### USING SCALE Invariant feature transform to verify lime scooter

*Samuel Yeboah*

The University of Texas at Austin

#### Abstract

Lime Scooters are an electric rentable scooter that have become increased in popularity over the past few years. The scooters can be rented using a phone application and ridden around before being parked on a curb or sidewalk for future use. The scooters allow for increased mobility in the cities where they are present. However, they also present a challenge because they must be parked in public spaces. Many scooters are parked in the street, thrown on the ground, or even left on balconies or in apartments. In this paper, I will discuss my novel approach to validating the parking assignments of lime scooters. My approach utilizes the SIFT algorithm to generate a key point database that will be used as a reference for comparing the parking assignments of recently finished lime rides.

**DESIGNING THE APPLICATION**

In designing my idea, I decided to create a web application that would implement my project vision. Images of the UI can be found on the following page.

I created a workflow that allows a Lime scooter rider to upload a picture of their scooter after they have parked it. First, the user uploads a picture of the lime scooter to the application web page. After submitting the picture for verification, the application prompts the user to determine whether they believe their parking job is correct. The application then proceeds to the results page which shows the SIFT algorithms resulting determination. The application allows for repeated verifications.

The actual image processing utilizes homographic transformation, the SIFT algorithm, and a library of images to determine whether the Lime scooter has been parked illegally. The algorithm classifies correct parking as a scooter that has a vertical orientation in a two-dimensional space. Utilizing the scale invariant feature transform algorithm from OpenCV, I created a library of reference images that will be used for comparison. For each picture, I created a key point map where the key points are concentrated on the lime scooter. After visually examining the reference library key points, I computed the descriptors from each reference picture and saved them into a list. Then, I filter the matches using the ratio test described in Lowe’s SIFT paper however I utilize a ratio of .6

The application runs the same SIFT algorithm on the input image to create key points and descriptors on the input image as well. Once both sets of key point images have been created, I utilize a brute force match to create a match set between the input image and the reference image. To determine whether the Lime is present, I set a minimum match count that determine whether the picture is present or not. Afterwards, I utilize the RANSAC algorithm in the findHomograph function to find the transformation from the input image to the output image to determine the vertical orientation comparison between the original image and the reference image. The homograph I validates whether the scooter is lying on the ground or standing upright. Also, the homograph is used augments the SIFT in recognizing whether a scooter is present in the image.

**3. Conclusion / What We Learned**

My final product is a web application that validates the parking job of a lime scooter. The largest takeaways from this product were that I learned a tremendous amount about object recognition, SIFT algorithm, and topography. I also learned about the difficulties in creating a digital image processing project. I was able to use the SIFT algorithm to generate relevant key points that were important useful in determining whether the lime scooter was present in the input image. The descriptors were created from the key points computed in the original image. I learned how to modify certain factors of the SIFT algorithm to achieve a consistent result across multiple images. For my project, I limited the number of key points that could be created to 3000. I also used the ratio test described in the Lowe’s SIFT paper to determine which matchings were relevant to my object matching. The application works moderately well however there are multiple areas for improvement. I can improve the image library by taking using images of scooters that have a reduced number of edges in the image. By taking a picture of the lime scooter on a blank background, it allows the SIFT algorithm the focus solely on the edges created by the Lime scooter instead of the surrounding background. This will improve the creation of key points in the reference library. I can improve the algorithm by using another method to determine the transformational differences between the input image and the reference. Besides improvements to the image recognition, I want to create an Ios or android application as the user interface instead of a web application.

A picture containing road, outdoor

Description automatically generatedA picture containing text, outdoor, green

Description automatically generatedA picture containing text

Description automatically generated

*Figure4: Key point mapping of one reference image*

*Figure 2: Key point mapping of one reference image*

*Figure 3: Home screen where a user can upload a picture*

*Figure 1: Validation page after the processImage.py script runs*

A picture containing text, green

Description automatically generated

*Figure 5: UI page that asks for user opinion on lime parking*