Mathematical Techniques of Image Processing

Chris Louder

October 29, 2012

Declaration

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Chris Louder

Abstract

This paper investigates various different types of image processing methods and models for image processing: a total variation based approach, a shock filters approach and morphological component analysis. Discretisations of the first two will be seen, as well as various different 1D and 2D examples of their results, and 2D+T (video) examples for Morphological component analysis. Their behaviour and results with different images is compared to contrast the different behaviour in the presence of noise or small scale texture.

Contents

Chapter 1

Introduction

Image processing is the mathematical analysis of an image. The image can be still or video and the output need not be an image. An attempt to reproduce an image without the glare of the Sun's reflection is a form of image processing. Producing a line graph of the varying light intensity in separate frames from a video clip is an equally valid image processing technique. Image processing has become far more advanced in recent years due to a wide variety of scientific advances. Cameras have improved dramatically in the last century when a grainy black and white camera was expensive: nowdays most people have a high-definition camera in their mobile phones. With these technological advances comes new possibilities and requirements. With bigger telescopes allowing deep space observation, less information is gathered than relatively close observations, and superior techniques to reconstruct missing information and better detect and remove noise is required. More advanced optical electronic computer vision systems are being used in a wide variety of fields: some examples are listed below.

- Navigation;
  - the mapping of underwater rocks or currents.
- Surveillance;
  - Counting the number of people entering the Olympic Stadium for example, or modelling the likely outward flow of people in the event of a

fire to detect hazardous areas.

- Scene reconstruction;
  - a computer generated 3-D model of a crime scene.
- Industrial object inspection;
  - Detecting air bubbles in manufactured parts, automatic processing of hand-written documents (i.e. a filled in form having the information automatically added into a database).
- Medical;
  - Functional magnetic resonance imaging for brain scans or real time displays of internal bodies for precise surgery.

While there are many other applications in computer vision, there are also uses in computer graphics. Computer vision is the attempt to electronically model the real world, while computer graphics is the attempt to artificially generate an image (that may or may not resemble the real world).

We will investigate different models for image processing, observing their different results and behaviour depending on the content of the image. We will also see examples of different applications these methods have, such as image inpainting.

Chapter 2

The Vese Osher Total Variation Model (VO)

We first look at an approach developed by Luminita Vese and Stanley Osher, which uses functional minimisation and partial differential equations to decompose a noisy image into two functions [21], one an approximation of the true image, the other an approximation of the visual noise. The model attempts to approximate the texture preserving model of Meyer [14] and solve it using the edge preserving total variation minimisation method from the Rudin Osher Fatemi (ROF) model [18]. The noise is approximated more simply (and less accurately) than in other techniques, using only two functions g_1, g_2, and for both the approximate image and noise, with each iterative step, only the values from the current iteration are stored, previous values are overwritten. "This is much simpler and much more efficient than in other techniques for textures," [21] as the model is designed to be faster. Examples of models given in the original paper include [18], [16] and [6] among others. The simplicity makes the algorithm faster, requiring less memory and processing power than other image processing techniques. As a result there are a few significant differences. The algorithm also does not preserve texture (defined in [21] as small scale repeated details/patterns) as it detects noise and texture similarly. Noise, although random, is also small scale detail so both are detected, modelled and removed in the same way. We will refer to noise when

looking at this model, though the removal of texture is still useful, The paper attempts to solve approximately the minimization problem

$$\inf_{u,g1,g2}\left\{G_p(u,g_1,g_2)=\int|\nabla u|+\lambda\int|f-u-\partial_x\ g_1-\partial_y\ g_2|^2\ dx\ dy \right. \\ \left. +\mu\left[\int(\sqrt{g_1^2+g_2^2})^p\ dx\ dy\right]^{1/p}\right\}, \quad (2.1)$$

where f is the original noisy image, u is the approximation of f, g_1, g_2 represent the noise, $\lambda, \mu > 0$ are tuning parameters, p is from the approximation of the L^∞ norm by L^p. Minimising the problem and taking $p = 1$ generates

$$\begin{aligned} u =& f-\partial_x g_1-\partial_y g_2+\frac{1}{2\lambda}div\left(\frac{\nabla u}{|\nabla u|}\right), \\ \mu\frac{g_1}{\sqrt{g_1^2+g_2^2}} =& 2\lambda\left[\frac{\partial}{\partial x}(u-f)+\partial_{xx}^2 g_1+\partial_{xy}^2 g_2\right], \\ \mu\frac{g_2}{\sqrt{g_1^2+g_2^2}} =& 2\lambda\left[\frac{\partial}{\partial y}(u-f)+\partial_{xy}^2 g_1+\partial_{yy}^2 g_2\right]. \end{aligned} \quad (2.2)$$

Values of $p > 1$ can be used, but introduce new terms to the algorithm, complicating the discretization and requiring more calculations. Vese and Osher commented that results with $p > 1$ values have been calculated, but show little difference compared with results from $p = 1$ calculations. As such only the p=1 case will be considered.

2.1 Discretization

The equations (2.2) are discretized using a method of finite differences and a fixed point iteration algorithm, based on other schemes from non-linear variational image processing algorithms. Using the traditional notation $f_{i,j} \approx f(ih, jh), u_{i,j} \approx u(ih, jh), g_{1,i,j} \approx g_1(ih, jh), g_{2,i,j} \approx g_2(ih, jh)$, it is noted that the space step of 1 pixel (i.e. h=1) removes the h term from each notation. The equations (2.2)

initially discretise to:

$$
\begin{aligned}
u_{i,j} = f_{i,j} &- \frac{g_{1,i+1,j} - g_{1,i-1,j}}{2h} - \frac{g_{2,i,j+1} - g_{2,i,j-1}}{2h} \\
&+ \frac{1}{2\lambda h^2} \left[\frac{u_{i+1,j} - u_{i,j}}{\sqrt{\left(\frac{u_{i+1,j}-u_{i,j}}{h}\right)^2 + \left(\frac{u_{i,j+1}-u_{i,j-1}}{2h}\right)^2}} \right. \\
&\left. - \frac{u_{i,j} - u_{i-1,j}}{\sqrt{\left(\frac{u_{i,j}-u_{i-1,j}}{h}\right)^2 + \left(\frac{u_{i-1,j+1}-u_{i-1,j-1}}{2h}\right)^2}} \right] \\
&+ \frac{1}{2\lambda h^2} \left[\frac{u_{i,j+1} - u_{i,j}}{\sqrt{\left(\frac{u_{i+1,j}-u_{i-1,j}}{2h}\right)^2 + \left(\frac{u_{i,j+1}-u_{i,j}}{h}\right)^2}} \right. \\
&\left. - \frac{u_{i,j} - u_{i,j-1}}{\sqrt{\left(\frac{u_{i+1,j-1}-u_{i-1,j-1}}{2h}\right)^2 + \left(\frac{u_{i,j}-u_{i,j-1}}{h}\right)^2}} \right],
\end{aligned}
\tag{2.3}
$$

$$
\begin{aligned}
\frac{\mu\ g_{1,i,j}}{\sqrt{g_{1,i,j}^2 + g_{2,i,j}^2}} = 2\lambda \Bigg[&\frac{u_{i+1,j} - u_{i-1,j}}{2h} - \frac{f_{i+1,j} - f_{i-1,j}}{2h} + \frac{g_{1,i+1,j} + g_{1,i-1,j} - 2g_{1,i,j}}{h^2} \\
&+ \frac{1}{2h^2} \left(2g_{2,i,j} + g_{2,i-1,j-1} + g_{2,i+1,j+1} - g_{2,i,j-1} - g_{2,i-1,j} - g_{2,i+1,j} - g_{2,i,j+1}\right) \Bigg],
\end{aligned}
\tag{2.4}
$$

$$
\begin{aligned}
\frac{\mu\ g_{2,i,j}}{\sqrt{g_{1,i,j}^2 + g_{2,i,j}^2}} = 2\lambda \Bigg[&\frac{u_{i,j+1} - u_{i,j-1}}{2h} - \frac{f_{i,j+1} - f_{i,j-1}}{2h} + \frac{g_{2,i,j+1} + g_{2,i,j-1} - 2g_{2,i,j}}{h^2} \\
&+ \frac{1}{2h^2} \left(2g_{1,i,j} + g_{1,i-1,j-1} + g_{1,i+1,j+1} - g_{1,i,j-1} - g_{1,i-1,j} - g_{1,i+1,j} - g_{1,i,j+1}\right) \Bigg].
\end{aligned}
\tag{2.5}
$$

Notice that the partial differentials have been discretised as

$$
\begin{aligned}
\partial_x g_{1,i,j} &= \frac{1}{2h} \left(g_{1,i+1,j} - g_{1,i-1,j}\right) \\
\partial_{xx} g_{1,i,j} &= \frac{1}{h^2} \left(g_{1,i+1,j} + g_{1,i-1,j} - 2g_{1,i,j}h^2\right) \\
\partial_{xy} g_{1,i,j} &= \frac{1}{2h^2} (2g_{2,i,j} + g_{2,i-1,j-1} + g_{2,i+1,j+1} - g_{2,i,j-1} - g_{2,i-1,j} - g_{2,i+1,j} - g_{2,i,j+1})
\end{aligned}
$$

Similarly for g_2 and partial differentials with respect to y. For the discretization of u, to prevent division by zero, a small term ϵ^2 is introduced into the square roots.

The discrete form is then linearised to:

$$
\begin{aligned}
u_{i,j}^{n+1} =& f_{i,j} - \frac{g_{1,i+1,j}^n - g_{1,i-1,j}^n}{2h} - \frac{g_{2,i,j+1}^n - g_{2,i,j-1}^n}{2h} \\
&+ \frac{1}{2\lambda h^2} \left[\frac{u_{i+1,j}^n - u_{i,j}^{n+1}}{\sqrt{\left(\frac{u_{i+1,j}^n - u_{i,j}^{n+1}}{h}\right)^2 + \left(\frac{u_{i,j+1}^n - u_{i,j-1}^n}{2h}\right)^2 + \epsilon^2}} \right. \\
&\left. - \frac{u_{i,j}^{n+1} - u_{i-1,j}^n}{\sqrt{\left(\frac{u_{i,j}^{n+1} - u_{i-1,j}^n}{h}\right)^2 + \left(\frac{u_{i-1,j+1}^n - u_{i-1,j-1}^n}{2h}\right)^2 + \epsilon^2}} \right] \\
&+ \frac{1}{2\lambda h^2} \left[\frac{u_{i,j+1}^n - u_{i,j}^{n+1}}{\sqrt{\left(\frac{u_{i+1,j}^n - u_{i-1,j}^n}{2h}\right)^2 + \left(\frac{u_{i,j+1}^n - u_{i,j}^{n+1}}{h}\right)^2 + \epsilon^2}} \right. \\
&\left. - \frac{u_{i,j}^{n+1} - u_{i,j-1}^n}{\sqrt{\left(\frac{u_{i+1,j-1}^n - u_{i-1,j-1}^n}{2h}\right)^2 + \left(\frac{u_{i,j}^{n+1} - u_{i,j-1}^n}{h}\right)^2 + \epsilon^2}} \right],
\end{aligned}
\tag{2.6}
$$

$$
\begin{aligned}
\frac{\mu\ g_{1,i,j}^{n+1}}{\sqrt{(g_{1,i,j}^n)^2 + (g_{2,i,j}^n)^2}} =& 2\lambda \left[\frac{u_{i+1,j}^n - u_{i-1,j}^n}{2h} - \frac{f_{i+1,j} - f_{i-1,j}}{2h} + \frac{g_{1,i+1,j}^n + g_{1,i-1,j}^n - 2g_{1,i,j}^{n+1}}{h^2} \right. \\
&\left. + \frac{1}{2h^2} \left(2g_{2,i,j}^n + g_{2,i-1,j-1}^n + g_{2,i+1,j+1}^n - g_{2,i,j-1}^n - g_{2,i-1,j}^n - g_{2,i+1,j}^n - g_{2,i,j+1}^n \right) \right],
\end{aligned}
\tag{2.7}
$$

$$
\begin{aligned}
\frac{\mu\ g_{2,i,j}^{n+1}}{\sqrt{(g_{1,i,j}^n)^2 + (g_{2,i,j}^n)^2}} =& 2\lambda \left[\frac{u_{i,j+1}^n - u_{i,j-1}^n}{2h} - \frac{f_{i,j+1} - f_{i,j-1}}{2h} + \frac{g_{2,i,j+1}^n + g_{2,i,j-1}^n - 2g_{2,i,j}^{n+1}}{h^2} \right. \\
&\left. + \frac{1}{2h^2} \left(2g_{1,i,j}^n + g_{1,i-1,j-1}^n + g_{1,i+1,j+1}^n - g_{1,i,j-1}^n - g_{1,i-1,j}^n - g_{1,i+1,j}^n - g_{1,i,j+1}^n \right) \right].
\end{aligned}
\tag{2.8}
$$

By introducing the shorthand notation's below, the above equations can be simplified, and then rearranged to yield the $n+1$ terms:

$$\begin{aligned}
c_1 =& \frac{1}{\sqrt{\left(\frac{u^n_{i+1,j}-u^n_{i,j}}{h}\right)^2+\left(\frac{u^n_{i,j+1}-u^n_{i,j-1}}{2h}\right)^2}},\\
c_2 =& \frac{1}{\sqrt{\left(\frac{u^n_{i,j}-u^n_{i-1,j}}{h}\right)^2+\left(\frac{u^n_{i-1,j+1}-u^n_{i-1,j-1}}{2h}\right)^2}},\\
c_3 =& \frac{1}{\sqrt{\left(\frac{u^n_{i+1,j}-u^n_{i-1,j}}{2h}\right)^2+\left(\frac{u^n_{i,j+1}-u^n_{i,j}}{h}\right)^2}},\\
c_4 =& \frac{1}{\sqrt{\left(\frac{u^n_{i+1,j-1}-u^n_{i-1,j-1}}{2h}\right)^2+\left(\frac{u^n_{i,j}-u^n_{i,j-1}}{h}\right)^2}}.
\end{aligned} \tag{2.9}$$

By solving equations(2.6-2.8)

$$\begin{aligned}
u^{n+1}_{i,j} = \left(\frac{1}{1+\frac{1}{2\lambda h^2}(c_1+c_2+c_3+c_4)}\right)\Bigg[f_{i,j} - \frac{g^n_{1,i+1,j}-g^n_{1,i-1,j}}{2h}\frac{g^n_{2,i,j+1}-g^n_{g_2,i,j-1}}{2h}\\
+\frac{1}{2\lambda h^2}(c_1u^n_{i+1,j}+c_2u^n_{i-1,j}+c_3u^n_{i,j+1}+c_4u^n_{i,j-1})\Bigg],
\end{aligned} \tag{2.10}$$

$$\begin{aligned}
g^{n+1}_{1,i,j} = \left(\frac{2\lambda}{\frac{\mu}{\sqrt{(g^n_{1,i,j})^2+(g^n_{2,i,j})^2}}+\frac{4\lambda}{h^2}}\right)\Bigg[\frac{u^n_{i+1,j}-u^n_{i-1,j}}{2h} - \frac{f_{i+1,j}-f_{i-1,j}}{2h} + \frac{g^n_{1,i+1,j}+g^n_{1,i-1,j}}{h^2}\\
+\frac{1}{2h^2}(2g^n_{2,i,j}-g^n_{2,i-1,j-1}-g^n_{2,i+1,j+1}-g^n_{2,i,j-1}-g^n_{2,i-1,j}-g^n_{2,i+1,j}-g^n_{2,i,j+1})\Bigg]
\end{aligned} \tag{2.11}$$

$$\begin{aligned}
g^{n+1}_{2,i,j} = \left(\frac{2\lambda}{\frac{\mu}{\sqrt{(g^n_{1,i,j})^2+(g^n_{2,i,j})^2}}+\frac{4\lambda}{h^2}}\right)\Bigg[\frac{u^n_{i,j+1}-u^n_{i,j-1}}{2h} - \frac{f_{i,j+1}-f_{i,j-1}}{2h} + \frac{g^n_{2,i,j+1}+g^n_{2,i,j-1}}{h^2}\\
+\frac{1}{2h^2}(2g^n_{1,i,j}-g^n_{1,i-1,j-1}-g^n_{1,i+1,j+1}-g^n_{1,i,j-1}-g^n_{1,i-1,j}-g^n_{1,i+1,j}-g^n_{1,i,j+1})\Bigg]
\end{aligned} \tag{2.12}$$

In the numerical model a space step of one (i.e. one pixel length) is taken. It is worth noting that in in image processing the space step is almost always set as 1

pixel (i.e. h=1) and therefore does not need to be included in the discretisations, though it has been included here for clarity. Values of the tuning parameters are varied from between 0.01 and 10. Dirichlet boundary conditions are applied. For calculating u, values at the boundary of the domain are extended by reflection. Images are processed in greyscale double precision form. The values of u, g_1 and g_2 were initialised as $u^0 = f$, $g_1^0 = -\frac{f_x}{2\lambda|\nabla f|}$, $g_2^0 = -\frac{f_y}{2\lambda|\nabla f|}$. Approximately 100 iterations are performed. Reproducing the code of the above technique has been attempted, though this has failed to reproduce similar results to those obtained originally. We will therefore show some results from the original paper, as well as some from extensions on this work in subsequent papers.

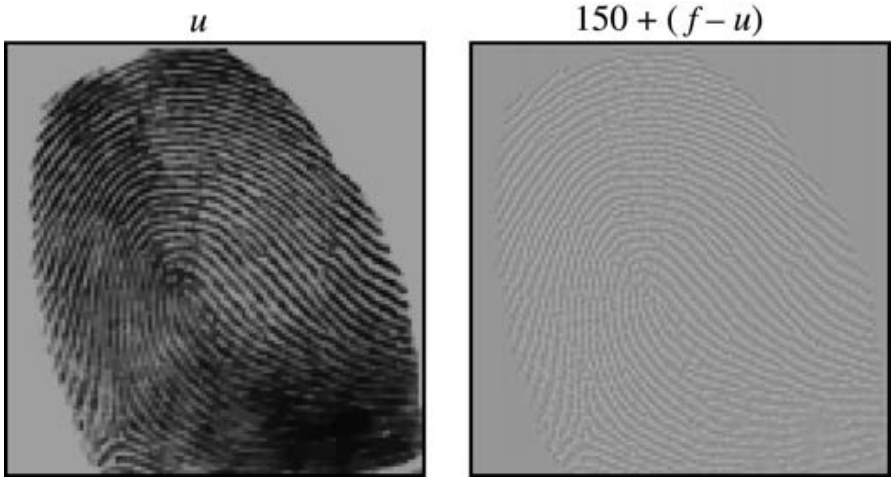
2.2 Numerical Results

Fig 2.1 demonstrates texture removal. In Fig 2.1c we see the resultant image u, alongside the extracted noise v (adding 150 makes the noise easier to see on its own). Alongside are $u + v$ and $(f - u - v)$, these display what remains of the original image, and the information that has been lost in the process respectively. Contrasted with the previously developed ROF model which was designed for noise removal, a lot more texture has been extracted, leaving a relatively plain image u, with most texture preserved in v. The texture extracted by the ROF model is displayed by $f - u$ as ROF does not store the extracted noise/texture.

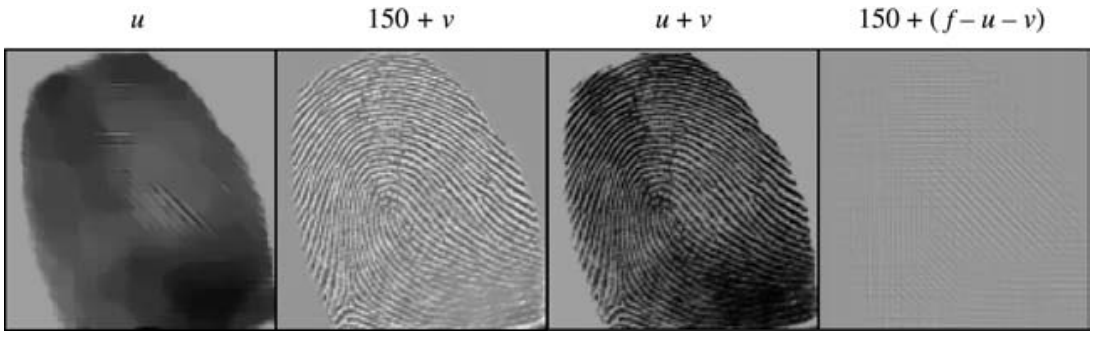
In Fig 2.2 we see the results for the same image with random zero mean noise added. Fig 2.2b shows that the ROF model has removed most of the noise successfully, though a lot of texture has been removed as well. In Fig 2.2c the consequences of high or low μ values is demonstrated. The low μ values have removed and stored most noise and texture together, failing to differentiate between the two. Higher values of μ has also removed most noise and more texture is preserved, though it is most noticeable that most noise and texture has been lost rather than stored. In fact so little noise has been stored that $u + v$ appears to be a better resultant image than u for noise removal texture preserved.

In Fig 2.3 we see the results for a real world textured image. The model has

(a) An initial fingerprint image.



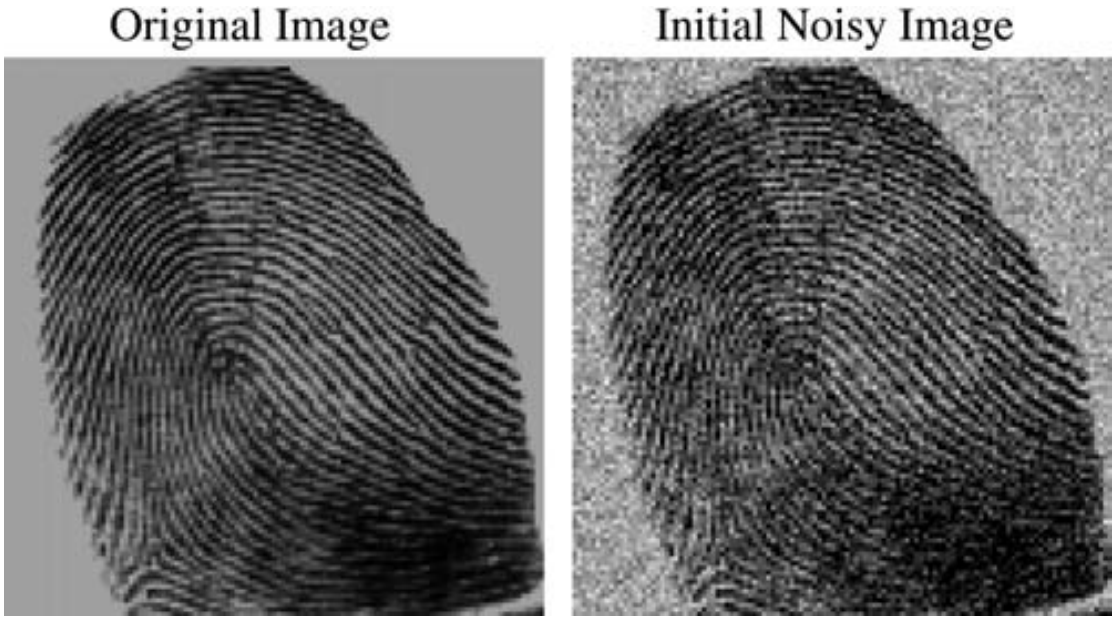
(b) Result using ROF model with $\lambda = 0.1$.



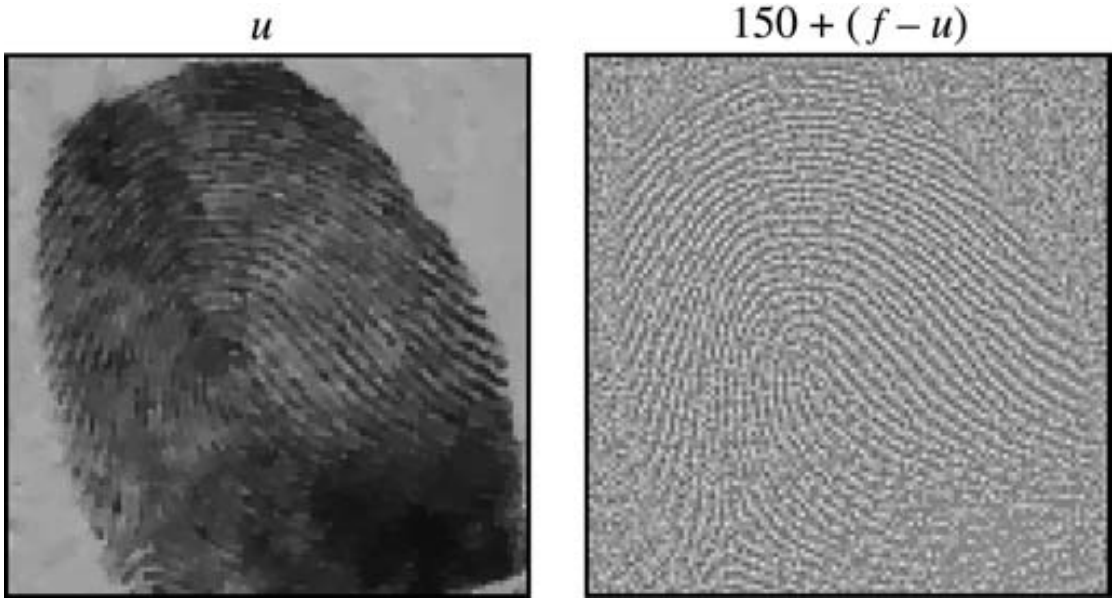
(c) Result using TVM model with $\lambda = 0.1$, $\mu = 0.1$.

Figure 2.1: Results for a clean, high detail image.

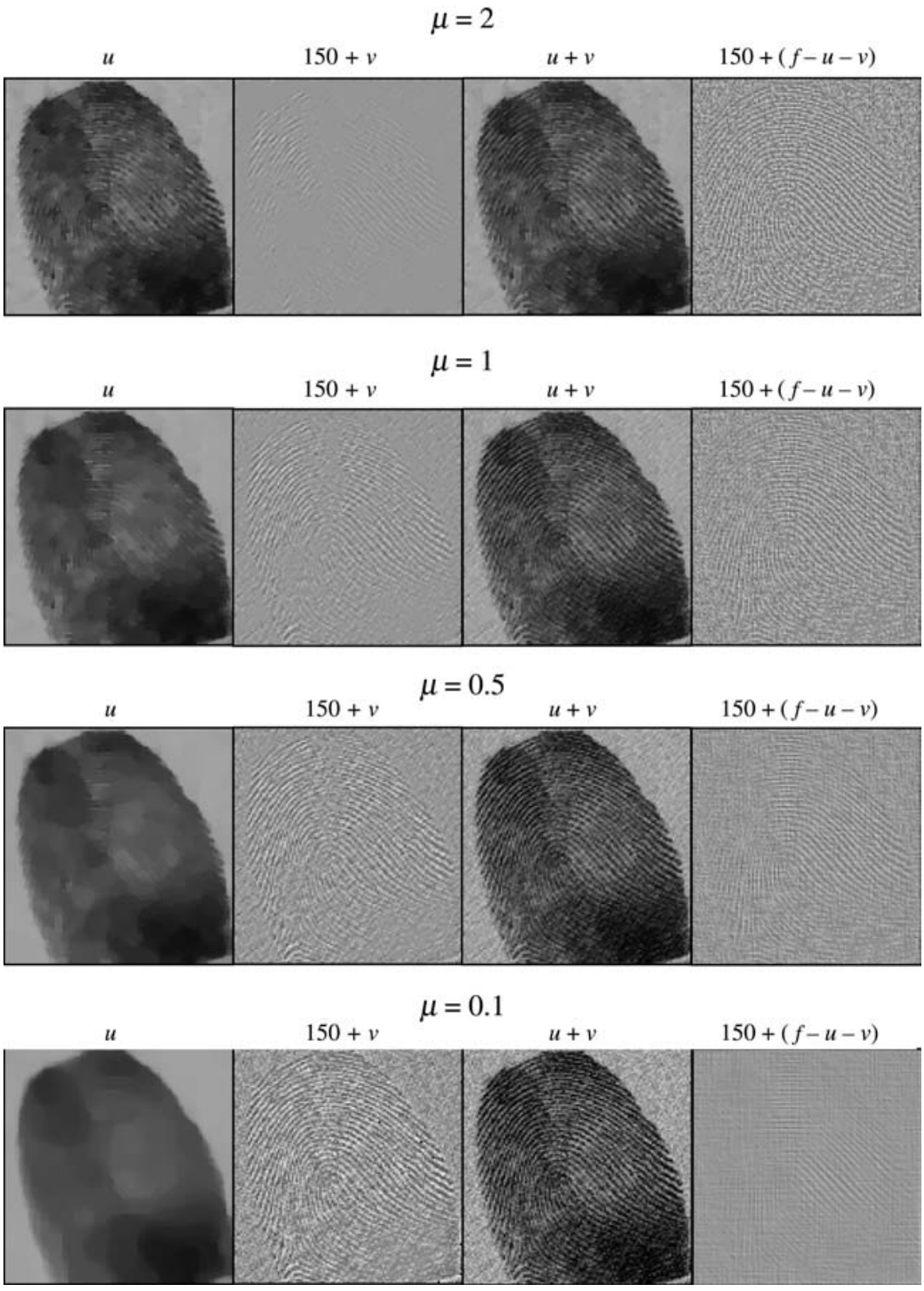
Reproduced from [21]



(a) Original fingerprint image (left) and a noisy version (right).



(b) Result using ROF model with $\lambda = 0.03$.

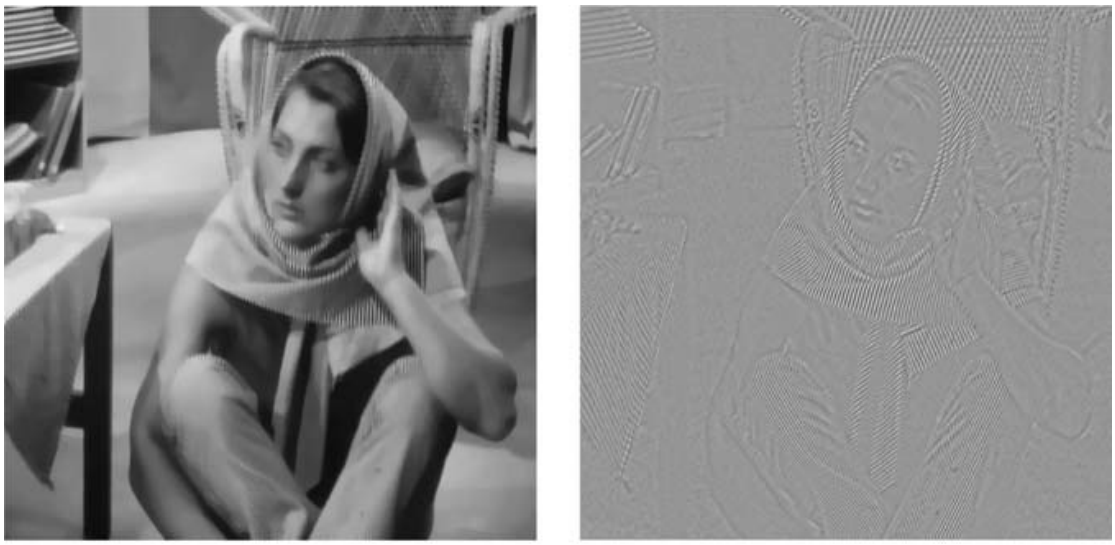


(c) Results using TVM model with $\lambda = 0.03$ and $\mu = 2; 1; 0.5; 0.1$.

Figure 2.2: Results for a noisy, high detail image.
Reproduced from [21]

f $u + v$

(a) Original Barbara image (left) and a noisy version (right).



u $150 + v$

(b) Result using TVM model with $\lambda = 0.2$, $\mu = 0.01$.

Figure 2.3: Results for a high texture image.
Reproduced from [21]

extracted texture effectively, most notably the stripes in the clothing and tablecloth. Little information has been lost, as there is no noticeable difference between f and $u+v$. Texture segmentation and extraction like this is not usually useful on it's own, but is useful in further image processing techniques like image inpainting (filling in gaps in images). The Total Variation Model was extended in [3] for this very purpose, separating the texture so the plain image and the extracted texture can be processed separately to attempt to rebuild the lost information using an image inpainting model from [2] and a texture syntheses model from [9]. An example of this is given in Fig 2.4.

Given the wide range of applications this Total Variation framework has, it is worth investigating where this implementation into image processing originally started. The original paper [15] was on a different form of processing than what we have seen here and as such, the results cannot be compared. Despite this the models are originating from similar mathematical problems, using total variation methods to try and solve them. It is therefore worth observing the relative success of the original model.

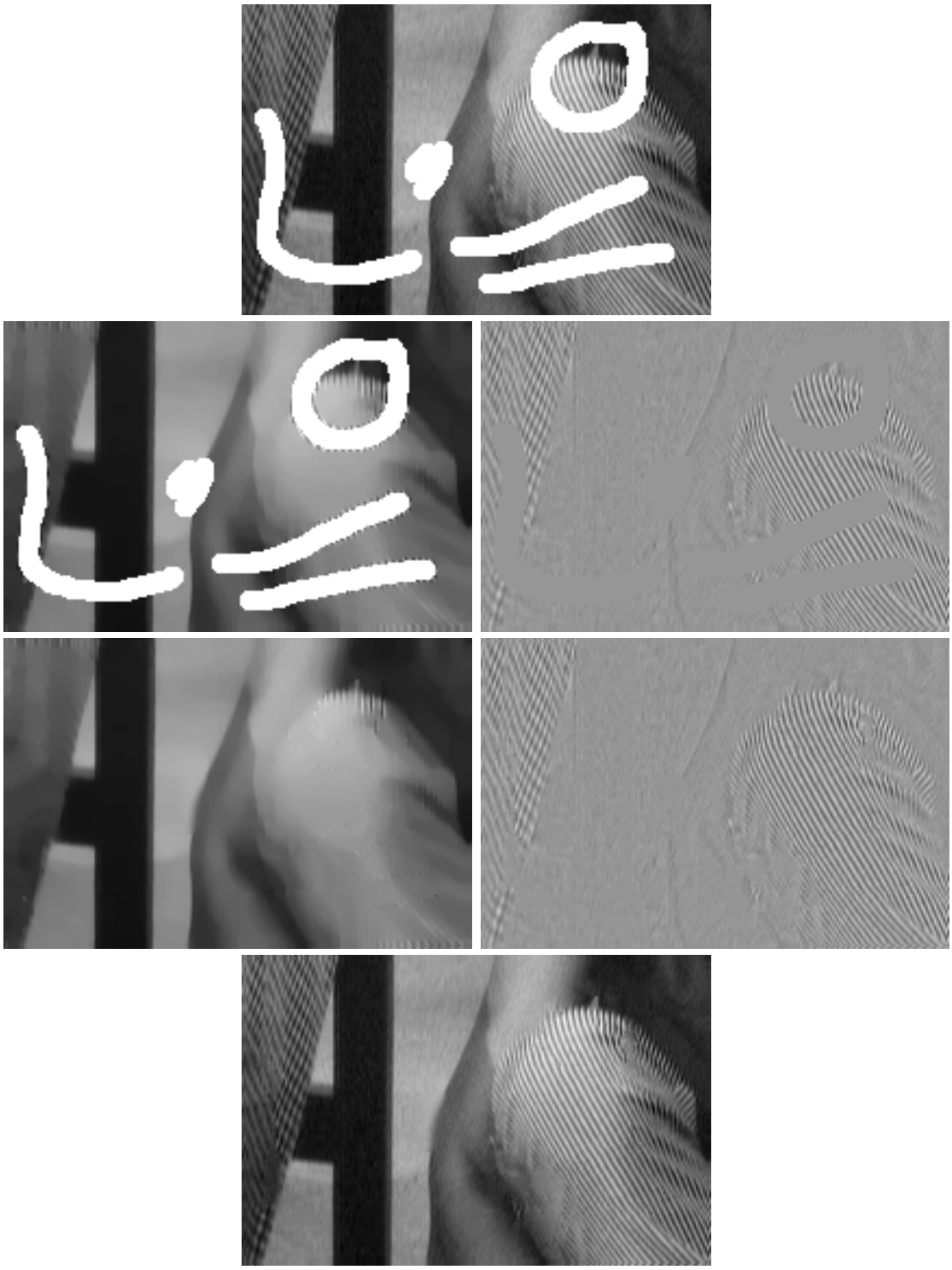


Figure 2.4: Example of image inpainting using the Total Variation Model to separate texture from the initial image.

Chapter 3

The Rudin and Osher Shock Filter Enhancement Model (SFE)

We now look at an image sharpening method using shock filters. By using non-linear time dependent partial differential equations we see the discretisation of a system where an initial image $u(x, y, 0)$ converges to that of a steady state solution $u(x, y, \infty)$. The partial differential equations satisfy a maximum principle, and the total variation of $u(x, y, t)$ is the same for all $t \geq 0$. The resultant image obtained is nonoscillatory and piecewise smooth.

Unlike the VO model from the Chapter 2 we are not attempting to diffuse the image and preserve sharp boundaries, we are reverse engineering a diffusion procedure to remove blurring and restore sharp boundaries. This attempt to recover/generate new information is an ill posed problem in the sense that information is being extracted from incomplete data. In addition to boundaries being sharpened, noise is amplified significantly and in an unstable manner. Noise could be present in the initial image, or just introduced by rounding errors from the discretisation of the scheme. If the image has discontinuities (which is usually the case) oscillations will occur near them (known as ringing) if an attempt to prevent noise amplification is made by removing high frequencies.

The techniques developed in [15] build in total variation preserving techniques into partially tested non linear shock filter schemes. Image sharpening is a logical

application of shock filters as images are essentially two dimensional functions with sharp discontinuities, (the boundaries between objects). Shock filters are designed to resolve discontinuities and the discontinuities are usually the most important part of an image (as opposed to say the texture within objects). It is difficult to discern what is within a blurry image, yet it is quite easy to see what is in an untextured (cartoon) image with sharp boundaries. People have claimed to see the Loch Ness Monster in blurry image for years, who cannot see what is going on in The Simpsons.

3.1 Shock Filters and Shock Capturing Methods

The following description of shock filters is from [15].

The filters use non-linear time-dependent partial differential equations. The evolution of the initial data $u_0(x)$ *as* $t \to \infty$ *through* $u(x,t), \quad t > 0$ *is the filtering process. The partial differential equations have solutions that satisfy a maximum principle and more. In fact the total variation of the solution for any fixed positive time is the same as that of the initial data, i.e., the operator is total variation preserving.*

The paper [15] looks at the 1D case and then extends the process to a 2D situation. A shock capturing scheme for image processing needs to be quite sophisticated as the aim is to restore sharp edges accurately from smeared images and without generating oscillations. Resolving edges sharply is critical as it is the whole point, but resolving them in an oscillatory way will cause ghost lines, a duller copy of edges appearing either side.

We first observe the 1D case, which transforms smooth curves into piecewise constant lines with sharp edges (as expected) and how any noise is sharpened as well. In addition to these examples there will also be an example of where significant diffusion has removed too much information for an accurate recovery.

3.2 Enhancement in One Dimension

We start with the following equation, which stems from previous work in shock calculations (though this is not made explicitly clear in [15])

$$u_t = -\left|u_x\right| F(u_{xx}) \tag{3.1}$$

where $x \in \mathbf{R}, \qquad 0 \leq t \in \mathbf{R}, \qquad u(x,0) = u_0(x)$ is the initial data and F is a Lipschitz continuous function satisfying

$$F(u) \begin{cases} > 0 & \text{if } u > 0 \\ < 0 & \text{if } u < 0 \\ = 0 & \text{if } u = 0. \end{cases} \tag{3.2}$$

The $-\left|u_x\right|$ term satisfies a maximum principle (meaning that the solution in non-oscillatory), in that at extrema $u_x = 0$ so $u_t = 0$. i.e. extrema are time invariant so

$$\begin{aligned} \max u(x,t) &= \max u(x,0) \\ \min u(x,t) &= \min u(x,0) \\ TVu(x,t) &= TVu(x,0) \qquad \forall x. \end{aligned}$$

While this still holds for

$$u_t = +\left|u_x\right| F(u_{xx}) \tag{3.3}$$

with a positive sign, the results are better, or rather more stable. The authors of [15] were informed by P. L. Lions (by private communication [15]) that his work in viscosity solutions had revealed that conservation form of solutions to 3.3 can result in singularities at extrema which then diminish the magnitude of the extrema as t increases. As a result reverse engineering such schemes can generate spurious extrema resulting in very unstable behaviour. The F function determines what type of extrema are encountered and is referred to as the edge switch.

3.2.1 One Dimensional Discretisation and Results

Equation (3.1) is discretised to

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{h} \left| m(u_{i+1}^n - u_i^n, u_i^n - u_{i-1}^n) \right| F\left(\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2} \right) \tag{3.4}$$

with time and space step $\Delta t, h$, and m is defined as

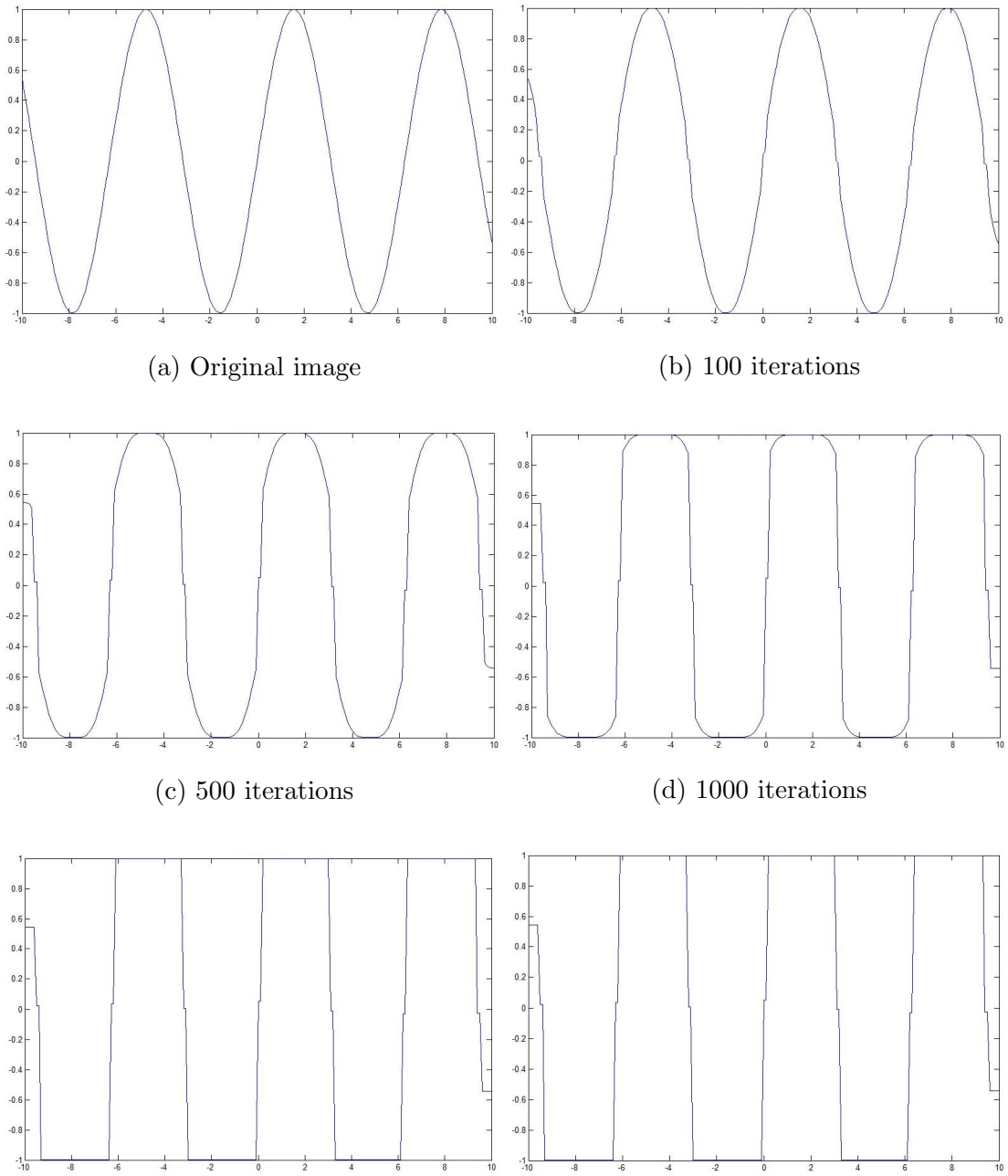
$$m(x,y) = \begin{cases} (\text{sign } x) \min(|x|, |y|) & xy > 0 \\ 0 & xy \leq 0. \end{cases} \tag{3.5}$$

F is taken in the following (self generated) examples to be $F(x) = \text{sign}(x)$.

From the plots of the sin graph, it can be seen that after only 500 iterations the sin wave curves are being straightened into vertical and horizontal lines. By 10000 iterations they have reached a steady state solution of a square wave. Note that the steady state is not necessarily the sought result, since any resultant image in between may be equally valid. The square wave is in keeping with expected behaviour, though an observer may expect more of a zig-zag of diagonal lines instead.

The next set of plots are of an x^2 curve containing two small errors on one side. The symmetric nature of x^2 helps compare the effects of the errors after processing. We can see how the errors are altering the shape resultant graph. After around 500-1000 iterations the errors are somewhat emphasised, with all symmetry lost by 5000 iterations. Here it is important to note that while no attempt to solve this problem (in 1D) has been found in subsequent papers in this area, (the point of this is to adapt the approach for 2D image processing), there were far more efficient and elegant methods available for sharpening 1D graphs available at the time. Indeed [15] refers to how they "could have done the same just by finding inflection points and extrema and then performing thresholding." They then state how this would be "a futile exercise if we tried to extend it to a two-dimensional calculation."

The plots 3(a)-(f) demonstrate the limitations of the enhancement process. An initial wave has been heavily diffused, reducing the magnitude of the oscillations.



(a) Original image

(b) 100 iterations

(c) 500 iterations

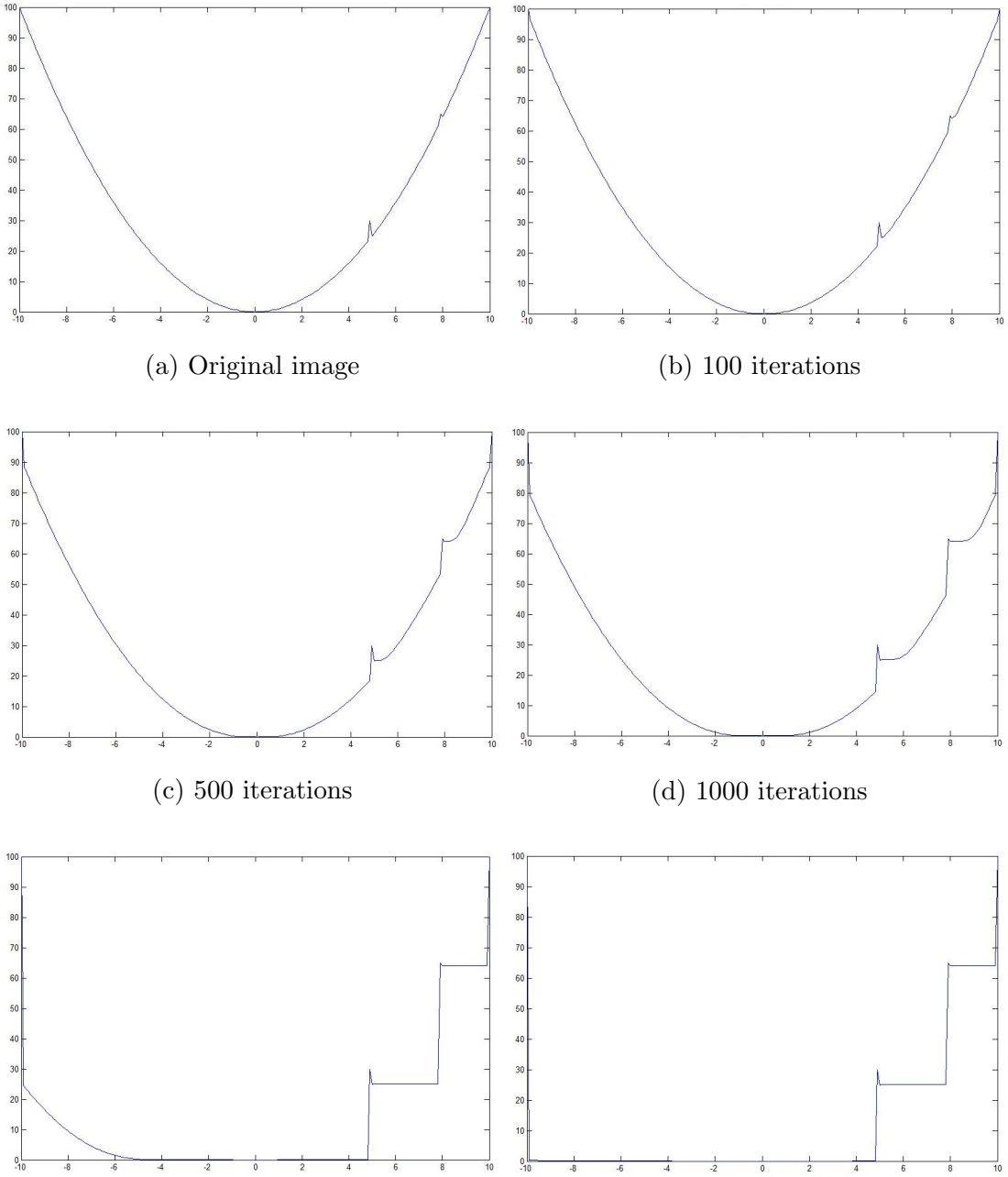
(d) 1000 iterations

(e) 10000 iterations

(f) 100000 iterations

Figure 3.1: Sin wave

Self generated



(a) Original image

(b) 100 iterations

(c) 500 iterations

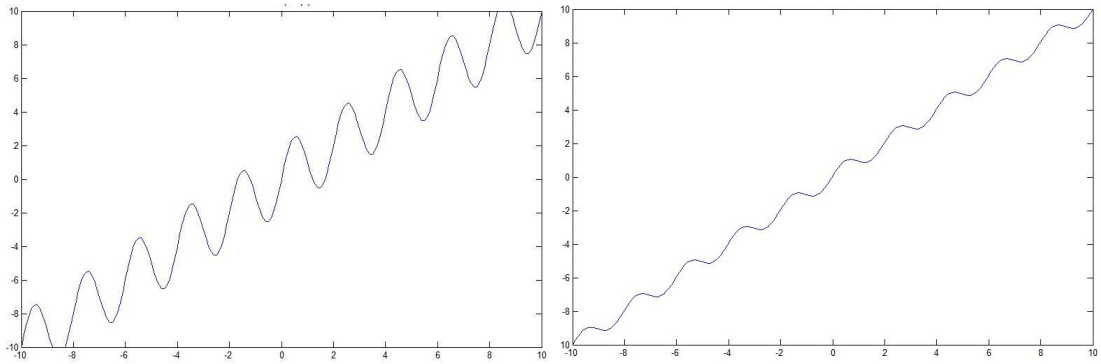
(d) 1000 iterations

(e) 5000 iterations

(f) 10000 iterations

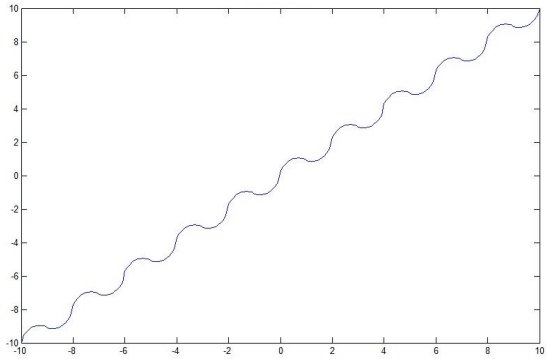
Figure 3.2: x^2 curve

Self generated

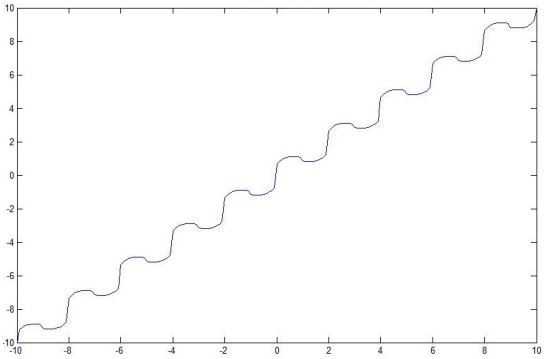


(a) Original image

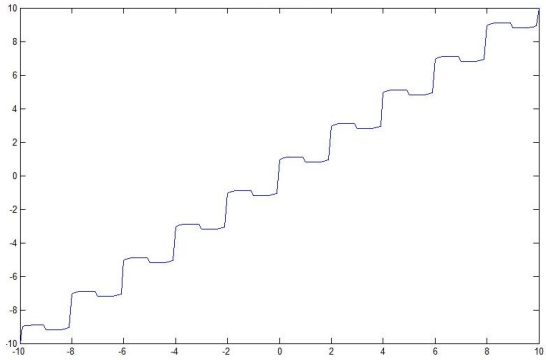
(b) Diffused image



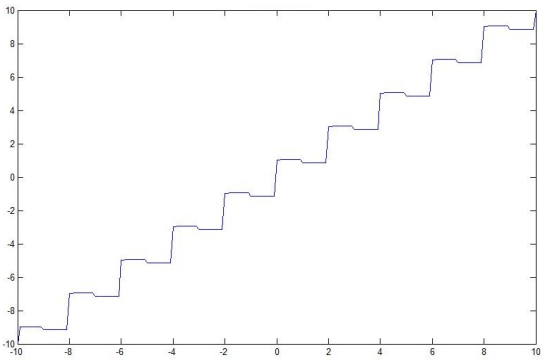
(c) 100 iterations



(d) 250 iterations



(e) 500 iterations



(f) 1000 iterations

Figure 3.3: $x + 2\sin(x \times \pi)$

Self generated

While the programs attempts have partially restored the waves, the loss of information is beyond recovery, with waves being squared off and the original wave magnitude remaining diminished. This is a result that stems from the ill posed nature of the problem and can never be completely overcome.

The program remained fast and efficient, performing around 300 iterations a second. While results would have been slower at the time of development, this is still very fast in comparison to other developed image processing techniques. Note that it is unlikely to be performing thousands of iterations, the 10000 and 100000 iteration cases being performed here to demonstrate the convergence of the scheme.

Table 3.1: Results for sin wave plots 1(b)-(f)

number of iterations	time to execute (seconds)
100	0.3410
500	1.7011
1000	3.4027
10000	34.0194
100000	339.7153

Computer specs: 3.16GHz Duo CPU, 3.21GB RAM

3.3 Extension to Two Dimensions

We now at the generalisation to two dimensional image enhancement,

$$u_t = -\sqrt{u_x^2 + u_y^2} F(\mathscr{L}(u)) \tag{3.6}$$

where $x, y \in \mathbf{R}$, $0 \leq t \in \mathbf{R}$, $u(x, y, 0) = u_0(x, y)$ is the initial data. F again satisfies the criteria given in (3.2). $\mathscr{L}(u)$ is a second order, (generally) non-linear elliptic operator. The $-|\nabla u|$ term determines the magnitude of the gradient, while $F(\mathscr{L}(u))$ is the edge detector and is again key as it determines where sharpening should occur. F is taken to be the same as in the one dimensional

case, (i.e. $F(\mathscr{L}(u)) = \text{sign}\,\mathscr{L}(u)$). Thus sharpening will occur when $\mathscr{L}(u)$ changes sign, and the accuracy of the sharpening process will depend on how good $\mathscr{L}(u)$ is at changing sign at the boundaries. Based on the 1D case a logical edge operator might be the Laplacian $\mathscr{L}(u) = u_{xx} + u_{yy}$. However shortly before [15], Rudin (one of the authors) had done extensive work on shock filters, edge detectors and their performance in the proximity of singularities. It was demonstrated in [17] that the Laplacian detector will miss "geometrically non-trivial singular boundaries." Based on this work a "better choice" is made, an "expression for the second derivatives of u in the direction of the gradient."

$$\mathscr{L}(u) = u_{xx} \cdot u_x^2 + 2 \cdot u_x u_y u_{xy} + u_{yy} \cdot u_y^2 \tag{3.7}$$

This choice is far from unique, many choices exist and will give different results depending on the type of blurring. This choice is based on the assumption that blurring was by a Gaussian convolution, and is also a trade-off between quality and speed.

3.3.1 Two Dimensional Discretisation and Results

Equation (3.6) is discretised to

$$u_{i,j}^{n+1} = u_{i,j}^n - \frac{\Delta t}{h}\sqrt{(m(\Delta_+^x u_{i,j}^n, \Delta_-^x u_{i,j}^n))^2 + (m(\Delta_+^y u_{i,j}^n, \Delta_-^y u_{i,j}^n))^2}F_{i,j}(\mathscr{L}(u)). \tag{3.8}$$

The $\Delta_\pm^x$ terms are as defined in [15], given below

$$\begin{aligned}
\Delta_+^x u_{i,j}^n &= u_{i+1,j}^n - u_{i,j}^n \\
\Delta_-^x u_{i,j}^n &= u_{i,j}^n - u_{i-1,j}^n \\
\Delta_+^y u_{i,j}^n &= u_{i,j+1}^n - u_{i,j}^n \\
\Delta_-^y u_{i,j}^n &= u_{i,j}^n - u_{i,j-1}^n
\end{aligned}$$

with the minmod operator m as defined in (3.5), and partial differentials discretised to

$$\begin{aligned}
u_x &= (1/h)(\Delta_+^x u_{i,j}^n + \Delta_-^x u_{i,j}^n) \\
u_{xx} &= (1/h^2)(\Delta_+^x \Delta_-^x u_{i,j}^n) \\
u_{xy} &= (1/2h^2)(\Delta_+^x \Delta_+^y u_{i,j}^n + \Delta_-^x \Delta_-^y u_{i,j}^n)
\end{aligned}$$

(u_y, u_{yy} defined similarly). Satisfaction of the CFL stability condition is achieved using

$$\sup\left(\frac{\Delta t}{h}F_{i,j}(\mathscr{L}(u))\right) \leq \frac{1}{4} \tag{3.9}$$

To remove the external function m and simplify coding the scheme is then modified to

$$\begin{aligned} u_{i,j}^{n+1} =& u_{i,j}^{n} \\ &- \frac{\Delta t}{h}\sqrt{((\Delta_+^x u_{i,j}^n)^+)^2 + ((\Delta_-^x u_{i,j}^n)^-)^2 + ((\Delta_+^y u_{i,j}^n)^+)^2 + ((\Delta_-^y u_{i,j}^n)^-)^2}F_{i,j}^-(\mathscr{L}(u)) \\ &- \frac{\Delta t}{h}\sqrt{((\Delta_+^x u_{i,j}^n)^-)^2 + ((\Delta_-^x u_{i,j}^n)^+)^2 + ((\Delta_+^y u_{i,j}^n)^-)^2 + ((\Delta_-^y u_{i,j}^n)^+)^2}F_{i,j}^+(\mathscr{L}(u)) \end{aligned} \tag{3.10}$$

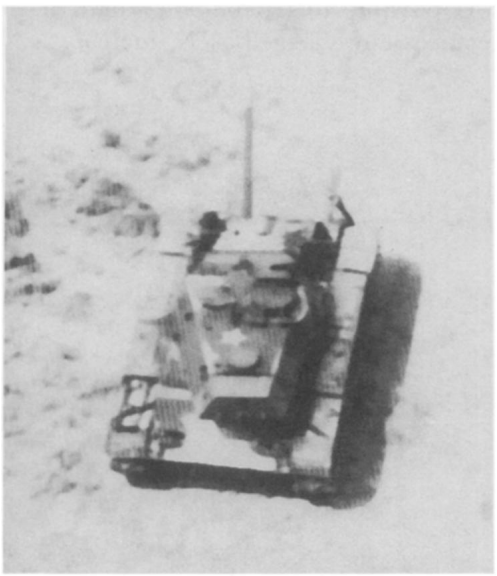
with $(\Delta u)^+ = \max(\Delta u, 0) \qquad (\Delta u)^- = \min(\Delta u, 0)$. This modified discretisation has the same CFL condition. The space step h is chosen as 1 pixel (i.e. h=1). We shall now see some results from the enhancement process, one from the original paper and some new results made by coding up the model, shown in Fig 3.5 - 3.7

With Fig 3.4, after only a few iterations the results of the enhancement are immediately obvious. The process has increased the contrast between certain features and their surroundings making them clearer, most notably the entry hatches on the turret or the ladder in the lower left corner (from the observers perspective). Although some detail has been lost as well. The boundary between the left side of the tank and the terrain is even harder to see than in the diffused image. Most terrain detail has been lost too. In Fig 3.5 again the enhancement has increased contrast, making certain features clearer, most notably the cameraman's gloves, side pockets and where his coat overlaps (this is less clear in the smaller scale figure here).

Looking at some colour results, Fig 3.6 we see a diffused natural scene with larger scale boundaries and smaller scale texture. The larger boundaries have been sharpened, though this has partially caused jagged edges and a separation of colours. Colour images are processed in RGB (Red Green Blue) format, where each pixel is stored as a sum of all three colours. The shock filter is applied separately to each colour, and different rounding errors between them causes slight

(a) Original tank image

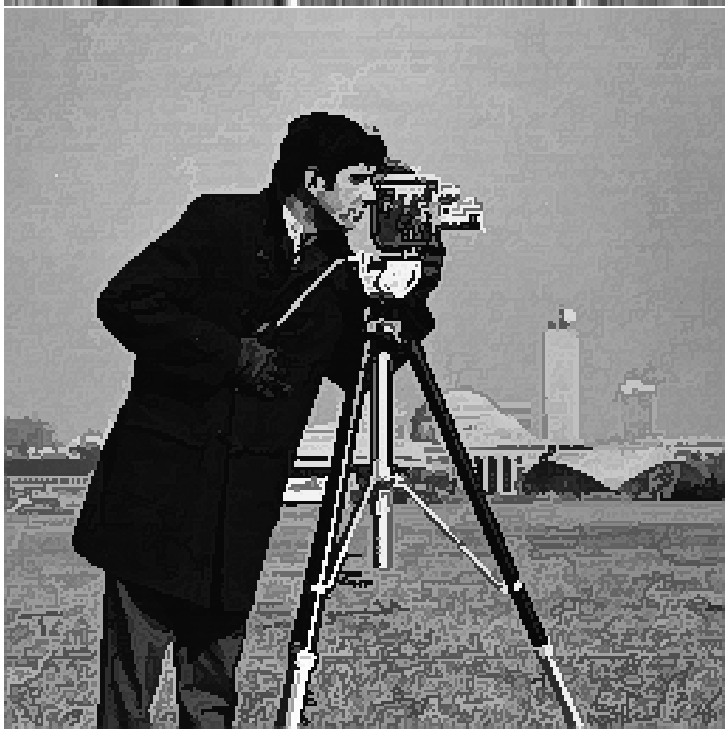
(b) Diffused tank image



(c) 13 iterations

Figure 3.4: Tank images and results obtained from [15].

(a) Cameraman



(b) 10 iterations

Figure 3.5: Original Cameraman image obtained from MATLAB.

(a) Autumn

(b) Autumn (diffused)

(c) 10 iterations

Figure 3.6: Original Autumn image obtained from MATLAB.

(a) Peppers

(b) Peppers (diffused)

(c) 10 iterations

Figure 3.7: Original Peppers image obtained from MATLAB.

discrepancies resulting in minor colour separation at boundaries. Note that the small scale texture is not restored, as little information of it is left by a diffusion of the image. The result is a painting-like image.

With Fig 3.7 there are lots of different colours, some sharing similar colour levels, such as the overlapping green peppers, some sharing partial colour levels, such as the red and orange peppers (orange consists of red and green in RGB), and totally distinct colours such as the red and green peppers. Comparing with Fig 3.6 due to the lack of multicoloured texture of objects within this figure, it suffers far less from colour separation. The more distinct colours are not all good news however, as higher iterations that were performed showed the process began to emphasise the glare of light reflection in some locations. This appears to affect colour images more than black and white images; notice how in Fig 3.5 the sharpened black and white in the camera stand does not suffer from glare. The shock filter model has demonstrated substantial success in the above examples, with interesting quirks and unwanted side effects of the sharpening process. The ill-posed nature of the problem means complications will always be present, such as the jagged edges. It will also always have its limitations, the lack of a unique solution means with more image distortion comes more reasonable solutions, making it harder for any model to 'guess' which is correct.

Chapter 4

A Final Comparison of Different Total Variation Models

We now see a short comparison of three total variation models. The VO model of Chapter 2, Meyer's model [14](the texture preserving element that was built into VO), and the more recent TV-L^1 model [23](a different total variation model stemming from the ROF model [18], the other element encompassed by the VO model). We will briefly see the key difference between these models and observe the results.

The ROF model was attempting to solve

$$\min_{u \in BV} \left\{ TV(u) + \lambda \left\| f - u \right\|_{L^2}^2 \right\}. \tag{4.1}$$

Here the TV(u) term reduces oscillations, while $\lambda \left\| f - u \right\|_{L^2}^2$ keeps the processed image u close to the original image f.

Meyer's model attempts to solve

$$\inf_{u \in BV} \left\{ \int |\nabla u|, s.t. \left\| f - u \right\|_G \leq \sigma \right\}, \tag{4.2}$$

or in Lagrangian relaxed form

$$\inf_{u \in BV} \int |\nabla u|, + \lambda \left\| f - u \right\|_G. \tag{4.3}$$

The Lagrangian relaxed version is included here for comparison, but it was the constrained version above that was used originally in [14]. The Lagrangian form of

Meyer's model cannot be solved simply with a partial differential equation method (as Euler-Lagrange equations cannot be formed for (4.3))

The VO model attempts to approximate (4.3) in a way that can be solved using partial differential equations. The approximation (2.1) is given again below for convenience.

$$\inf_{u,g1,g2} \left\{ VO_p(u, g_1, g_2) = \int |\nabla u| + \lambda \int |f - u - \partial_x\ g_1 - \partial_y\ g_2|^2\ dx\ dy \right. \\ \left. + \mu \left[\int (\sqrt{g_1^2 + g_2^2})^p\ dx\ dy \right]^{1/p} \right\}. \quad (4.4)$$

(The G term from (2.1) has been replaced with VO in the above equation to avoid confusion with the norm of Meyer's model.)

The TV-L^1 model, similar to the ROF method, it modifies the minimisation problem to

$$\min \left\{ TV(u) + \lambda \left\| f - u \right\|_{L^2}^2 \right\}, \quad (4.5)$$

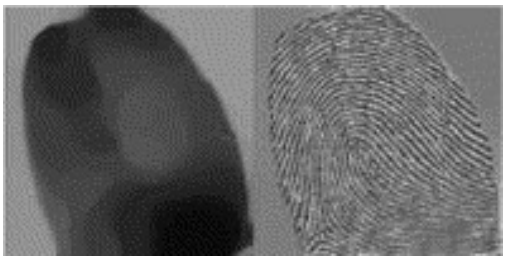
which yields the Lagrangian relaxed form

$$\min_{u \in BV} \int_\Omega |\nabla u| + \lambda \int |f - u|, \quad (4.6)$$

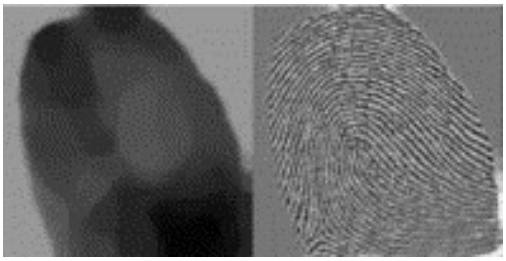
where Ω is a bounded convex open set Ω. It was stated in the original [18] paper that "the L^1 norm of the gradient is the appropriate space. This is basically the space of functions of bounded total variation: BV." Despite this the L^2 norm was used as it is easier to work with. Advances in dealing with the L^1 norm (and its associated complications) since then in a wide range of papers such as [1] and [7] among others, have demonstrated the better approximation that the L^2 norm also give rise to, including other useful properties such as texture extraction by scale [22].

4.1 Comparison of models results

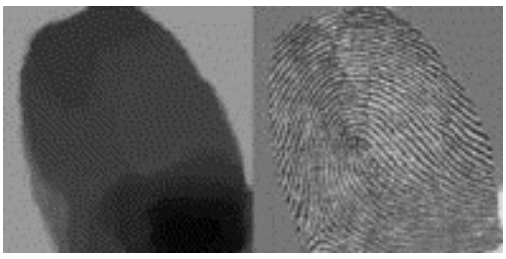
(a) Clean fingerprint image



(b) Meyer ($\sigma = 35$)



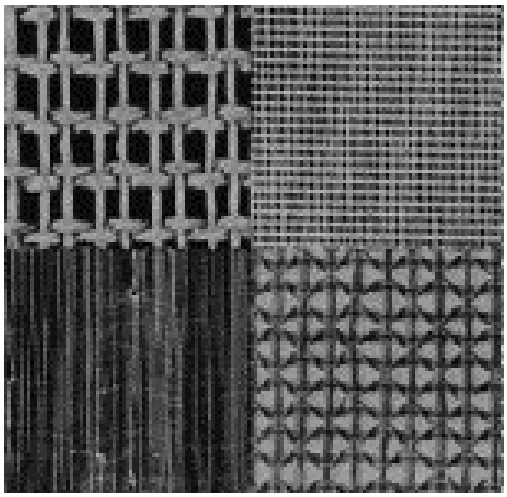
(c) VO ($\mu = 0.1$)



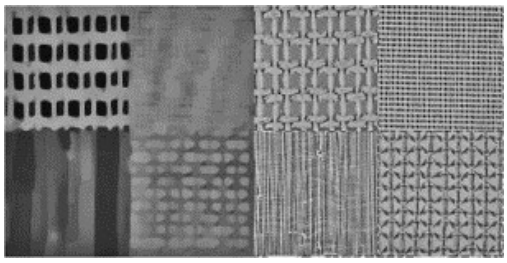
(d) TV-L^1 ($\lambda = 0.4$)

Figure 4.1: Reproduced from [22].

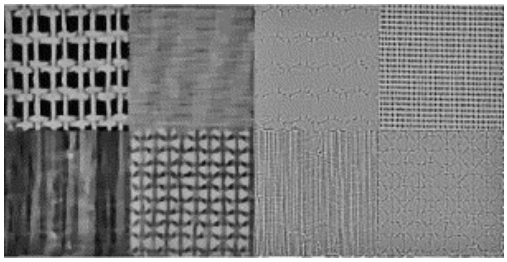
Here we see a fingerprint processed to extract the texture for analysis. Fingerprint analysis is a comparison of the ridges from a finger, so the ideal result in this example would be separate the ridges sharply and accurately but leave any light changes in the cartoon element. Here the Meyer and VO models give similar results, though the TV-L^1 has extracted the varying light levels with the texture. While this small difference might not seem significant to the human eye and visual comparison would be unimpaired, an algorithm meant to compare a print to those stored in a database may suffer more.



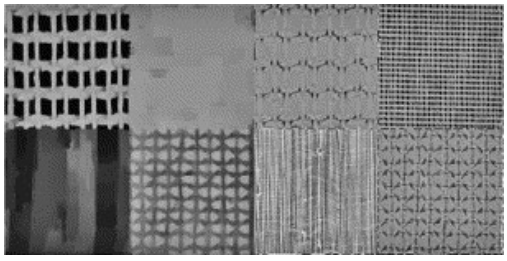
(a) Clean 4texture image



(b) Meyer ($\sigma = 50$)



(c) VO ($\mu = 1.0$)

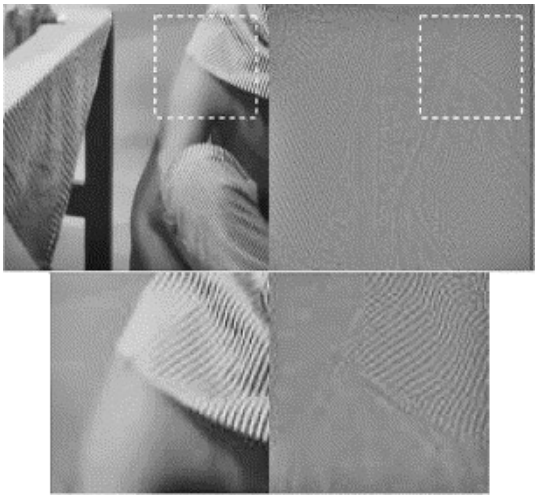


(d) TV-L^1 ($\lambda = 0.8$)

Figure 4.2: Reproduced from [22].

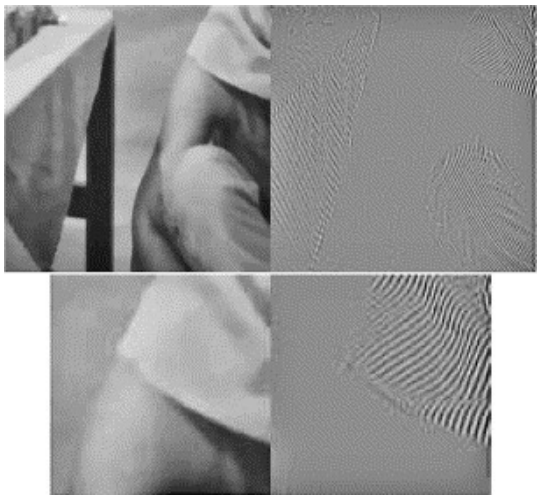
Given the varying texture levels, the parameters were chosen to ensure the extraction of the texture in the upper right box. For this quadrant the TV-L^1 model has outperformed the other models. However for the upper left and lower right boxes Meyer's model has extracted more texture than the others, and is the only one that has managed to extract the knots leaving smooth lines. The VO model performs quite poorly in this example, the upper right and lower left quadrants show less texture separation than the other models, and the other two quadrants have kept most texture in the cartoon part.

(a) Clean Barbara image

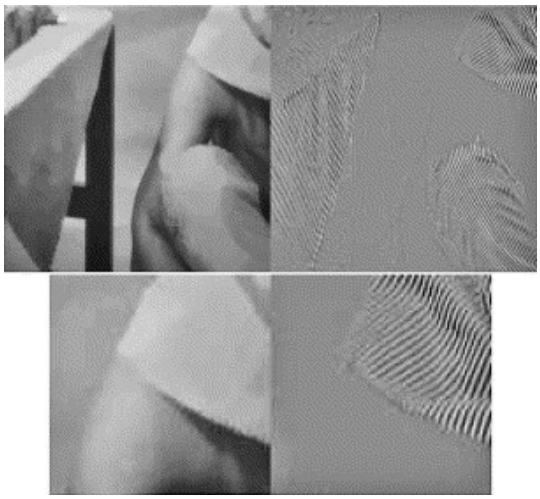


(b) Meyer ($\sigma = 6$)

(c) Meyer ($\sigma = 15$)



(d) VO ($\mu = 0.5$)



(e) TV-L^1 ($\lambda = 0.8$)

Figure 4.3: Reproduced from [22].

In this example the Meyer model begins to have difficulties. It seems unable to extract most texture without bringing parts of smooth objects with it. There are "...two different σ's, one very conservative and then one very aggressive..." [22]. For the conservative form 4.3b most texture is not extracted, where as with 4.3c the outlines of smooth objects such as the arm or the table leg have also been extracted. The VO and TV-L^1 perform rather similarly, most texture and only texture being separated from the image. The only noticeable difference is the ripple in in the sleeve has been extracted also by the TV-L^1 model but remains for the VO model.

Chapter 5

Morphological Component Analysis (MCA)

So far we have seen a model for diffusing still images to remove noise or texture, and a model for sharpening the edges of a diffused image, both focusing on speed and efficiency rather than quality of results. We have also seen a comparison of a couple of other variational models for different textured images. All models displayed varying degrees of success, performing well with some images and poorly with others. The differences in the results are because different images have different kinds (and scales) of texture, and different models work better with different textures. Recall Fig 4.1, the TV-L^1 model extracted the varying light level, which was undesirable in the example. However it is perfectly valid to view varying light levels as an added texture (or even noise) to the underlying image. While the light extraction was undesirable, that does not make its extraction wrong. Until now we have simply assumed the image-texture representation $f = u + v$, but all different textures and noise are just lumped into one function v. A somewhat better definition is one that defines a sum of layers, each layer being its own piece of texture or noise. For the purposes of the new approach in this Chapter, we will now reformulate this breakdown.

A given overall signal (in our case the original image) consists of a sum of component signals to be separated (the component textures, objects, lighting effects,

etc.) composed by the sum

$$f = \sum_{k=1}^{K} f_k. \tag{5.1}$$

Separation of images into separate components is not a new form of texture separation. Independent Component Analysis (ICA) and sparsity methods have been used to calculate this breakdown, with varying levels of success. The Matching Pursuit method is an example of one such approach, first developed by Mallat and Zhang in 1992, it uses a greedy algorithm which decomposes and refines the input signal into a linear expansion of waveforms [13]. These kinds of approaches are combinatorially complex and very expensive. An alternative to these is Morphological Component Analysis, an adaptation of the Basis Pursuit method that separates features within an image with different morphological aspects [20].

The MCA model requires two assumptions to hold true.

1. For each signal f, there exists a dictionary of bases $\mathscr{D} = \{\Phi_1, \ldots, \Phi_N\}$ (that may be overcomplete) such that each component signal f_k is sparse in $\Phi_i \in \mathscr{D}$ for some i.

2. For each signal component f_k that is sparse in Φ_i, f_k is non-sparse (or at least less sparse) in $\Phi_j \in \mathscr{D}$ for all $j \neq i$.

Thus choosing the dictionary wisely is key, as the dictionary is what discriminates between the different image layers. The speed of the model will also heavily depend on the dictionary chosen; the greater the difference between the sparse and non sparse representations of a component, the more effectively it will be separated in terms of speed and accuracy.

5.1 The MCA Model

Here we will look at an example of an MCA model, presented in [20]. MCA models are not unique as MCA can be applied to different types of problems and usually have small differences depending on the desired result, or to incorporate known

information about the input signal. An example of known information would be if added noise is known to be Gaussian, uniform, zero-mean, etc.

We are attempting to find the most sparse representation over the dictionary $\mathscr{D}$ of a given signal f. Thus we are attempting to solve

$$\begin{aligned} &\arg \min_{\{\alpha_1,\ldots,\alpha_K\}} \sum_{k=1}^{K} \|\alpha_k\|_0 \\ &\text{subject to: } f = \sum_{k=1}^{K} \Phi_k \alpha_k. \end{aligned} \tag{5.2}$$

$\|\alpha_k\|_0$ denotes the ℓ_0 pseudo-norm of the vector α_k (that is, the number of non-zero coefficients in α_k). If our previous two assumptions about the dictionary hold, this will lead to an accurate separation of the signal components. While this might lead to the desired solution, (5.2) is difficult to solve as the problem is non-convex. The suggested way to overcome this is by replacing the ℓ_0 norm with the ℓ_1 norm. A solution can then be formulated by relaxing the constraint in (5.2), then by rewriting the constrained form in Lagrangian form we obtain

$$\arg \min_{\{\alpha_1,\ldots,\alpha_K\}} \sum_{k=1}^{K} \|\alpha_k\|_0 + \lambda \left\| f - \sum_{k=1}^{K} \Phi_k \alpha_k \right\|_2^2. \tag{5.3}$$

The above gives an example of where the model is tailored to a specific type of problem. The use of the ℓ_2 norm is applied if the residual is assumed to behave as zero-mean Gaussian noise. Different models could assume different behaviour, like Laplacian (ℓ_1) or uniform distribution (ℓ_∞).

This model also further notes the memory hungry nature the algorithm will have depending on the total length of all components α_k due to all the matrices involved. As the dictionary is usually (very) over-complete, the introduced redundancy increases the overall matrix size. If the redundancy factor is 10 (i.e. the total length of all components α_k is 10 times the length of the input signal), then 10 times as much memory as the input signal will be required for manipulating the solution. By reformulating the problem the signal components $f_1, \ldots, f_K$ become the unknowns, which reduces the memory required.

$$\arg \min_{\{f_1,\ldots,f_K\}} \sum_{k=1}^{K} \|\mathbf{T}_k f_k\|_1 + \lambda \left\| f - \sum_{k=1}^{K} f_k \right\|_2^2, \tag{5.4}$$

where $\mathbf{T}_k$ is the basis transformation of Φ_k ($\alpha_k = \mathbf{T}_k f_k$). Further an advantage of the reformulation is that constraints can be added to each signal component leading to an addition term in (5.4),

$$\arg \min_{\{f_1,\ldots,f_K\}} \sum_{k=1}^{K} \|\mathbf{T}_k f_k\|_1 + \lambda \left\| f - \sum_{k=1}^{K} f_k \right\|_2^2 + \sum_{k=1}^{K} \gamma_k C_k(f_k), \tag{5.5}$$

where C_k implements constraints on f_k.

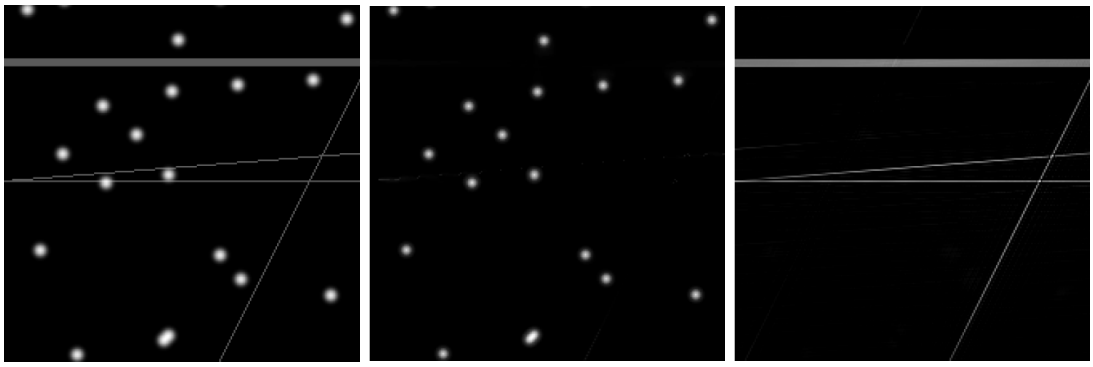
This can then be solved by external numerical solvers. The paper this model originates from uses the Block-Coordinate Relaxation Method [20]. The discretisation will not be shown for this, though examples of discretised models and algorithms implementing this can be found in [20], [5] and [4] among others.

While all this might seem like a lot of hassle, separating each object and texture into separate layers, the appeal is not simplicity of the model, it is the simplicity it allows and induces in various applications. Demonstrations of the application of MCA applied to real world imaging can be found in such papers as fingerprint separation [12], astronomical imaging [19] and cancer screening [11].

5.2 A Few Numerical Results

The following result is an example from [20], demonstrating the ability to separate Gaussian dots from straight lines in a relatively empty image. For this example the authors used two transforms, the wavelet transform as its isotropy makes it efficient at Gaussian detection, and the ridgelet transform which is optimal for extracting lines. The efficiency of these transforms at this was discovered in previous work by the authors.

Fig 5.1 is an example of astronomical imaging. Most stars will appear as Gaussian distributed, with dust, gas and other remnants in space adding separate noise. Separation of these features must be very accurate for astronomical imaging as most features are very small. The authors have also demonstrated in [10] the texture separation and inpainting capabilities of MCA, seen in Fig 5.2 and 5.3.



(a) Original Image. (b) Extracted Gaussians. (c) Extracted lines.

Figure 5.1: Lines and Gaussians image. Reproduced from [20].
Gaussians extracted with the *à* trous wavelet.
Lines extracted using Ridgelet.

(a) Original Image. (b) Separated cartoon. (c) Separated texture.

Figure 5.2: Textured Barbara image. Reproduced from [20].
Separation obtained using curvlet transform and cosine transform [10].

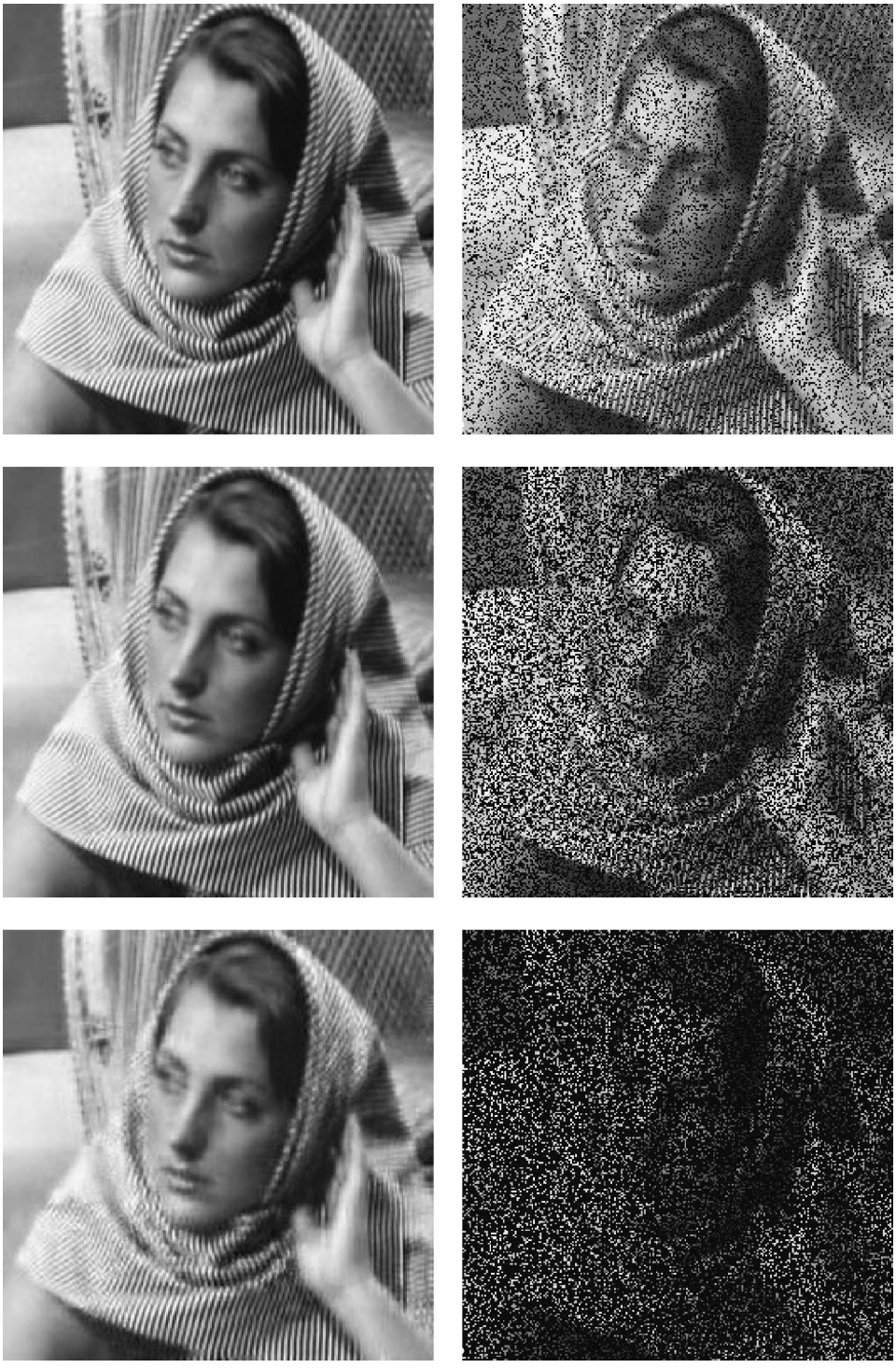


Figure 5.3: Barbara image with missing information. Reproduced from [20]. Separation obtained using curvlet transform and cosine transform [10]. $20\%, 50\%, 80\%$ of pixels removed, with restored image to the left.

5.3 MCA in Video Processing

We now look at MCA moving beyond stills and its application in video processing. Moving into three dimensions (2D+T) leads to more complexity in the problem and more possibilities to improve how the input video is processed. This adds the new form of texture caused by movement alone. A smooth moving object can become textured in video depending on the movement pattern. In fact what we mean by texture has still not been actually defined. Most papers have just referred to terms like most useful information or semantically meaningful information. For rigorous mathematical based models this is somewhat unsatisfactory, but giving any mathematical definition of texture is a notoriously difficult problem. Mathematical definitions require quantifiable features with clear cut off points. What is meaningful information depends on the observer and the purpose of their observation, it's situational and subjective. MCA being used in video processing is relatively new and a recent paper [8] makes a real step towards this. Stating "around 75% of major publications do not specify their definition of dynamic texture," while not actually defining texture, it defines several different classes that texture will fit into to separate the variety of textures that exist. This Taxonomy is promising, separating textures by the different properties they present.

1. Classification of texture begins with its origins, is the texture natural (created by natural process), artificial (human created) or synthetic (computer generated).

2. The texture is then classed as static (still) or dynamic (moving).

3. Dynamic then undergoes further breakdown by the cause of this motion. Is the motion the result of an internal force (e.g. a moving car), an external force (e.g. a tree blowing in the wind) or acquisitional (e.g. movement of the clouds blocking sunlight).

4. The motion is further separated by texture and motion patterns. Is the texture rigid or deformable, meaning does the texture just move or does it

rearrange itself as well. Is the motion stochastic or deterministic, meaning is the motion predictable or mostly random.

5. Finally the motion has its number of modes determined. This is referring to when there are different motions within the same texture. A large scale tidal wave with smaller scale ripples within is an example of dynamic texture with two modes. This also poses added complexity when they overlap and interact with each other.

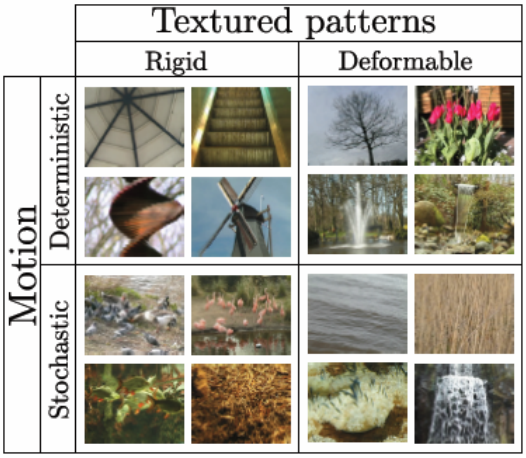


Figure 5.4: Examples of texture and motion separation. Reproduced from [8].

From this the authors then gave the following definition of dynamic textures.

A natural, artificial or synthetic image sequence may contain a static texture component and/or a dynamic texture component. This latest one is composed of at least one dynamic texture.

A dynamic texture is a textured pattern that can be rigid or deformable. This pattern has a motion induced by a force which can be internal, external or created by camera motions. This motion can be deterministic or stochastic. Dynamic textures are composed of modes, which may overlap, characterized by repetitive spatial and temporal phenomena.

In addition to the different motions that can occur within the same texture it is worth noting that separate dynamic textures that are transparent (partially or totally) can overlap, which is not the same thing. The MCA model used in [8] for video processing is similar to what was seen in section 5.1,though they do not use

Lagrangian form.

$$\begin{aligned} &\min_{f_1,\ldots,f_K} \sum_{k=1}^{K} \left\| \Phi_k^T f_k \right\|_p^p \\ &\text{such that } \left\| f - \sum_{k=1}^{K} f_i \right\|_2 \leq \sigma \end{aligned} \tag{5.6}$$

This is then solved by hard thresholding the residual

$$r_k^{(n)} = f - \sum_{i \neq k} f_i^{(n-1)}, \tag{5.7}$$

$$\alpha_k^{(n)} = \delta_{\lambda^{(n)}} \left(\Phi_k^T \left(r_k^{(n)} \right) \right), \tag{5.8}$$

where $r_k^{(n)}$ is the residual and $\delta_{\lambda^{(k)}}$ is the hard threshold function with threshold $\lambda^{(k)}$. Hard thresholding sets to zero any element whose absolute value is lower than the threshold $\lambda^{(k)}$. An outline of the algorithm is presented in [8].

5.4 Results of Video Processing

Here we will see a few results of the MCA model for video processing from [8], and comment on their level of success. We will also see the difference between a 2D+T processing and a 2D (frame by frame) processing.

Fig 5.5 is split into three separate areas highlighted for comparison. (a) shows how the texture of the lake has been separated. Ripples in the water have been extracted as texture in the 2D+T process, but the 2D has left ripples behind in the cartoon component. (b) captures how the model handles objects with the video. It can be seen how the 2D case has extracted smooth parts of the duck to the texture component, and the 2D+T has done so to a far smaller extent. (c) reveals how well texture is preserved in the texture component, and again the 2D+T excels ahead. The water ripples are a highly dynamic texture and the 2D approach entirely misses the dynamic behaviour.

Fig 5.6 is of two overlaying dynamic textures, a water fountain which overlays with water ripples around it. The 2D+T model has attempted to separate the textured ripples in the water from the transparent smooth water flow from the

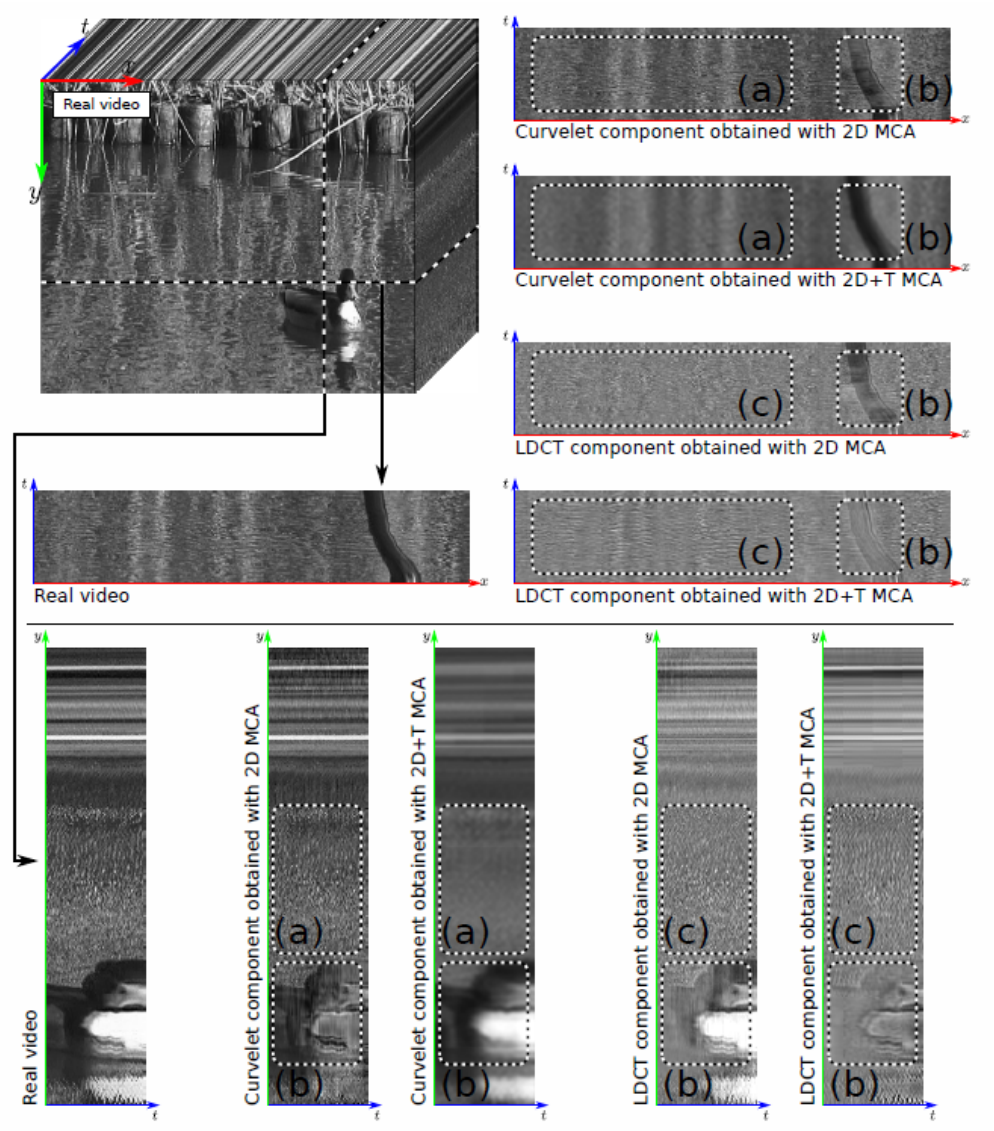


Figure 5.5: Reproduced from [20].

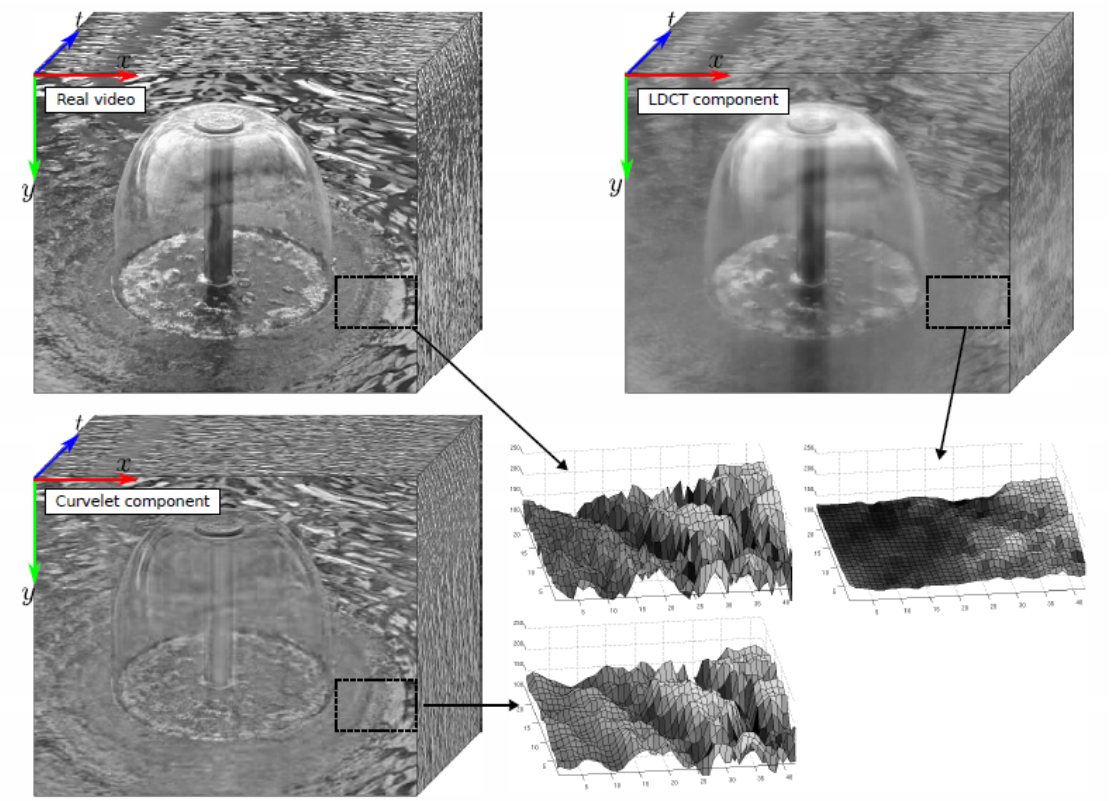


Figure 5.6: Reproduced from [20].

fountain. There is also a surface plot for a section in the real video and its components. As can be seen there is a rather smooth surface in one component, and most bumps and peaks are in the texture surface plot.

These results demonstrate that the MCA model can be extended to the video processing are is capable of extracting various different features and components. It can process video taking advantage of all three dimensions and is far more effective than a similar two dimensional approach that handles videos frame by frame without interpreting temporal features.

As MCA was designed to be an alternative to the computationally expensive ICA and sparsity methods mentioned earlier, it is worth just noting the computation time needed for these kinds of component separations. There is a short section in [8]on time considerations and a table of the time taken on two different machines is given.

	Platform 1 (32 bits)	Platform 2 (64 bits)
ATSLc	$\approx$ 3h47	$\approx$ 1h56
ATSEc	$\approx$ 5h22	$\approx$ 2h44

Table 5.1: Average computation time for MCA to decompose 5 second video from DynTex database. ATSLc and ATSEc are thresholding strategies implemented in the MCA algorithm.

This timing is relatively large for 5 seconds of video. The fastest of the two, ATSLc would take approximately a day to process 1 minute of video. A video longer than a few minutes would most likely need to be split between multiple platforms. This is commented on by the authors, who also point out that the algorithm is multi core compatible (multiple platforms can be set up to split the process between them).

Chapter 6

Conclusions and Further Work

We have investigated image processing techniques including total variation, shock filters and MCA. The total variation model has been discretised and its results for different images and textures seen. The shock filter enhancement model has also been discretised and shown to restore sharp edges, with notable side effects like jagged edges occurring. Total variation models have been contrasted to show how they outperform each other depending on the textured image they are processing. MCA methods have demonstrated the breaking down of images into multiple layers, and an extension to videos has given good results for texture separation. A taxonomy for dynamic textures has also been defined.

Work in Total variation minimisation for image processing continues, with other additions to solving the problem being implemented such as the H^{-1} norm and Bregman equations. Despite this potential and continued research total variation minimisation methods in image processing remain mostly unused. Most applied models have been specifically tailored to their applications, and total variation remains general in its applications. Attempting to develop an application targeted model could be a potential for further development within the total variation minimisation framework. Different total variation models have outperformed each other which suggests a total variation model could be tailored to a specific problem.

The shock filter model is somewhat outdated, while its work has been incorpo-

rated into other methods, there does not appear to be much point in attempting to improve the original model, as superior sharpening techniques have already been discovered.

Morphological Component Analysis has demonstrated the greatest amount of promise for future work. Relatively new, with many application and capabilities, and development of applications to dynamic textures and video processing only occurring in the last few years, the wealth of possibilities and the quality of the results suggest further research in this area is likely to yield improved models that could be easier to solve or more easily tailored to an application. An area of MCA that needs further work is the dictionary selection. A wider array of dictionaries already confirmed to be suited to representing certain types of noise or texture would ease the task of most MCA algorithm designs. MCA is also still afflicted with a long processing time, short clips can be done by standard computers, but prolonged videos require days or supercomputers to process. While MCA is more about the quality of the results that speed and simplicity, a more efficient solver for the minimisation problem would be another very useful improvement to the model.

Bibliography

[1] S. Alliney. Digital filters as absolute norm regularizers. *Signal Processing, IEEE Transactions on*, 40(6):1548–1562, 1992.

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. 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.

[3] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *Image Processing, IEEE Transactions on*, 12(8):882–889, 2003.

[4] J. Bobin, JL Starck, J. Fadili, Y. Moudden, and DL Donoho. Morphological component analysis: New results. *IEEE Transactions on Image Processing-revised*, 2006.

[5] J. Bobin, J.L. Starck, J.M. Fadili, Y. Moudden, and D.L. Donoho. Morphological component analysis: An adaptive thresholding strategy. *Image Processing, IEEE Transactions on*, 16(11):2675–2681, 2007.

[6] A. Chambolle and P.L. Lions. Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76(2):167–188, 1997.

[7] T.F. Chan and S. Esedoglu. Aspects of total variation regularized l 1 function approximation. *SIAM Journal on Applied Mathematics*, 65(5):1817–1837, 2005.

[8] S. Dubois, R. Péteri, and M. Ménard. Decomposition of dynamic textures using morphological component analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(2):188, 2012.

[9] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[10] M. Elad, J.L. Starck, P. Querre, and D.L. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis (mca). *Applied and Computational Harmonic Analysis*, 19(3):340–358, 2005.

[11] X. Gao, Y. Wang, X. Li, and D. Tao. On combining morphological component analysis and concentric morphology model for mammographic mass detection. *Information Technology in Biomedicine, IEEE Transactions on*, 14(2):266–273, 2010.

[12] R. Geng, Q. Lian, and M. Sun. Fingerprint separation based on morphological component analysis. *Computer Engineering and Applications*, 44(16):188–190, 2008.

[13] S.G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12):3397 –3415, dec 1993.

[14] Y. Meyer. *Oscillating patterns in image processing and nonlinear evolution equations: the fifteenth Dean Jacqueline B. Lewis memorial lectures*, volume 22. Amer Mathematical Society, 2001.

[15] S. Osher and L.I. Rudin. Feature-oriented image enhancement using shock filters. *SIAM Journal on Numerical Analysis*, 27(4):919–940, 1990.

[16] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, 1990.

[17] L.I. Rudin. Images, numerical analysis of singularities and shock filters. 1987.

[18] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[19] J.L. Starck, D.L. Donoho, and E.J. Candès. Astronomical image representation by the curvelet transform. *Astronomy and Astrophysics*, 398(2):785–800, 2003.

[20] J.L. Starck, Y. Moudden, J. Bobin, M. Elad, and DL Donoho. Morphological component analysis. In *Proceedings of the SPIE conference wavelets*, volume 5914, page 191. Citeseer, 2005.

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

[22] W. Yin, D. Goldfarb, and S. Osher. A comparison of three total variation based texture extraction models. *Journal of Visual Communication and Image Representation*, 18(3):240–252, 2007.

[23] W. Yin, D. Goldfarb, and S. Osher. The total variation regularized l. [1] model for multiscale decomposition. *Multiscale Modeling & Simulation*, 6(1):190–211, 2007.