CS5680 – Fall Semester 2017
Final Project Report
Austin Derbique A01967241

Paper Details

Name of Paper: CrackTree: Automatic crack detection from pavement images

Authors: Qin Zou, Yu Cao, Qingquan Li, Qingzhou Mao, Song Wang

Publication Source: Science Direct, Available on Utah State University – University Libraries

Publication Year: 12 November 2011

# of Citations: 162

Paper URL: https://www-sciencedirect-com.dist.lib.usu.edu/science/article/pii/S0167865511003795

Implementation

Components of System

1. **shadowMask = FindShadow(rgb image).** This component takes an input of a rgb image and returns a binary image mask of where the shadow is.
2. **noShadowIm = RemoveShadow(grayscale_image, shadowMask).** As followed in the paper, this component takes an input of an enhanced grayscaled image and binarized shadow mask image and performs computations to return an image with no shadow. Because the algorithm for detecting a shadow was not mentioned in the paper, I modified one from here: https://www.mathworks.com/matlabcentral/fileexchange/54647-shadow-detection
3. **CrackMap = GenerateCrackMap(noShadowIm).** The input for this is an image with the shadow removed. The output is the map of crack seeds.
4. **Tensor Field = Find_features(CrackMap, sigma_value).** This part takes the input of the crack seed map and creates a tensor field. With direction and magnitude of each crack. The output is a 4D array called T.
5. **Voting Components = Convert_tensor_ev(Tensor Field).** This part takes the 4D array and votes on where the cracks are likely to occur. The output is a map with these cracks.
6. **Pruned curves = Calc_ortho_extreme(Tensor Field, radius,size).** This takes the crack map after being tensor voted and performs operations to narrow down the results to which are the likely cracks. The result is re which will eventually be overlayed with the original image.

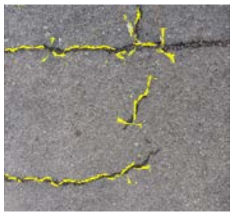
Improvements

The addition of a median filter on top of the Gaussian filter helped with reducing the amount of false positives for crack seed detection. Another improvement I made was condensing operations down into fewer for loops, therefore saving on running time.

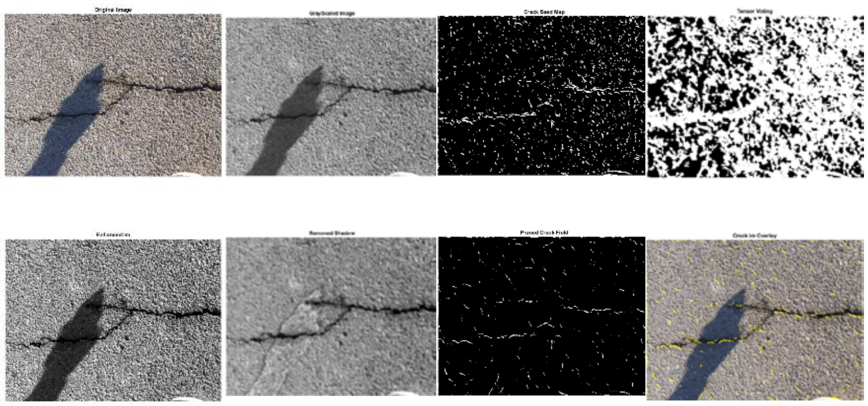
Changes

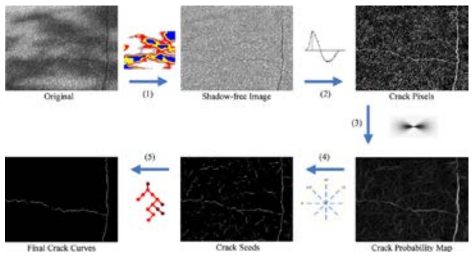
While this paper gave the overview of the algorithms used, it was very broad in defining specifics. This led to me trying different values such as thresholds, radius, widths, etc. The paper used a minimum spanning tree at the end, which was not properly documented. This made it impossible to know exactly how the paper had implemented a solution. This meant that I used a pruning algorithm that detected the orthonormal extremes. This was pulled and modified out of a library for tensor fields.

Part 1: Experimental Results



← My experimental output





Geodesic Shadow Removal **Tensor Voting & Pruning** **Paper Implementation**

The Top left figures show the geodesic shadow removal that I implemented from the paper. The center figures show the crack map and tensor voting along with pruning. The image to the left shows the final result of the program I implemented. I realized that a lot of values had to be tweaked to get the charactericts I wanted. That makes me question the paper into how well it actually did.

Part 2: Other results to show effectiveness of system

Using a timer, the program took on average 4 seconds to run, which is quicker than the results in the paper. For more results and figures, look inside the results folder.

Source Code

All source code can be found in the implementation code folder.

References

Shadow Detection: https://www.mathworks.com/matlabcentral/fileexchange/54647-shadow-detection

Tensor Voting: https://www.mathworks.com/matlabcentral/fileexchange/21051-tensor-voting-framework