Software Engineering in the Mariokart System

Wim Looman
wgl18@uclive.ac.nz

Coauthors: Simon Richards, Zachary Taylor and Henry Jenkins
scr52@uclive.ac.nz zjt14@uclive.ac.nz hvj10@uclive.ac.nz

Supervisor: Dr. Andrew Bainbridge-Smith
andrew.bainbridge-smith@canterbury.ac.nz

Department of Electrical and Computer Engineering
University of Canterbury
Christchurch, New Zealand

***Abstract*—Something amazing about engineering a software system.**

I. Introduction

The aim of this project was to explore the possible methods of tracking a marker using computer vision. The objective was to implement the chosen system on an autonomous go-kart for use in following a leading person or vehicle that possessed this marker. The system had to interface with a laptop that was mounted on this go-kart, the system had to operate in real time recognizing the marker, finding its location and outputting the path to travel on with minimal lag. The system had to be able to work in the environment of a mobile go-kart that could travel at up to 50 KMPH and was powered by a large electric dc motor. The kart was expected to be able to operate in both indoor and outdoor environments in a large variety of conditions. [**?**], [**?**]

II. Computer Vision Solutions Theroy

A large number of algorithms exist for tracking an object in a video feed. Some of the more common were examined to determine how appropriate they would be for performing the task of tracking an object which the go-kart could follow. To perform this task the algorithm had to be able to reliably give the position of the object relative to the kart including an accurate estimate as to the distance from the kart. It also had to work even when the object was moving quickly across the camera and at a large distance from the camera. It had to be robust to the large variations in lighting conditions and backgrounds that it would experience when mounted to the kart. It had to be able to be performed in real time giving out information on the position of the objects location with little lag and at a high refresh rate

Color detection was the first process looked at. This approach took the cameras image and converted it to HSV (hue, saturation and value) space [1]. In this space each pixel was classified by its hue or color, its saturation which is the intensity of the color and its value which represents how light or dark the color is. This hue was then taken to vary by very small amounts for the object during the tracking. For the most simple form of color tracking the image was then tresholded by a color in the range of the object being tracked. This would leave only the desired color present on the screen. A more robust form of color detection is the camshaft algorithm. This algorithm takes in a region around the object to track and creates a histogram of the hues present. It then scans successive frames for regions whose color distributions closely match the histogram.

Chessboard detection is often used in augmented reality applications to get the position of a marker relative to a camera so that a 3d model can be displayed with the correct size and orientation [4]. This algorithm uses a chessboard of known size as its marker. It thresholds the image, applies a Harris corner detection to find the corners in the image this is combined with a Hough transform to find the lines on the chessboard. Once all the corners and lines have been found the image is searched for a pattern of them that would give the chessboard [3]. This method considers all of the intrinsic parameters of the camera as well as the extrinsic transformation for the chessboards location in the scene. This means that the position of the chessboard marker can be accurately computed even taking into account the curving effect of the cameras lens at the corners.

A second method for detecting a marker in a scene is the SURF (speeded up robust features) algorithm. This algorithm operates along very similar lines to the SIFT (scale invariant features transformation) algorithm except operating at a greater rate, hence the reason for the ”speeded up” in its name. The surf algorithm matches points that are common to two separate images. In this case the images are a unique marker and an image of a scene in which the marker is located. The algorithm operates by selecting a large number of unique points. Finding these points involves finding areas of an image that are unique in there surroundings and that can be easily identified regardless of changes to the viewpoint [5]. By themselves these points could not be located in a new image so instead a small window surrounding each point is also saved. Each new frame of the scene is then scanned for areas that match these points.

SURF uses Harr like features to make this match.

Once the points are matched a second algorithm RANSAC (RANdom SAmple Consensus) can be used to extract the size and orientation of the object from the scene. RANSAC itself is not a computer vision algorithm but an iterative method to find parameters of a mathematical model within noisy data where redundancy in the number of points exists [7]. RANSAC works by randomly selecting four points in the image and from this working out the transformations the picture has undergone to fit these four points in the scene. Using this transformation a calculated location is found for all the other points. The distance between where a point is calculated to have been and where it actually lies in the image is found. This gives an estimation of the error in the transformations that the picture has undergone. If this error is small the transformation is kept and taken as correct, if it is large the process is repeated with four new points.

III. Implementation of Solutions

Color detection, SURF and Chessboard detection were all then implemented to examine their performance for tracking an object in the required environment. To aid in this implementation extensive use was made of the opencv computer vision libraries. These libraries were used as they contained implementations of a large number of computer vision algorithms that had been thoroughly tested by a large community and optimized to run as effectively and efficiently as possible.

The first algorithm implemented was the simple color based approach. In this approach a bright red ball was used as the marker to track. For this approach the scene was thresholded for a value that was approximately red and opening (that is erosion followed by dilation) was used to fill any holes in the image of the ball and remove background noise. A rough estimation of the balls location was then made using its x and y position in pixels from the center of the camera. The z location of the ball was taken to be proportional to the inverse of the size of the ball in pixels. These measures of x, y and z were extremely crude however they provided adequate for the testing that was performed on the color method.

The SURF algorithms marker was a printout of a ————————. This image was used as it contained a large amount of strong lines and detail that meant that a large number of unique and easily identifiable points could be located on the image.

Two chessboards were constructed and tested with the alg

IV. Path Following

Once the kart had located a marker a method for following the marker had to be implemented. The simplest solution of simply driving towards the marker had many drawbacks. An example would be that if the kart was following the marker and the marker moved forwards and then after a few meters turned to the right. The following kart would cut the corner and potentially smash straight into an obstacle. A second problem would be that if the marker left the karts field of vision the kart would be forced to immediately come to a stop as it no longer had anything to track.

The algorithm constantly updated the absolute position of the kart. It did this by using the karts speed sensor and a sensor that gave the angle of the karts wheels. Using this information and recording the time between runs of the kart the position and orientation of the kart could be found. Once the location of the kart had been found the relative position and orientation of the chessboard marker if present was also recorded. After this had been done the absolute position of the marker was calculated. The current and all previous positions were stored in an array.

The core of the path following algorithm was very simple and executed as follows. 1) The kart would select the first point on the path 2) The kart would set the steering to point at the point and drive towards it 3) When the kart came within a minimum distance of the line (set to the turning radius of the kart) or the point became too old the kart would deem itself to have reached the point 4) The kart would then repeat the process for the next point

By setting the criteria for meeting the point to twice the turning radius of the kart this meant that the kart would always be able to turn to be within this distance of the next point regardless of where it lay. The kart skipping a point if it became older then a set value (in testing set to 10 seconds) meant that if the kart began to lag behind the object it was tracking it would start cutting corners in order to catch up to the target. This meant that if the target was slowly moving through tight turns the system would work on matching its path and when the target was moving quickly the system would priorities keeping up with the target.

V. Performance of Solutions

The color detection algorithm required easily the least processing power of all the computer vision methods looked at running at a solid 20 fps only limited by the speed of the camera it was receiving the images from. It also performed the best under motion blur detecting the marker even when it was moved extremely quickly in front of the camera. The system however had the major drawback of being far less robust then the other methods of object detection, even small changes in lighting such as adding an additional light to a room caused the hue of the image to change sufficiently to require recalibration. The background also could not contain significant amounts of a similar color or this would be detected instead of the marker. A further problem was that the position calculations epically the depth were extremely crud. Lighting changed the detected size of the object to such an existent that the estimated distance could change by more than 50

The SURF algorithm also performed poorly in many situations. The algorithm ran slowly on the 2.1 GHz test laptop averaging 5 fps. The SURF algorithm was also largely affected by distance and motion blur. On the 340 by 480 pixel test camera the image had to in most cases take up around 50

Chessboard detection operated slowly if the chessboard had not been located in the previous frame only operating at a rate as low as 2 fps. Once the chessboard was found however the algorithm used the previous location to find it

more quickly raising the frame rate to around 15 fps. The chessboard showed some sensitivity to motion blur requiring the corners of the board to be clearly distinguishable for detection to occur. The system was by far the most robust to changes in lighting conditions with detection occurring in all environments without recalibration being required. Its detection of the position was the most accurate of all the tested solutions with an error of less than 10

VI. Conclusion

Three computer vision solutions were looked at for tracking a marker for use in allowing a kart to follow a lead vehicle. Color detection was found to be too sensitive to changes in lighting conditions for it to be used in a robust tracking system. The SURF algorithm ran slowly and its inability to handle even small amounts of motion blur made it inappropriate for the application of being mounted to a moving vehicle. Chessboard detection was the only system that was robust enough to detect a marker and give its position reliability and it was slightly hindered by it own inability to deal with large amounts of motion blur. The output of these systems were combined with sensors measuring the speed and wheel angle of the kart to produce a system that followed the path the marker had taken rather than the marker giving more accurate following.