Rochester Institute of Technology

Imaging Science Undergraduate Senior Project
Final Report

Text Inpainting and Its Application in Video CAPTCHA Text Removal

Author:
Huaijin (George) Chen

Project Committee:
Dr. Richard Zanibbi, Advisor
Dr. Carl Salvaggio
Dr. Jeff Pelz

May 20, 2013

Abstract

Video CAPTCHA improves the usability while reducing the attack success rate compared to a conventional text-based CAPTCHA, however remains vulnerable to OCR attacks. Thus, we proposed to use inpainting algorithms to remove the contained text in the video frames, so that the OCR success rate can be reduced without introducing unwanted artifacts that displease the endusers. We implemented Bertalmio's Image Inpainting for our particular case of text inpainting, and were managed to prove through OCR attacks on inpainted text-containing image that inpainting can potentially help to improve the security of Video CAPTCHA.

Contents

1 Introduction

1.1 Objectives

The motivation of this senior project is to improve the reliability of the current Video CAPTCHA by implementing an image inpainting algorithm to eliminate visible text within the video, thus reduce the risk of being bypassed using OCR attacks while avoiding visual artifacts to be shown to the endusers. We completed the following work to prototype a pipeline that we take in images or decomposed video frames that contain text, inpaint the specified text region, and eventually evaluate the inpainting performance.

1. Investigated four major inpainting algorithms, and chose Bertalmio's Image Inpainting [1] for our inpainting step
2. Concluded a proper stop point of the iteration and a optimal dilation size for the text region mask for our particular case of inpainting, because those are not specified in the Bertaalmio's paper [1]
3. Developed framework to pre-process the inpainted images for the OCR attack
4. OCR attacked the pre-processed images and measured the attack success rate using the string edit distance.
5. Assessed the inpainting through perceptual appearance of the inapinted image.

1.2 Conventional artwork inpainting

In art world, the word "inpainting" referred to the process of precise restoring of the lost or damaged part of paintings. This technique has been widely used by skilled art conservators to recover valuable paintings Figure 1. A typical inpainting process, in both traditional and modern sense, has the goal of restoring the unity of the image.

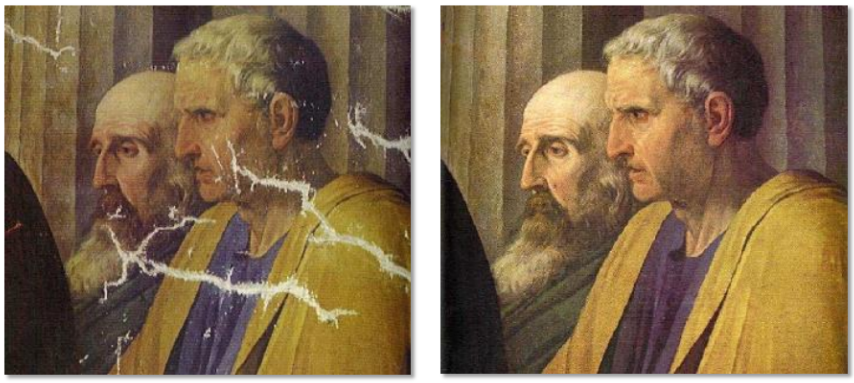


Figure 1: An inpainting example shown in [8]

As described in [8], a typical workflow of inpainting is consisted of the following steps:

1. Defining the inpainting region
2. Since structure near the boundary needs to be propagating into the area, contour lines intercept at the boundary are extended into the region.
3. Different areas inside the inpainting region segmented by the contour lines are filled with the corresponding colors matched at the boundary.
4. Textural details are extended into the region.

1.3 Digital Image inpainting

Bertalmio's [1] *Imaging Inpainting* is arguably the first paper on digital image inpainting. The algorithm proposed in the paper exactly follows the fundamental inpainting principles stated above.

Given the inpainting region and boundary of the inpainting region, the goal of such algorithm is to prolong the isophote lines (the contour lines of different gray-scale level) arriving at the boundary, while maintaining the angle of arrival (Figure 2).

Figure 2: Contour line propagation process

Therefore, this algorithm proceeds drawing from the boundary inward in this way, while curving the prolongation lines progressively to prevent them from crossing each other. [1] A sample inpainting process from the paper is shown in Figure 3.

Figure 3: Inpainting process shown in Bertalmio's paper [1]

1.4 CAPTCHA

CAPTCHA, or Completely Automated Public Turing test to tell Computers and Humans Apart, is a human-easy and AI-hard test that widely used nowadays to avoid machine generated spams or attacks to online services while letting human users to pass through.

The most common use cases of CAPTCHA can be preventing bots registering thousands of accounts at free email services, avoiding cheating at online polls or preventing a login portal from dictionary attacks.

1.4.1 Text-based CAPTCHA

The most commonly used CAPTCHAs are the text-based ones shown in Figure 4, where the users are prompted to type the words that are contained in the figure. The words are in a twisted fashion, so that it is not easy for the computer to perform a OCR, but it should be easy for human to recognize. However, the two major disadvantages about this method are low human success rate and the insecurity against OCR attack.

1.4.2 Video CAPTCHA

To solve the problems conventional text-based CAPTCHAs have, Kluever and Zanibbi [9] developed a novel video-based CAPTCHA method.

In this method, the system will take in a video clip from user-contributed video sites, i.e. YouTube[1], then ask the user to watch the clip and type in three keywords about the video. After the keywords are entered, the system will then compare them with the keywords tagged on YouTube. If the result match, then the

[1]So far this application is for internal academic-purpose use, we have not started looking at copyright issues yet. But once it goes public, this will be a important issue to concern.

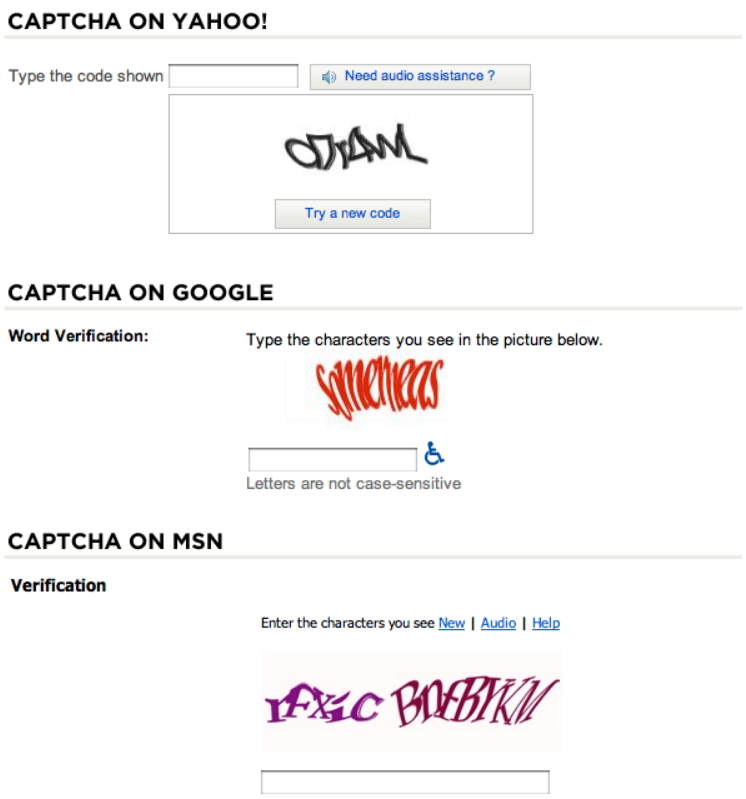


Figure 4: Text-based CAPTCHAs [18]

test is passed. This method efficiently increased the human success rate while reducing machine-attack pass rate, because scene understanding is an easy task for humans, but a considerably difficult task for machines.

However, there are still potential problems of this method. The most obvious one as mentioned above is that videos are very likely to contain not only the scene but also the text, which can be highly related to the tagged keywords. The existence of text in the scene makes it possible for the computer to crack the CAPTCHA by performing an OCR attack. In such case, the OCR attack might be even easier than text-based CAPTCHA, because the text remains unprocessed. Therefore it is crucial that we can remove the text sufficiently and naturally for a valid video CAPTCHA

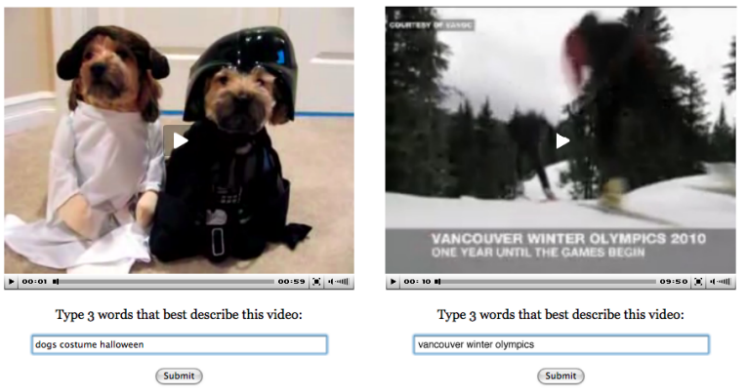


Figure 5: Samples of video CAPTCHA, the right might lead to higher attack success rate due to the contained text [15]

Also this method brings the possibility of word frequency attack. For instant, keywords "music" and "video" have much higher chance to appear [9] than other tags. Using these two keywords to attack gives you higher probability to succeed. So we need to survey the keywords tagged on YouTube, and exclude those words from the list of valid keywords.

2 Methodology

2.1 Inpainting Algorithms

We examined four inpainting algorithms which will be discussed in this section, including the *Image Inpainting* [1] and *Simultaneous structure and texture image inpainting* [2] by Bertalmio, et al, *Region filling and object removal by exemplar-based image inpainting* [6] by Criminisi, et al, and Total Variation-based method proposed by Chambolle [3].

2.1.1 Image Inpainting by Bertalmio et al [1].

We first implemented Bertalmio et al's *Image Inpainting* [1]. This iterative algorithm is basically consisted of two steps: Inpainting and Diffusion.

Inapinting Step

Given the nature of the manual inpainting procedures, Bertalmio, et al developed a iterative digital inpainting algorithm that can be generally described below using [Equation 7]. Given a 2D greyscale image I of size $M \times N$,

$$I^{n+1}(i,j) = I^n(i,j) + \Delta t I_t^n(i,j) \tag{1}$$

where (i,j) are the pixel coordinates, which needs to be inpainted within the inpainting regionΩ, $I^n(i,j)$ is the input image at each iteration and $I_t^n(i,j)$ is the update to the input image, and Δt is the rate of improvement. Therefore at each iteration, the output $I^{n+1}(i,j)$ is the improved version of the input $I^n(i,j)$. Ideally at as n increases, the result is getting finer, to a certain n that the improvement is too small to be noticed by a human observer, thus should be considered as the stopping point of the algorithm.

As stated in the previous section, we need to smoothly propagating information $L^n(i,j)$ from outside the inpainting region Ω into the region Ω across the inpainting boundary $\delta\Omega$ along the direction $\vec{N^n}(i,j)$, so that

$$I_t^n(i,j) = \delta\vec{L}^n(i,j) \cdot \vec{N^n}(i,j) \tag{2}$$

where $\delta\vec{L}^n(i,j)$ is a measure of the change in the information $L^n(i,j)$.

To make to propagation smooth, $L^n(i,j)$ should be a image smoothness estimator. Thus we can use the discrete Laplacian shown in 3

$$L^n(i,j) = I_{xx}^n(i,j) + I_{yy}^n(i,j) \tag{3}$$

where the subscripts represent the derivatives. Therefore, the change $\delta\vec{L}^n(i,j)$ of the discrete Laplacian $L^n(i,j)$ is calculated using

$$\delta\vec{L}^n(i,j) = (L_n(i+1;j) - L_n(i-1,j), (L_n(i;j+1) - L_n(i,j-1), \tag{4}$$

The direction $\vec{N^n}(i,j)$ is the key element is our propagating process, simply using the direction that is perpendicular to $\delta\Omega$ can be problematic because as [Figure 6] shows, lines arriving at the boundary can be falsely inpainted. Therefore, the isophotes direction is chosen since isophotes tend to align with $\vec{N}$.

We use then a time varying estimation of the isophotes direction field suggested in the paper. For any point (i,j) in the region, the discretized gradient vector $\nabla I^n(i,j)$ gives the direction of largest spatial change, while its 90 degrees rotation $\nabla_\perp I^n(i,j)$ is the direction of smallest spatial, therefore is the isophotes direction.

So our field $\vec{N}$ then can be calculated by $\vec{N}(i,j,n) = \nabla_\perp I^n(i,j)$. We are using a time-varying estimation that is coarse at the beginning but progressively achieves the desired continuity at $\delta\Omega$, in stead of a fixed field

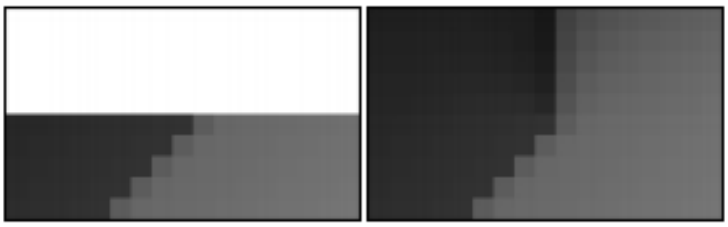


Figure 6: Failure case of inpainting, if the inpainting is done along the direction that is perpendicular to the isophote direction, as shown in [1]

$\vec{N}(i,j)$ that is set at the beginning. We choose not to normalize our direction field as the paper suggested to help ensure the numerical stability of the algorithm, the reasons of which are explained in [11], [14].

Hence the calculation of the isophotes direction becomes

$$\frac{\vec{N^n}(i,j,n)}{|\vec{N^n}(i,j,n)|} := \frac{-I_y^n(i,j), I_x^n(i,j)}{\sqrt{(I_y^n(i,j))^2 + (I_x^n(i,j))^2}} \tag{5}$$

To summarize, we estimate a variation of the smoothness, by using a discretization of the 2D Laplacian, and project this variation into the isophotes direction, where the projection only affect pixels within the region Ω. As a result, the discrete calculation of $I_t^n(i,j)$ becomes

$$I_t^n(i,j) = \left(\delta\vec{L^n}(i,j) \cdot \frac{\vec{N^n}(i,j,n)}{|\vec{N^n}(i,j,n)|}\right) \tag{6}$$

Finally, we multiply it by a slope-limited version of the norm of the gradient of the image $|\nabla I^n(i,j)|$

$$I_t^n(i,j) = \left(\delta\vec{L^n}(i,j) \cdot \frac{\vec{N^n}(i,j,n)}{|\vec{N^n}(i,j,n)|}\right) |\nabla I^n(i,j)| \tag{7}$$

We used the slope-limited version of the norm of the gradient of the image because a central differences realization would turn the scheme unstable. So for each pixel, we calculate

$$|\nabla I^n(i,j,n)| = \begin{cases} \sqrt{(I_{xbm}^n)^2 + (I_{xfM}^n)^2 + (I_{ybm}^n)^2 + (I_{yfM}^n)^2}, & \text{when } \beta^n > 0 \\ \sqrt{(I_{xbM}^n)^2 + (I_{xfm}^n)^2 + (I_{ybM}^n)^2 + (I_{yfm}^n)^2}, & \text{when} \beta^n < 0 \end{cases}$$

where

$$\beta^n(i,j) = \delta\vec{L^n}(i,j) \cdot \frac{\vec{N^n}(i,j,n)}{|\vec{N^n}(i,j,n)|} \tag{8}$$

The subindexes b and f denote backward and forward differences respectively, while the subindexes m and M denote the minimum or maximum, respectively, between the derivative and zero.

Diffusion Step

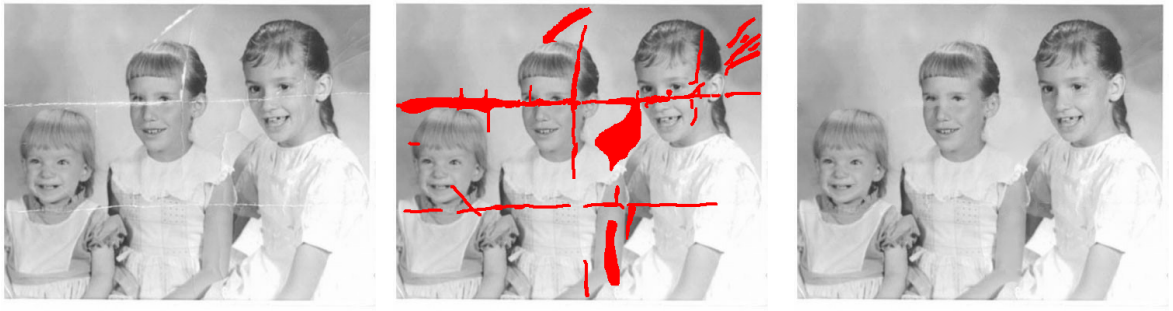
To further propagate information into the region and ensure the smoothness of the propagation, a diffusion step is suggested to be applied after each iteration of inpainting by the paper. Anisotropic diffusion is used in order to achieve goal without losing sharpness in the reconstruction. The discrete anisotropic diffusion can be described using the following equation.

$$\frac{dI}{dt}(x,y,t) = g_\epsilon \kappa(x,y,t)|\nabla I(x,y,t)| \tag{9}$$

where(x,y) is the pixel coordinate in the $\delta\Omega$, which is the dilation of Ω with a ball of radius ϵ

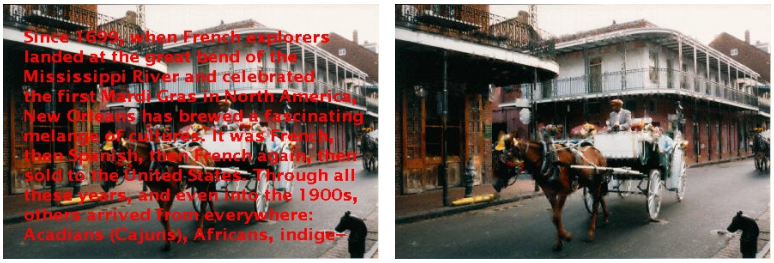
Implementation

We implemented this algorithm, we set the total iteration T = 3000, with inpainting iteration t = 17 and diffusion iteration d = 2. The inpainting results are shown below in Figure 7 and 8.



(a) The original image (b) Marked inpainting region (c) Inpainting result

Figure 7: Result of the Bertalmio's inpainting algorithm on damaged image



(a) The original image (b) Inpainting result

Figure 8: Result of the Bertalmio's inpainting algorithm on text-imposed image

2.1.2 Simultaneous structure and texture image inpainting

Though performing well in most cases, a major disadvantage of the original Bertalmio's method is that the details can't not be well preserved as we propagate information from outside the boundary $\delta\Omega$ into the inpainting region Ω through iteratively filling and blurring process. A revised version of the original inpainting algorithm are proposed by Bertalmio, et al. [2] later in 2003 titled *Simultaneous Structure and Texture Image Inpainting*.

Decomposition

The key innovation of this algorithm is that it first decomposes the target image to two sub-images: a structural sub-image and a textural sub-image. For the structural sub-image, the algorithm will treat the inpainting task the same as the original algorithm, while for the textural sub-image, it uses texture synthesis technique to directly duplicate the fine details from the surrounding region.

As we can see, the critical step in the new inpainting algorithm is the image decomposition. Bertalmio, et al based their decomposition algorithm on Vese, et al's texture modeling method descirbed in *Modeling Textures with Total Variation Minimization and Oscillating Patterns in Image Processing* [16].

In this final project, we will focus on the optimization component of decomposition step in the image inpainting algorithm mentioned above.

Let $I : \mathbb{R}^2 \to \mathbb{R}$ be a given image $I \in L^2(\mathbb{R})$, we can decompse it into two sub-image: 1) structural part that looks like the cartoon of the image $u(x, y)$, 2) textural part $v(x, y)$ that contains the fine detail of the image. Then,

$$I(x, y) = u(x, y) + v(x, y) \tag{10}$$

We can take the problem of reconstructing u from I as a minimization problem in the space of functions of bounded variation, allowing for edges:

$$inf_{u \in BV} \left[F(u) = \int |\nabla u| + \lambda ||v||^2_{L^2}, I = u + v \right] \tag{11}$$

where $\lambda > 0$ is a tuning parameter. The first term here in the enegy function is a regularizing term, and the second term is afidelity term, so that we can remove noise and small details while keeping important features and sharp deges.

The paper suggests that a small λ in the enegy function will remove the texture. Since the goal is to extract both the $u \in BV$ component (structural cartoon), and the v component (oscillating function representing texture of noise) from I, the model then should be conducted using a different space of function. The idea is that if (2) is used , then v will not just contain oscillations, but also undesired brightness deges. The following definition is introduced:

Let G denote the Banach space consisting of all generalized function $v(x, y)$ which can be written as

$$v(x, y) = \partial_x g_1(x, y) + \partial_y g_2(x, y), g_1, g_2 \in L^\infty(\mathbb{R}^2) \tag{12}$$

induced by the norm $||I||_*$ defined as the lower bound of all L^∞ norms of the unction $|g|$ where $g = (g_1, g_2)$, $|g(x, y)| = \sqrt{g_1(x, y)^2, g_2(x, y)^2}$ and where the infimum is computed over all decompositions (3) of I.

Therefore the now model becomes

$$inf_{u \in BV} \left[F(u) = \int |\nabla u| + \lambda ||v||_*, I = u + v \right] \tag{13}$$

The authors provided the solution of a variant of this model using only of simple partial differential equations.

$$inf_{u,g_1,g_2} \left[G_p(u, g_1, g_2) = \int |\nabla u| + \lambda \int |I - u - \partial_x g_1 - \partial_y g_2|^2 dxdy + \mu \left[\int \left(\sqrt{(g_1^2 + g_2^2)} \right)^p dxdy \right]^{\frac{1}{p}} \right] \tag{14}$$

where λ, μ are tuning parameters, and $p \rightarrow$. The first term ensures that $u \in BV$, the second term makes I approximately equal to $u + div(g_1, g_2)$ while the third term is a penalty on the norm in G of $v = div(g_1, g_2$

For the $p = 1$ case used in the paper, the corresponding Euler-Lagrange equations are

$$u = I - \partial_x g_1 - \partial_y g_2 + \frac{1}{w\lambda} div \left(\frac{\nabla u}{|\nabla u|} \right) \tag{15}$$

$$\mu \frac{g_1}{\sqrt{g_1^2 + g_2^2}} = 2\lambda \left[\frac{\partial}{\partial_x}(u - I) + \partial_{xx}^2 g_1 + \partial_{xy}^2 g_2 \right] \tag{16}$$

$$\mu \frac{g_2}{\sqrt{g_1^2 + g_2^2}} = 2\lambda \left[\frac{\partial}{\partial_y}(u - I) + \partial_{xy}^2 g_1 + \partial_{yy}^2 g_2 \right] \tag{17}$$

Inpainting and texture synthesis

Once the decomposition is finished, we then can inpaint the structural sub-image using the algorithm proposed in Bertalmio et al's original inpainting paper [1] and conduct texture synthesis for the textural sub-image. The smiler patch searching and replacing process can be done by looking and comparing the adjacent the patches outside the inpainting region Ω and replacing it with the corresponding patches at same relative geometric location. The scheme is shown in Figure 9a, and an example result given in the paper is shown in Figure 9b

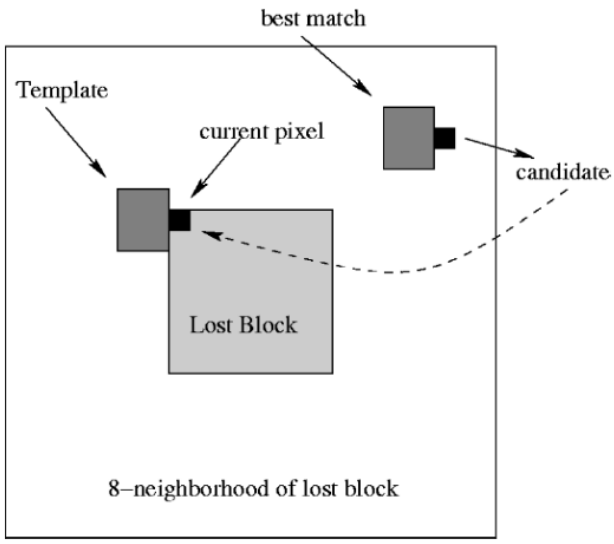
2.1.3 Region filling and object removal by exemplar-based image inpainting

2.1.4 Inpainting by minimizing the total variation (TV)

Given the fact that the minimization procedures used in the other inpainting algorithms we examined are brute force search based, we studied TV-based inpainting algorithm with the hope that it can increase the efficiency of the inpainting, because TV minimization can be easily modeled as convex optimization problem. We then can have a large variety of highly efficient convex optimization tools to choose from.

Definition

According to [5] and [17], for a real-valued continuous function f, defined on an interval $[a, b] \in R$, its total variation on the interval of definition is a measure of the one-dimensional arc-length of the curve with



(a) Texture synthesis process [2]

(b) top left: original input image; top right final inpainting result; middle left: decomposed strutral sub-image; middle right: decomposed textural sub-image; bottom left: inpainted structral sub-image; bottom right: texture-synthesised textural sub-image [2]

Figure 9: Concept of texture synthesis process in [2] (left), and the sample inpainting result (right)

parametric equation. $x \rightarrow f(x)$, for $x \in [a, b]$. In the case of image processing, we can define the total variation as the L1 norm of the magnitude of the gradient

Formulation of the problem

As proposed by Rudin et al [14], if we assume the noise or damages in the input signal (i.e. image) to be sparse and excessive, then it tends to increase the total variation, as we expect the ideal signal to be smooth. Therefore, we can remove the unwanted noise by reducing the total variation, while ensure that it deviates from the original signal within a certain range. In this way, we can remove the unwanted detail while still preserving important details such as edges.

Thus, for a input signal x, we can derive a de-noised signal s, when the total variation of the input signal x is minimized:

$$s = \min_{x} ||t(x)||_{TV} \tag{18}$$

such that

$$||b - Ax||_2 \leq \epsilon \tag{19}$$

where x is the calculated signal, $t(x)$ is the transfomation that convert the 2-D image into 1-D signal, $|| \cdot ||_{TV}$ is the L1 norm of the magnitude of the gradient, b is the original signal with noised added, A is a identifier for the target inpainting area.

Implementation

We can chose Douglas Rachford method [7] to solve the optimization problem, as it can optimize the objective functions with the form of $\min_{x \in R} f_1(x) + f_2(x)$

We set or objective function as

$$\min_{x \in R^N} f_1(x) + f_2(x) \tag{20}$$

where

$$f_1 = ||t(x)||_{TV} \tag{21}$$

and

$$f_2 = i_{c2}(x) = \begin{cases} 0, & if||Ax - b||_2 \leq \epsilon \\ 1, & \text{otherwise} \end{cases} \tag{22}$$

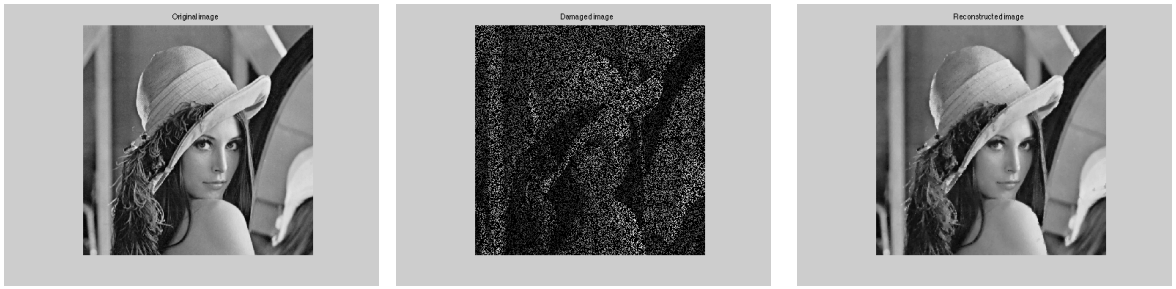
Our designed variable is simply the signal x.

We used MATLAB convex optimization tool box **UNLocBox** to implement this inpainting algorithm. Following the instruction of the toolbox manual [13], we define the objective function as above and the proximate solvers required by the program as following:

$$prox_{(f_1,\gamma)}(z) = \underset{x}{\operatorname{argmin}} \frac{1}{2}|x - z|_2^2 + \gamma||z||_{TV} \tag{23}$$

$$prox_{(f_2,\gamma)}(z) = \underset{x}{\operatorname{argmin}} \frac{1}{2}|x - z|_2^2 + \gamma i_S(x) \tag{24}$$

where $i_S(x)$ has the same piece wise definition as (22), according to the user manual, for the form $s = \min_z ||x - z||_2^2 \; s.t \; ||w \cdot z||_1 \leq \epsilon$, the integrated *proj_B2* command for projection on B2-ball operator can be used. We damaged 50% of pixels of the Lena grayscale image, and use the TV-based inpainting algorithm to inpaint the image, and the result is shown below in Figure 10



(a) The original image without overlaid noise

(b) 50% random noise damaged image

(c) Inpainting result of the 50% noise-damaged image

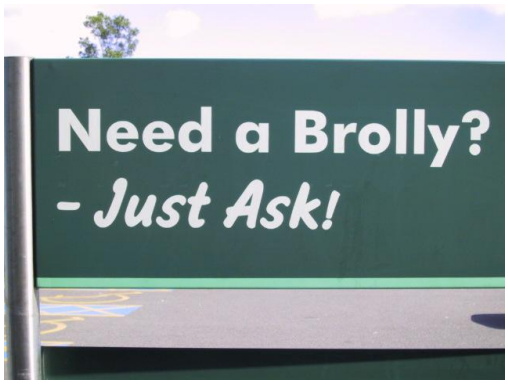
Figure 10: Result of the TV-based inpainting on gray-scale image

2.2 Implementing and modifying Bertalmio's *Image Inpainting*

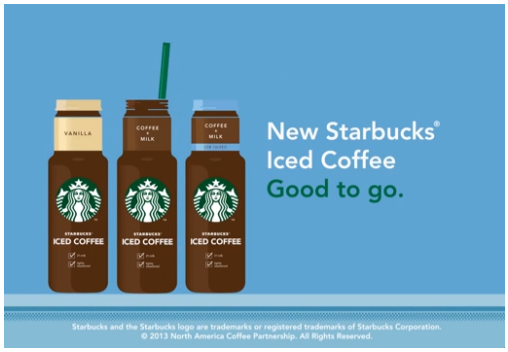
We chose to incorporate Bertalmio's original image inpainting algorithm [1] to our text removal pipeline because of the low complexity of the algorithm. More importantly, this algorithm deliver's assumption of distinguishable contour lines agrees with the fact that text in images or video frames of natural scenes or artificially imposed to an image or video frames usually are of bold color on a smooth background (Figure 11). As the original inpainting algorithm was not proposed for the purpose of text removal, certain modifications have to be made to accommodate our need.

2.2.1 Number of total iteration

The original paper suggests 17 times inpainting followed by 2 times of diffusion in each loop to achieve a good result, however, it doesn't specify a certain number of total iterations or a stop point criterion. Figure 7c shows the result of inpainting an aged photo with cracks, as was used in the paper, at 3000 times of total inpainting iterations, which is quite visually satisfying. However, we noticed that after a certain number of iteration, no changes seemed to be made to the image. Therefore, it is necessary for us to find a good timing to stop the iteration.



(a) Text in natural scene image



(b) Artificially imposed text on a video frame

Figure 11: Samples of text seen in images or videos

To testify our hypothesis, we inpainted the text-imposed image, and printed out the image at each iteration and plotted out the Mean Square Error (MSE) between the result at each iteration and original images in Figure 12 and 13.

(a) Original (b) T=5 (c) T=10 (d) T=15

(e) T=20 (f) T=25 (g) T=30 (h) T=35

Figure 12: Samples of text seen in images or videos

The resulting image and the MSE curve did confirm us that the inpainting algorithm converges at a certain point. To conclude a proper stop point, we inpainted randomly selected 109 images from the ICDAR datasets with defined text inpainting region, and plotted out the MSE value as well as the difference in MSE between the current iteration and previous iteration, and the results are shown in Figure 15

From Figure 15c, we set MSE difference of 0.1×10^{-4} as the stop criterion of the inpainting procedure. Also, since most of the images converge before 1500th iteration, we set our maximum iteration to 1500.

2.2.2 Text mask Size

Another important factor we found that affects the final inpainting results greatly is the size of the defined inpainting region (text mask in our case). For a small mask, it's possible that the mask is smaller then the actual region, resulting in the final inpainted image containing obvious rings of the original texts, making it vulnerable to OCR attack (Figure **??**). On the other hand, if the dilation is too big, though the texts are fully covered, it's hard for the inpainting algorithm to propagate information into the inpainting region, thus making the results contain considerable amount of visual artifacts in the inpainting region, as well as making it harder for the algorithm to converge.

Therefore, we ran the inpainting algorithm on different text mask size to find the optimal dilation size. We dilated the masks with disk of radius of 1 pixel, 5 pixels, and 1 percent, 5 percent of the biggest bounding

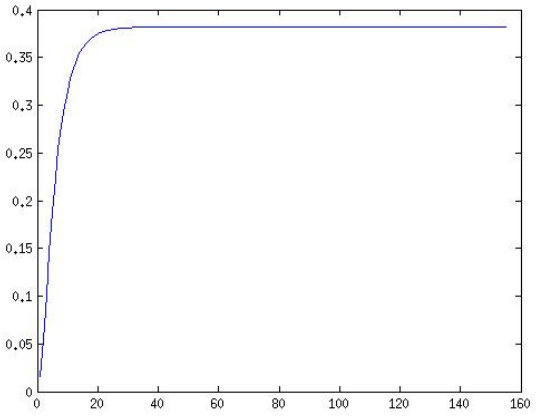
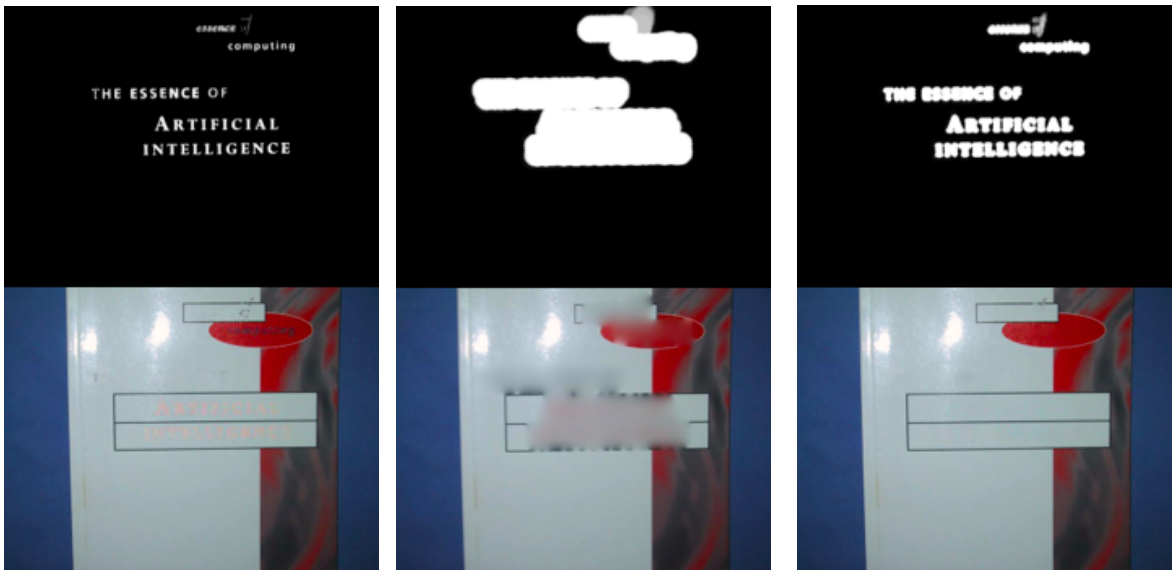


Figure 13: MSE of the red channel of the inapinting text-imposed image at each iteration



(a) Image inpainted with a too-small inpainting region
(1-pixel dilation)

(b) Image inpainted with a too-big inpainting region
(5-percent max width of bounding box dilation)

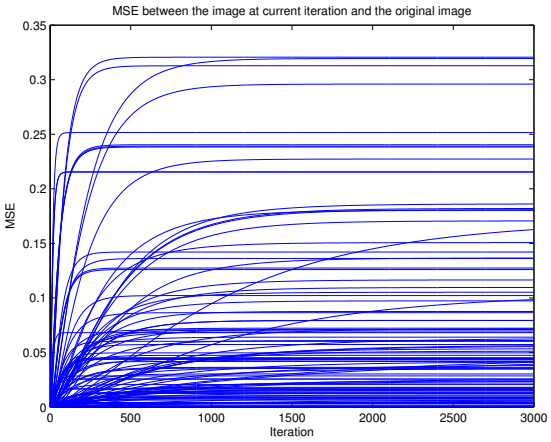
(c) Image inpainted with a properly sized inpainting region
(1-percent max width of bounding box dilation)

Figure 14: Image inpainted with different size of defined inpainting region

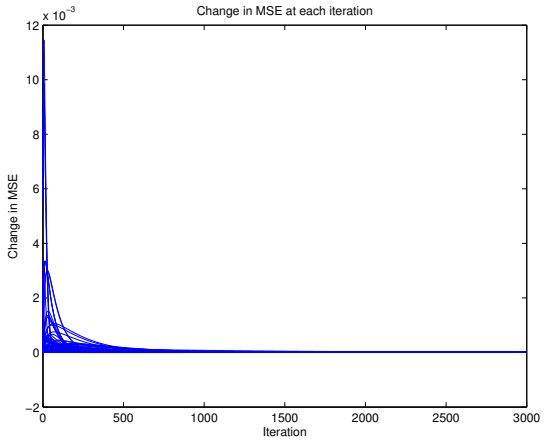
box within the image, and inpainted 50 randomly selected ICDAR testing images with those differently sized masks. We visually observed the results and found the 1 percent width of bounding box dilation delivered the best result. Therefore, we chose this size as our mask size for our future experiments and evaluations.

2.2.3 Color Images

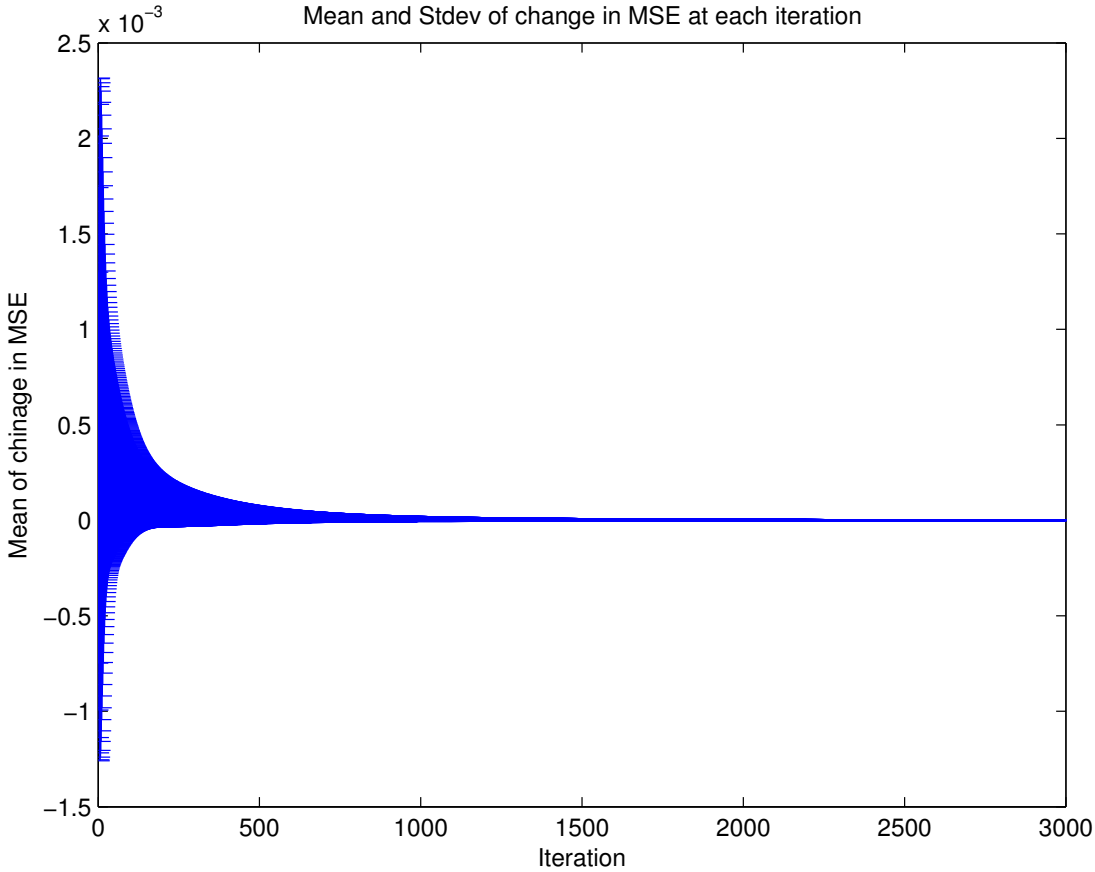
Since most of our input images are color image while our algorithm works with grayscale images, we simply decompose the color images into red, green and blue channels and perform the inapinting on each channel.



(a) MSE of the images at each iteration



(b) Difference in MSE of the image at each iteration



(c) Mean difference in MSE of the images, plotted with the standard deviation

Figure 15: MSE values at each iteration as well as the difference in MSE between the current iteration and previous iteration of the randomly selected 109 ICDAR testing image

3 Experimental Results and Evaluation

3.1 Pipeline

As shown in Figure 16, our framework of text inpainting and evaluation is consisted of the following steps.

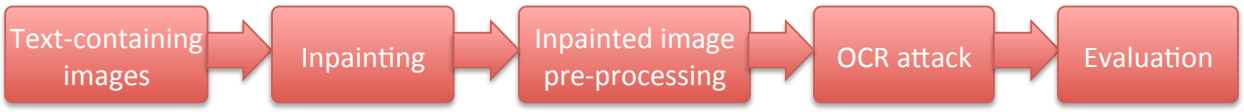


Figure 16: Pipeline of our proposed system

3.1.1 Testing datasets

The testing datasets we used for our experiments were the ICDAR robust reading and text locating testing datasets[2], which contain 251 images of natural scenes that contains in total 517 lines of text (five images without text were also included) We resize the oversized image (longest size $>$ 480 pixels) to 480p resolution which resembles the resolution of a typical online video. The text regions are provided by [15] and dilated with a disk that has radius of 1 % of width the of the biggest text bounding box within in the image. (Figure 17)

(a) Image for inpainting

(b) Dilated text region mask

Figure 17: Sample image from the ICDAR datasets and dilated text mask

3.1.2 Inpainting algorithm

Our pipeline starts with a text-containing image and the defined corresponding text regions. We inpaint the text region with the modified Bertalmio's algorithm demonstrated above, where the algorithm stops when the difference in MSE is smaller than 0.1×10^{-4} after 1500 iterations.

3.1.3 Pre-processing of the inpainted image

The following pre-processing procedures were applied to the inpainted images to ensure OCR success rate. A sample pre-processing result is shown in Figure 18

1. Extracting the text regions of the image to smaller images

[2] http://algoval.essex.ac.uk/icdar/Datasets.html

2. Mean shift filtering to reduce the noise and segment the text out [4]
3. Convert the segmented color images to gray-scale
4. Image binarization using Otsu's method [12]

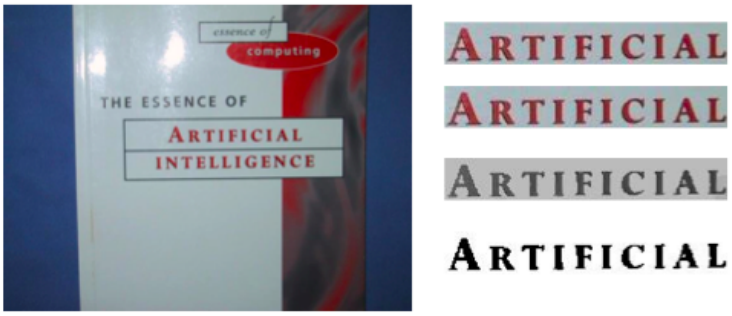


Figure 18: Sample pre-processing result, left column: original image, right column from top to bottom: extracted text region, mean shifted image, gray-scale image, Otsu's method binarization

3.1.4 OCR attacks

Finally, we ran OCR attacks on the pre-processed images. We chose the Tesseract OCR engine[3] for its better performance among the open source OCR engines. For the parameters of the OCR, we set the language to English and page segmentation to treat the image as a single text line.

3.1.5 Evaluation

We used Levenshtein distance (a.k.a string edit distance)[10] as the metric for OCR attack assessment. Since by definition, the Levenshtein distance is the minimum number of single-character edits between two strings, greater Levenshtein distance between the inpainted image and the original image represents a lower OCR success rate, thus a better performance of the inpainting algorithm

3.2 Results

In Figure 19, we showed five samples of inpainting results, and the corresponding OCR results, including the recognized text and the Levenshtein with and without inapinting are shown in Table 1.

Table 1: Sample OCR Attack Results

Inpainted Image	Text Ground Truth	OCR Before Inapinting	Distance Before	OCR After	Distance After
Example 1	ARTIFICIAL	ARTIFICIAL	0	-	10
Example 2	No mobile	No mobile	0	-	9
Example 3	Panasonic	Panasonic	0	-	9
Example 4	Need a Bolly?	Need a Bolly?	0	-	14
Example 5	WHSmith	WHSmith	0	n HSmith	4

As one can see in Figure 19 and Table 1, before the inpainting, the OCR attack rate is fairly high, but after the inpainting, the characters are barely recognizable, increasing the Levenshtein distance to the character length of text line.

Plotting out the histograms of the Levenshtein distance between the ground truth text and OCR results with and without inpainting for all 517 text lines in the 250 testing images in Figure 22 and 23, we can

[3] https://code.google.com/p/tesseract-ocr/

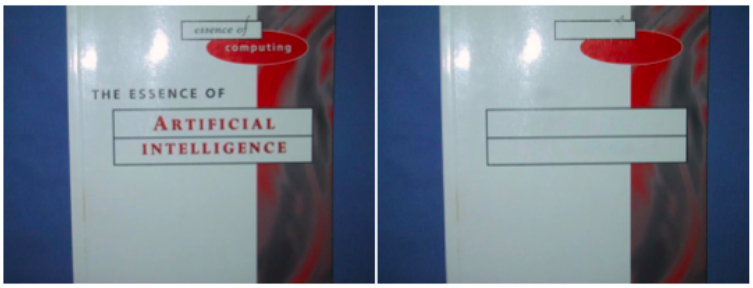
clearly see that inpainting successfully reduced the OCR attack success rate from 38.29% to 3.8%, if we count exact match between OCR result and ground truth as a successful attack.

To quantitatively evaluate the visual impact of the inpainting algorithm, we manually overlaid text on the image and perform inpainting over the text region. We then compared the result image with the original image using Peak Signal-to-Noise Ratio(PSNR), and Perceptual Image Difference [19] utility under the setting of 2.2 display Gamma correction, 100 cd/m^2, and 45 degree visual angle. The result are shown in Figure 21. The PSNR of the inpainted image compared to the original image is 27.56dB, and the PDIFF utility indicated that 53998 pixels are different under the specified observing environment.

We also applied the inpainting algorithm a text-overlaid video to simulate a video CAPTCHA scenario, and the result is shown in Figure 20. As one can see, visually it is fairly satisfying.

4 Conclusion

We proposed and developed a pipeline that utilize image inpainting technique to remove the text within the images, which has the potential application of removing the text in a video CAPTCHA to avoid OCR attacks. Therefore, we test the efficacy of the our text inpainting in preventing OCR attack, and result is encouraging: it can successfully reduce the chance of the video CAPTCHA being comprised, as the OCR success rate was greatly reduced after we applied inpainting to the target image.



(a) Inpainting example 1

(b) Inpainting example 2

(c) Inpainting example 3

(d) Inpainting example 4

(e) Inpainting example 5

Figure 19: Inpainting result samples

Figure 20: Video Inpainting Results. Inpainted with 51 total iterations and inpainting number of iterations was set to 17 and number of diffusion iterations was set to 2.

(a) Text overlaid image

(b) Original image

(c) Inpainting result

Figure 21: Inpainting the text overlaid image

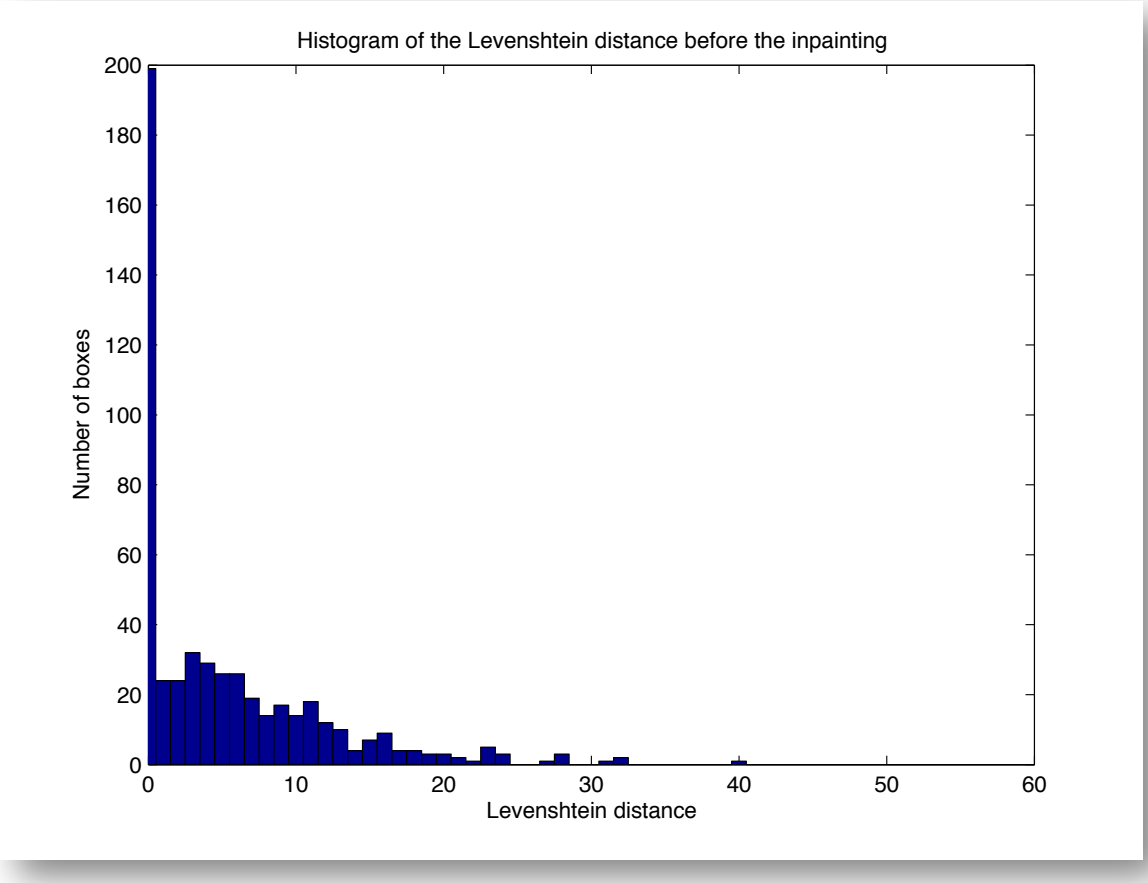


Figure 22: Histogram of the Levenshtein distance between the ground truth text and the OCR text before the inpainting

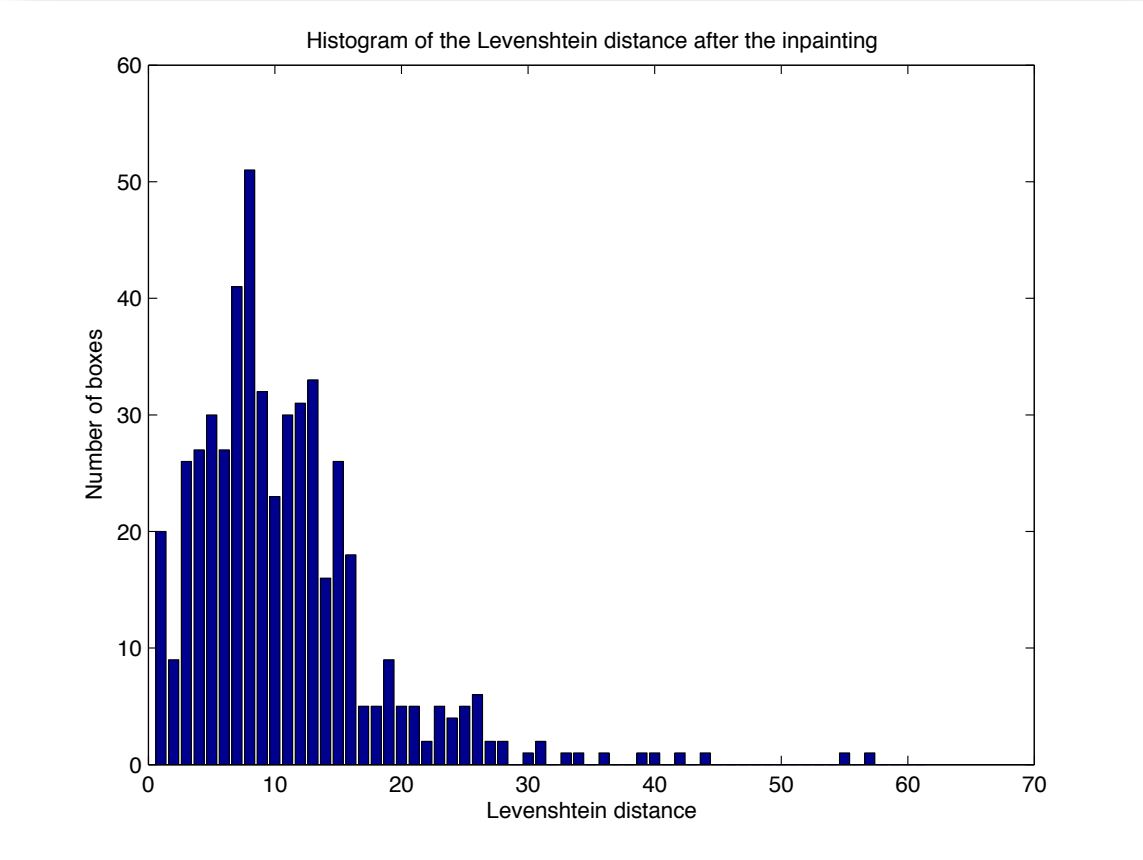


Figure 23: Histogram of the Levenshtein distance between the ground truth text and the OCR text after the inpainting

References

[1] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[2] Marcelo Bertalmio, Luminita Vese, Guillermo Sapiro, and Stanley Osher. Simultaneous structure and texture image inpainting. *Image Processing, IEEE Transactions on*, 12(8):882–889, 2003.

[3] Antonin Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1-2):89–97, 2004.

[4] Yizong Cheng. Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):790–799, 1995.

[5] James A Clarkson and C Raymond Adams. On definitions of bounded variation for functions of two variables. *Transactions of the American Mathematical Society*, 35(4):824–854, 1933.

[6] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 13(9):1200–1212, 2004.

[7] Jim Douglas and HH Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 82(2):421–439, 1956.

[8] G. Émile-Mâle. *La Restauration Des Peintures de Chevalet (Restoration of easel paintings)*. Office du Livre, 1976.

[9] Kurt Alfred Kluever and Richard Zanibbi. Balancing usability and security in a video captcha. In *Proceedings of the 5th Symposium on Usable Privacy and Security (SOUPS '09)*, 2009.

[10] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

[11] Antonio Marquina and Stanley Osher. Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal. *SIAM Journal on Scientific Computing*, 22(2):387–405, 2000.

[12] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[13] Shuman David Vandergheynst Pierre Perraudin Nathanal and Puy Gilles. *UNLocBox: Short User Guide*. LTS2 EPFL, 2012.

[14] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[15] Dave Snyder. *Text detection in natural scenes through weighted majority voting of DCT high pass filters, line removal, and color consistency filtering*. PhD thesis, Rochester Institute of Technology, 2011.

[16] Luminita A Vese and Stanley J Osher. Modeling textures with total variation minimization and oscillating patterns in image processing. *Journal of Scientific Computing*, 19(1-3):553–572, 2003.

[17] Wikipedia. Total variation - wikipedia, the free encyclopedia, 2013. [Online; accessed 10-May-2013].

[18] Luke Wroblewski. A sliding alternative to captcha. `http://www.lukew.com/ff/entry.asp?1138`.

[19] Hector Yee, Sumanita Pattanaik, and Donald P Greenberg. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Transactions on Graphics (TOG)*, 20(1):39–65, 2001.