the size and shape of each cell when computing HOG features, the size of the window in the sliding window algorithm (and therefore how many HOG cells it included), and the threshold for detection.

The second classification algorithm we implemented was a logistic regression classifier, again using the HOG method to obtain image features. The idea behind binary classification via logistic regression is quite simple. The features of the training images are extracted using the HOG algorithm to yield a feature matrix, ***θ***. The features of the candidate window in our testing data are then extracted to yield ***x***. The data is then fitted along a sigmoid curve given by the following equation:



The decision boundary is set at 0.5, indicating a 50% probability that the window contains a hold, given the extracted HOG features, ***x***. h***θ***(***x***) < 0.5 indicates a negative detection, and h***θ***(***x***) ≥ 0.5 indicates a positive detection.

As well, it is important to note that because we used three different sizes of processed images for our training data, we developed, trained, and tested three different models for our SVM and logistic regression classifiers. When running our classifier on test data, each of the three models was used to evaluate the image, and then the output from each model was combined into a single output image with boxes drawn around positive detections.

In addition, an unexpected and reoccurring problem that arose was that of multiple, overlapping detections around the same hold. This is most likely due to the nature of the sliding window algorithm, whereby a given window will contain part or all of a hold that the classifier is able to detect, but the subsequent iteration of the algorithm does not move the window far enough to preclude the same hold from appearing in the next window. To deal with this, we employed a technique known as Non-maximum Suppression (NMS). The principle idea behind this technique is take a series of overlapping detection windows that correspond to the same object, calculate the intersection of each window with all the remaining windows divided by their total union, and selects the individual window with the best intersection over union score as the one that captures the true detection of the object of interest. In our results, this process is depicted by multiple red detection boxes around a hold, with the correct window selected by the NMS algorithm depicted as a green box.

**Results**

Blahblahblah

**Discussion and Conclusions**

Blahblahblah