Real-time Video Alignment and Fusion Using Feature Detection on FPGA Devices

A Thesis

Submitted to the Faculty

of

Drexel University

by

Robert Haywood Taglang

in partial fulfillment of the

requirements for the degree

of

Master of Science in Computer Engineering

June 2017

Table of Contents

List of Tables

List of Figures

Abstract

Real-time Video Alignment and Fusion Using Feature Detection on FPGA Devices
Robert Haywood Taglang
Prawat Nagvajara, Ph.D.

Video fusion functions as a way to combine the important or useful parts of two or more sequences of images. The scenario presented is the use of Laplacian fusion to produce a single video composed of the fields of view of two cameras whose areas of focus differ substantially. This is not a useful real-time strategy unless the frames can be aligned. This thesis presents a system for detecting features using an FPGA implementation of SURF (Speeded-Up Robust Features), and aligning video streams by applying a transform generated from the key features.

1 Background

1.1 Introduction

The fusion of data from two or more sensors has been well-researched *citations needed*, though these approaches typically discuss the process of fusing images which have been pre-aligned. Pre-computed transforms used to align the frames of two cameras are not robust to variations. Some approaches have made use of additional hardware sensors in order to correct against these variations [3]. The approach presented in this thesis seeks to perform this correction completely in hardware using feature detection on a FPGA.

The design choice to use a FPGA rather than a GPU or some other software based approach was made due to the advantages gained from operating with embedded hardware, namely higher portability and lower overall power consumption.

1.2 Laplacian Fusion

Laplacian pyramids of images have their origin as a strategy for image encoding [2]. A gaussian blur is applied to the image, and the image is downsampled to half of its original size. This process can be repeated on the resulting image to create a sequence of images representing the original in different scale spaces. This sequence of blurred and downsampled images is known as a Gaussian pyramid. An illustration of a Guassian pyramid for an example image can be seen in Figure 1.

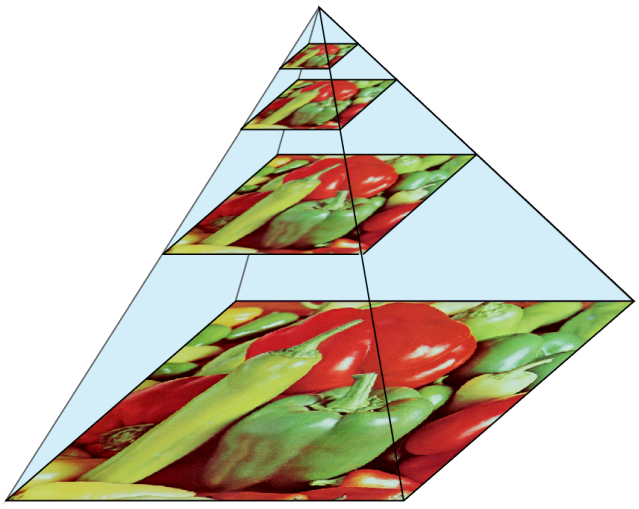


Figure 1: Gaussian pyramid of an example image

The Laplacian pyramid is one which can be used for reconstruction of the original image.

At each level above the lowest level of the Gaussian pyramid, the level below is upsampled to match the scale of the current level. The difference between the upsampled image and the current scale level image is known as the Laplacian of the image. The sum of the upsampled lower level and the Laplacian is the original image. At a single level, the Laplacian can be thought of as the error introduced by applying a Gaussian and Box filter. A diagram illustrating a Laplacian pyramid can be seen in Figure 2

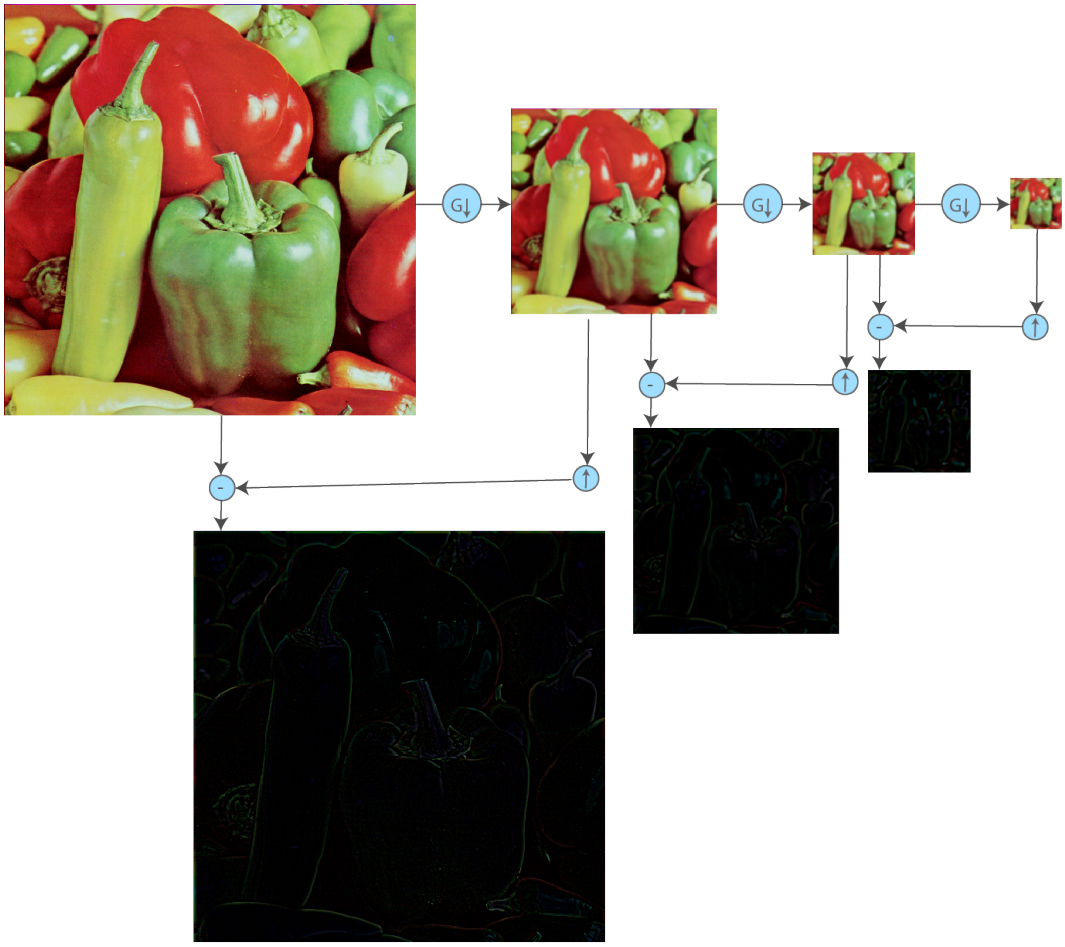


Figure 2: Laplacian pyramid of an example image

The property of the Laplacian that makes it ideal for fusion is its ability to capture the high frequency components in an image through the use of very simple kernel operators that are easily implemented in hardware. The difference between a blurred image and the original will have higher magnitude in the areas where the image was sharpest.

The fusion of two images can be thought of as a function of the two images X and Y of dimension $M \times N$ where $Z = f(X, Y)$, a single image of dimension $M \times N$. A naive approach to fusion would be to compute the Laplacians and use their magnitudes to select

a pixel from either X or Y as shown in Equation 1.

$$Z(i,j) = \begin{cases} X(i,j) & |L(X(i,j))| \geq |L(Y(i,j))| \\ Y(i,j) & otherwise \end{cases} \tag{1}$$

This approach does not account for variations in colorspace between the two images. Consider the images in Figure 3. The more saturated image will likely have a higher valued Laplacian in some parts simply because it is brighter, therefore having higher magnitudes at individual pixels. This approach also will not facilitate smooth stitching of the images. Contiguous regions of selection from one image will be adjacent to regions from the other with no transition, producing a grainy effect at areas of high frequency. The result of this naive fusion can be seen in Figure 4a which exhibits the flaws of this approach.

(a) Blurred on the left; more saturated (b) Blurred on the right; more neutral

Figure 3: Two images of the same scene with variations in sharpness and colorspace

A more correct approach would involve using the Laplacian in a weighted sum to combine the pixels of the images, rather than simply selecting them, as shown in Equation 2.

$$Z(i,j) = \frac{|L(X(i,j))|}{|L(X(i,j))| + |L(Y(i,j))|} \cdot X(i,j) + \frac{|L(Y(i,j))|}{|L(X(i,j))| + |L(Y(i,j))|} \cdot Y(i,j) \tag{2}$$

The result of this weighted sum approach can be seen in Figure 4b which exhibits a reduction in graininess from the naive approach.

(a) Fusion using the naive approach (b) Fusion using weighted sum

Figure 4: Fusion of the images in Figure 3 using a naive selection and weighted sum approach

1.3 Speeded-up Robust Features (SURF)

The generation of features for use as marker points in alignment utilizes the SURF algorithm from Bay et al [1]. SURF is composed of two parts: a discrete approximation for computing Hessian determinants, and the generation of rotation invariant feature descriptors for detected feature points.

SURF is typically used for its applications in object recognition, where the feature descriptor is used to facilitate a match between what is observed and some known set of feature points and descriptors. The descriptor largely serves as a way of discriminating against false positives. In terms of using SURF for fusion, the detected feature points will be matched across two images with the assumption that the subject is the same and that the images do contain spatially coherent matches. Given this assumption, it can be concluded that the feature descriptor is not necessary for alignment, only the feature points computed using Hessian determinants.

1.3.1 Computation of Hessian Determinants

The Hessian determinant is the determinant of a Hessian matrix, which is a matrix composed of the spatial partial second derivatives of a function $f : \mathbb{R}^n \to \mathbb{R}$. It is of the general form shown in Equation 3. In the image domain, $f : \mathbb{R}^2 \to \mathbb{R}$. The particular form of the Hessian matrix in $\mathbb{R}^2$ with dimensions x_1 and x_2 is shown in Equation 4.

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{3}$$

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} \tag{4}$$

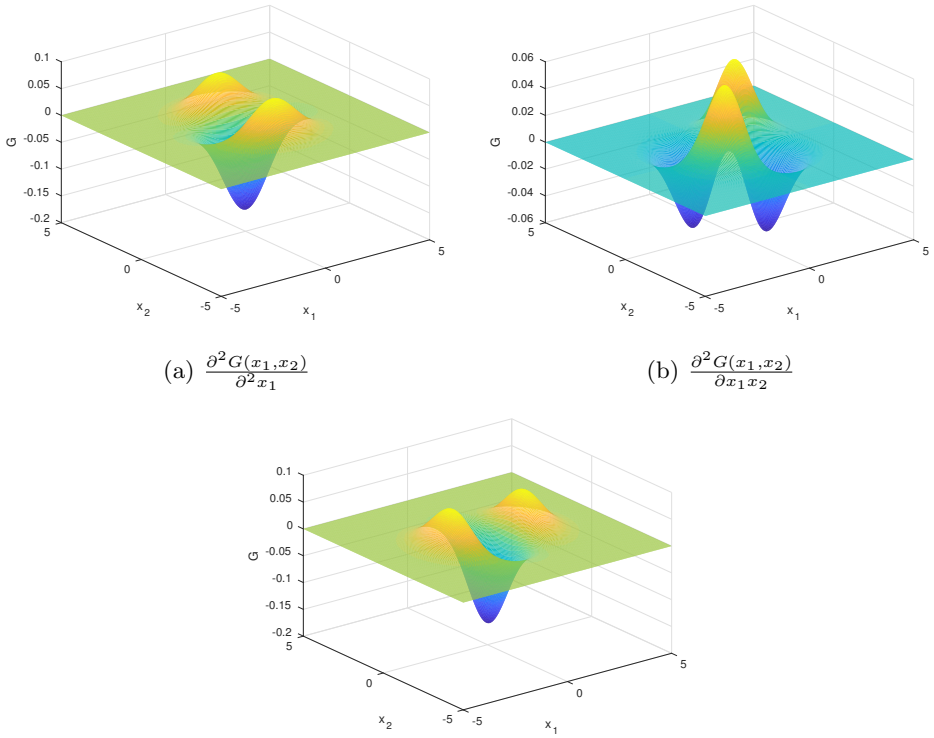
The second order derivative used can be computed via convolution of the Gaussian second order derivative at any point, x, in the image. The formulas for the Gaussian second order derivative for each partial with respect to x_1^2, $x_1 x_2$ and x_2^2 can be seen in Equations 5, 6, and 7 respectively.

$$\frac{\partial^2 G(x_1, x_2, \sigma)}{\partial^2 x_1} = (-1 + \frac{x_1^2}{\sigma^2}) \frac{e^{-\frac{x_1^2 + x_2^2}{2\sigma^2}}}{2\pi\sigma^4} \tag{5}$$

$$\frac{\partial^2 G(x_1, x_2, \sigma)}{\partial x_1 x_2} = \frac{x_1 x_2}{2\pi\sigma^6} e^{-\frac{x_1^2 + x_2^2}{2\sigma^2}} \tag{6}$$

$$\frac{\partial^2 G(x_1, x_2, \sigma)}{\partial^2 x_2} = (-1 + \frac{x_2^2}{\sigma^2}) \frac{e^{-\frac{x_1^2 + x_2^2}{2\sigma^2}}}{2\pi\sigma^4} \tag{7}$$

3D surface plots of these equations where $\sigma = 1$ can be seen in Figure 5. In order to compute these functions quickly, SURF approximates them with cropped, discrete, kernels.



(a) $\frac{\partial^2 G(x_1,x_2)}{\partial^2 x_1}$ (b) $\frac{\partial^2 G(x_1,x_2)}{\partial x_1 x_2}$

(c) $\frac{\partial^2 G(x_1,x_2)}{\partial^2 x_2}$

Figure 5: 3D surface plots of the Gaussian second order derivative functions where $\sigma = 1$

1.3.2 SURF Implementations for FPGA Devices

1.4 Iterative Closest Point Algorithm

1.5 Singular Value Decomposition(SVD)

1.5.1 SVD Implementations for FPGA Devices

2 Implementation

2.1 Streaming Kernel Operators

2.1.1 Hessian Kernel with Accumulator

2.1.2 Box Kernel Approximation of Single Level Laplacian Pyramid

2.2 Computation of Transform from Detected Features

2.3 Application of Transform to Real-Time Data

3 Results

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer visionECCV 2006*, pages 404–417, 2006.

[2] P. Burt and E. Adelson. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, 31(4):532–540, April 1983.

[3] S. Chappell, A. Macarthur, D. Preston, D. Olmstead, B. Flint, and C. Sullivan. Exploiting Real-time FPGA Based Adaptive Systems Technology for Real-time Sensor Fusion in Next Generation Automotive Safety Systems. In *The IEE Seminar on Target Tracking: Algorithms and Applications 2006 (Ref. No. 2006/11359)*, pages 61–68, March 2006.