Universität des Saarlandes

Bachelor Thesis

Feature Selection for Diffusion-based Inpainting

Author:
Daniel Gusenburger

Supervisor:
Prof. Joachim Weickert

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Mathematical Image Analysis Group
Department of Computer Science

March 9, 2020

UNIVERSITÄT DES SAARLANDES

Abstract

Faculty of Mathematics and Computer Science
Department of Computer Science

Bachelor of Science

Feature Selection for Diffusion-based Inpainting

by Daniel Gusenburger

Contents

Abstract iii

1 Motivation 1

2 Fundamentals 3

2.1 Mathematical concepts 3

2.2 Corner detection methods 6

2.3 Inpainting 7

3 Implementation 9

4 Results 11

5 Outlook 13

Chapter 1

Motivation

Chapter 2

Fundamentals

In this chapter I will go over the fundamental mathematical concepts fom image processing used in my thesis. We will see the basic ideas of diffusion based inpainting as well as take a somewhat deeper look at the most relevant methods of corner/feature detection.

2.1 Mathematical concepts

Talking about corner detection and inpainting requires some basic knowledge of multivariable calcus so I am going to introduce the fundamental and most important concepts here.

First, we have to define what an image is mathematically. Most of the time in image processing, a (grey value) image is defined as a continuous function $f : \Omega \to [0, 255]$ where the *image domain* $[0, n_x] \times [0, n_y] =: \Omega \subset \mathbb{R}^2$ is a 2-dimensional cuboid in $\mathbb{R}^2$. In reality however, images are not really continuous functions but rather discrete sets of pixels. These pixels are normally assumed to lie on a rectangular equidistant grid, where the distance between each pixel is h_x in x-direction and h_y in y-direction. In our case, the distances in both directions are equal, making it a quadratic equidistant grid with grid size $h := h_x = h_y$. We define points on the grid as

$$\mathbf{x}_{i,j} = (x_i, y_j) = (ih, jh) \qquad \forall (i, j) \in [0, n_x - 1] \times [0, n_y - 1] \tag{2.1}$$

The pixels that are known are then defined as

$$f_{i,j} := f(\mathbf{x}_{i,j}) = f(x_i, y_j) \qquad \forall (i, j) \in [0, n_x - 1] \times [0, n_y - 1] \tag{2.2}$$

Partial derivatives and the Gradient

NEEDS WORKING ON !!!!

The partial derivative of a continuous function in multiple variables is roughly

defined as differentiating this function in one variable while keeping the remaining variables constant. Geometrically, this can be seen as finding the slope of a surface in a certain direction. More formally, the partial derivative of a continuous function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$\frac{\partial f}{\partial x}(x, y) = f_x(x, y) := \lim_{h \to 0} \frac{f(x + h, y) - f(x, y)}{h} \tag{2.3}$$

The partial derivative in y-direction is defined analogously. Using these definitions, one can define the so called *gradient* of f as

$$\text{grad} f(x, y) = \nabla f(x, y) := \left(\frac{\partial f}{\partial x}(x, y), \ \frac{\partial f}{\partial x}(x, y) \right)^\top \tag{2.4}$$

As we will see later, the gradient plays a very important role in corner detection algorithms. How ever, as we recall, images typically do not have an infinite amount of pixels, meaning that they can not be differentiated using the existing data. To still be able to compute the gradient we have to resort to approximating it numerically.

Numerical differentiation

Schemes for numerical differentiation are derived from the *Taylor polynomial* of the respective function. For the sake of simplicity and readability, I'm going to derive such a scheme for a one-dimensional function as it is basically the same for a multivariable function.

Let $f : (a, b) \to \mathbb{R}$ be a $n + 1$ times continuously differentiable function, i.e. $f \in C^{n+1}$ and $x_0 \in (a, b)$. The Taylor polynomial of degree n for f is then defined as

$$T_n(x, x_0) := \sum_{k=0}^{m} \frac{(x - x_0)^k}{k!} f^{(k)}(x_0) \tag{2.5}$$

Recalling Taylor's theorem, $f(x)$ can be approximated using this polynomial, leaving an error of

$$R_n(x, x_0) := \frac{(x - x_0)^{n+1}}{(n + 1)!} f^{(n+1)}(x_0 + \Theta(x - x_0)) \qquad \Theta \in (0, 1) \tag{2.6}$$

Now, let u be a function from which we only know some points on a grid with grid size h as defined in 2.2. To approximate the first derivative of u at position i we want to find some coefficients α, β, γ such that

$$u_i' \approx \alpha u_{i-1} + \beta u_i + \gamma u_{i+1} \tag{2.7}$$

Using 2.2 and 2.5 we can derive an approximation for u_{i-1} and u_{i+1}:

$$u_{i-1} \approx u_i - hu'_i + \frac{1}{2}h^2u''_i \tag{2.8}$$

$$u_{i+1} \approx u_i + hu'_i + \frac{1}{2}h^2u''_i \tag{2.9}$$

Inserting 2.8 and 2.9 in 2.7, we get the following term after sorting the coefficients:

$$u'_i \approx (\alpha + \beta + \gamma)u_i - h(\alpha - \gamma)u'_i + \frac{h^2}{2}(\alpha + \gamma)u''_i \tag{2.10}$$

Solving the arising linear system gives us the following coefficients:

$$\alpha = \frac{1}{2h} = -\gamma \tag{2.11}$$

$$\beta = 0 \tag{2.12}$$

And therefore the approximation

$$u'_i \approx \frac{u_{i+1} - u_{i-1}}{2h} \tag{2.13}$$

which is called a *central difference scheme*

Convolution

The Structure Tensor

Another very important piece of mathematics is the so called *structure tensor*. It serves as an operator that yields information about the surroundings of a certain location. One can for example by just looking at the eigenvalues of this tensor figure out whether the current location belongs to a corner or not. Hence it is used in a large quantity of corner detection algorithms. More on the exact nature of the algorithms can be found in the next section.
Since differentiation is inherently unstable, it is good practice to first smooth the image before computing the derivatives to get rid of high frequencies and noise. That means, replacing the original image f by a smoothed version u given by a convolution with a gaussian kernel K_σ where σ denotes the standard deviation of the Gaussian.

$$u := (K_\sigma * f) \tag{2.14}$$

Putting it all together, the structure tensor for the smoothed image u looks like this:

$$J_\rho = K_\rho * \left(\nabla u \nabla u^\top\right) = \begin{pmatrix} K_\rho * u_x^2 & K_\rho * u_x u_y \\ K_\rho * u_x u_y & K_\rho * u_y^2 \end{pmatrix} \tag{2.15}$$

The second gaussian convolution with the so called *integration scale* ρ is added so that the structure tensor incorporates more information about the surrounding region.
The eigenvalues of the structure tensor have some very interesting and unique properties that help greatly in distinguishing corners from non-corners.

Diffusion

The easiest way to explain diffusion (in this case *homogeneous* or *linear* diffusion) is to say that it describes the way a drop of ink propagates in a glass of water. More generally, diffusion describes how the concentration of some substance (?) changes over time. In a mathematical way, a simple form of diffusion is given by the *partial differential equation* (PDE)

$$u_t = u_{xx} \tag{2.16}$$

in the one dimensional case and

$$u_t = \Delta u := u_{xx} + u_{yy} \tag{2.17}$$

in the higher dimensional case. Δu is called the *Laplace operator*.

2.2 Corner detection methods

There are many different ways to find corners in an image. The approach I chose computes the components of the structure tensor at each position in the image and then uses some kind of measure to evaluate the *cornerness* at each location. Two measures I used to find relevant features are the *Foerstner-Harris* (2.18) and the *Tomasi-Kanade* (2.19) measures

$$\text{Harris}(J) = \frac{det(J)}{tr(J)} = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} \tag{2.18}$$

$$\text{Tomasi}(J) = min(\lambda_1, \lambda_2) \tag{2.19}$$

where λ_1 and λ_2 are the eigenvalues of the structure tensor. While the Tomasi measure explicitly computes the eigenvalues to find the smallest one,

which can be pretty expensive, especially since it has to be done for each pixel, the Harris measure only approximates the smallest eigenvalue.

2.3 Inpainting

Cries in painting.

Chapter 3

Implementation

Chapter 4

Results

Chapter 5

Outlook