Generating Mosaics: A recreation of the "Novel Artificial Mosaic Generation Technique"

Aishwarya Pani Tabitha Roemish

***Abstract*—In this paper, we recreate the novel algorithm for creating Mosaics as described in a computer vision research paper using Computer Vision techniques. The algorithm used in the research paper uses Gradient Vector Flow to determine the angle of each tile placed and uses a unique set of heuristics to fill in the mosaic with non-overlapping tiles which results in a realistic mosaic like image. For our project we interpret and code the proposed algorithm using C++ and the OpenCV library. We take an image as an input, as shown in Fig. 1, and output an image stylized as a mosaic. The paper discusses the algorithm's key steps, including luminance extraction, Robert's Gradient application, GVF edge map generation, non-maximum suppression, and tile placement heuristics.**

While the results differ from the original research paper, the project provides valuable insights into gradient analysis, edge detection, and the process of creating mosaics using computer vision. Suggestions for future work involve refining tile laying techniques and exploring alternate approaches for achieving uniform tile placement. Overall, the project enhances our understanding of computer vision and creating art through mosaic generation.

***Index Terms*—Computer Vision, Mosaic**

I. Introduction

FOR or this project we recreated the algorithm proposed in "A Novel Artificial Mosaic Generation Technique Driven by Local Gradient Analysis"[1] using C++ and the OpenCV library. The research paper provides pseudo code for their algorithm which includes five major steps: 1) Get Luminance, 2) Use Robert's Gradient, 3) Get a gradient vector flow edge map, 4) Use non-maximum suppression, 5) Place tiles in a heuristic approach. We will discuss how we implemented each step using the OpenCV library and what is accomplished by each of these methods. We then provide an example of our results compared to the research paper and after that discuss possible future work.

II. Methods

While the paper provided direction on how to implement their mosaic approach, the path from pseudo code to actual code can pose a lot of structural and practical questions. In this section we discuss the instructions in the the paper and how we implemented those instructions. We also explain what each step accomplishes for the final output image.

A. Luminance

In this step we remove the color information so that we are only dealing with the luminance of the image. Using OpenCV's cvtColor method we converted the image to grayscale using COLOR_BGR2GRAY. This allows us to simplify the processing of the image.

Fig. 1. Original Lena Image

B. Robert's Gradient

Robert's Gradient is the process of using 2 X 2 kernels to traverse the image.

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

By convolving the image with these kernels we can obtain the magnitude by finding the sum of squared differences between diagonal pixels. This exaggerates the differences between the diagonal pixels and produces an image with edges defined. This process is heavily influenced by noise in the image. The paper uses this edge detector specifically saying,

> "Notice that in the implementation gradient computation is performed using Robert's Kernel. This choice is more noise sensitive and hence incorporates in the final mosaic a little, aesthetically pleasant, randomicity."[1]

In our results, it's hard to say if using Robert's gradient gave our final output a more aesthetic quality. A lot of the aesthetics are degraded by the gaps between our tiles but using Robert's gradient was a good way to see, in a simple way, how the edges of images are defined using magnitude.

C. Gradient Vector Flow

One of the novel methods in this mosaic creation technique is using gradient vector flow to position tiles. The gradient

vector flow map takes the changes in the x and y coordinates at the gradient defined edges of the image and then ”generalized diffusion equations” are used to calculate GVF forces. These GVF Forces are propagated over the entire image. In this way a sort of topographic map is created for the image so that values near 1 are the high altitude regions, i.e. the edges, while values near zero are like low altitude regions, i.e. areas away from image edges and the GVF forces are pointing towards the high altitude regions on either size of those regions. [4] What this does for mosaics is provide edge corresponding angle laying information for non-edge tiles.

The gradient vector field was designed by the authors of ”Snakes, shapes, and gradient vector flow”[3] and while we found one example of how to achieve this map image[4], it was beyond our grasp to make the process work. What we settled on was taking the gradient orientation of each tile to determine the angle for tile placement. To achieve this we used the Scharr method in OpenCV which takes in an image and will return the change in x coordinates and y coordinates. With the change in the x and y coordinates we could use arctan2 to find the angle of each tile. This resulted in more sporadic angling in our tiles while the mosaic in the research paper[1] contained more rows of uniform tile placement.

D. Non-Maximum Suppression

Non-Maximum Suppression thins out overlapping points. For the proposed algorithm, it allows tile placement to focus only on the most important edge points. For our project, OpenCV did not provide a stand-alone function for Non-Maximum Suppression so the solution was to write a method that looked at two points on either side of a base point. These points were set according to the gradient angle of the base point. If the base point magnitude was greater than that of the two neighbor points it was retained as a value, otherwise it was set to zero. In this way, if a point was not of greater than or equal magnitude to its neighbors, it was effectively removed. Which left only the most edge like edge points for tile placement.

E. Placing Tiles

The tile placement heuristics were the second novel method in the mosaic algorithm. Essentially, the edge tiles are separated from the other tiles in the image using a threshold. Then those tiles are placed, one by one, starting from the tile with the largest gradient. Once an edge tile is placed, perpendicular tiles are placed on either side of that edge tile based on a second smaller threshold. This process continues until all edge tiles are reviewed and placed (or not). Once that is completed, the remaining tiles are placed. The thresholds divide the tiles into edge tiles, adjacent tiles and the rest. By laying the edge and adjacent tiles first, the image edges are clearly defined by the tile structure which adequately mimics the appearance of an ancient mosaic.

Within the algorithm there were questions that came up such as, what should the thresholds be, why are tiles being placed and then checking for placement and what does ”skip positioning” mean? In our implementation we decided to do a percentage of the overall image magnitude to determine the threshold values. We use a map that contains RotatedRect objects and use that map to determine if tile placement will work and then place the tile if the answer is true. We took skip positioning to mean do not place tile. The minute details of laying tiles in an image, we believe, accounted for the greatest differences between our image and the research paper mosaic of Lena which will be evident in the results. We were not sure what method the authors used to reduce the gaps between tiles. Using the gradient vector field to obtain uniform angles probably accounted for some of the smaller gaps, but they had a method of laying tiles that we did not know and thus were not able to achieve the same effect as in their mosaic. We settled on a brute force method of tile laying with some basic heuristics, but this did not seem to be as effective as we hoped.

III. Results

To test our results we did a visual review. We compared our final output image, as shown in Fig. 3, to the output provided in the research paper, as shown in Fig. 2. As previously mentioned, there were some challenges that resulted in very different image outputs including using the angle of each tile instead of using a GVF Field to determine the angle and a less effective tile laying algorithm.

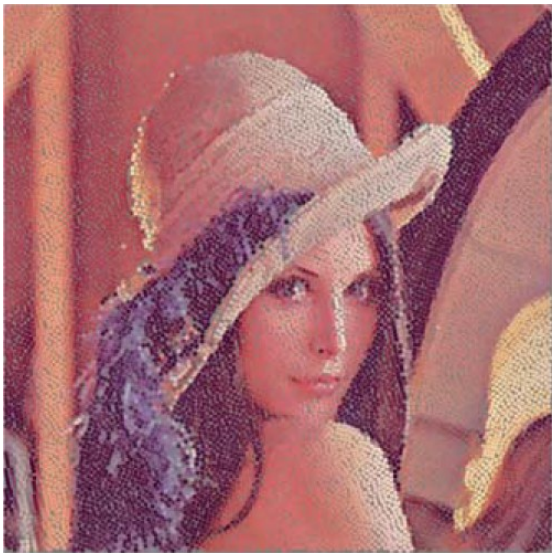

Fig. 2. Mosaic from Research Paper[1]

Compared to Fig.2, you can see that our final output image (Fig. 3) suffers from large gaps but it is hard to say how different it would appear if we could minimize our gap issues. Also to speak to our successes, we did effectively obtain the edge tiles and place them according to those edge angles. While the gaps make the image appear less aesthetically pleasing, it is a vast improvement from the images we were generating initially, as shown in Fig 4, which had more grooves. We believe we did follow the tile placing heuristics of the research paper effectively. Finally, what seemed like a simple idea, generate a mosaic from an image, was deceptively tricky as there are a lot of details that needed to be accounted

for and implementing those details resulted in a lot of code. So simply getting a mosaic resembling image as our final output is a sort of success.

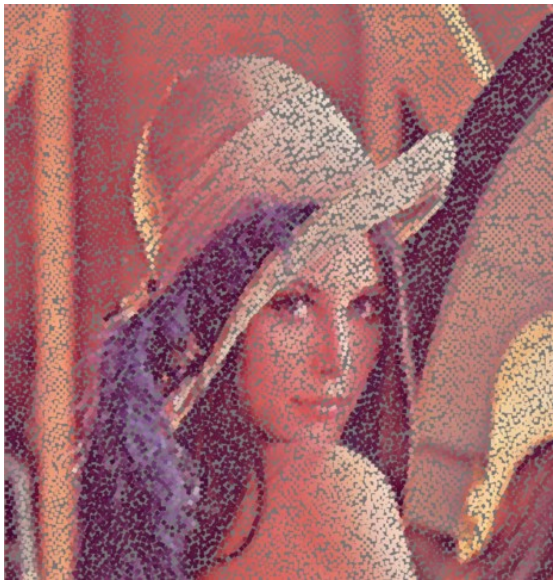

Fig. 3. Our Final Output Mosaic

Fig. 4. Lena with grooves

IV. Future Work

While we were not able to get the gradient vector flow map to work, an alternate idea would be to alter the tile laying algorithm such that, when placing the adjacent tiles, rather than using the angle of the tile like we did, we could use the angle of the edge tile. This would generate a result much like the GVF map in that all tiles moving away from the edges would be facing the same angle until a tile was found that was out of the adjacent tiles threshold. This would result in more uniform tile laying around the image edges. We also have some theories on improving our brute force tile laying algorithm that involve finding the largest overlaps instead of finding the first overlap. This might result in tiles moving closer to the nearest tiles, thus reducing gaps.

V. Conclusion

In the end, we were not able to follow the whole algorithm completely or with as much success as the authors in the research paper. We did not get the gradient vector field which might have given us more uniform edges and we did not unlock the tile laying prowess they possessed which left large gaps between our tiles. There were successes and failures in this project but all of it was educational. The OpenCV library had a lot of useful methods and structures that we were able to use. We learned a great deal about gradients including their direction and orientation and how they are used to define edges in images. This was a challenging project but it was fun to learn about computer vision while trying to create art.

References

[1] Battiato, Sebastiano & Blasi, Gianpiero & Gallo, Giovanni & Guarnera, Giuseppe & Puglisi, Giovanni, *A Novel Artificial Mosaic Generation Technique Driven by Local Gradient Analysis*, 2008

[2] Rosebrock, Adrian. "Image Gradients with OpenCV (Sobel and Scharr)" May 12, 2021 https://pyimagesearch.com/2021/05/12/image-gradients-with-opencv-sobel-and-scharr/

[3] Prince, L., Xu, C. "Snakes, shapes, and gradient vector flow" IEEE Transactions on Image Processing 7(3) (1998) 359–369

[4] Prince L., Jerry, Xu, Chenyang "Active Contours, Deformable Models, and Gradient Vector Flow" https://iacl.ece.jhu.edu/Projects/gvf/