Image Dilution using Harris Corner Detection and Geometric Kernels

1st Aiden James
Department of Computer Science
Department of Mathematics
University of North Carolina Charlotte
Charlotte, NC, USA
ajames61@uncc.edu

2nd Xingjie Li
Department of Mathematics
University of North Carolina Charlotte
Charlotte, NC, USA
xli47@uncc.edu

***Abstract*—Image files contain large amounts of data. An image is essentially a matrix of values, often represented as RGBA (red, green, blue, alpha) arrays. A large three dimensional matrix with a height and width likely in the hundreds can take long for a program to read. However, many applications may only require specific key features to understand an image. The majority of pixel data is relatively unimportant when determining the contents of the image file. In fact, such extra data can sometimes deceive the machine, or in the case of a hybrid image [1], the human viewer. Using only basic matrix calculations and matrix convolution, specifically Harris Corner Detection and Edge Detection kernels, we have extracted the key features of an image in order to dilute its data. The results show that any confusion, such as noise or the low frequencies of a hybrid image, becomes weaker, and that the computation is very efficient and robust.**

keywords *– Harris Corner Detection, Edge Detection Kernel, Dilution, Denoising, Hybrid Image, Matrix Convolution*

I. Introduction

Because an image is essentially a matrix of RGBA vectors, it contains extraneous data. Using only simple matrix calculations, we have converted the image to a binary matrix of the same size which retains most of the important features. In the code itself, these images consist of only ones and zeroes; one in the matrix is considered a white pixel, and any zero is considered a black pixel. Working with a sparse binary matrix is much easier for a machine to compute. As well as this, the high frequencies of an image are outlined which makes confusion, such as noise or the low frequencies of a hybrid image, become weaker [2].

We may obtain the binary representation of this image by using matrix convolutions and kernels. Our first approach uses the edge and line kernels [3]. First, the image is convoluted with an edge kernel, and any pixel above a pre-selected threshold is set to one, otherwise it is set to zero. After this initial convolution, various kernels employed to detect lines and corners are applied with different weights in each cell, giving each white pixel and its eight neighboring pixels a score and finding their sum. If any of these kernels have a high enough sum, the pixel is kept, otherwise it is deleted. This filters out any white pixels that are not adjacent to other white pixels, however it does not filter out groups of pixels including common imaging noise. Most detail is kept, along with some leftover noise.

James was supported by the UNC Charlotte OUR program and Li was partially supported by the NSF DMS-1847770.

To overcome these issues, we improve the algorithm by using Harris corner detection for filtering. Harris corner detection [4] is an algorithm designed to determine whether a pixel in a given image is a corner, edge, or center pixel. We will discuss how this works in sections II and III. While Harris corner detection is normally used only for finding the corners [2], [5], we have used it for filtering an image to extract a binary reduced representation. Any pixel with a high likelihood of being a corner or an edge is set to one in the binary reduced matrix. Rather than attempt to filter out noise, we check that the pixel is a part of an edge before including it in the binary matrix. Some pixels may be considered corners if they are close to a corner pixel, so to check for this we also apply an edge kernel and make sure it is above a certain threshold.

The effect when using a hybrid image as the input is that the moderate to high frequencies are kept, making it easier to see the sharper of the two images more easily. Hybrid images appear different depending on the distance from them because the high detail features are not visible from far away. What this means is that the sharper image one sees when they are closer is higher contrast, and will be retained when the image is filtered. This means that the lower frequency components are filtered out and the illusion is removed; only a white outline of the high frequency image is left.

Our main contribution of this work is summarized below: We have used Harris corner detection to dilute an image to its most important components. Anything that we detect as a corner or an edge is set to one in a binary reduced matrix. The resulting image is further streamlined by an edge kernel while its important features are reserved. This sparse matrix allows for the image to be read later much more efficiently.

This paper will be organized as follows: in section II, we will review the mathematical formulations of Harris corner detection and matrix convolution with edge kernel; in section III, we will demonstrate the algorithm used for image denoising and dilution; in section IV, we will test and discuss the performance of this algorithm on several benchmark examples and the comparison with other methods; in section V, we will conclude this work and propose some future work.

II. Overview of Methods

We first review the Harris corner detection method. In Harris corner detection, I_x and I_y are set to image derivatives in x and y directions respectively. In the simulation, we approximate them at pixel (i, j) by the finite difference:

$$I_x(i,j) := \frac{1}{2}\left(I(i+1,j) - I(i-1,j)\right),$$
$$I_y(i,j) := \frac{1}{2}\left(I(i,j+1) - I(i,j-1)\right).$$

Next, we defined the difference matrix $M\big|_{(i,j)}$ by averaging the derivatives at pixel (i, j) and its neighbors via a Gaussian window function ω:

$$M\big|_{(i,j)} = \sum_{x,y\in \mathsf{Neigh}(i,j)} \omega(x,y)\begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}, \tag{1}$$

where a Gaussian window function $\omega(x, y)$ is applied with pre-selected window size L. In the simulation, we used the default Gaussian window function. Next, we compute the Harris Corner Detection R score of pixel (i, j) with a pre-selected k value:

$$R := \mathsf{det}(M) - k\big(\mathsf{trace}(M)\big)^2. \tag{2}$$

The number output R has three possible outcomes, either being close to zero, medium, or very large, depending on the pixel values, pre-selected $k = 0.05$ and threshold value. Large values are corners, medium values are edges, while near zero values are more likely interiors. For the purposes of our new application for this algorithm, anything not considered near-zero is filtered out, as corners and edges are both desirable features to keep. The main strength of this system is that it finds the edges of objects without relying solely on contrast, allowing unimportant pixels to easily be filtered out. However, this method also creates much larger areas of interest, as the windowing function causes surrounding pixels to also be read as corners. To mitigate this issue, we also checked that an applied edge kernel, which is introduced in (3), met a certain threshold on the pixel.

As a comparison, we also implement the method for dilution without Harris corner detection. First, the image is convoluted with an edge kernel. Then, every pixel is either set to a 1 or a 0 based on if it has enough luminance. After this, multiple kernels are applied to check for possible lines. If any one of these meets the threshold, the pixel is kept. This removes a lot of noise, but clumps of pixels usually remain, unlike the Harris corner method. This method only detects linear edges or corners, with single pixel deviations considered part of the important information. More detail is kept, however less noise is filtered out.

Unlike this system of many kernels, Harris corner detection is not dependent on a set number of configurations. Although the edge filters have some weights to allow for variation in the lines they detect, Harris corner detection views the image in a more non-local and nonlinear way than the edge and geometric kernel algorithm. Curves are also detected as part of the edge when using Harris corner detection.

III. Methodology

For each method, we first make an empty binary matrix. Applying Harris corner detection citation needed, we are able to determine the likelihood that a pixel is part of an edge or corner. If the odds of this are high, we apply an edge detection kernel

$$E_{\text{Edge}} := \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{3}$$

to the pixel, and if its contrast meets a threshold then the corresponding entry in the binary matrix will be set to one. Otherwise, the pixel will be set to zero. The simple use of only two numbers, as well as the fact that this leads to a sparse matrix, allows for any later calculations to be performed much faster. The exported images are multiplied by 255 to show clearly what is stored in the data, however this would not be needed if the image is intended to only be read by a computer.

Before trying the corner detection method, we originally tried using an edge kernel and a set of other geometric kernels, which we designed to be 5-by-5 instead of 3-by-3 in contrast to the edge kernel. This allowed us to view the wider context of the surrounding pixels to see if they formed geometrical elements. An edge kernel is applied directly to the image. Pixels in the convoluted image whose values were above a threshold of ten were kept in the binary matrix. Then, the following geometric kernels were applied in every applicable orientation:

$$E_{\text{Line}} := \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}, \tag{4}$$

$$E_{\text{Diagonal}} := \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}, \tag{5}$$

and

$$E_{\text{Corner}} := \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 2 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{6}$$

If any one of these kernels or their rotations convoluted with a value of $T_4 = 6$ or higher, the pixel was kept. The goal was similar to our use of Harris corner detection, which was to filter out any potential noise by only keeping pixels that were likely parts of an edge.

A. Algorithms

Here, we summarize the pseudo code of the algorithm used in this work. We first state the algorithm for the Harris corner filter combined with edge kernel

Algorithm 1: Harris Corner and Edge kernel Filter

Input: Image, Edge kernel, $k = 0.05$, Threshold T_R for R, Edge kernel threshold T_E

Output: Binary Matrix with the same size as the image matrix

```
1: for each pixel at coordinates $(i,j)$ do
2:     $I_x = \frac{1}{2}(I(i+1,j) - I(i-1,j))$
3:     $I_y = \frac{1}{2}(I(i,j+1) - I(i,j-1))$
4: end for
5: for each pixel at coordinates $(i,j)$ do
6:     $M\big|_{(i,j)} = \sum_{x,y} \omega(x,y) \begin{bmatrix} I_xI_x & I_xI_y \\ I_xI_y & I_yI_y \end{bmatrix}$
7:     $R = \mathsf{det}(M) - k \times \mathsf{trace}(M)^2$
8:     Calculate the applied edge kernel on the pixel
9:     if $R$ meets the threshold $T_R$ and the edge kernel meets its threshold $T_E$ then
10:        Set the corresponding cell in the binary matrix to a value '1'
11:    else
12:        Set the cell to a value '0'
13:    end if
14: end for
```

In the simulation, we set $T_E = 10$ and T_R for each individual example, respectively. Next, we state the algorithm for the edge kernel and linear kernel filter without Harris corner method.

Algorithm 2: Edge Kernel and Geometric Kernels Filter

Input: Image, Edge kernel, Several geometric kernels, Threshold T_3, Geometric kernel threshold T_4

Output: Binary Matrix with the same size as the image matrix

```
1: convolute image with edge kernel $E_{\text{edge}}$ (3)
2: convert convoluted image to grayscale
3: create a binary matrix the same size as the image matrix
4: for each pixel $(i,j)$ in image do
5:     if the pixel's luminance meets the threshold $T_3$ then
6:         set the corresponding cell in the binary matrix to a value '1'
7:     else
8:         set the cell to a value '0'
9:     end if
10: end for
11: for each white pixel in binary matrix do
12:     apply each geometric kernel to the pixel and store the value
13:     if any single geometric kernel value meets a threshold $T_4$ then
14:         remove the pixel
15:     end if
16: end for
```

In the simulation, we employed one edge kernel (3) and three geometric kernels (4)-(6) and set thresholds $T_3 = 10^{-2}$ and $T_4 = 6$.

IV. Results

We tested both algorithms

- Algorithm 1: Harris corner and Edge kernel Filter
- Algorithm 2: Edge kernel and Geometric Kernel Filter

on several images of various types and found *consistent results between them if we set the thresholds T_R for Harris corner R score manually for each image*. Automation of this threshold is something we plan on implementing in the future.

A. Dilute the image of a group of giraffes.

We first tested our algorithm on the image of giraffes [6], as their features, such as their patterns, are obfuscated with 20% added Gaussian noise, and there is high contrast grass in the background. (See Figure 1). Notice that only using an edge kernel yielded a very noisy result, whereas the Harris corner filter alone did not distinguish between important features, such as the fur patterns and the noise. The background of the image is retained in the output. It is notable that the extra edge kernel check allowed the filter to retain more detail in the pattern. The final edge kernel check mitigates the the possibility of incorrectly identified corners.

The giraffe image was also processed using the old method of edge and geometric kernels. As shown in Figure 1, the applied 20% Gaussian noise was too much for this system, but lowering the noise to only 1% yields a better image. More detail is retained, some noise is unfortunately kept in the background. Even with sharper details, the higher presence of noise is an undesirable result.

B. Dilute several hybrid images of various types.

We then tested the several hybrid images: (1) a portrait of Harry Potter and Einstein [7]; (2) a video screenshot of a hand gesture surrounded by a kitchen background of similar color [8]; (3) a vase in shadow (taken by the author Aiden James). The results are summarized in Figure 2.

We found that the most optimal algorithm is Harris corner detection combined with the edge kernel check. The edge kernel is important because details may be lost using Harris corner detection. Pixels are considered corners even if they are simply close to corner pixels. The main strength of Harris corner detection is that noise is easily detected as not part of an edge. In contrast to this, the method using the edge and geometric kernels (4)-(6) is less likely to detect noise.

The original concept of our research was to use this to extract only one object from a hybrid image. Hybrid images may appear to be one of two images, so removing the low frequency components would make it easier to detect the sharper of the two images. We tested the filters on a hybrid image of Einstein and Harry Potter [7], and only the white outline of Einstein was kept. Gaussian noise with a standard deviation of 20 (out of 255) was applied to every test image to ensure that this program would work with imperfect input

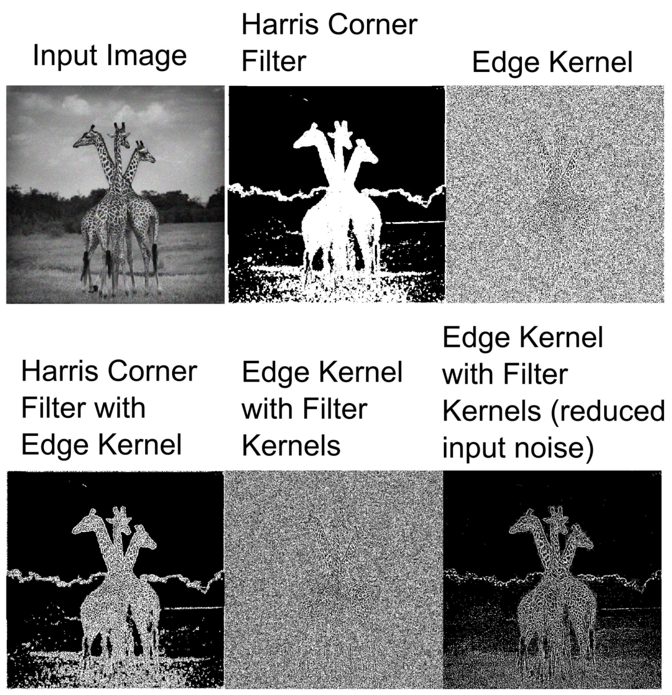
Input Image
Harris Corner Filter
Edge Kernel
Harris Corner Filter with Edge Kernel
Edge Kernel with Filter Kernels
Edge Kernel with Filter Kernels (reduced input noise)

Fig. 1. Comparison of image dilution of different filters upon the same image with 20% Gaussian noise: applying the Harris corner detection filter only; applying the edge kernel convolution only; applying the proposed algorithm; applying the edge and geometric kernels; applying the edge and geometric kernels with only 1% Gaussian noise.

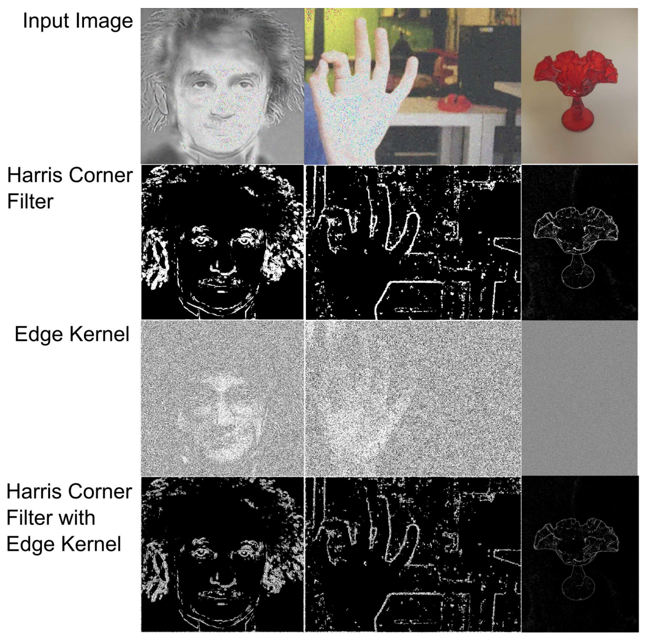

Fig. 2. Comparison of image dilution of different types of images using three methods: **Row 2** is obtained by applying the Harris corner detection filter only; **Row 3** is obtained by applying the edge kernel and three geometric kernels convolution; **Row 4** is obtained by applying the proposed algorithm combining Harris corner with edge kernel.

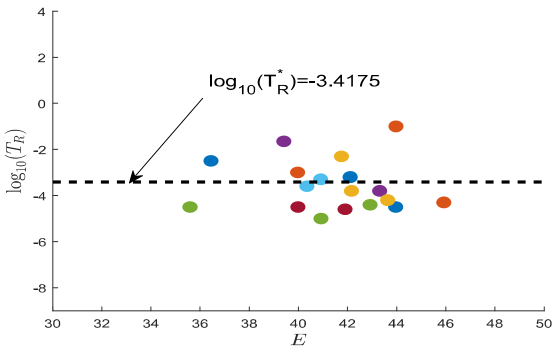

Fig. 3. The average edge convolution value E verse $\log_{10}(T_R)$ of 20 images. The image source can be found at [7].

data. The edge kernel alone left too much noise for the image to be interpreted correctly. Harris corner detection, on the other hand, was able to filter out almost all of the noise. There is no resemblance of Harry Potter in the filtered output.

The image of a hand gesture contains a lot of noise as well as a confusing background of a kitchen. The image also has a very low resolution. Most of the hand is kept in the output, however parts are missing where it overlaps with the table that is similar in color. A lot of background elements are kept, but they appear low-fidelity.

In the third picture, the vase is outlined very well despite the overlaid shadow and added noise because of the image's high resolution and plain background.

C. *Empirical study of threshold T_R for Harris corner.*

Currently, the threshold for the Harris corner filter must be decided manually. A good starting point for the filter is $T_R^* = 10^{-3.4175}$. However, we have not determined the formula for automatically calculating this variable. We found the value of T_R^* because we tried to set a mathematical relation between the average edge convolution value E

$$E = \frac{1}{M \times N} \sum_{(x,y)\in \mathsf{Img}} \Big(E_{\text{Edge}} * \mathsf{Img} \Big)(x,y)$$

and T_R, where Img is the grayscale image matrix of size M-by-N and "$*$" denotes the matrix convolution. We have calculated the average of the best thresholds T_R based on the empirical study of 20 images, which is around $T_R^* = 10^{-3.4175}$. We plotted the average edge convolution value E verse $\log_{10}(T_R)$ of 20 images in Figure 3. To automatically calculate the threshold, it may be easier to use the average of the R score values, but we have not tested this.

Most of the time, similar images have similar ideal thresholds. In the future, we may find a way to calculate this automatically.

V. Conclusion

Using elementary matrix operations, it is possible to extract only the most important features of an image. Unlike existing methods of feature extraction, our methods only require simple calculations to achieve this. Using a new application of

Harris corner detection as a dilution filter, along with a post-processing edge kernel, we diluted the image to an outline of its key details. Using another method involving a set of weighted geometric kernels, we were able to filter remaining noise out of an edge convolution. In both of these methods, the image was converted to a binary matrix, making any calculations much more efficient due to the lack of extraneous information.

More efforts needs to be put into investigating the threshold calculation, especially in regards to the Harris corner detection algorithm. While it is generally a predictable number close to $T_R \approx 10^{-3.4175}$ according to the empirical experiments of 20 images, perfecting the automatic calibration of this number is something we will work towards. Since there is seemingly little correlation between the average edge value and the ideal threshold in the Harris corner algorithm, more mathematical possibilities should be explored.

References

[1] Aude Oliva, Antonio Torralba, and Philippe G. Schyns. Hybrid images. *ACM Transactions on Graphics*, 25, 2006.

[2] Jie Chen, Li hui Zou, Juan Zhang, and Li hua Dou. The comparison and application of corner detection algorithms. *Journal of Multimedia*, 4, 2009.

[3] Utkarsh Sinha. Image convolution examples, 2010. Article found at https://aishack.in/tutorials/image-convolution-examples/.

[4] C. Harris and M. Stephens. A combined corner and edge detector. *Proceedings of the Fourth Alvey Vision Conference*, 1988.

[5] Olfa Haggui, Claude Tadonki, Lionel Lacassagne, Fatma Sayadi, and Bouraoui Ouni. Harris corner detection on a numa manycore. *Future Generation Computer Systems*, 88, 2018.

[6] Image sourced from https://www.lirent.net/2012/04/black-and-white-animal-photography/.

[7] Image sourced from http://cvcl.mit.edu/hybrid_gallery/gallery.html.

[8] Image source from https://nofilmschool.com/2017/06/watch-4-ways-you-can-reduce-and-avoid-grainy-footage,.