

Implementing DSP Algorithms Using Spartan-3 FPGAs

This article presents two case studies of FPGA implementations for commonly used image processing algorithms — feature extraction and digital image warping.

by Paolo Giaccon
Graduate Student
Università di Verona, Italy
paolo.giaccon@students.univr.it

Saul Saggin
Undergraduate Student
Università di Verona, Italy
saul.saggin@students.univr.it

Giovanni Tommasi
Undergraduate Student
Università di Verona, Italy
giovanni.tommasi@students.univr.it

Matteo Busti
Graduate Student
Università di Verona, Italy
matteo.busti@students.univr.it

Computer vision is a branch of artificial intelligence that focuses on equipping computers with the functions typical of human vision. In this discipline, feature tracking is one of the most important pre-processing tasks for several applications, including structure from motion, image registration, and camera motion retrieval. The feature extraction phase is critical because of its computationally intensive nature.

Digital image warping is a branch of image processing that deals with techniques of geometric spatial transformations. Warping images is an important stage in many applications of image analysis, as well as some common applications of computer vision, such as view synthesis, image mosaicing, and video stabilization in a real-time system.

In this article, we'll present an FPGA implementation of these algorithms.

Feature Extraction Theory

In many computer vision tasks we are interested in finding significant feature points — or more exactly, the corners. These points are important because if we measure the displacement between features in a sequence of images seen by the camera, we can recover information both on the structure of the environment and on the motion of the viewer.

Figure 1 shows a set of feature points extracted from an image captured by a camera. Corner points usually show a significant change of the gradient values along the two directions (x and y). These points are of interest because they can be

uniquely matched and tracked over a sequence of images, whereas a point along an edge can be matched with any number of other points on the edge in a second image.

The Feature Extraction Algorithm

The algorithm employed to select good features is inspired by Tomasi and Kanade's method, with the Benedetti and Perona approximation, considering the eigenvalues α and β of the image gradient covariance matrix. The gradient covariance matrix is given by:

$$H = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$

where I_x and I_y denote the image gradients in the x and y directions.

Hence we can classify the structure around each pixel observing the eigenvalues of H:

- No structure : $\alpha \approx \beta \approx 0$
- Edge : $\alpha \approx 0, \beta \gg 0$
- Corner : $\alpha \gg 0, \beta \gg 0$



Figure 1 – Feature points extracted from an image captured by a camera

Using the Benedetti and Perona approximation, we can choose the corners without computing the eigenvalues.

We have realized an algorithm that, compared to the original method, doesn't require any floating-point operations. Although this algorithm can be implemented either in hardware or software, by implementing it in FPGA technology we can achieve real-time performance.

Input:

- 8-bit gray-level image of known size (up to 512 x 512 pixels)
- The expected number of feature points (wf)

Output:

- List of selected features (FL). The type of the output is a 3 x N matrix whose:
 - First row contains the degrees of confidence for each feature in the list
 - Second row contains the x-coordinates of the feature points
 - Third row contains the y-coordinates of the feature points

Semantic of the Algorithm

In order to determine if a pixel (i, j) is a feature point (corner), we followed Tomasi and Kanade's method.

First, we calculate the gradient of the image. Hence the 2 x 2 symmetric matrix $G = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$ is computed, whose entries derive from the gradient values in a patch around the pixel (i, j).

If the minimum eigenvalue of G is greater than a threshold, then the pixel (i, j)

is a corner point. The minimum eigenvalue is computed using an approximation to avoid the square root operation that is expensive for hardware implementations.

The corner detection algorithm could be summarized as follows:

The image gradient is computed by mean of convolution of the input image with a predefined mask. The size and the values of this mask depend on the image resolution. A typical size of the mask is 7 x 7.

- For each pixel (i, j) loop:

$$a_{i,j} = \sum_k^N (I_x^k)^2$$

$$b_{i,j} = \sum_k^N I_x^k I_y^k$$

$$c_{i,j} = \sum_k^N (I_y^k)^2$$

where N is the number of pixels in the patch and I_x^k and I_y^k are the components of the gradient at pixel k inside the patch.

$$P_{i,j} = (a - t)(c - t) - b^2$$

where t is a fixed integer parameter.

- If ($P_{i,j} > 0$) and ($a_{i,j} > t$), then we retain pixels (i,j)
- Discard any pixel that is not a local maximum of $P_{i,j}$
- End loop
- Sort, in decreasing order, the feature list FL based on the degree of confidence values and take only the first wf items.

Implementation

With its high-speed embedded multipliers, the Xilinx® Spartan™-3 architecture meets the cost/performance characteristics required by many computer vision systems that could take advantage of this algorithm.

The implementation is divided into four fundamental tasks:

1. Data acquisition. Take in two gradient values along the x and y axis and

compute for each pixel three coefficients used by the characteristic polynomial. To store and read the gradient values, we use a buffer (implemented using a Spartan-3 block RAM).

2. Calculation of the characteristic polynomial value. This value is important to sort the features related to the specific pixel. We implemented the multiplications used for the characteristic polynomial calculus employing the embedded multipliers on Spartan-3 devices.
3. Feature sorting. We store computed feature values in block RAM and sort them step by step by using successive comparisons.
4. Enforce minimum distance. This is done to keep a minimum distance between features; otherwise we get clusters of features heaped around the most important ones. This is implemented using block RAMs, building a non-detect area around each most important feature where other features will not be selected.

Spartan-3 Theoretical Performance

The algorithm is developed for gray-level images at different resolutions, up to 512 x 512 at 100 frames per second.

The resources estimated by Xilinx System Generator are:

- 1,576 slices
- 15 block RAMs
- 224 LUTs
- 11 embedded multipliers

The embedded multipliers and extensive memory resources of the Spartan-3 fabric allow for an efficient logic implementation.

Applications of Feature Extraction

Feature extraction is used in the front end for any system employed to solve practical control problems, such as autonomous navigation and systems that could rely on vision to make decisions and provide control. Typical applications include active video surveillance, robotic arms motion,

measurement of points and distances, and autonomous guided vehicles.

Image Warping Theory

Digital image warping deals with techniques of geometric spatial transformations.

The pixels in an image are spatially represented by a couple of Cartesian coordinates (x, y). To apply a geometric spatial transformation to the image, it is convenient to switch to homogeneous coordinates, which allow us to express the transformation by a single matrix operation. Usually this is done by adding a third coordinate with value 1 (x, y, 1).

In general, such transformation is represented by a non-singular 3 x 3 matrix H and applied through a matrix-vector multiplication to the pixel homogeneous coordinates:

$$\begin{bmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} H_{1,1}x + H_{1,2}y + H_{1,3} \\ H_{2,1}x + H_{2,2}y + H_{2,3} \\ H_{3,1}x + H_{3,2}y + H_{3,3} \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} x'/w' \\ y'/w' \\ 1 \end{bmatrix} \Rightarrow (x'/w', y'/w') \quad (1)$$

The matrix H, called homography or collineation, is defined up to a scale factor (it has 8 degrees of freedom). The transformation is linear in projective (or homogeneous) coordinates, but non-linear in Cartesian coordinates.

The formula implies that to obtain Cartesian coordinates of the resulting pixel we have to perform a division, an operation quite onerous in terms of time and area consumption on an FPGA. For this reason, we considered a class of spatial transformations called "affine transformations" that is a particular specialization of homography. This allows us to avoid the division and obtain good observational results:

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} A_{1,1}x + A_{1,2}y + A_{1,3} \\ A_{2,1}x + A_{2,2}y + A_{2,3} \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \Rightarrow (x', y') \quad (2)$$

Affine transformations include several planar transformation classes as rotation, translation, scaling, and all possible combinations of these. We can summarize the affine transformation as every planar transformation where the parallelism is pre-

served. Six parameters are required to define an affine transformation.

Image Warping Algorithms

There are two common ways to warp an image:

- Forward mapping
- Backward mapping

Using forward mapping, the source image is scanned line by line and the pixels are copied to the resulting image, in the position given by the result of the linear system shown in equation (2). This technique is subject to several problems, the most important being the presence of holes in the final image in the case of significant modification of the image (such as rotation or a scaling by a factor greater than 1) (Figure 2).

The backward mapping approach gives

better results. Using the inverse transformation A^{-1} , we scan the final image pixel by pixel and transform the coordinates. The result is a pair of non-integer coordinates in the source image. Using a bilinear interpolation of the four pixel values identified in the source image, we can find a value for the final image pixel (see Figure 3).

This technique avoids the problem of holes in the final image, so we adopted it as our solution for the hardware implementation.

Implementation

Software implementations of this algorithm are well-known and widely used in applica-

tions where a personal computer or workstation is required. A hardware implementation requires further work to achieve efficiency constraints on an FPGA.

Essentially, the process can be divided in two parts: transformation and interpo-

lation. We implemented the first as a matrix-vector multiplication (2), with four multipliers and four adders. The second is an approximation of the real result of the interpolation: we weighted the four pixel values approximating the results of the transformation with two bits after the binary point. Instead of performing the calculations given by the formula, we used a LUT to obtain the pixel final value, since we divided possible results of the interpolation into a set of discrete values.

Spartan-3 Theoretical Performance

We designed the algorithm using System Generator for DSP, targeting a Spartan-3 device. We generated the HDL code and synthesized it with ISE™ design software, obtaining a resource utilization of:

- 744 slices (1,107 LUTs)
- 164 SRL16
- 4 embedded multipliers

The design can process up to 46 fps (frames per second) with 512 x 512 images. Theoretical results show a boundary of 360+ fps in a Spartan-3-based system.

Applications of Image Warping

Image warping is typically used in many common computer vision applications, such as view synthesis, video stabilization, and image mosaicing.

Image mosaicing deals with the composition of sequence (or collection) of images after aligning all of them respective to a common reference frame. These geometrical transformations can be seen as simple relations between coordinate systems.

By applying the appropriate transformations through a warping operation and merging the overlapping regions of a warped image, we can construct a single panoramic image covering the entire visible area of the scene. Image mosaicing provides a powerful way to create detailed three-dimensional models and scenes for virtual reality scenarios based on real imagery. It is employed in flight simulators, interactive multi-player games, and medical image systems to construct true scenic panoramas or limited virtual environments.

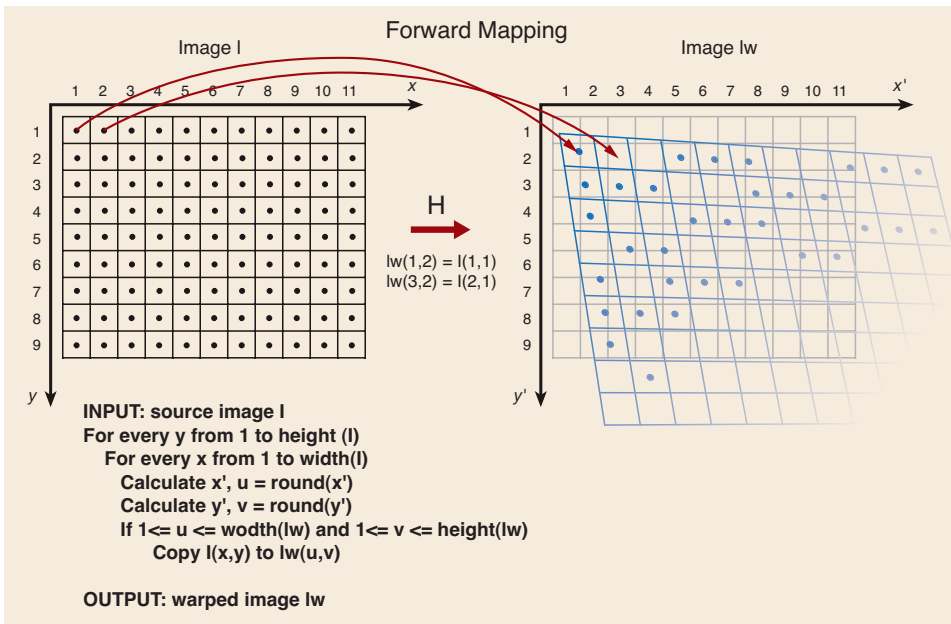


Figure 2 – Forward mapping with a scaling factor greater than one

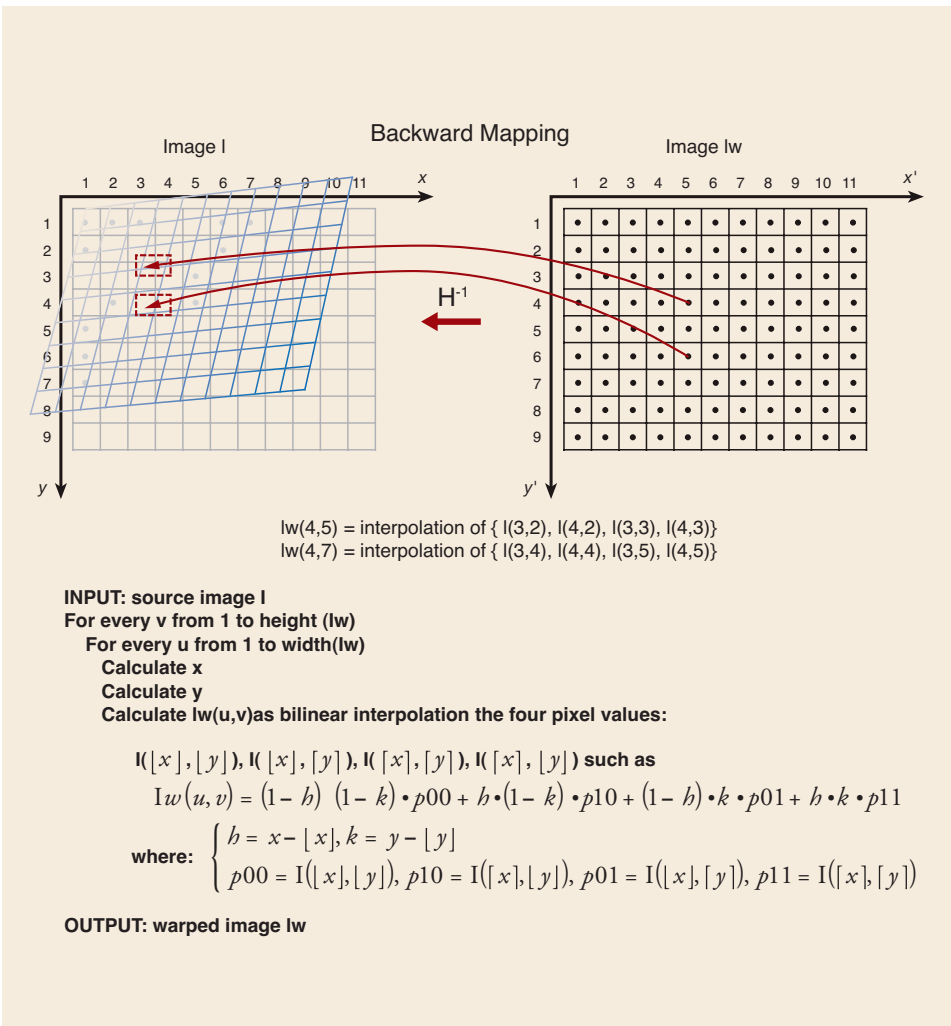


Figure 3 – Backward mapping with a scaling factor greater than one

Conclusion

The challenge is to design efficient, effective, and reliable vision modules with the highest possible reliability.

Ultimodule, a Xilinx XPERTS partner, and the VIPS Laboratory at the Università di Verona have defined a foundation platform for computer vision using Ultimodule's system-on-module family. The platform provides a stereovision system for real-time extraction of three-dimensional data and a real-time image-processing engine implementing most of the algorithms required when an application relies on vision to make decisions and provide control.

The platform supports applications that require high performance and robust vision analysis, both in qualitative and computational terms (real-time), including active video surveillance, robotic arm motion and control, autonomous vehicle navigation, test and measurement, and hazard detection. The platform provides modules with all required system control logic, memory, and processing hardware, together with the application software. Interconnecting modules allow fast development of a complex architecture.

The platform leverages Xilinx Spartan-3 devices, which are an optimal choice for image processing IP cores because of their flexibility, high performance, and DSP-oriented targeting. The Spartan-3 family provides a valid, programmable alternative to ASICs. This characteristic, coupled with its low cost structure, adds considerable value when time to market is crucial.

For more information about feature extraction, you can e-mail the authors at paolo.giacon@students.univr.it or saul.saggin@students.univr.it. For more information about image warping, you can e-mail matteo.busti@students.univr.it or giovanni.tommasi@students.univr.it.

We are grateful for the support from our advisor, Professor Murino, in the Vision, Image Processing, and Sound (VIPS) Laboratory in the Dipartimento di Informatica at the Università di Verona, and contributions from Marco Monguzzi, Roberto Marzotto, and Alessandro Negrente.