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 [11] [9], 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 [4]. 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 [3]. 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.

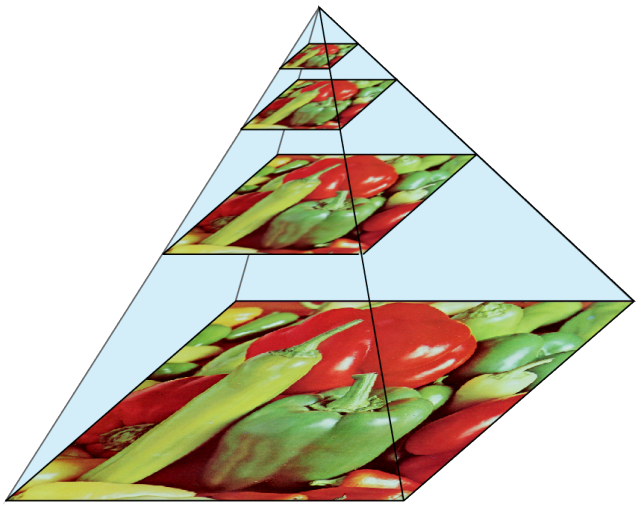


Figure 1: Gaussian pyramid of an example image

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

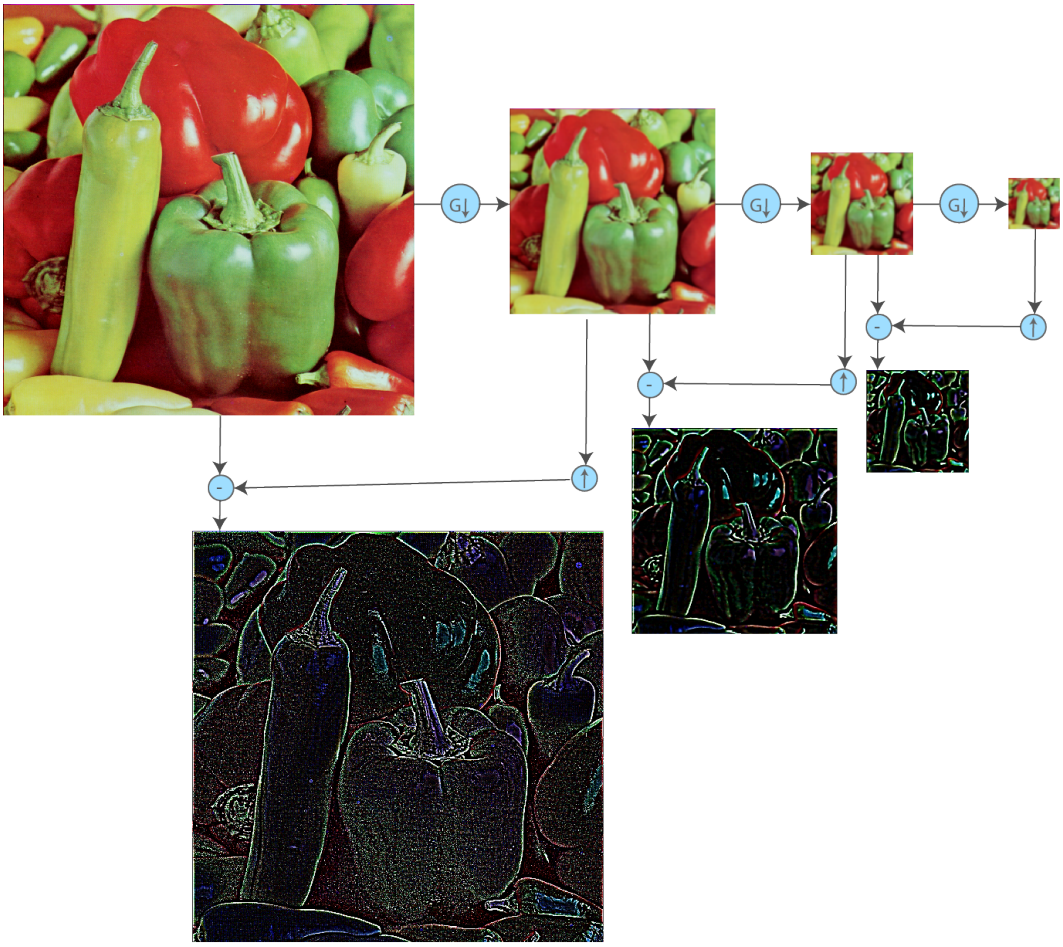


Figure 2: Laplacian pyramid of an example image

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 [2]. 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.

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

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

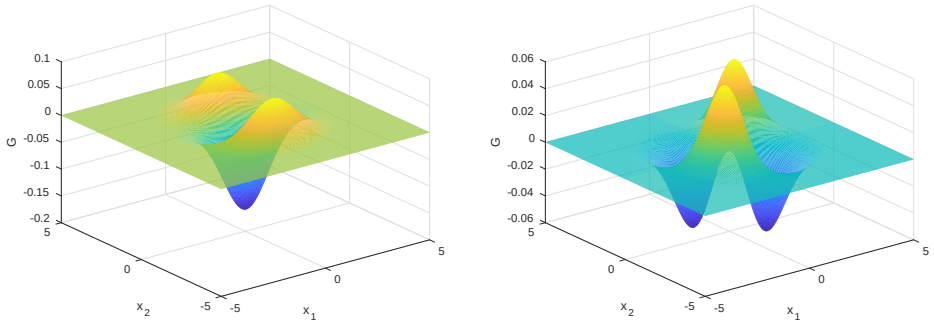
$$\frac{\partial^2 G(x_1, x_2, \sigma)}{\partial^2 x_2} = (-1 + \frac{x_2^2}{\sigma^2}) \frac{e^{-\frac{x_1^2 + x_2^2}{2\sigma^2}}}{2\pi\sigma^4} \tag{7}$$

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.

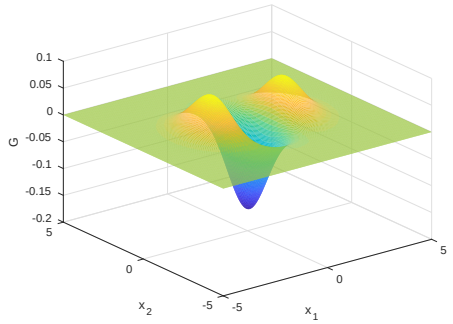
The Gaussian second order derivatives can be approximated as 9×9 kernels with $\sigma = 1.2$. These kernels can be seen in Figure 6. By adjusting σ, Hessian determinants can be computed in different scale spaces. This is a concept that SURF draws from the Scale-Invariant Feature Transform (SIFT) from Lowe et al [10].

By detecting features in different scale spaces, SIFT and by proxy, SURF are robust to changes in scale. SIFT accomplished this by downsampling the image to detect at lower order scale spaces. SURF improved on this approach for speed by instead scaling the kernel.

Another speed optimization presented in SURF takes advantage of the form of the discrete kernels. Since the approximated kernels are composed of rectangles of constant value, they can be decomposed into a set of box filters. Box filters can be computed quickly with the use of integral images. The general form of the integral image of an $M \times N$ image I is



(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 $\sigma = 1$

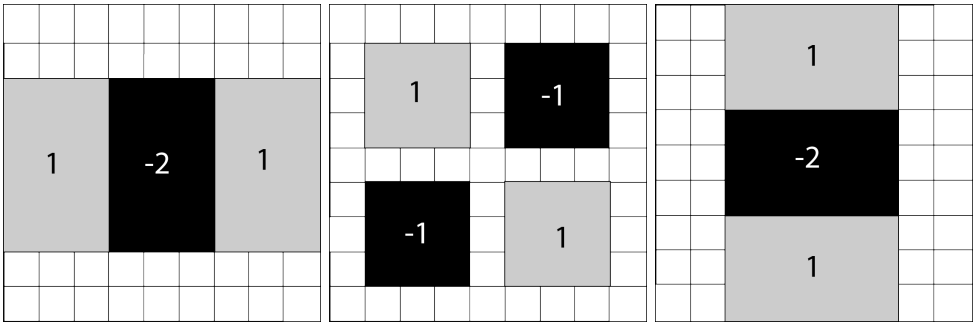
shown in Equation 8.

$$\int I = \begin{bmatrix} I(0,0) & \sum_{m=0}^{1} I(0,m) & \cdots & \sum_{m=0}^{M} I(0,m) \\ \sum_{n=0}^{1} I(n,0) & \sum_{n=0}^{1}\sum_{m=0}^{1} I(n,m) & & \vdots \\ \vdots & & \ddots & \vdots \\ \sum_{n=0}^{N} I(n,0) & \cdots & \cdots & \sum_{n=0}^{N}\sum_{m=0}^{M} I(n,m) \end{bmatrix} \tag{8}$$

Integral images decrease the computational complexity of finding the sum of an area in the input image. The computational complexity of strict kernel convolution at each point scales with the size of the kernel and is of the order $O(N^2)$ where the kernel is $N \times N$. A box filter can be decomposed into finding the sum of an area in the image and scaling it. Finding the sum from the integral image can be performed in $O(1)$ as shown in Equation 9.

$$\sum_{n=w}^{y}\sum_{m=x}^{z} = \int I(y,z) - \int I(w-1,z) - \int I(y,x-1) + \int I(w-1,x-1) \tag{9}$$

This makes the computation of the Hessian matrix scale only in terms of the image size,



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

Figure 6: 9×9 discrete approximations of the Gaussian second order derivative with $\sigma = 1.2$

yielding no additional penalty for operating on different scale spaces.

1.3.2 SURF Implementations for FPGA Devices

The relatively low computational complexity makes SURF a popular choice for FPGA applications. Battezzati et al. present an architecture using accumulators for computing the integral image pipelined through the Hessian computation and storing detected feature points in a FIFO cache [1]. These are matched against a stored set of feature points. Chen et al. present improvements on this approach by parallelizing the computation of different scale spaces [5]. The implementation in this thesis follows these approaches with some additional improvements for speed based on the use case of matching against another image rather than a stored set of features.

1.4 Iterative Closest Point Algorithm

The crucial step in aligning the two images is the computation of an affine transform mapping one image into the space of the other. Once the images are aligned, they can be fused. A combination of SURF and the iterative closest point algorithm are used to compute this transform. Iterative closest point was first proposed by Chen and Medioni as a method for aligning 3-D point cloud data [6]. In its simplest form, the algorithm follows the following steps:

1. Each point in the set of points to be transformed is matched against the closest (usually Euclidean distance) point in the reference set of points.

2. A transformation is estimated to minimize the distance between the transform set and their matches in the reference set of points.

3. The transform is applied to the points.

This process is repeated, converging to the local minimum that is the match between the two point sets.

Let $X_p(i)$ be be the i^{th} point in the set of reference points onto which the transform points, $X(j)$, where j is the index of the closest point to $X_p(i)$ in X, will be projected. The general form for this transformation M is shown in Equation 10, and the expanded matrix form can be seen in Equation 11. In the expanded matrix form M is decomposed into R, a 2×2 rotation matrix, and T, a translation offset.

$$X_p(i) = X(j) \cdot M \tag{10}$$

$$\begin{bmatrix} X_p(i)_1 \\ X_p(i)_2 \\ 1 \end{bmatrix} = \begin{bmatrix} X(j)_1 & X(j)_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_{11} & R_{12} & T_1 \\ R_{21} & R_{22} & T_2 \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

This equation only solves for M for a single point relation, but can be restructured to contain the whole set of N points as shown in Equation 12. In this equation P is a function $P : i \to j$ mapping the closest points in each set.

$$\begin{bmatrix} X_p(0)_1 \\ X_p(0)_2 \\ X_p(1)_1 \\ X_p(1)_2 \\ \vdots \\ X_p(N)_1 \\ X_p(N)_2 \end{bmatrix} = \begin{bmatrix} X(P(0))_1 & X(P(0))_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X(P(0))_1 & X(P(0))_2 & 1 \\ X(P(1))_1 & X(P(1))_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X(P(1))_1 & X(P(1))_2 & 1 \\ \vdots & & & & & \vdots \\ X(P(N))_1 & X(P(N))_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X(P(N))_1 & X(P(N))_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_{11} \\ R_{12} \\ T_1 \\ R_{21} \\ R_{22} \\ T_2 \end{bmatrix} \tag{12}$$

Solving for M in this way maps all points in X to their closest points in X_p with the error minimized in a least-squares sense.

In this form, the transform M will include shear transformations and non-uniform scaling. The computation can be simplified by forcing the second basis vector in R to be orthogonal to the first. The two cameras are expected to be be physically in the same plane, and as such, non-uniform scaling and shear transformations are not expected for

alignment. By setting $R_{21} = -R_{12}$ and $R_{22} = R11$, the computation of M can be reduced as shown in Equation 13.

$$\begin{bmatrix} X_p(0)_1 \\ X_p(0)_2 \\ X_p(1)_1 \\ X_p(1)_2 \\ \vdots \\ X_p(N)_1 \\ X_p(N)_2 \end{bmatrix} = \begin{bmatrix} X(P(0))_1 & X(P(0))_2 & 1 & 0 \\ X(P(0))_2 & -X(P(0))_1 & 0 & 1 \\ X(P(1))_1 & X(P(1))_2 & 1 & 0 \\ X(P(1))_2 & -X(P(1))_1 & 0 & 1 \\ \vdots & & & \vdots \\ X(P(N))_1 & X(P(N))_2 & 1 & 0 \\ X(P(N))_2 & -X(P(N))_1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_{11} \\ R_{12} \\ T_1 \\ T_2 \end{bmatrix} \tag{13}$$

The performance of iterative closest point can be further improved by making it more sensitive to errors. Chetverikov introduces a variant of iterative closest point referred to as trimmed iterative closest point (TrICP) [7]. TrICP is more robust to errors by eliminating points that introduce error into the matching. Some detected features points will not have correspondences between images. By eliminating these points, the overall error can be reduced to compute a more accurate transform.

1.5 Singular Value Decomposition (SVD)

Let Equation 13 be of the form $X = Q \cdot M$. In order to compute compute the transform M, the equation must be restructured as in Equation 14.

$$Q^{-1}X = M \tag{14}$$

Q is not a square matrix, and as such is not invertible, but its pseudoinverse can be used in this instance. The pseudoinverse is computed through the use of singular value decomposition. The use of singular value decomposition to compute the transform is the source of the least squares fitting achieved by iterative closest point.

1.5.1 SVD Implementations for FPGA Devices

Ledesma-Carillo et al. present a hardware efficient algorithm for computing singular value decompositions on large matrices uisng one-sided Jacobi rotations [8].

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] N. Battezzati, S. Colazzo, M. Maffione, and L. Senepa. SURF algorithm in FPGA: A novel architecture for high demanding industrial applications. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 161–162, March 2012.

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

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

[4] 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.

[5] W. Chen, S. Ding, Z. Chai, D. He, W. Zhang, G. Zhang, Q. Peng, and W. Luo. FPGA-Based Parallel Implementation of SURF Algorithm. In *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pages 308–315, December 2016.

[6] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, April 1992.

[7] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. The Trimmed Iterative Closest Point algorithm. In *Object recognition supported by user interaction for service robots*, volume 3, pages 545–548 vol.3, 2002.

[8] L. M. Ledesma-Carrillo, E. Cabal-Yepez, R. d J. Romero-Troncoso, A. Garcia-Perez, R. A. Osornio-Rios, and T. D. Carozzi. Reconfigurable FPGA-Based Unit for Singular Value Decomposition of Large m x n Matrices. In *2011 International Conference on Reconfigurable Computing and FPGAs*, pages 345–350, November 2011.

[9] Hui Li, B. S. Manjunath, and S. K. Mitra. Multi-sensor image fusion using the wavelet transform. In *Proceedings of 1st International Conference on Image Processing*, volume 1, pages 51–55 vol.1, November 1994.

[10] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[11] Wencheng Wang and Faliang Chang. A Multi-focus Image Fusion Method Based on Laplacian Pyramid. *Journal of Computers*, 6(12), December 2011.