Automatic Text Removal from Images Using Mumford-Shah Inpainting

Addison Howenstine, Hunter Lee, Aditya Srinivasan, Josh Xu

Duke University

ECE 590: Image Processing

Introduction

It is common for text to obscure photos, which may often detract from the image itself. For example, timestamps, captions, and watermarks are often unwanted but present. Consider a situation in which someone accidentally left the timestamp feature enabled on their digital camera but forgot to set the correct date; the photographer would likely want to remove all of the incorrect timestamps.

Our project solves these common problems with a GUI used to streamline a two-step process: detecting text and inpainting it. Text may manifest differently in different pictures and this adds difficulty to removing the text. The GUI offers the user choices for possible algorithms that may help to remove the text from the image for varying situations.

Approach & Methods

I. Input Image

- Our most general input image is one with obscuring text. Without loss of generality, the text can be of any font, weight, size, and color, and appear in any part of the image.

II. Detect Text & Obtain Mask

Methods of detecting text:

- **Edge Detection (Prewitt) —** The Prewitt operator calculates the gradient of the image intensity at each pixel and uses that information to detect edges at points where the change is most abrupt. Then the edges are made thick which effectively generates a mask we can use for inpainting.
- **Otsu's Method —** In the special case where the image has primarily two colors, with the text being a different color from the underlying image, Otsu can be used to automatically threshold the histogram and perform a segmentation, giving us the desired text binary mask.
- **Color-Select Masking —** This algorithm takes user input for the text color and creates a mask surrounding areas with a concentration of this color value.
- **User Generated Mask —** As a last resort, if a user finds the program is having difficulty isolating the text specifically, the user can input a premade mask.

III. Run Inpainting Algorithm

- The Mumford-Shah functional is defined as follows, where **I** is the particular image, D is some domain of the image, J is the image's model, and B are the boundaries associated with that model:

$$E[J,B] = C\int_D (I(\vec{x}) - J(\vec{x}))^2 d\vec{x} + A\int_{D/B} \vec{\nabla} J(\vec{x}) \cdot \vec{\nabla} J(\vec{x}) d\vec{x} + B\int_B ds$$

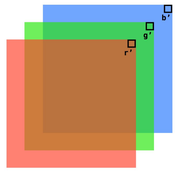
- This functional is optimized by approximating it with another functional, proposed by Ambrosio and Tortorelli.

$$u = \operatorname{argmin}_u \frac{\alpha}{2}\int \Omega\, \chi^2\, |\nabla u|^2\, dx + \beta \int \Omega \vdash\left(\epsilon|\nabla\chi|^2 + \frac{(1-\chi)^2}{4\epsilon}\right)\dashv dx$$

- This is solved iteratively via alternating solutions of the Euler-Lagrange equations for u and x.

IV. Post-processing

- If median filters were used, the output image may need to be sharpened.
- For initial, unique images, inspect results and decide whether a better text detection method would have been better.

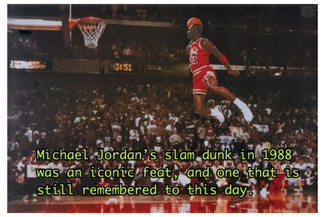


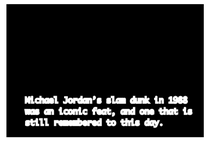
Given:
Pixel color value: RGB(r, g, b)
where 0 ≤ r, g, b ≤ 255

Mapping:
if (r − 50 < r' < r + 50)
and (g − 50 < g' < g + 50)
and (b − 50 < b' < b + 50)
then RGB -> (255, 255, 255) *white*
else RGB -> (0, 0, 0) *black*

Results

Edge Detection (Prewitt)

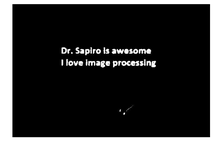




Areas of high gradient contrast are found and edge lines are thickened to overlap text. Works well for text on images with few significant edges.

Otsu's Method

Significant color differences are detected using Otsu's Method and a mask is created from the less common color area.



Color-Select Masking

The color of text to be removed is input by the user and a mask is created surrounding this text. Most useful for consistent color selection, for example a specific camera's timestamp color.

User Generated Mask

When other methods do not result in satisfactory results, a user can input a manually generated mask for inpainting.

Discussion

As can be seen, each text masking technique has its own strengths and weaknesses for various image styles.

We found edge detection to be the most versatile text masking technique. However, it was also the most prone to inpainting non-text contours which the user did not want to be inpainted.

Otsu's method and color-select masking were both useful for segmenting a single-colored text. Otsu's method was more useful in automatically detecting this color change while color-select masking required user input but produced more robust results.

Next steps:

The next steps for this project would go in a number of directions.

- Finding ways to improve the performance of the inpainting algorithm such that it would work better with examples where the text colors blended with the background of the photo.
- Training the program to not only detect the text automatically but to decide which text masking technique to use on a specific image.

Conclusion

The techniques used in this project are useful for automatically detecting undesired text in an image and removing this text from the image. Useful applications of our program include automatically masking text in forms such as watermarks, timestamps, overlaid meme text, and captions. Various text identification methods were found to be useful for various situations and a helpful GUI was generated to allow a user to make an informed decision about the best method to choose. In the background, a mask can be created and applied to an image with the Mumford-Shah algorithm to remove text from the image and return it to the user in a more satisfactory form with the text removed.

References

Schonlieb, Carola-Bibiane, Partial Differential Equation Methods for Image Inpainting, Cambridge University Press, November 2015.

Parisotto, Simone and Schonliebe, Carola, MATLAB Codes for the Image Inpainting Problem, GitHub repository, MATLAB Central File Exchange, September 2016.

Esedoglu, S., & Shen, J. (2002). Digital inpainting based on the Mumford–Shah–Euler image model. European Journal of Applied Mathematics, 13(04).