HERO X CHALLENGE

PROJECT REPORT

Tyre Wear Estimation using Computer Vision

Author
Mohit VAISHNAV
vaishnavmohit1@gmail.com
Edinburgh, UK

Contents

List of Figures

1 Principle

Depth information of the tread is required to be measured which needs 3D reconstruction of the tyre surface. Once this information is available, depth can be easily measured. Stereo images are essential for such reconstructions, which gives the perception of depth.

2 Assumption

Some of the assumptions taken to complete this project are as follows:

- Camera is moved approximately parallel to the axis of the tyre minimizing the vertical motion.
- First and second images are taken about 1m from the surface of the tyre. The second image taken is taken at about 10-20 cm away from the first image.
- Field of View(FOV) is assumed to be at 45 degree in angle.
- Focal length is approximated using the above FOV.
- Disparity value terms of pixels is less than y px.

3 Procedure

First a pair of stereo images are acquired from normal smart phone camera by possibly sliding the axis as horizontally as possible. The distance could be approximated to about 10-20 cm. Once the images are in place, they form the a system of un-calibrated pair of images which are required to be rectified. OpenCV platform is used for solving Computer Vision related problems. For illustration purpose the procedure can be followed with a pair of test images Fig 1.

These images cannot be used as a whole because tread is too far from visibility and it is required to get the tyre shape as perpendicular as possible without any curvature. Hence the images are modified as shown in Fig. 2. Tyre is cropped in such a manner that it is very close to the observation point and tread are clearly visible. Dimension of both the cropped images has to be same otherwise it will have to be programmed to resize into similar shape before carrying out any operation. In this particular program above mentioned procedure is carried out manually but it could be

done automatically by creating a window of specific fixed size and explicitly taking information from that particular region of the image. As it has already been assumed to move the camera horizontally without any vertical movement hence the path extracted should be of a similar region.

(a) Left Image

(b) Right Image

Figure 1: Pair of test images

(a) Left Image

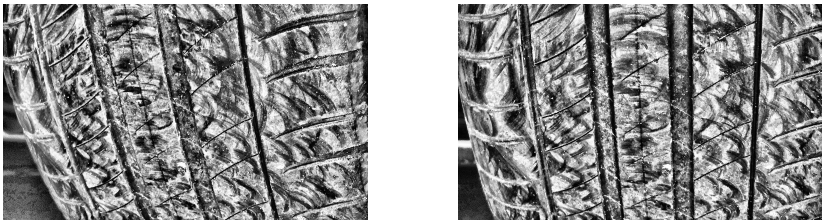
(b) Right Image

Figure 2: Close look-up Pair of test images

Before carrying out any operation, images have to be converted to gray scale which can be performed as follows:

For obtaining the better results, image equalization is performed on the input image which are useful in finding more detailed keypoints and descriptors. After image equalization is performed they looks like in Fig. 3:

Next, a set of matching points are estimated for which both of $SURF$ and $SIFT$ were used and because of better performance later was preferred. SIFT approximates Laplacian of Gaussian with difference of Gaussian for finding scale space whereas SURF approximates Laplacian of Gaussians with Box filter. Earlier times processors were not fast enough to computer in a real time using SIFT hence SURF was developed. Both the algorithms are used at present. In case of SURF, Hessian parameter has to be determined which relates to the Hessian filter output for the point of interest.



(a) Left Image (b) Right Image

Figure 3: Histogram Equalization of Pair of test images

A larger value will relates to more salient points which are fewer in number in contrast to smaller value giving numerous points. For this practical purpose Hessian optimal Threshold has to be taken.

Once the keypoints and descriptors are obtained for both the images which are further used for the $FLANN$ based matcher [1]. FLANN trains $flann :: Index\_$ on train descriptor collection and call its nearest search methods to find best matches. It is usually faster than brute force method when working on a large train collection. These keypoints could be observed in Fig. 4 when computed on tyre after histogram equalization. Use of this step could be seen from Fig. 5 where there are less number of points obtained for a set of tyre. Number of keypoints for matching purpose has to be same as the argument in calculating fundamental matrix require equal number of points.

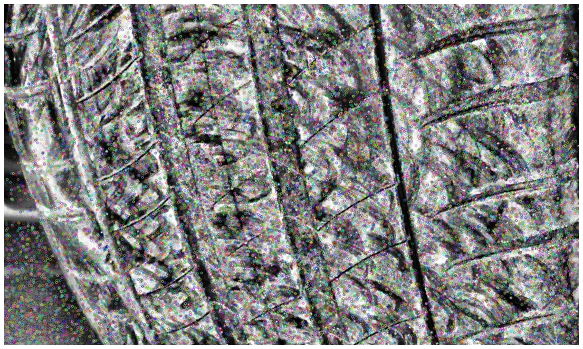


Figure 4: Kepypoints on tyre after histogram equalization

Using the user defined minimum and maximum distances, good points are picked up which acts as a set of input for the calculation of fundamental matrix. For its estimation, various options could be explored but $LMEDS$

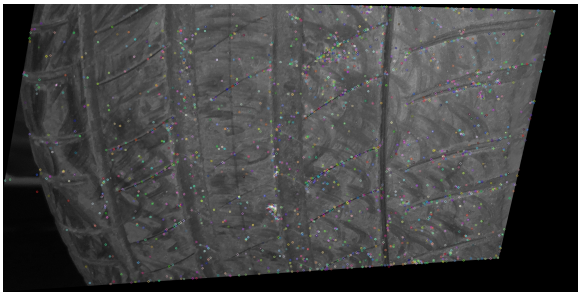
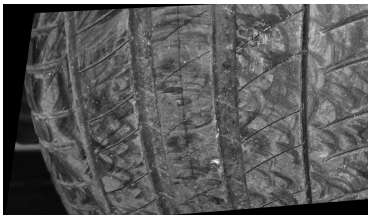


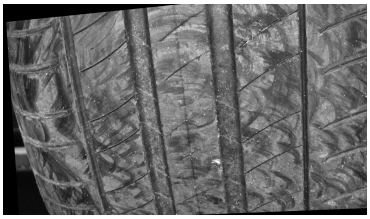
Figure 5: Kepypoints on a Normal tyre

and $8POINT$ gave the best results.

All these variables acts as an input for the rectification of images, giving transformation matrix (H) corresponding to the pair of stereo images. These corresponding images are rectified using that matrix and saved in the folder name "rectified" for analysis purpose.



(a) Rectified Image 1



(b) Rectified Image 2

Figure 6: Example showing Rectified Images

In some cases this rectification was not working appropriately. The reason could be the blurry or not so clear input image, deviation in the movement while taking photos, not many good keypoints to match, not much information over the tyre surface. In one of the test cases rectified images can be seen as in Fig 6. Few alterations working were either using $LMEDS$ or *Ransac*.

The above finishes the first half of the problem statement where uncalibrated images were now transformed into a pair of stereo images.

For rectification, two popular methods are, *StereoSBGM* [2] and *StereoBM* from which earlier one was preferred. This algorithm matches the block but could be done for the pixel by setting the block size as 1. It uses a simple Birchfield Tomasi sub pixel metric form. Parameter associated could

be updated once there is availability of the large amount of data set. Normalization of the obtained image plays a crucial role in obtaining correct values of the depth image. Images could also be again passed through the histogram equalizer for enhancement but this is optional. Fig. 7 shows the disparity map of the test images.

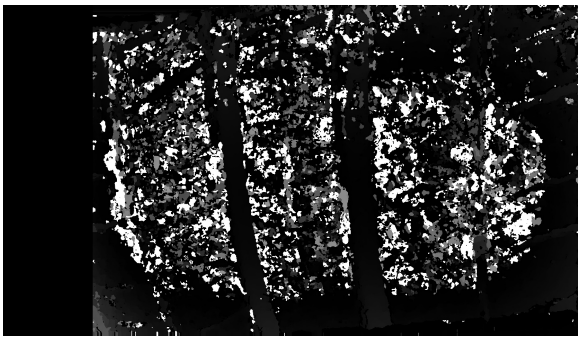


Figure 7: Disparity Map

Moving ahead, careful analysis of depth map is carried out. With the data set available, it was observed that any information where the disparity values comes out to be more than threshold (y) is taken as outliers. Because of the various restrictions imposed on the user, this assumption can be safely made. Once this is done, hardest part has arrived where the disparity value needs to be transformed back to give the actual information about the depth of tread of tyres. In *Python* function reciprocal does the inverse of a matrix element wise while in case of $C++$ it has to be first reciprocated then used further. A very small value is added in the step to omit the case of possibility of dividing by 0.

$$Z = \frac{f \times B}{d}$$

where Z = distance along the camera Z axis f = focal length (in pixels) B = baseline (in metres) d = disparity (in pixels)

Below mentioned is the procedure to do that task.

Term 1.2 comes into the picture because of the assumption of FOV as 45°. Normalization factor is taken as the size of the original image as all the calculations are done wrt to the cropped image whereas focal length and all the properties of image work wrt the original image. Horizontal displacement is assumed to be in the range of 10-20 cm for the safe calculation.

Varying this parameter changes the result but not significantly if taken in this range. So it could be made constant to know the approximation of the depth of tread of tyre.

Last part of the algorithm deals with the type of tyre we are working with. It takes the input argument as the maximum tread width possible for any type of tyre 8.33 (in case of car). It gives the depth information of only those required regions and rest is negated. This is one of the fast approximation to obtain the results in a real time scenario. Either histogram could be created for this kind of information or an averaging could be done which gives the information for average depth of the tyre tread acting as a benchmark for the decision.

4 Results and Conclusion

This algorithm has been tested for a small data set and it works well, as seen in Fig. 11, where the correlation between actual and estimated depth is good enough. However, it can be said that more data has to be tested before production software could be used. Extreme conditions like in Fig. 9 are not conducive for practical purpose because tyre images are not visible properly.

One of the issue was to look for the average depth with in the tread. Either we have to know the region in which depth have to be known and carry out the similar procedure for that region or we can visualize using the Histogram of the values as seen in Fig. 12. Other alternative of this is to use the disparity map and color code (Fig. 13). Another issue is about the depth which keeps on changing at each point on the tyre because of the unequal wear of it.

5 Future Work

Enumeration of the improvement could be as follows:

1. Integrate the sectional profile of a tyre rather than the average behaviour.
2. Rather than manually cropping the image, it could be made automatic.
3. Focal Length is approximated in the implementation but EXIF data from the images can be used to estimate the exact focal length to obtain precise value.

4. Robustness could be increased by integrating the applicability of this code on different types of tyres available widely. For this different profiles could be created to include its functionality over different types of tyres.

6 Acknowledgement

This work is done under the academic supervision of Hugues Talbot, Professeur, Centre for Numerical Vision (CVN), CentraleSupélec, Université Paris-Saclay.

References

[1] https://docs.opencv.org/2.4/modules/features2d/doc/common_interfaces_of_descriptor_matchers.html?highlight=flannbasedmatcher#flannbasedmatcher

[2] https://docs.opencv.org/3.3.1/d2/d85/classcv_1_1StereoSGBM.html

[3] https://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

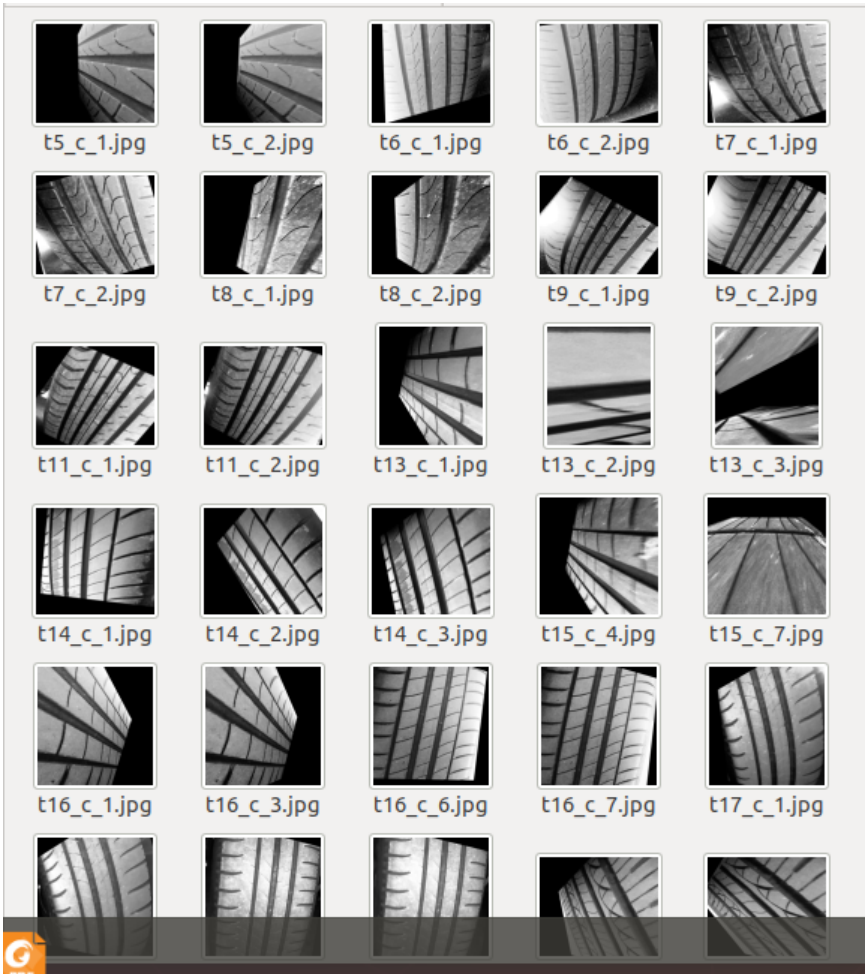
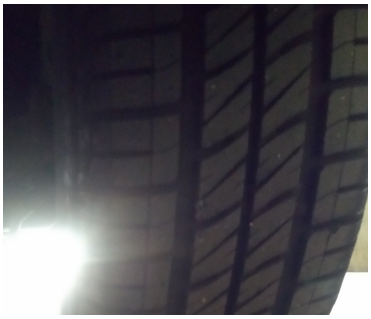


Figure 8: Rectified Images using C++



(a) Light in background

(b) Blurry Image

Figure 9: Different Condition of Images

Figure 10: Disparity Examples

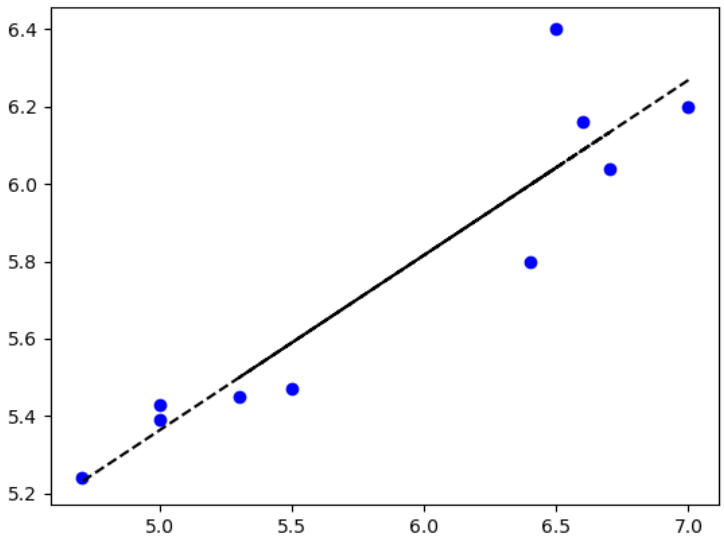


Figure 11: Result (Measured v/s Actual depth)

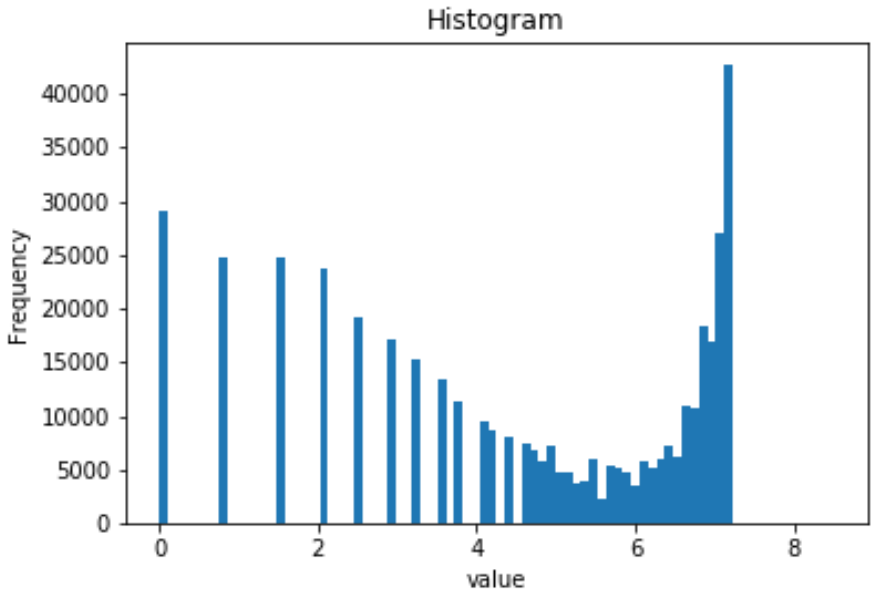


Figure 12: Histogram

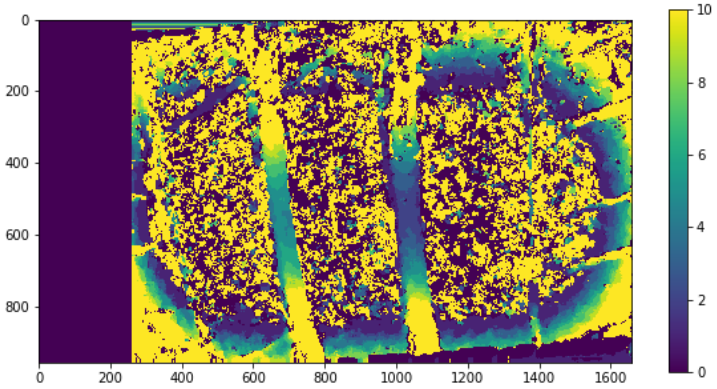


Figure 13: Distance Perception