FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Implementação de rectificação de imagens estéreo num sistema embutido baseado em FPGA

João Guilherme Pereira Rodrigues

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Supervisor: Prof. João Canas Ferreira

Junho 2009

© João Guilherme Pereira Rodrigues, 2009

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2008/2009

A Dissertação intitulada

"IMPLEMENTAÇÃO DE RECTIFICAÇÃO DE IMAGENS ESTÉREO NUM SISTEMA EMBUTIDO BASEADO EM FPGA"

foi aprovada em provas realizadas em 20/Julho/2009

o júri

Presidente Professor Doutor José Manuel Martins Ferreira
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Arnaldo Silva Rodrigues de Oliveira
Professor Auxiliar Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Professor Doutor João Paulo de Castro Canas Ferreira
Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - **JOÃO GUILHERME PEREIRA RODRIGUES**

Faculdade de Engenharia da Universidade do Porto

Resumo

Para se obter percepção da profundidade em visão computacional é necessário processar pares de imagens estéreo. Esse processamento é computacionalmente muito pesado, dificultando significativamente a sua realização em tempo-real. Este processo requer a procura de correspondências dos objectos entre as duas imagens, tarefa que é bastante simplificada se as imagens estiverem rectificadas.

O processo de rectificação de imagens estéreo consiste normalmente em dois passos distintos, de requisitos computacionais diferentes. O cálculo das transformações a realizar requer um largo poder computacional sem requisitos de tempo real e a aplicação dessas transformações tem estritos requisitos de tempo real. Estas fases não costumam, por isso, ser implementadas em conjunto no mesmo sistema.

Neste projecto é aproveitado o poder crescente das FPGAs e é implementado um sistema que conjuga eficazmente ambas as fases. Neste documento são explicados os processos necessários para a concretização de cada passo e apresentado o estudo realizado sobre diferentes implementações e algoritmos já existentes.

É proposto um novo método que calcula as transformações necessárias, recorrendo apenas à análise de um par de imagens estereoscópicas. O método proposto é implementado numa FPGA, assim como um módulo que aplica em tempo real essas transformações.

O sistema realizado é capaz de rectificar em tempo-real imagens de duas câmaras de vídeo a 25 FPS com uma resolução de 640 x 480 pixéis. A FPGA utilizada é uma Xilinx XC3S1500, mas o sistema é escalável, facilmente aplicado em imagens de resolução superior ou com uma cadência superior.

Os resultados obtidos são bastante animadores, obtendo-se imagens com disparidade vertical inferior a 2 pixéis. Fica, assim, provada a capacidade de as FPGAs realizarem ambas as operações eficientemente.

Abstract

In order to obtain depth perception in computer vision, one needs to process pairs of stereo images. This process is computationally challenging to be carried out in real-time, because it requires the search for matches between objects in both images. Such process is significantly simplified if the images are rectified.

Stereo images rectification is usually achieved in two separate steps, with different computational requirements. The calculation of the transformations to be applied to each image has no real-time requirements, but is very demanding. On the other hand, applying those transformations to the video images is very restricted by real-time constraints. Therefore, these two stages are not usually implemented in the same system.

In this project, both stages are effectively implemented in a system, due to the increasing power of FPGAs. This report explains the necessary steps to implement both stages and discusses different algorithms and implementations described in the technical literature.

A new method is presented, that calculates the required rectification based on a pair of stereoscopic images. An FPGA-based implementation of the method is described and evaluated, such as a module that applies the calculated changes in real time.

The system is capable of performing real-time rectification on the images of two video cameras, with a resolution of 640 x 480 pixels and a frame rate of 25 FPS. The FPGA used for implementation is a Xilinx XC3S1500, but the system is scalable, and easily applied to images of higher resolution or a higher rate.

The results are quite satisfactory, with output images having a maximum vertical disparity of 2 pixels, proving that full images rectification can be efficiently achieved in a FPGA.

Agradecimentos

Quero agradecer por nenhuma ordem em especial,

Ao meu orientador João Canas Ferreira, e professor José Carlos Alves.

Aos meus colegas de laboratório, Joãozão, Pedro, Boneca, Pereira, Nuno e Alfredo, por terem aturado e contribuído nas inúmeras sessões de brainstorming.

À Xavier.

À minha família, namorada e amigos, por perceberem que em breve vão voltar a aturar-me.

Ao Google, que é meu amigo.

Aos senhores da República do Churrasco, pelas excelentes refeições que nos proporcionaram.

À Fonera, por disponibilizar um bom modelo para as imagens analisadas.

Ao Billy, o meu amigo imaginário.

João Guilherme Pereira Rodrigues

"I just had an epiphany..."

Every FPGA Developer

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

Lista de abreviaturas

BPS	Bits por Segundo
DDR	Double-Data-Rate Synchronous Dynamic Random Access Memory
DEEC	Departamento de Engenharia Electrotécnica e de Computadores
FEUP	Faculdade de Engenharia da Universidade do Porto
FIFO	First-In, First-Out
FPS	Frames por segundo
MIEEC	Mestrado Integrado em Engenharia Electrotécnica e de Computadores
IO	Input - Output
SAD	Sum of Absolute Differences
SRAM	Static Random Access Memory
SVD	Singular Value Decomposition

Lista de símbolos

X	Coordenada horizontal de um ponto
Y	Coordenada vertical de um ponto
U	Coordenada horizontal do mesmo ponto após transformação
V	Coordenada vertical do mesmo ponto após transformação
W	Peso das coordenadas de um ponto
H	Matriz de Transformação
F	Matriz Fundamental
e	Epipólo na imagem referência
A^T	Transposta da matriz A
A^{-1}	Inversa da matriz A

Capítulo 1

Introdução

Actualmente o sistema de processamento de imagens é uma tarefa que exige elevados recursos computacionais e é normalmente realizado por sistemas dedicados especialmente desenhados para o efeito. Por outro lado, o crescente aumento da densidade lógica dos sistemas reconfiguráveis tornam as FPGAs potenciais plataformas para o desenvolvimento de módulos de processamento de vídeo.

O problema de rectificação de imagens aqui abordado está presente em todos os sistemas de estereoscopia e é muitas vezes ignorado. Contudo, este pode deteriorar significativamente os resultados desses sistemas, mesmo em situações em que as câmaras se encontram calibradas.

Um bom sistema de rectificação das imagens, previamente ao processamento, pode melhorar os resultados dos mais variados sistemas.

1.1 Motivação

A estereoscopia tem sido alvo de muita investigação nas últimas décadas e nas mais variadas áreas. Obter informação de profundidade através de imagens planas é uma potencialidade apelativa para áreas como a robótica, a militar (como é o caso descrito em [2]), a cinematográfica, a automóvel e até mesmo a área médica, pois facilitaria a análise de raios-X (investigado em [3]) ou as intervenções à distância que, na última década, se tornaram alvo de investigação.

Contudo, quando um par de imagens estéreo é analisado computacionalmente ou observado por olhos humanos, existem vários aspectos que podem não representar fielmente a experiência de visualização 3D real, como defendido por Holliman em [4]. Caso existam demasiados aspectos imperfeitos, o resultado pode ser desconfortável para o olho humano ou resultar em erros de cálculo em análise por computador. Para identificar e corrigir algumas dessas irregularidades podem ser aplicadas diversas técnicas às imagens, sendo uma delas a rectificação. O processo de Rectificação de Imagens consiste em transformar as coordenadas de duas imagens, de modo a que todos os objectos se encontrem horizontalmente alinhados entre estas.

A rectificação de imagens estéreo num ambiente não calibrado é um processo de extrema exigência computacional e de elevados requisitos temporais. Por conseguinte, não é habitual a

implementação completa deste processo, sendo normalmente divido em duas fases distintas e implementadas em sistemas diferentes: uma fase de computação exigente, onde é calculada a transformação necessária e outra fase com requisitos de tempo real, em que essa transformação é aplicada.

Nesta dissertação pretende-se aproveitar a crescente capacidade das FPGAs para implementar as duas fases completas num único sistema, aumentando a utilidade do processo de rectificação de imagens.

1.2 Objectivos

O objectivo principal do projecto é implementar um sistema de rectificação de imagens estéreo em FPGA, para serem usadas posteriormente num sistema de Estereoscopia. O sistema possui como únicas entradas as imagens de duas câmaras estéreo, e assim tem de calcular todos os dados necessários à rectificação a partir de um ou mais pares de imagens. Posteriormente é necessário rectificar em tempo-real as imagens dos vídeos das duas câmaras.

O método em vista é integrado no hardware disponível, baseado em sensores CMOS de 640 x 480 pixéis, acoplados a um circuito baseado numa FPGA Xilinx Spartan-3 XC3S1500. O sistema implementado é escalável, facilmente implementável noutro hardware ou com requisitos de vídeo diferentes, sem com isso prejudicar os resultados.

Numa primeira fase é projectada uma biblioteca de funções e de módulos de hardware que facilitam a utilização de recursos do hardware fornecido. Esta biblioteca dá um certo grau de abstracção aos restantes módulos, facilitando o desenvolvimento destes.

Posteriormente é proposto um novo método para cálculo das transformações necessárias e que maximiza a visibilidade da área comum às duas imagens. Este cálculo é realizado com vista a facilitar a posterior transformação das imagens e é especializado em imagens estéreo. O método proposto é sujeito a testes de simulação e validação e são obtidos excelentes resultados em simulação.

De seguida, esse método é implementado, assim como um módulo que aplica em tempo real as transformações calculadas ao vídeo das câmaras. São também realizados testes para comparar a precisão dos resultados entre a simulação e a implementação.

Os resultados obtidos são bastante satisfatórios, estando ao nível dos de outros métodos executados em processadores com um maior poder computacional.

1.3 Estrutura

Este documento está organizado de forma a proporcionar ao leitor uma compreensão simples e intuitiva de todo o processo de rectificação.

Assim, no capítulo 2 é dada uma breve introdução aos conceitos básicos como base aos algoritmos utilizados. É apresentada e explicada a sequência de processamento necessária e são avaliados diferentes métodos e implementações divulgadas nos meios técnicos.

No capítulo 3 são introduzidos os materiais, hardware e programas de desenvolvimento utilizados. É explicada a implementação realizada dos módulos de suporte, responsáveis pelo funcionamento do sistema no hardware em questão.

No capítulo 4 é explicado o desenvolvimento do método proposto e do sistema responsável pela implementação das transformações.

No capítulo 5 são apresentados os testes realizados para validação dos algoritmos utilizados e expostos os resultados obtidos do sistema.

No capítulo 6 o trabalho é concluído e são mencionadas propostas para implementações futuras.

Capítulo 2

Estado da Arte

Neste capítulo é efectuada uma breve explicação dos conceitos básicos relacionados com rectificação de imagens estéreo e os seus diferentes problemas.

É também apresentada a pesquisa realizada a documentos técnicos, com o intuito de o leitor perceber mais aprofundadamente os diferentes problemas e ter uma perspectiva alargada das diferentes soluções já propostas.

2.1 Conceitos Básicos

Esta dissertação foi escrita de maneira a não ser necessário nenhum conhecimento prévio por parte dos leitores acerca de rectificação de imagens, no entanto algumas bases de matemática e álgebra são esperadas. Alguns dos conhecimentos necessários são:

- Multiplicação de Matrizes;
- Matriz Transposta;
- Matriz Inversa;

Nas seguintes secções será dada uma breve explicação dos conceitos básicos necessários à compreensão dos algoritmos existentes e aqui implementados.

2.1.1 Coordenadas Homogéneas

A transformação de imagens, isto é, o processo de alterar o modo como uma imagem é visualizada ou imprimida num ecrã, geralmente requer cálculos sobre as coordenadas de cada pixel da mesma. Por exemplo, uma translação na vertical que corresponde a mover a imagem para cima dy pixéis, em computação corresponde a realizar um cálculo $Y' = Y + dy$, em que Y é a coordenada vertical da imagem original, e Y' a coordenada obtida após translação. Uma ampliação ou escalonamento vertical em torno da origem por um factor sy corresponde a realizar $Y' = sy \cdot Y$.

Com a necessidade de representar coordenadas e transformações de coordenadas de forma clara, foi inventado no século XIX um sistema de representação de coordenadas denominado

Coordenadas Homogéneas. Este sistema permite simplificar as transformações de coordenadas comuns, implementando-as como uma simples multiplicação matricial. Mais precisamente, para imagens em 2D uma transformação ou sequência de transformações pode ser representada por uma matriz quadrada F de tamanho 3 x 3, e realizá-la corresponde a multiplicar essa matriz pela coordenada do ponto que se deseja transformar.

Mais informações e explicações sobre coordenadas homogéneas podem ser encontradas em [5] e [6].

Para a compreensão desta dissertação, é apenas necessário ter conhecimento dos seguintes factos:

- No sistema de coordenadas homogéneo, um ponto é representado por três coordenadas $[U,\ V,\ W]$, em que U e V são os equivalentes a X e Y no sistema cartesiano e W é o peso dessas coordenadas.

- A passagem de uma coordenada cartesiana para homogénea é realizada igualando W a 1:

$$[X,\ Y] = [12,\ 41] \rightarrow [U,\ V,\ W] = [12,\ 41,\ 1]$$

- Para a passagem inversa é necessário dividir cada coordenada pelo seu peso, ou seja:

$$[U,\ V,\ W] \rightarrow [X,Y] = [U/W,\ V/W]$$

  Este facto permite o tratamento de pontos no infinito, representados com $W = 0$, que será importante para o desenvolvimento do sistema a implementar.

- A representação de uma recta em coordenadas homogéneas $[A,\ B,\ C]$ corresponde em coordenadas cartesianas à seguinte equação.

$$X \cdot A + Y \cdot B + C = 0$$

  Note que existe uma infinidade de pontos para os quais a última equação se verifica, pontos esses que formam a recta desejada.

- Tal como em coordenadas cartesianas, para descobrir um ponto de uma recta é necessário resolver o seguinte sistema homogéneo linear.

$$[A,\ B,\ C] \cdot [X,\ Y,\ 1]^T = 0\ ,\ ou\ ,\ [A,\ B,\ C] \cdot [U,\ V,\ W]^T = 0$$

- Uma multiplicação de qualquer representação em coordenadas homogéneas por um escalar, resulta numa representação que em coordenadas cartesianas corresponde ao mesmo ponto ou recta. Ou seja, se k for uma constante, a seguinte equação aplica-se.

$$[U,\ V,\ W] = [X,\ Y] \leftrightarrow k \cdot [U,\ V,\ W] = [X,\ Y] \quad pois$$

$$k \cdot [U,\ V,\ W] = [kU,\ kV,\ kW] \rightarrow [X,Y] = [kU/kW,\ kV/kW] = [U/W,\ V/W]$$

- Para se realizar uma qualquer transformação definida por F é necessário efectuar a multiplicação
$$[U',\ V',\ W']^T = F \cdot [U,\ V,\ W]^T$$

- Como a multiplicação de matrizes é associativa, uma sequência de transformações correspondentes a
$$[U',\ V',\ W']^T = F_1 \cdot F_2 \cdot F_3 \cdot [U,\ V,\ W]^T$$
pode ser simplificada realizando previamente a multiplicação
$$F = F_1 \cdot F_2 \cdot F_3$$
sendo posteriormente apenas necessário realizar
$$[U',\ V',\ W']^T = F \cdot [U,\ V,\ W]^T$$

2.1.2 SVD

Ao longo desta dissertação serão realizados muitos cálculos com matrizes. Destes, o mais importante baseia-se na resolução de sistemas homogéneos lineares.

Estes sistemas matriciais são do formato $A \cdot x = 0$ e são muito comuns: por exemplo a solução ao problema "dados vários pontos de uma recta, encontrar as coordenadas homogéneas que definem essa mesma recta" é equivalente a resolver o sistema

$$v[U,\ V,\ W] \cdot [A,\ B,\ C]^T = 0$$

em que $v[U,\ V,\ W]$ é uma matriz de tamanho N x 3 gerada através das coordenadas de N pontos. A resolução deste sistema homogéneo de 3 variáveis requer no mínimo 2 equações, neste caso coordenadas de dois pontos.

Para resolver este género de sistemas podem ser usados métodos presentes em livros introdutórios à álgebra linear, como a Eliminação Gaussiana. No entanto, estes algoritmos são muito afectados por pequenos erros nos dados, obtendo resultados pouco favoráveis em sistemas de computação, visto que nestes os dados contêm sempre pequenos erros, como os de arredondamento. Sistemas de processamento de imagem não fogem a estes erros, e até incluem mais erros provocados por distorções na imagem como *demosaicing*[1].

No caso de alguns sistemas posteriormente apresentados nesta dissertação não existe certeza quanto à veracidade de todos os dados utilizados nos cálculos. Os dados são uma estimativa, podendo parte deles estar completamente errados. O algoritmo utilizado de resolução do sistema de

[1] O problema de *Demosaicing* será introduzido posteriormente

equações homogéneas deveria ser capaz de resolver o problema.

SVD

Existem alguns métodos de resolução, sendo o mais utilizado o SVD - *Singular Value Decompostion*. Este método realiza a *Least Mean Square* da solução, ou seja, tenta achar a solução que apresente a soma do menor erro quadrático, e devolve a solução e o grau de confiança dessa solução. Este factor é o ideal para computação, obtendo resultados úteis mesmo com alguns dos dados errados.

Na resolução de um sistema homogéneo de N variáveis os métodos tradicionais de resolução requerem $N-1$ equações. O método por SVD pode ser utilizado com qualquer número de equações superior ou igual a $N-1$, obtendo a solução que minimiza o erro de todas as equações.

Este método apresenta maior precisão e menores erros de arredondamento para a resolução de matrizes com valores próximos do 0, sendo assim comum normalizar as matrizes para valores no intervalo $[-1,\ 1]$.

Este método é de fácil implementação em computação, como descrito em [7].

Podem ser encontradas mais informações sobre o cálculo e vantagens da SVD em [8]. Para a compreensão desta dissertação apenas será necessário saber que ao ser realizada a SVD de uma matriz A se obtém a decomposição $A = U \cdot E \cdot V^T$, em que a matriz diagonal E exprime o erro da solução encontrada. A solução encontrada que resulta no menor erro quadrático para o sistema $A \cdot x = 0$ encontra-se na matriz V, na mesma coluna que o valor mais baixo da matriz diagonal E. Caso A seja uma matriz quadrada, a mesma coluna correspondente da matriz U corresponde à solução do sistema $A^T \cdot x = 0$, não sendo necessário efectuar outro SVD se este resultado for necessário, como será o caso nesta dissertação.

Os conceitos básicos previamente introduzidos deram uma noção dos cálculos matemáticos realizados nesta dissertação. Irão de seguida ser introduzidos conceitos de uma natureza menos matemática.

2.1.3 Estereoscopia

Estereoscopia é o processo de percepção visual que nos permite ter a sensação da profundidade a que um objecto se encontra, a partir de duas imagens diferentes desse mesmo objecto. Após o estudo de um mesmo objecto nas duas imagens, é possível calcular a sua distância relativa às câmaras através de um processo simples de triangulação.

A implementação deste processo em computação baseia-se fundamentalmente em dois cálculos distintos, como introduzido em [9]:

1. Achar as correspondências entre os objectos das duas imagens e retirar a disparidade da coordenada em pixéis;
2. Realizar a triangulação.

O segundo cálculo é de simples implementação e rápido processamento, já o primeiro tem uma complexidade que varia com a posição das câmaras: se as câmaras estiverem perfeitamente alinhadas, a apontar precisamente na mesma direcção e tiverem a mesma distorção ocular, o problema é simplificado a uma busca uni dimensional - numa linha paralela à recta epipolar de cada imagem, como explicado na secção sec:GeometriaEpipolar e em [10]; se um destes requisitos não for válido, que é o mais comum visto ser praticamente impossível alinhar perfeitamente duas câmaras, será necessária uma busca num espaço bidimensional. Dado que uma busca de correspondências em 2 dimensões exige muito processamento é necessário realizar-se, em sistemas de visão estéreo, um processo prévio de calibração das câmaras ou imagens.

A este processo chama-se Rectificação de Imagens, e com o intuito de o simplificar as imagens são normalmente analisadas segundo uma Geometria Epipolar.

2.1.4 Geometria Epipolar

Imagens podem ser descritas segundo várias geometrias. A Geometria Epipolar é a mais utilizada em casos de imagens estéreo, pois facilita a compreensão e cálculos necessários à rectificação. Esta explica e denota as relações existentes entre os objectos a 3 dimensões e a sua projecção no sensor da câmara (2 dimensões).

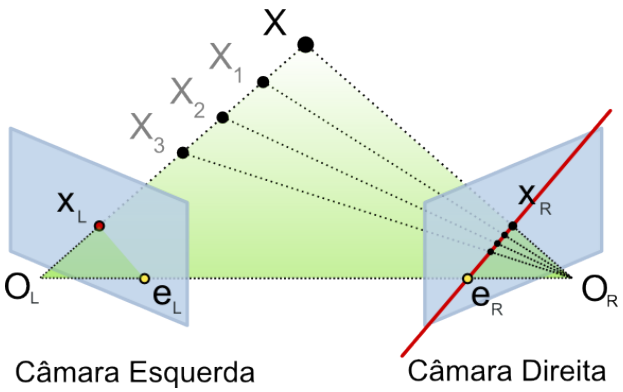


Figura 2.1: Geometria epipolar em imagens não rectificadas.

A figura 2.1 dá um exemplo de algumas limitações de projecções nos sensores das câmaras: a câmara da esquerda não consegue distinguir os pontos X, X_1, X_2 e X_3, ao contrário da câmara da direita. O_L e O_R representam o ponto de foco das câmaras, X o objecto de interesse, e X_L e X_R a sua projecção nos sensores das câmaras. Para simplificar, o sensor é representado à frente da câmara, sendo que geralmente na realidade se encontra por detrás do ponto de foco (lente).

Aos pontos que correspondem à intersecção da recta que passa nos dois focos das câmaras com os sensores destas, chama-se epipólos (E_L e E_R). Ambos os pontos de foco e os epipólos (O_L, O_R, E_L e E_R) se encontram sobre a mesma recta. A recta $O_L - X$ é vista pela câmara da esquerda como um único ponto, pois está alinhada com o seu ponto focal (O_L). No entanto, a câmara da direita vê esta recta como uma linha ($E_R - X_R$), denominada de Recta Epipolar da câmara da direita. Do mesmo modo, a recta $E_L - X_L$ é uma recta epipolar da câmara da esquerda.

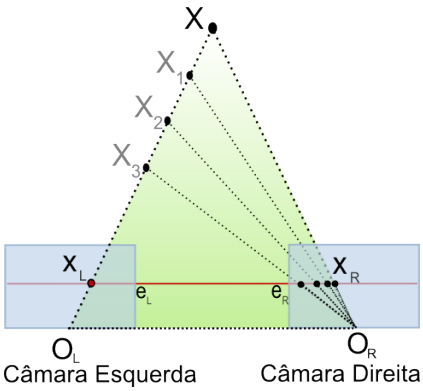


Figura 2.2: Geometria epipolar em imagens rectificadas.

Esta geometria epipolar é simplificada se os semi-planos dos sensores das câmaras coincidirem perfeitamente no mesmo plano, como mostra a figura 2.2. Neste caso, as linhas epipolares coincidem entre as câmaras ($E_L - X_L = E_R - X_R$), e são sempre paralelas à recta que passa em O_L e O_R. Com apenas mais uma rotação consegue-se garantir que as rectas epipolares ficam alinhadas com o eixo horizontal da imagem. Isto significa que para cada ponto numa imagem, o ponto correspondente na outra imagem se encontra à mesma altura (mesma coordenada vertical).

Fisicamente não é possível garantir que duas câmaras estejam alinhadas, mas é possível alinhar virtualmente as imagens resultantes destas. Esta operação chama-se de Rectificação de Imagens Estéreo, e é realizada através da aplicação de uma matriz de transformação H às coordenadas de cada pixel das imagens.

2.1.5 Rectificação

O processo de Rectificação de Imagens consiste em transformar duas imagens de modo a que todos os objectos se encontrem à mesma altura nas duas imagens. Assim é simplificada a busca de correspondências, reduzindo-a a uma busca a uma dimensão segundo uma recta horizontal. Isto corresponde a tornar as linhas epipolares das imagens perfeitamente horizontais, como mostra a figura 2.2.

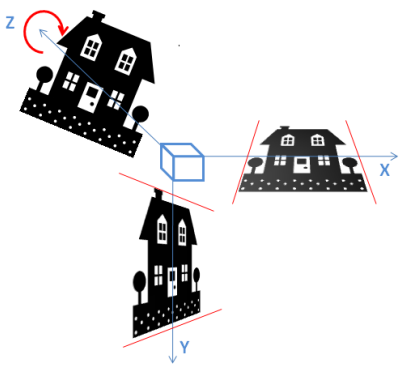


Figura 2.3: Rotações possíveis em imagens sobre os diferentes eixos

De um modo geral, a rectificação permite transformar duas imagens de quaisquer duas câmaras que apontem para objectos comuns em duas imagens perfeitamente alinhadas, efectuando apenas rotações em X, Y e Z, como mostra a figura 2.3, assim como translações e multiplicações por um factor de escala (para tornar as imagens do mesmo tamanho). Estas transformações correspondem a colocar ambas as imagens num único plano, e horizontalmente alinhadas.

Na realidade, a rotação, escalonamento e translação segundo o eixo X não serão conhecidas nem calculadas, porque o seu cálculo é impossível de realizar sem conhecimento de informações externas às imagens, como distância entre câmaras, etc. No entanto, estas transformações só alteram as coordenadas X dos pontos das imagens, não sendo portanto necessárias na rectificação.

Para facilitar a notação deste processo é utilizada a geometria epipolar descrita anteriormente.

2.2 Matriz de Transformação

Na secção anterior foi visto que será necessário efectuar transformações a nível de coordenadas (rodar, esticar, mover, etc) às imagens de ambas as câmaras. Como já foi referido, é vantajoso representar estas transformações em forma de matrizes, facilitando a computação, e qualquer número de transformações pode ser representado sobre uma única matriz.

Neste relatório chamar-se-ão a estas matrizes de Matrizes de Transformação, ou simplesmente de H, e o seu cálculo e precisão serão muito importantes para a qualidade dos resultados. Para a calcular será necessário realizar uma série de operações, sendo uma delas o cálculo da Matriz Fundamental.

A Matriz Fundamental (F) é uma matriz de tamanho 3 x 3 que relaciona duas imagens de uma mesma cena, indicando onde os pontos ou objectos de uma imagem se devem encontrar na outra. Sendo $Q = [X,\ Y,\ 1]^T$ e $Q' = [X',\ Y',\ 1]^T$ duas coordenadas distintas do mesmo objecto em duas imagens diferentes, a operação $l = F \cdot Q$ resulta numa recta na qual o ponto Q' se encontra na segunda imagem, ou seja, a sua recta epipolar (descritas na secção 2.1.4).

Existe alguma documentação de diversos métodos de calcular estas matrizes. Por exemplo, Mohr e Triggs [11] dão uma boa introdução à geometria epipolar, matriz fundamental e cálculo desta através apenas de correspondências entre as imagens. Estes autores propõem um fluxo geral para o aperfeiçoamento da matriz fundamental, da seguinte forma:

1. Extrair pontos das imagens;
2. Encontrar algumas correspondências iniciais utilizando um método qualquer;
3. Utilizar o algoritmo linear descrito por eles para calcular a matriz fundamental;
4. Iterar os últimos dois passos utilizando um algoritmo de optimização e eliminar correspondências indesejadas.

Já o algoritmo descrito por estes para calcular a Matriz Fundamental é semelhante ao apresentado por Hartley [12]. O método proposto utiliza também correspondências entre objectos nas

duas imagens para calcular a matriz fundamental, evitando assim o conhecimento prévio de qualquer parâmetro das câmaras. O autor dá muita importância à matemática por detrás do método, e apresenta boas bases matemáticas para o cálculo da Matriz Fundamental e também da Matriz de Transformação.

O método de cálculo da Matriz Fundamental de Mohr e Triggs [11] e Hartley [12] é semelhante, e consiste no seguinte: em parágrafos anteriores foi explicado que a operação $l = F \cdot Q$ resulta numa recta l' na qual se encontra Q'. Vimos também na secção 2.1.1 que em coordenadas homogéneas que uma recta $[A,\ B,\ C]^T$ define uma infinidade de pontos $[X,\ Y,\ 1]^T$ para os quais $X \cdot A + Y \cdot B + C = 0$. Assim, e como Q' se encontra sobre a recta l', podemos afirmar que $Q'^T \cdot l' = 0$ e, portanto, $Q'^T \cdot F \cdot Q = 0$ para quaisquer coordenadas Q e Q' de objectos correspondentes entre ambas as imagens.

Para resolver este sistema basta expandir a multiplicação, resultando em

$$X \cdot X' \cdot F_{1,1} + X \cdot Y' \cdot F_{1,2} + X \cdot F_{1,3} + Y \cdot X' \cdot F_{2,1} + Y \cdot Y' \cdot F_{2,2} + Y \cdot F_{2,3} + X' \cdot F_{3,1} + Y' \cdot F_{3,2} + F_{3,3} = 0$$

Esta equação homogénea tem 9 incógnitas, logo serão necessárias pelo menos 8 coordenadas de correspondências para o seu cálculo. Para simplificação podem ser escritas sobre a forma $X_v \cdot F = 0$ em que X_v é uma matriz cujas linhas são formadas pelas correspondências entre as imagens no formato

$$X_v = \begin{vmatrix} X_1 \cdot X_1' & X_1 \cdot Y_1' & X_1 & Y_1 \cdot X_1' & Y_1 \cdot Y_1' & Y_1 & X_1' & Y_1' & 1 \\ X_2 \cdot X_2' & X_2 \cdot Y_2' & X_2 & Y_2 \cdot X_2' & Y_2 \cdot Y_2' & Y_2 & X_2' & Y_2' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & 1 \end{vmatrix}$$

e F é um vector com os elementos da Matriz Fundamental

$$F^T = [F_{1,1}, F_{1,2}, F_{1,3}, F_{2,1}, F_{2,2}, F_{2,3}, F_{3,1}, F_{3,2}, F_{3,3}]$$

Os autores aconselham a utilização do SVD descrito na secção 2.1.2 para a resolução deste sistema. Para isso, a matriz X_v necessita de ter pelo menos 8 linhas, ou seja, ser criada a partir de 8 correspondências perfeitas. Devido às características do SVD, esta matriz pode ser constituída por mais de 8 linhas, aumentando a fiabilidade dos resultados quando as correspondências não são perfeitas. A fiabilidade do cálculo da SVD também aumenta com a disparidade das correspondências encontradas, isto é, estas devem estar espalhadas por toda a imagem e não todas numa mesma zona.

Mohr [11] propõe um passo adicional para garantir algumas características em F. Esse passo baseia-se em realizar a SVD de F, forçar o valor singular menor, E, a 0 e recalcular $F = U \cdot E \cdot V^T$. Este passo serve para garantir uma Matriz Fundamental válida matematicamente, mas que traduz de forma menos precisa a informação epipolar. Assim, para o caso de se querer calcular e correctamente, este passo não deve ser realizado.

Este autor não apresenta nenhuma solução para o cálculo da Matriz de Transformação, H. Uma proposta para este cálculo é dada por Hartley [12], baseando-se apenas na Matriz Fundamental

e nas correspondências, que será agora descrito. Como foi referido na secção 2.1.4, as rectas epipolares cruzam-se todas num ponto, denominado de epipólo - e. Então, a equação $e \cdot l = 0$ é verdadeira para qualquer recta epipolar, e portanto $l' = F \cdot e$ resulta num plano constituído por qualquer recta, ou seja, na recta definida por $[0,\ 0,\ 0]$. O sistema $F \cdot e = 0$ pode então ser resolvido por SVD resultando nas coordenadas do epipólo e.

Como já foi explicado, todas as rectas epipolares passam por e. Forçar estas a serem paralelas entre si e perfeitamente horizontais corresponde a colocar o epipólo no infinito, ou seja, no ponto $[k,0,0], k \neq 0$. Corresponde a aplicar as matrizes

$$R = \begin{vmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

que roda a imagem θ graus colocando o epipólo sobre um ponto $k \cdot [f,\ 0,\ 1];\ k, f \neq 0$, e

$$G = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{-1}{f} & 0 & 1 \end{vmatrix}$$

em que coloca o epipólo no infinito, ou seja, em $[k \cdot f, 0, 0], k \cdot f \neq 0$. Estas transformações são necessárias mas não suficientes, visto que nos indica as rotações necessárias no eixo Y e Z e não o factor de escala nem translação vertical.

Portanto, Hartley [12] propõe uma fórmula geral para o cálculo de H como sendo $H = M \cdot G \cdot R \cdot T$, em que T é uma matriz de translação movendo o centro da imagem para a origem, R é a matriz de rotação em Z, G é a matriz de rotação em Y colocando o epipólo no infinito, e M é uma matriz obtida através da minimização das distâncias entre as correspondências após a aplicação das matrizes $G \cdot R \cdot T$. A proposta de Richard I. Hartley para o cálculo de M baseia-se na minimização da distância das correspondências. Este facto faz sentido quando se pretende rectificar imagens quase planas, com pouco informação de profundidade, como fotos aéreas ou de background. No caso de se querer rectificar imagens estéreo de forma escalável, que funcione com todo o tipo de equipamentos de estereografia, esta limitação não pode estar presente, sendo necessária a utilização de outro método. A hipótese mais simples é contar para a minimização da distância simplesmente a discrepância em Y, mantendo assim toda a informação de profundidade presente nas imagens originais. Como será falado posteriormente, este último passo pode ser realizado com uma simples SVD, e a precisão do método em geral melhorado com a normalização e posterior desnormalização das coordenadas da imagem.

O método para cálculo de H utilizado nesta dissertação foi baseado no proposto por Richard I. Hartley em [12] e será falado na secção 4.2.3.

2.3 O Problema das Correspondências

Como foi já explicado, parte do problema de rectificação consiste na busca de correspondências entre as duas imagens, isto é, para alguns elementos de uma imagem tem de se descobrir as coordenadas do mesmo elemento na outra imagem. Para isso, é necessário tomar algumas decisões quanto ao algoritmo a utilizar:

- Que elementos se querem detectar - Cantos, esquinas, ou objectos inteiros;
- Como se vai avaliar o candidato;
- Como procurar o candidato ou elemento correspondente na outra imagem.

É preciso ter em conta que alguns problemas podem ocorrer, como o facto de um elemento apenas existir numa das imagens (elemento obstruído por outro elemento) ou haver correspondências erradas devido a elementos parecidos. Alterações na luminosidade ou na ampliação também podem gerar erros. Muitos métodos diferentes são comparados e avaliados em [13]. No trabalho de Daniel Scharstein [13] e de [14] foram estudados os principais problemas neste género de algoritmos e os diferentes métodos normalmente utilizados.

Alguns algoritmos de busca de correspondências são baseados em correlações e outros em busca de características específicas. Os primeiros funcionam através da correlação entre a zona de uma imagem e várias zonas da outra, à procura da zona de maior correlação. Os pixéis centrais às zonas são então considerados correspondentes. Este método trata correctamente diferenças em luminosidade, mas é de computação muito lenta e pesada. São possíveis optimizações, implementando apenas busca local ou correlacionando por FFT e paralelizando na FPGA. Estes métodos por correlação obtêm bons resultados em zonas heterogéneas com muita informação visual. No entanto, se utilizados em padrões e imagens repetitivas resultam por vezes em correspondências erradas. O custo computacional de realizar a correlação aumenta exponencialmente com o tamanho da janela a analisar.

Outros métodos baseiam-se principalmente em características dos elementos. Estes analisam ambas as imagens e listam as zonas com muita informação visual, como esquinas e fronteiras. Aplicando pesos diferentes a cada elemento encontrado e realizando buscas locais aos elementos da outra imagem, consegue-se realizar correspondências com uma taxa elevada de sucesso. Os resultados deste tipo de método dependem de algumas variáveis, como a escolha de características a procurar e a zona de procura. Não obtém bons resultados para zonas com muitas texturas ou com muito ruído, apesar de poder ser robusto a alterações de luminosidade ou de zoom. Estes métodos, ao contrário dos baseados em correlação, não permitem achar correspondências para a imagem completa, sendo sempre limitados a alguns pontos espalhados pelas imagens. Este facto não é limitativo no intuito desta dissertação, pois como já foi explicado só são necessárias algumas dezenas de pontos para calcular a matriz.

Ambos os métodos podem ser melhorados através do cálculo iterativo da matriz fundamental, beneficiando assim do facto de as linhas epipolares serem já estimadas: a busca pelo elemento na

outra imagem pode ser reduzida a uma zona próxima da linha epipolar estimada, como discutido em [14].

Outro melhoramento possível de se implementar em qualquer algoritmo de busca consiste em realizar a procura em pirâmide, redimensionando a imagem previamente. A ideia básica consiste em procurar primeiro as correspondências em imagens mais pequenas e menos ruidosas (baixa resolução). Isto torna as buscas seguintes em buscas mais locais, à medida que se vai aumentando a imagem. Na última iteração as imagens a serem procuradas são as imagens originais e a zona de procura será muito localizada facilitando o processo. Este processo poderá aumentar a fiabilidade dos pares encontrados, mas aumenta também a complexidade do algoritmo e, portanto, a carga computacional necessária.

O trabalho de Pilu e Lorusso [14] introduz um novo método de aperfeiçoamento dos resultados. Este método não requer qualquer calibração ou informação prévia e baseia-se no algoritmo já introduzido, o Single Value Decomposition (SVD). O algoritmo utilizado baseia-se na busca de características em ambas as imagens e permite a realização, de base, de um trade-off entre desempenho e complexidade. O seu núcleo de processamento baseia-se nos seguintes passos:

1. Detectar qualquer número de características utilizando um método qualquer;

2. Criar uma matriz de pesos entre todas as características encontradas numa imagem e as da outra, em que a proximidade e a similaridade deverão ser os pesos. Foi proposta a equação

$$G_{ij} = e^{\frac{-(C_{ij}-1)^2}{2 \cdot y^2}} \times e^{\frac{-r_{ij}{}^2}{2 \cdot s^2}}$$

   em que c_{ij} é a correlação (similaridade) entre a característica i e j, r_{ij} é a distância entre essas mesmas características, e y e s são constantes a definir que pesam a importância de serem similares, e a sua proximidade, respectivamente ;

3. Calcular a SVD da matriz $G = T \cdot D \cdot U^T$;

4. Substituir os valores da diagonal de D por 1, e realizar o inverso da SVD: $P = T \cdot D \cdot U^T$.

Como se pode verificar nos passos anteriores, este algoritmo utiliza as propriedades da SVD para organizar as características e eliminar correspondências erradas. A matriz P final amplifica as qualidades entre pares. Para retirar correspondências desta matriz basta retirar o valor mais alto da matriz, sendo a linha e a coluna as características i e j. Deve-se repetir o processo, eliminando essa linha e coluna e retirando o novo máximo, até se ter pares suficientes ou a matriz se tornar nula. O trade-off entre desempenho e complexidade é realizado na recolha das características: quanto mais características se retirar, maior será a matriz G e o processamento necessário. O processamento baseia-se principalmente no cálculo da SVD e da matriz G, mas esta última pode ser facilmente simplificada. Os autores propõem outros métodos para eliminar más correspondências, que poderiam ser adicionados a este algoritmo, no entanto esses métodos estão sempre limitados

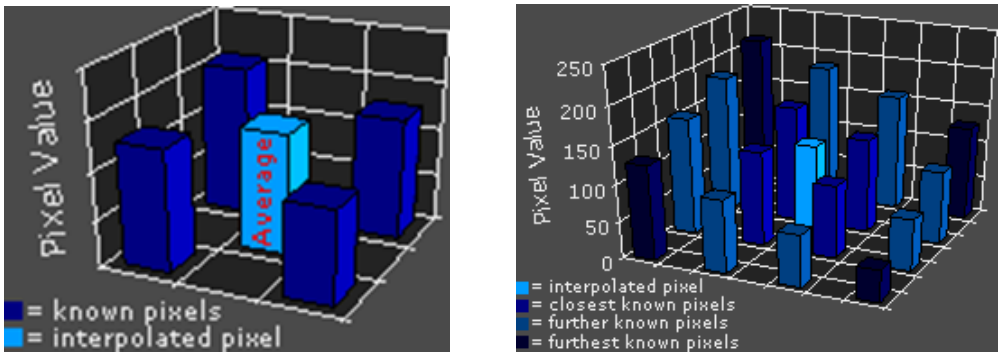
a alterar os valores da matriz G. Este facto torna difícil a utilização de informação epipolar para melhorar a precisão dos pares encontrados.

O método utilizado nesta dissertação para a busca de correspondências é um misto de muitos outros métodos estudados e será detalhadamente explicado na secção 4.2.1.

2.4 Reconstrução das Imagens Finais

Um problema sempre presente no processo de rectificação de imagens é a reconstrução das imagens finais. Este problema é mencionado em [1]. Após a matriz de transformação ser aplicada nas imagens, o que corresponde a rotações e escalonamentos, os pixeis resultantes não correspondem a outros pixeis nas imagens finais. Isto deve-se ao facto de todos os dispositivos de visionamento (monitores, etc) representarem os pixeis numa grelha rectangular, ou seja, mesma distância entre os pixeis na imagem toda.

Assim, a matriz de transformação indica que um certo pixel na imagem final corresponde ao pixel na imagem original encontrado entre outros pixeis, numa posição decimal. A precisão decimal é importante para haver a mínima perda de informação. Como na imagem inicial só nos são dados os pixeis em posições inteiras, temos de reconstruir a informação sub-pixel utilizando algoritmos especializados.



(a) Bi-linear (b) Bi-cúbica

Figura 2.4: Diferentes métodos de Interpolação. (Retirada de [1])

Tal processo é designado de interpolação e existem basicamente dois tipos de algoritmos interpoladores:

Adaptativos procuram por características importantes em redor do pixel a reconstruírem - esquinas, etc. Estes algoritmos diminuem, assim, os erros tentando preservar essas características. Como são mais avançados e variados, os algoritmos deste tipo são normalmente patenteados pela empresa que os inventou;

Não adaptativos que reconstroem a informação com base nos pixeis vizinhos. Quanto mais vizinhos se utilizarem, mais correcto pode ser o pixel reconstruído, mas maior o tempo de processamento. Estes algoritmos são os mais comuns e os mais utilizados são:

- Vizinho mais próximo: rápido mas com maus resultados. Consiste em escolher e copiar simplesmente o pixel mais próximo da posição interpolada. Isto resulta numa imagem final "pixelizada", com muitos erros visuais;
- Bi-linear: este método de interpolação considera os 2x2 vizinhos mais próximos da posição a interpolar. Calcula o valor do pixel com base numa média pesada dos valores desses 4 vizinhos, como se pode ver na Imagem 2.3. Os pesos de cada vizinho dependem da sua distância à posição a interpolarem. O resultado final é uma imagem mais suave e com menos erros visuais que o método anterior, mas com alguma perda de informação;
- Bi-cúbico: parecido com o bi-linear, mas utiliza uma janela de 4x4 vizinhos, como exemplificado na Imagem 2.4. Requer mais tempo de processamento que os anteriores, mas resulta numa menor perda de informação.

Em 2004 El-Kahmy e outros [15] propuseram um novo algoritmo de interpolação. Este algoritmo é baseado em interpolação bi-cúbica, mas adiciona pesos adaptativos aos vizinhos levando à preservação de esquinas e outras características. O algoritmo está bem documentado, e os resultados apresentados no artigo mostram que este obtém resultados melhores que com os algoritmos bi-linear e bi-cúbico. No entanto é mais pesado computacionalmente.

2.5 Implementações Anteriores

Como foi referido no capitulo 1, o objectivo desta dissertação é implementar rectificação de imagens estéreo num sistema baseado em FPGA e vídeo de resolução 640 x 480 a 25 FPS. Isto consiste em dois passo importantes e sempre relacionados: calcular a matriz de transformação e aplicá-la. Foi realizada uma pesquisa por implementações com características semelhantes e alguns sistemas foram encontrados.

Em [16] a equipa chinesa do Instituto de Tecnologia de Pequim desenvolveu um sistema de dimensões reduzidas capazes de realizar estereoscopia em tempo real. A implementação consiste numa FPGA Xilinx Virtex 2 com 2 milhões de portas lógicas, e um circuito auxiliar com três sensores CMOS de resolução 640 x 480 pixeis. O sistema realiza a rectificação e a estereoscopia nas imagens obtidas pela câmara e consegue obter resultados satisfatórios: 30 FPS a 640 x 480. Apesar de a rectificação do projecto beneficiar do facto de ser tri-ocular, o que diminui a carga computacional em cada par de imagens, os resultados finais não são tão bons quanto o esperado. Isto deve-se ao facto de ser necessário o conhecimento prévio da matriz de calibração para cada câmara, o que pode gerar erros, e o facto de a reconstrução final da imagem rectificada ser baseada

em interpolação bi-linear. Esta reconstrução final deveria tentar preservar esquinas e outras características das imagens, que poderão ser fundamentais ao processamento futuro dessas imagens. O facto de os autores terem conseguido implementar os dois processos em FPGA com resultados satisfatórios de desempenho dá uma boa motivação à realização de um trabalho mais especializado e cuidado do processo de rectificação.

Recentemente, Xinting Gao e outros [17] implementaram auto-rectificação de imagens estéreo num sistema baseado em IC3D, um SIMD especializado em processamento de vídeo. O sistema tem características boas pois implementa também o cálculo de mapas de profundidade, no entanto, funciona apenas em vídeos de tamanho 320 x 240 a 30 FPS. A rectificação implementada por Xinting Gao também é um factor limitativo, pois implementa apenas translações. O autor optou por impor esta restrição devido à quase calibração do sistema em que estava a trabalhar.

Muitos outros sistemas de rectificação foram desenvolvidos, sendo a maioria implementados em PC e portanto não cumprindo requisitos de tempo-real.

O facto de não haver nenhum sistema já desenvolvido levou a que fosse necessária a junção de diversos métodos e algoritmos, e a sua passagem e adaptação para FPGA e para a placa de desenvolvimento. No entanto, foi dada muita importância em manter o sistema o mais escalável possível, sem prejudicar muito o desempenho.

2.6 Resumo dos Algoritmos Propostos

Ao longo deste capítulo foram apresentados os principais métodos e algoritmos existentes na literatura. Na tabela 2.1 estes algoritmos são resumidos e as suas entradas, saídas, vantagens e desvantagens são apresentadas. Como é visível na tabela, foram encontrados somente dois projectos com características semelhantes.

Tabela 2.1: Resumo da literatura estudada

Autor	Entradas	Saídas	Vantagens	Desvantagens
Pilu, Lorusso	Par de Imagens Vector de características a testar	Vector de Correspondências	Eficiente e rápido Baseado em SVD	
Mohr, Triggs	Vector de correspondências	Matriz Fundamental	Algoritmo simples e linear	Desactualizado
Hartley	Vector de correspondências	Imagens rectificadas	Bem documentado	Processamento final das imagens por interpolação linear
El-Kahmy outros	Imagem a interpolar	Imagem interpolada	Preserva características	Pesado computacionalmente
Jia, outros	Imagens e Matriz de Calibração das Câmaras	Mapas de Profundidade	Implementado em FPGA	Necessita da matriz de calibração das câmaras
Xinting, Richard, Ben	Imagens	Mapas de profundidade	Inclui cálculo de H	Implementado em SIMD, pouco escalável, apenas translações, imagens de dimensões reduzidas.

Capítulo 3

Plataforma de Desenvolvimento

Neste capítulo é dada uma descrição geral do projecto, o material utilizado, programas e ambiente de desenvolvimento, programa de testes, assim como a razão das escolhas destes.

No capítulo anterior foi dada uma panorâmica geral sobre o sistema a implementar. Este resume-se a:

- Obtenção de imagens de duas câmaras;
- Análise das imagens e cálculo da matriz de transformação;
- Aplicação em tempo real da matriz calculada ao vídeo proveniente das câmaras;
- Projecção das imagens rectificadas num monitor ou gravação em memória.

Para além destas 3 funções básicas, deve ser implementando um sistema de teste e confirmação de todos os passos realizados.

Com estes requisitos em mente, foi analisado o material disponível e tomadas as seguintes escolhas de entre os programas a utilizar para desenvolvimento e teste.

3.1 Câmaras

Para a obtenção de imagens estéreo foram disponibilizadas duas câmaras digitais do mesmo modelo, permitindo assim minimizar discrepâncias na aquisição das imagens, como da intensidade luminosa dos pixéis, distorção ocular e níveis de focagem.

O modelo utilizado é o C3188A da Quasar Electronics com sensor OV7620 da OmniVision disponível em [18], ilustrado na figura 3.1.

Este modelo fornece uma saída digital com débito configurável de 8bit ou 16bit. Esta configuração e outras, como controlo de luminosidade e outros pré-processamentos, é realizada através de um barramento I2C e já está implementada nos módulos obtidos de projectos de outros anos. Para efeito de sincronização, em trabalhos anteriores foi usado um relógio externo às câmaras, proveniente da FPGA, em vez do relógio local de cada uma delas. A resolução de cada sensor

Figura 3.1: C3188A da Quasar Electronics

Figura 3.2: Suporte das câmaras

CMOS é de 640 X 480 pixéis a cores e a cadência com o relógio externo de 25 MHz utilizado em trabalhos anteriores é de 25 FPS, o que resulta numa cadência de pixéis de 12,5 MHz. Apesar destas características, a implementação proposta nesta dissertação tenta ser o mais escalável possível, principalmente no que toca à resolução das imagens.

Para simplificar o manuseamento e orientação das câmaras, foi fornecido juntamente com as câmaras uma PCB (placa de circuito impresso) onde se encontram montadas. Esta PCB comunica com a placa de desenvolvimento através de um cabo de 140 pinos IO.

Como se pode visualizar na figura 3.2, as câmaras encontram-se colocadas próximo uma da outra, a apontar numa direcção ligeiramente convergente, num ambiente quase rectificado. Estes factos permitem que seja desprezada a distorção ocular que iria complicar o método de cálculo da matriz de transformação. Esta distorção tenta ser a mais reduzida possível pelo fabricante das objectivas, mas geralmente é suficiente para provocar distorção radial visível nas imagens, como é visível na imagem 5.5. Como em sistemas de estereografia os objectos encontram-se geralmente em zonas semelhantes, esta distorção pode ser desprezada na maioria dos casos.

Uma desvantagem da utilização deste suporte ou PCB é o facto de só estar disponível um barramento de 8bit para os dados de cada câmara, o que torna impossível a aquisição da informação de cor a 25FPS. Esta limitação veio facilitar o processamento das imagens, no entanto veio também dificultar a busca de correspondências pois só parte da informação das imagens está disponível: a informação de luminosidade, normalmente denominada por imagem a preto e branco.

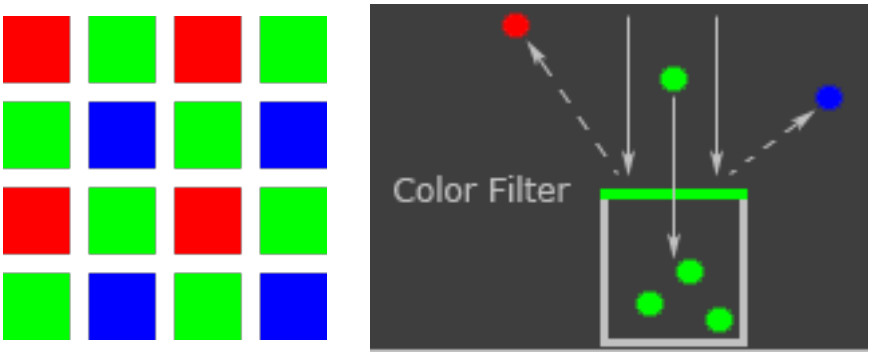


Figura 3.3: Filtro Bayer

Figura 3.4: Imagem original

Figura 3.5: Imagem no Sensor

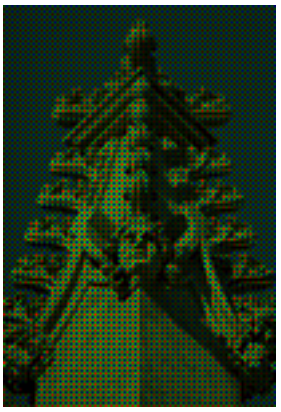


Figura 3.6: Imagem captada

Uma outra limitação das câmaras disponíveis reside no facto de o sensor CMOS ser baseado num filtro Bayer. Este filtro é normalmente utilizado pelas câmaras digitais e permite que um sensor CMOS de luminosidade (preto e branco) capte também informação de cor, como é explicado em [1]. Isto é feito com a aplicação de um filtro colorido sobre os pixéis do sensor, como mostra a figura 3.3. Assim, para uma imagem real como a 3.4, um sensor normal obteria a imagem 3.5, e com um filtro Bayer obtém 3.6. O inconveniente é que com a utilização de um filtro de cor, a informação das três cores não está presente em todos os pixéis, sendo necessário interpolar dos pixéis vizinhos. Este processo chama-se de Demosaicing.

No caso desta dissertação em que o barramento de dados permite apenas a utilização da informação de luminosidade, a utilização deste filtro vem apenas danificar a imagem, dificultando a análise da imagem. Não nos é indicado o algoritmo de Demosaicing realizado internamente pelas câmaras, mas mesmo com a utilização de um bom método de interpolação a imagem sofre sempre um efeito passa-baixo, dando uma sensação de desfocagem em alguns objectos O problema é ainda agravado quando se utiliza imagens estéreo, de duas câmaras, pois uma aresta afiada numa imagem pode ser detectada como uma aresta suave na outra, dependendo da sua cor e posição no sensor da câmara. A figura 3.7 mostra 4 possíveis imagens obtidas de uma mesma esquina.

Assim os algoritmos a implementar têm de ser adaptados, para se basearem unicamente em informação de luminosidade e serem capazes de lidar com imagens com propriedades diferentes.

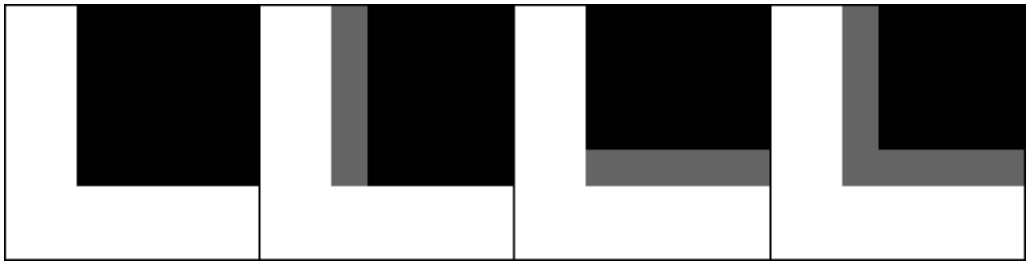


Figura 3.7: Problema de Demosaicing. Quatro possíveis representações da mesma esquina

3.2 Placa VGA adicional

Junto com as câmaras e a placa de desenvolvimento a seguir descrita, foi também fornecida uma placa VGA externa. Apesar de a placa de desenvolvimento já conter uma saída VGA, esta tem historial de causar incompatibilidades de sincronismo com o monitor. Assim a placa VGA externa permite ultrapassar esse problema. As imagens guardadas são imagens de 24 bits, ou seja 8 bits por pixel por cor. A placa contém 3 memórias FIFO, uma para cada cor. As FIFOs têm um tamanho de 512KBytes cada uma, mas apenas 307.200 bytes são utilizados por cor, que correspondem aos pixéis visíveis de uma imagem. Cada FIFO possui dois sinais de relógio, de leitura e de escrita, que permite que estas operações sejam feitas de forma independente, em momentos e até em frequências de relógio diferentes.

Esta placa contém também um VDAC que converte o sinal digital para analógico.

A comunicação desta com a FPGA é realizada através de um cabo de 50 pinos IO. A saída VGA desta placa está ligada a um monitor CRT de 15"e funcionará com as definições VGA padrão: 640x480, 60Hz.

3.3 Placa de Desenvolvimento

Para a realização deste projecto foi fornecida uma placa de desenvolvimento da Avnet, Avnet Spartan-3 Development Board (3S1500) Rev B. Esta placa vem já equipada com interfaces e recursos variados. Anteriormente já foram realizados projectos nesta placa que implementam aquisição e processamento de vídeo, o que facilita a configuração da placa e comprova a correcta propagação dos sinais necessários. Os periféricos encontrados são muito diversificados, como pinos de acesso genérico, portas de dados (USB, Ethernet, RS232, PS2, VGA, etc.), ecrãs oled, displays, interruptores e leds.

Dos recursos deste kit de desenvolvimento destacam-se:

- Núcleo reprogramável FPGA Spartan 3 XC3S1500-FG676;
- Vários relógios externos 100 MHz;
- 2 MB SRAM e 16 MB Flash;
- 32 MB DDR-SDRAM;
- Conector 50 pinos I/O;
- Conectores AvBus de 140 pinos I/O;
- Conector RS232;
- 8 LEDs e 8 Switchs;
- Portas de programação do tipo JTAG e *Paralel Cable IV* da *Xilinx* .

Os conectores de 50 e de 140 pinos serão utilizados respectivamente pela placa VGA e pelas câmaras, como foi descrito nas secções anteriores.

Após os testes preliminares da placa, reparou-se que a programação através do cabo JTAG é muito lenta, optando-se por utilizar o *Paralel Cable IV* para futuras implementações. Este cabo é capaz de programar a FPGA em menos de 3 segundos, fazendo com que seja desperdiçado menos tempo na realização de testes.

Os *Switchs* disponíveis na placa mostraram-se muito úteis para a sincronização dos sinais de controlo dos vídeos, nomeadamente da leitura das imagens e envio para a placa VGA. Esta sincronização será descrita no capítulo seguinte.

Os LEDs serão úteis durante a fase de desenvolvimento, permitindo um feedback rápido e simples de alguns sinais e estados internos dos módulos. O conector RS232 será muito utilizado em todas as partes do desenvolvimento, como principal meio de comunicação entre utilizador e placa de desenvolvimento.

A placa de desenvolvimento possui também vários tipos de memórias, com diferentes capacidades: SRAM de 2 MBs e barramento de 32 bits, FLASH de 16MBs e barramento também de 32 Bits, e DDR-SDRAM de 32 MBs e barramento de 16 Bits. A memória SRAM partilha o mesmo barramento de dados que a FLASH, logo a utilização conjunta ficará restrita a velocidades mais lentas. A FPGA presente nesta placa apenas contém 64 KBs de Block-RAMs. Como para a busca de correspondências é necessária a presença de duas imagens integrais em memória e cada imagem ocupa 302.000 Bytes, é necessária a utilização de uma destas memórias. As memórias DDR geralmente são capazes de velocidades de acesso superior devido a efectuarem operações em rajadas, ou seja, são capazes de com um comando enviar até 8 barramentos de dados em ciclos consecutivos. Isto resulta no máximo em 16 pixéis de 8 bits a cada 9 (1 de comando) ciclos de relógio a 100 MHz. A memória SRAM por outro lado é assíncrona e permite velocidades de acesso aleatório até 83 MHz. Devido a atrasos nos sinais de controlo e de sincronismo esta é apenas implementada a 50 MHz. Mesmo a esta frequência consegue-se uma velocidade de leitura e escrita superior à da memória DDR, de 4 pixéis de 8 bits a cada ciclo de relógio de 50 MHz. Devido à simplicidade de implementação e acesso aleatório decidiu-se utilizar unicamente a memória SRAM. A sua utilização e implementação do módulo de controlo serão descritas no capítulo seguinte.

Será agora dada uma breve descrição do núcleo reprogramável desta placa de desenvolvimento, uma FPGA Spartan 3 XC3S1500-FG676.

3.3.1 FPGA

A placa de desenvolvimento tem como núcleo principal um chip do tipo FPGA Spartan 3 XC3S1500-FG676. É muito comum utilizar-se chips FPGA em placas de desenvolvimento, devido à sua grande capacidade de reconfiguração. Esta reconfiguração, tanto interna como a nível das portas IO, permite as mais variadas funções e aplicações e, também, a conexão aos mais variados recursos externos sem necessitar de recorrer a drivers externos. Este integrado é formado por

um elevado número de células lógicas programáveis que podem ser interligadas e configuradas um número ilimitado de vezes, e de maneira a realizar qualquer tipo de lógica ou função.

A capacidade de reprogramação dos chips da FPGA torna-os mais baratos e rápidos de desenvolver que protótipos em tecnologia ASIC. O seu consumo energético e velocidade, apesar de piores, são suficientes para a maioria das aplicações/funcionalidades. Por outro lado, os processadores sequenciais como RISC (Reduced Instruction Set Computer) são mais baratos e rápidos a desenvolver, mas a sua velocidade de execução é demasiado lenta para implementar um sistema com restrições fortes de tempo-real.

No âmbito desta dissertação é necessário simplesmente perceber alguns pontos focados de seguida.

A configuração de uma FPGA é feita através de um software próprio do fabricante, que utiliza a linguagem Verilog para implementar a lógica desejada. Esse software será introduzido em 3.4.

A frequência máxima de execução de determinada lógica depende da sua complexidade e da forma como é implementada. Esta é definida pelos atrasos máximos das linhas de dados e lógica implementada, e existem diversas formas de a aumentar, como simplificar funções, pipelining, etc. Na utilização de recursos externos a FPGA, como as memórias SRAM, é importante também ter em conta os atrasos das linhas externas. Estes não são calculados pelas ferramentas de síntese, mas podem ser minimizados através da utilização de registos nos IOB. Um IOB é um In/Output Block, e como o nome indica são células da FPGA situadas na periferia do chip, com acesso directo aos PINs de saída. Um registo no IOB é a forma mais simples de garantir tempos mínimos de Setup e de Hold nos pinos da FPGA. As ferramentas geralmente utilizam os IOBs automaticamente para a comunicação com módulos externos.

Para aumentar o desempenho as FPGAs possuem normalmente blocos internos de hardware dedicados para as funções mais comuns, como RAM, processador, multiplicadores, etc. Este género de funções é frequentemente utilizado e ao ser implementado em hardware dedicado permite melhorar o seu desempenho e área ocupada, libertando recursos para fins mais diversos.

O modelo disponível na placa de desenvolvimento é uma Spartan 3 da *Xilinx* , como tal, este modelo não possui processador dedicado. No entanto o software do fabricante permite de forma rápida e simples a integração de um *Softcore*, denominado de MicroBlaze.

3.3.2 Softcore na FPGA

MicroBlaze é um *Softcore* desenhado especialmente para as FPGAs da *Xilinx* . É um microprocessador implementado unicamente com os blocos disponíveis na FPGA e permite a execução de instruções sequenciais numa FPGA, diminuindo o tempo de desenvolvimento de programas com menores requisitos temporais. Em termos de instruções disponíveis, o MicroBlaze é muito semelhante a um processador RISC.

As ferramentas incluem um compilador de código C para linguagem máquina, que é armazenado na memória e executado sequencialmente. A memória utilizada pode ser as BRAMs internas da FPGA ou um módulo de memória compatível interno ou externo, como SRAM, DDR-SDRAM, etc.

Para as comunicações o MicroBlaze utiliza um sistema versátil de comunicações: para módulos de BRAM local utiliza um barramento dedicado, chamado de LMB; para outras comunicações podem ser configurados vários barramentos de outros 3 standards, o PLB, OPB, e FSL. O PLB e OPB são geralmente utilizados para comunicações com módulos externos, como blocos de memória. Já o FSL - Fast Simplex Link, como o nome indica, é muito rápido e simples de implementar, tornando-o ideal para as comunicações com os outros módulos gerados pelo utilizador. Para ajudar nas comunicações por FSL é implementada uma FIFO em paralelo com o MicroBlaze, ficando esta responsável pelas comunicações com os outros módulos. Esta FIFO simplifica os sinais de sincronismo, permitindo até ser implementada uma versão assíncrona do barramento. Os módulos auxiliares gerados pelo utilizador podem ter diversas funções, como acelerar cálculos pesados computacionalmente, ou implementando drivers de comunicação com módulos externos.

O MicroBlaze torna-se assim uma boa base para realizar o cálculo da Matriz de Transformação, pois permite a utilização de código em C disponível na documentação para as funções mais complexas. O atraso introduzido devido ao baixo desempenho não é limitativo, pois essa parte do projecto não tem restrições de tempo-real. Pela sua simplicidade, o barramento FSL será o utilizado no desenvolvimento do projecto.

Apesar da execução de instruções ser mais lenta do que num CPU normal, o MicroBlaze beneficia bastante por estar implementado numa FPGA, permitindo assim distribuir processamento por módulos mais especializados.

Na FPGA disponível para este projecto, o MicroBlaze tem acesso a no máximo 64KBytes de memória BRAM e funciona no máximo a 50 MHz. A sua implementação ocupa cerca de 1400 Slices de um total de 13400, ou seja, cerca de 10% da área total da FPGA. Já nos recursos dedicados, com a unidade de cálculo com vírgula flutuante activada, utiliza 7 multiplicadores de um total de 32.

3.4 Sistema de Desenvolvimento

Para o desenvolvimento de todo o projecto serão utilizadas as ferramentas fornecidas pelo fabricante. A placa da Avnet contém um chip FPGA da *Xilinx* e disponibiliza os ficheiros necessários a iniciar um novo projecto nas ferramentas da *Xilinx* .

Como já referido, o projecto aqui desenvolvido terá duas fases distintas de processamento: um cálculo da matriz de transformação possivelmente em MicroBlaze; e a rectificação em tempo real das imagens de vídeo. Assim será necessário utilizar várias ferramentas distintas, disponíveis nos pacotes *Xilinx ISE Design Suite 10.1* e *Xilinx EDK* (Embedded Development Kit) [19]. Estes pacotes consistem num conjunto de ferramentas de software, fornecidas pela *Xilinx* que suporta as várias fases de um projecto, desde a sua criação, desenvolvimento, algum teste, sintetização e implementação, tanto em *Softcore*s como em síntese lógica. Deste conjunto os programas mais utilizados foram:

***Project Navigator*:** Utilizado para a criação, desenvolvimento e sintetização da parte lógica do sistema: Rectificação de imagens e módulos de suporte ao MicroBlaze;

***Xilinx Platform Studio*:** Utilizado para todo o processo de implementação do *Softcore* MicroBlaze. Este programa inclui compilador da linguagem C e permite sintetizar módulos criados pelo Project Navigator.

***CORE Generator*:** Ferramenta de criação de módulos auxiliares, pré-desenhados e optimizados pelo fabricante. Estes módulos são parametrizáveis, permitindo gerar módulos específicos às nossas necessidades sem *overhead* excessivo.

***iMPACT*:** É uma ferramenta integrada no XPS e no Project Navigator, que permite a configuração da FPGA com o código gerado por estes, através de uma variedade de cabos de ligação (incluindo o *Paralel Cable IV* utilizado).

3.5 Ferramentas para Simulação e Validação

Simulação e teste é uma tarefa muito importante e morosa no desenvolvimento de sistemas.

Para a simulação dos módulos de hardware, as ferramentas de desenvolvimento apresentadas na secção anterior incluem a capacidade de integrar ferramentas de simulação. O software utilizado nesta dissertação foi o **ModelSim XE III 6.1e**, da Mentor Graphics.

Estes simuladores necessitam da criação prévia de ficheiros de teste aos módulos criados, denominados de Testbenchs. Um testbench é um ficheiro escrito em linguagem Verilog, no qual se invoca o módulo que se pretende testar. Este permite aplicar um sinal de estímulo nos pinos de entrada do módulo, simulando o comportamento real do sistema e permitindo observar os resultados obtidos. Durante a simulação é registado o estado e transições de todos os registos e ligações internas do módulo, tornando possível a sua análise temporal.

Como a simulação por testbench é morosa de ser realizada, esta foi realizada apenas para os módulos mais simples de suporte. Estes módulos juntamente com o MicroBlaze permitem que seja realizado um depuramento rápido de qualquer outro módulo, a partir de um terminal série de um computador.

Quanto à simulação e teste dos algoritmos matemáticos a implementar, a escolha residia principalmente entre dois candidatos: Matlab, da Mathworks, ou Excel da Microsoft. O Excel não é uma ferramenta normalmente escolhida para simulação de algoritmos, pois as bibliotecas de funções já implementadas é reduzida e a programação com código é complicada. Após uma busca pela internet foi encontrada em [20] uma macro de Excel que realiza de forma fácil e rápida todas as operações matriciais necessárias à implementação dos algoritmos estudados (SVD, inversa, etc.). Por isto, e pela simplicidade de utilização e de edição de valores numéricos, foi escolhido o Excel como programa principal de simulação de algoritmos.

A figura 5.4 é um exemplo de uma folha de Excel utilizada para aprendizagem e testes. Na imagem são visíveis as células a verde que se deve preencher com a deformação a provocar sobre uma imagem. São calculadas instantaneamente as coordenadas de correspondências a utilizar-se nos testes do cálculo da matriz e a azul é representada a matriz correcta de transformação.

O gráfico actualizado também instantaneamente permite-nos ter uma noção mais prática dessa deformação.

3.6 Desenvolvimento dos Módulos de Suporte

Como foi referido nas secções anteriores, o projecto será implementado no núcleo da placa de desenvolvimento, uma FPGA. Nesta será necessário implementar o módulo de transformação das imagens e também um *Softcore* que realizará o cálculo das Matrizes de Transformação. Estas duas implementações principais serão introduzidas no capítulo seguinte.

Nesta secção serão explicados os módulos de suporte que foi necessário desenvolver. A função principal destes é simplificar a interacção entre os recursos externos e os módulos principais. Nesta secção será explicada a biblioteca de funcionalidades implementadas por estes módulos, que vêm introduzir uma certa abstracção no acesso aos recursos disponíveis na placa de desenvolvimento.

Numa perspectiva temporal de toda a implementação do sistema, foi decidido implementar-se primeiro os módulos de suporte, com o intuito de simplificar a depuração e evitar a criação de alguns Testbenchs.

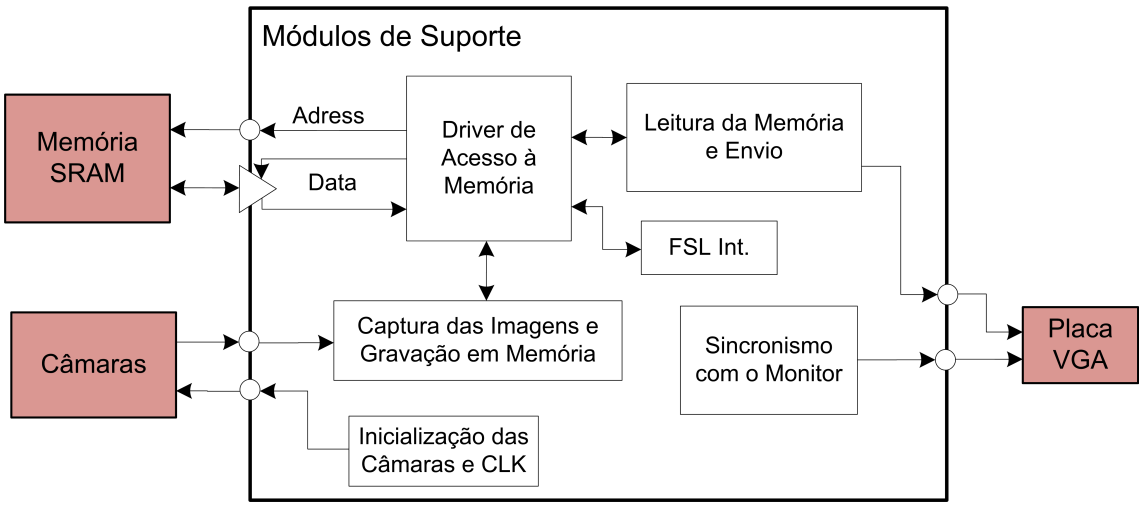


Figura 3.8: Módulos de suporte

Na figura 3.8 estão representadas as funcionalidades implementadas pelos módulos explicados nesta secção. Os módulos a vermelho são os recursos externos à FPGA, apresentados anteriormente neste capítulo.

Cada funcionalidade descrita pode corresponder a um ou mais módulos Verilog. Isto é devido à capacidade de se organizar o código Verilog em módulos simples e instanciá-los uma ou mais vezes dentro de outros módulos. O intuito desta dissertação não é mencionar a organização da implementação, mas sim as funcionalidades implementadas, pelo que as referências a código Verilog serão evitadas.

3.6.1 Inicialização das Câmaras

Como o nome indica, esta funcionalidade é necessariamente executada quando se reinicia a placa. Os módulos base que a implementam foram obtidos de um projecto da cadeira de PSDI (Projecto de Sistemas Digitais) leccionada na FEUP, [21]. Como o projecto de onde foram retirados os módulos só incluía uma câmara, foi necessário realizar-se algumas alterações.

Com a utilização desses módulos em duplicado detectou-se alguma falta de sincronismo nos sinais de controlo provenientes das câmaras. Após alguma pesquisa detectou-se que as câmaras, apesar de terem o mesmo barramento de dados e serem do mesmo modelo, nem sempre concluíam a inicialização no mesmo ciclo de relógio. Assim, foi decido alterar o módulo de maneira a que este não configure as duas câmaras independentemente, mas sim com o mesmo barramento de dados e assim obrigatoriamente ao mesmo tempo. Após esta alteração, os problemas de sincronismo com as câmaras ficaram resolvidos.

O funcionamento das câmaras pode ser configurado em qualquer momento através de um barramento I2C, com a alteração de alguns registos. As funcionalidades possíveis de serem modificadas através desta configuração são muito variadas e podem ser consultadas na Data Sheet do sensor OV7620, disponível em [18]. O módulo obtido de inicialização das câmaras já incluía a comunicação I2C para configuração das câmaras, facilitando assim qualquer configuração extra a realizar.

Durante a realização de diversos testes reparou-se que as zonas escuras das imagens obtidas encontravam-se muito saturadas. Assim, decidiu-se alterar o controlo de luminosidade interno das câmaras. Após a alteração dos registos correspondentes por I2C, as imagens ficaram mais claras e com as zonas escuras menos saturadas.

3.6.2 Sincronismo com o Monitor

Outro módulo necessário é o de sincronização com o monitor. Como já foi referido, a placa VGA disponível simplifica em muito esta sincronização, bastando enviar-lhe alguns sinais de sincronização e a imagem a mostrar no monitor. Este módulo foi também retirado do projecto de PSDI.

Como a placa VGA contém FIFOs com dois sinais de relógio independentes para leitura e escrita, a leitura pode ser completamente independente da escrita. Assim, os sinais que controlam a leitura da imagem das FIFOs e envio para o monitor são gerados por este módulo e não dependem de nenhum outro módulo ou sinal. A escrita das imagens nas FIFOs também é de implementação fácil, bastando o envio de dados ser síncrono com o sinal de relógio, e o sinal de activação de escrita ser activado quando se desejar escrever.

Neste projecto foram utilizados sinais de relógio de escrita e leitura a 25 MHz. Isto corresponde a uma frequência de refrescamento do monitor de 60 Hz, ou seja, cada FIFO é lida 60 vezes por segundo. A escrita nas FIFOs não está restringida a nenhuma cadência de imagens, mas as restrições temporais do projecto são de, no mínimo, 25 Imagens por segundo que foi a cadência realizada.

3.6.3 Driver de Acesso à Memória

Como foi referido nas secções anteriores, será necessário utilizar as memórias SRAM de 2MB externas à FPGA para armazenar as imagens a processar.

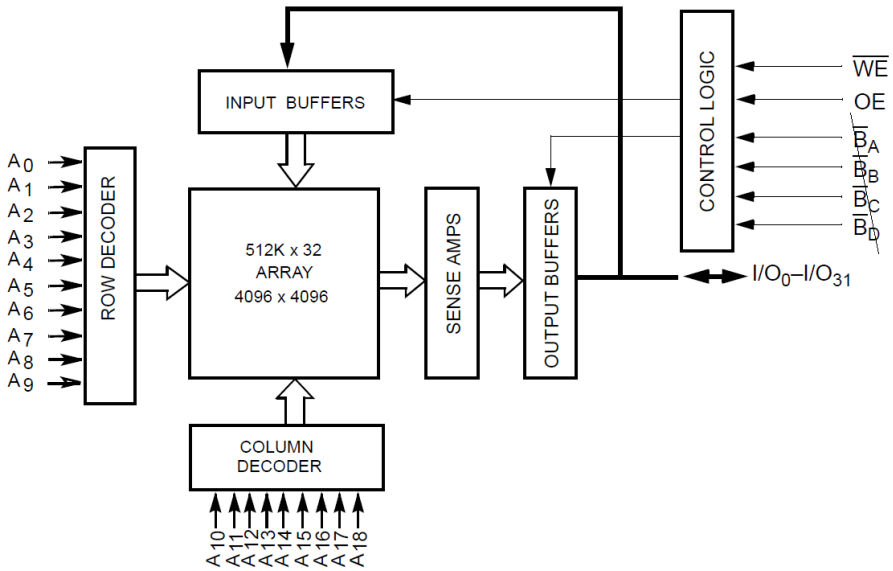


Figura 3.9: Organização e barramentos da SRAM

Estas são memórias RAM estáticas do modelo CY7C1062AV33 da Cypress [22] e estão organizadas em 512K palavras de 32 bits. Assim, existe um barramento de 19bits para seleccionar o endereço da palavra a ler e um barramento de dados de 32bits. Para simplificar o controlo de memória na gravação e leitura de imagens, foi utilizada a seguinte distribuição: $Endereo = pos, Y, X'$, em que X' são os 8 bits mais significativos da coordenada X da imagem[1], Y são os 9 bits da coordenada Y, e pos é um código de 2 bits diferente para cada câmara. Foi utilizado $pos = 00$ para a imagem da câmara da direita, e $pos = 01$ para a imagem da esquerda. As outras duas posições são utilizadas para informação visual e testes, como será descrito posteriormente. Como os valores de X' e Y são no máximo 159 e 479, respectivamente, a memória não é toda utilizada, porém o restante não foi aproveitado por não haver necessidade.

Como se pode observar na figura 3.9, o barramento de dados é único e responsável pela transmissão e recepção de dados. Este facto complica a implementação deste módulo, tornando necessário um driver de controlo responsável por gerir estes barramentos. Este driver terá de garantir que só um acesso à memória é realizado em cada instante e que os timings são cumpridos.

Para cumprir os timings necessários ao bom funcionamento foi analisada a Data Sheet da memória disponível em 3.9. Como os tempos de operação mínimos são de 12 ns e a frequência máxima de relógio da FPGA é de 100 MHz, decidiu-se implementar os ciclos de escrita e leitura em 2 ciclos de relógio, 20 ns. Para compensar o facto de a memória ser externa à FPGA foram registadas as entradas e saídas nos IOB, minimizando o atraso externo entre os pinos da FPGA e

[1] Os 2 bits menos significativos são ignorados pois corresponde à posição dentro da palavra guardada em memória, que é de 4 bytes

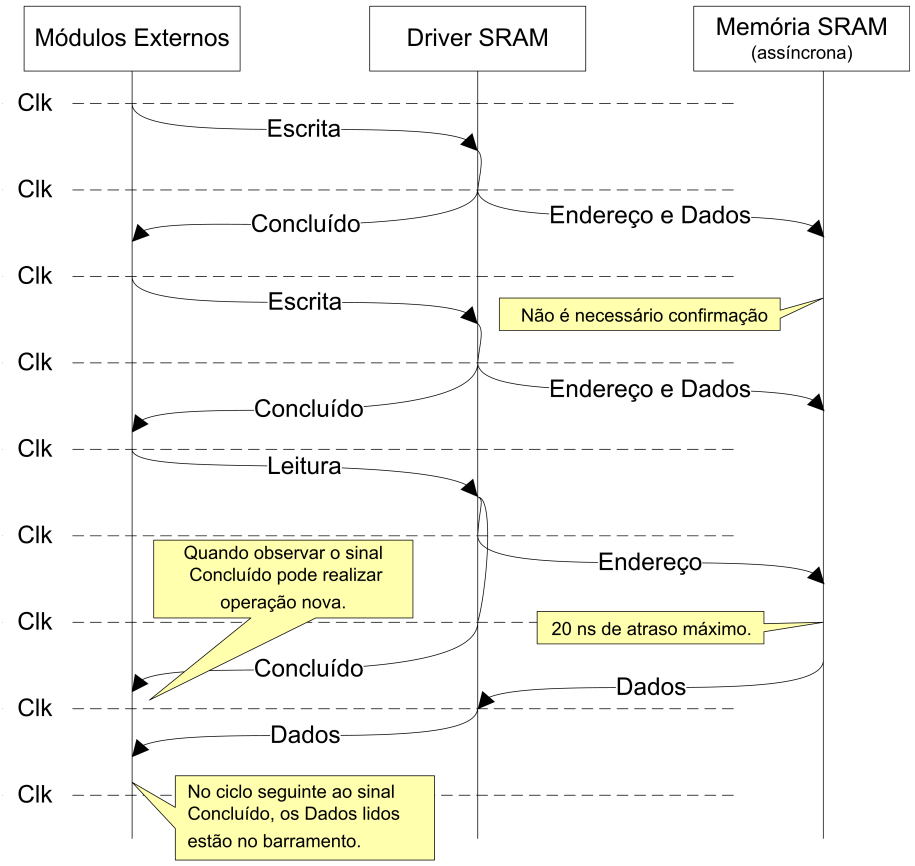


Figura 3.10: Funcionamento e timings do driver SRAM

os pinos da memória. O diagrama 3.10 representa os timings mínimos nas diferentes operações. Como é visível, com um relógio de 100 MHz, período de 10 ns, o funcionamento externo máximo numa operação de leitura é de 20 ns. Isto equivale a 12 ns para a operação da memória e 8 ns para o atraso das linhas entre FPGA e memória, 4 ns para cada lado. Os testes realizados deram positivos, mostrando que os 4 ns são suficientes.

Como é visível no diagrama, um módulo externo pode realizar operações distintas de escrita a cada 2 ciclos de relógio. Numa operação de leitura, os dados demoram 4 ciclos a estarem disponíveis, mas pode ser realizada outra operação por outro módulo após 2 ciclos sem perda de dados.

O driver de controlo da memória tem um funcionamento crítico para o sistema, logo foram realizados testes exaustivos para garantir o bom funcionamento com os timings implementados.

Para permitir que vários módulos "agendem" operações, tanto de leitura como de escrita, foram criados quatro canais de comunicação para com este módulo: dois para operações de leitura e dois de escrita. Para evitar conflitos, cada canal tem a sua prioridade fixa. Será necessário evitar situações de Starving[2], em que um certo canal tem de esperar tempo excessivo para obter permissão.

[2]Starving - Situação em que o pedido de um dos módulos nunca é executado por haver módulos de prioridade superior com pedidos excessivos.

A cada módulo é associado um ou mais canais e cada canal tem uma função e uma prioridade diferente. Os canais permitem que um módulo que utilize dois canais de leitura seja capaz de realizar uma operação de leitura a cada 2 ciclos, como se fossem dois módulos distintos.

Os canais foram distribuídos pelos seguintes módulos segundo esta prioridade:

1. Canal de escrita das imagens obtidas das câmaras.
2. Canal de leitura partilhado com o módulo de Leitura da memória para a VGA e o de transformação da imagem direita.
3. Canal de leitura partilhado com o módulo de FSL e o de transformação da imagem esquerda.
4. Canal de escrita utilizado pelo módulo FSL e pelo módulo de transformação para a realização de testes.

Os factores mais importantes para a escolha das prioridades foram os de atraso máximo na concretização. A escrita das imagens, como veremos na secção seguinte, ocorre a cada 32 ciclos de relógio e dura 4 ciclos (2 operações). A escrita e leitura pelo módulo FSL não possui duração máxima nem mínima, nem tempo de espera máximo, sendo apenas necessário evitar starvings. O módulo VGA não possui também restrições temporais de execução, mas o mesmo canal é partilhado com o módulo de transformação das imagens e este necessitará de 24 ciclos a cada 32, com atraso máximo de 8 ciclos. A utilização dos canais do módulo de transformação serão os mais críticos, mas este funcionará apenas em paralelo com o módulo de escrita das câmaras, ficando assim disponíveis 4 ciclos em cada 32.

3.6.4 Câmaras -> Memória

Este módulo é o responsável pela recepção das imagens das câmaras e gravação das mesmas na memória. Cada câmara tem o seu barramento de dados de 8 bits, que transmite com uma cadência de 12.5 MHz, o que é equivalente a 1 pixel cada 8 ciclos de relógio por cada câmara. Como a gravação em memória é realizada em grupos de 32 bits, este módulo tem um contador interno e procederá à gravação de um grupo de 4 pixéis por imagem a cada 32 ciclos. Este módulo é também o responsável por descartar os dados obtidos durante os sinais de controlo.

Através dos interruptores DIP 6 7 e 8, o utilizador consegue controlar o desfasamento entre o sinal de relógio das câmaras e da escrita em memória, evitando erros de sincronização. Este controlo é necessário pois o cabo de conexão entre as câmaras e a placa introduz um atraso, apesar de raramente haver necessidade de serem alterados estes interruptores.

3.6.5 Memória -> VGA

O sistema geral não necessita deste módulo para realizar a sua função, mas este é importante, pois dá informação ao utilizador em tempo-real do que está a ser executado na FPGA. Essa informação é transmitida em formato de três imagens, denominada por leitura a 3 cores. Assim, a imagem guardada na memória na posição 00 (câmara da direita quando módulo de gravação está

activo) é enviada para a FIFO que corresponde à cor vermelha, a posição 01 (imagem da câmara esquerda) é enviada para a FIFO responsável pela cor azul, e a posição 10 da memória SRAM, alterada exclusivamente pelo *Softcore* através do módulo Verilog, é enviada para a FIFO de cor verde. Este modo de visualização é mais informativo do que a visualização de uma imagem de cada vez a preto e branco.

Figura 3.11: Óculos 3D

A visualização da imagem da câmara direita a vermelho e a da esquerda a azul, permite ainda a visualização da imagem a 3D com a utilização de uns óculos de lentes coloridas, como mostra a imagem 3.11. O intuito de utilizar os óculos 3D era para testar a qualidade da rectificação: se a imagem no monitor aparecesse nítida a 3D estaria correctamente rectificada; em caso contrário, apareceria com fantasmas, isto é, objectos semi-transparentes e duplicados. Este teste não pode ser utilizado pois o cérebro humano tem a capacidade de rectificar as imagens automaticamente, visualizando-se fantasmas no monitor apenas em situações de extrema deformação das imagens.

Para o teste do bom funcionamento deste módulo, uma coluna dinâmica estreita e de cor verde percorre constantemente o ecrã. Desta forma é possível saber quando este módulo está em execução e se as FIFOs estão a gerar sinais correctos para o monitor. Alguns erros no módulo de sincronismo com o monitor foram detectados e corrigidos desta forma.

Assim como o módulo anterior, este módulo contém um desfasamento nos sinais de relógio controlados pelos interruptores DIP 1 2 e 3. Raramente há necessidade de serem alterados.

3.6.6 FSL Interpreter

Devido à necessidade de inclusão de um *Softcore* na FPGA, foi implementado este módulo. É assim o responsável pelas comunicações entre o *Softcore* e os restantes módulos.

Como o *Softcore* será o responsável pela maioria dos cálculos necessários neste projecto, é este módulo que faz a gestão global de todo o sistema, tendo assim capacidade de activar e desactivar módulos e sinais de controlo.

Este módulo comunica com o MicroBlaze através de um barramento FSL. Para a largura do barramento decidiu-se utilizar 32bits por ser a largura do barramento de dados da memória. Decidiu-se que destes 32 bits, 12 seriam para a funcionalidade ou comando a executar, e os outros 20 ou seguintes ciclos de 32 seriam dados a transmitir. Assim, por exemplo, uma ordem de escrita em memória necessita de duas comunicações FSL: uma com 12 bits de comando e 19 de endereço a gravar e outra com os 32bits a gravar na memória.

Com o intuito de poupar memória e tempo de execução ao *Softcore*, foram implementadas em Verilog e neste módulo algumas funções de apoio. Como o compilador do *Softcore* compila automaticamente todas as bibliotecas necessárias à execução do código, ao passar uma função do C para o módulo FSL (Verilog) é poupado o espaço ocupado pela biblioteca. As funções mais simples poderão demorar mais tempo a executar em Verilog que em C, pelo que é necessário realizar um trade-off entre desempenho e memória ocupada.

Algumas funções implementadas no módulo FSL foram:

- Funções de escrita e leitura da memória, com barramento de 32 bits.
- Controlo de alguns módulos, permitem entre outros activar e desactivar uma ou ambas as câmaras e a transformação das imagens.
- Uma função denominada de “Wait” substitui o habitual “sleep”, que aproveitando o sinal de sincronismo vertical do monitor implementa facilmente uma função de espera, em que a unidade de tempo base é o número de frames.
- Função para ler um grupo de 5x5 pixéis em torno de uma coordenada dada.
- Função de limpar toda a imagem da posição 10 (verde).
- Função para desenhar um $+$ ou um $\times$ na coordenada dada.
- Função de recepção das Matrizes de Transformação. Esta guarda as matrizes em registos, podendo ser utilizados pelo módulo de transformação apresentado no capítulo seguinte.

Para além destas funções houve ainda necessidade de implementar um módulo baseado no algoritmo CORDIC, para cálculo de Raízes Quadradas. Este cálculo é muito moroso em *Softcore* e a sua implementação ocupa bastante memória. A implementação do CORDIC na FPGA foi parametrizada para ter a maior precisão possível, que corresponde a 48bits de entrada e 24 de saída. Ambos são representados como inteiros sem sinal, ficando a cargo do *Softcore* a passagem de números decimais para inteiros, como veremos na secção seguinte.

Outras funções, como “Incrementar LEDs”e “PING”, que desempenhavam papéis unicamente de simulação e sincronismo foram comentadas após a conclusão das simulações, com o intuito de poupar memória no *Softcore*.

3.6.7 Integração com o EDK

Após o desenvolvimento dos módulos de suporte, e durante o desenvolvimento do módulo interpretador de FSL, foi exportado o projecto para o EDK com o intuito de integrar o *Softcore* na FPGA.

As ferramentas da *Xilinx* mostraram alguma incompatibilidade na sintaxe dos ficheiros, nomeadamente nas portas Tri-State. O Project Navigator suporta a declaração de portas InOut, enquanto o EDK obriga a declaração de 3 portas: In de dados, Out de dados e Out de (des)activação de alta impedância. Visto que o projecto possui várias portas InOut, foi necessário alterar o TopLevel do projecto.

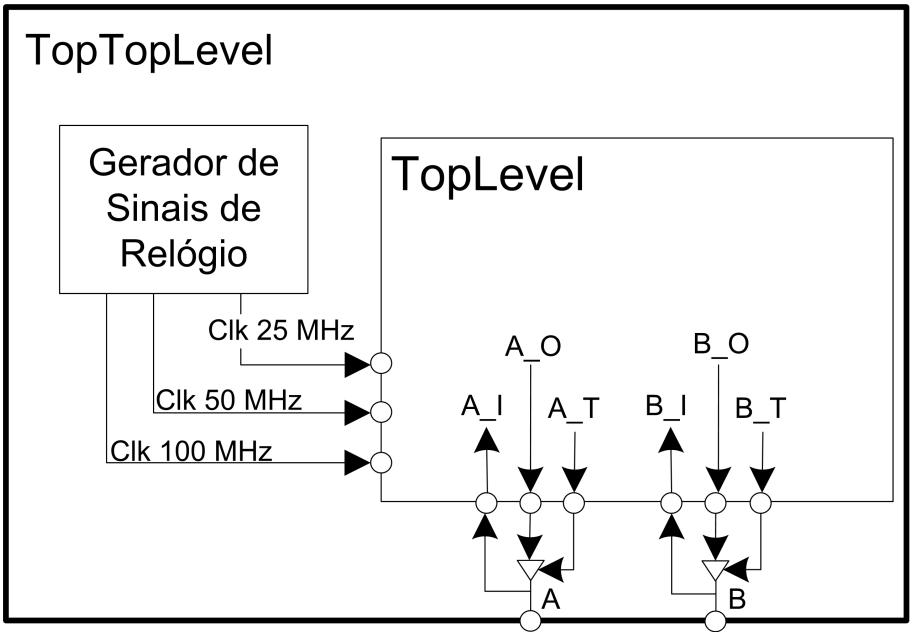


Figura 3.12: TopTopLevel do sistema

Para manter a compatibilidade com o Project Navigator e com o EDK decidiu-se utilizar dois ficheiros TopLevel, como mostra a figura 3.12. Isto tornou possível o teste e edição dos módulos Verilog através do Project Navigator, sem a necessidade de criar um projecto novo no EDK.

Outro problema existente na integração consiste no gerador de sinais de relógio. Para garantir a sincronização entre os sinais de relógio do Microblaze, da comunicação FSL e do resto do sistema, estes devem ser gerados por um único gerador de sinal. A maneira mais simples é utilizar o gerador já existente em todos os projectos do EDK. Contudo, este facto dificulta o teste dos módulos no Project Navigator. A solução mais simples encontrada foi a de inclusão de um gerador de sinais de relógio no TopLevel externo ao EDK, denominado de TopTopLevel na figura 3.12. Assim, é possível editar os ficheiros em ambos os lados, sem ser necessário a criação de um novo projecto.

Para a configuração dos pinos externos da FPGA com os recursos externos foi necessária a utilização de um ficheiro UCF. O ficheiro UCF foi obtido do trabalho do aluno Marc Antunes no seu projecto de dissertação no ano 2006/2007. Este incluía já configuração de todas as portas necessárias à realização deste projecto.

Para facilitar a interface entre o utilizador e a FPGA, foi implementado um controlador XuartLite. Este é um controlador de comunicação série por RS232, logo foi utilizado o programa Hyperterminal para a comunicação por parte do computador. A configuração de velocidade utilizada foi a máxima suportada pela porta, 115.200 bps, no sentido de diminuir os tempos de comunicação.

No sentido de aumentar a memória disponível, tentou-se implementar um controlador de memória externa que permitisse ao MicroBlaze utilizá-la como memória de instruções e dados. Devido a incompatibilidades da placa de desenvolvimento com as últimas versões do software da *Xilinx* não foi possível a sua concretização.

3.6.8 Softcore: Funções Base e Menu de Testes

Após a integração do MicroBlaze na FPGA foi necessário escrever em C a biblioteca de funções realizadas em parceria com o FSL Interpreter. Essas funções são bastante simples, bastando na maioria dos casos uma escrita e leitura do barramento FSL. Outras merecem mais atenção, como é o caso da de cálculo das Raízes Quadradas.

Como vimos na secção anterior, a função implementada com o algoritmo CORDIC para calcular Raízes Quadradas tem como entrada e saída um número inteiro. Como os cálculos no *Softcore* são realizados utilizando variáveis de vírgula flutuante, de 32 bits, foi necessário proceder a uma transformação destes números para inteiros e vice-versa. Esta transformação tem também de ter em conta que é necessário realizar duas escritas no barramento FSL para transmitir todos os 48 bits a calcular: uma escrita de 16bits+12 de comando e outra de 32 bits. Assim, a transformação de vírgula flutuante para inteiro sem sinal de 48bits e posterior transformação de 24 bits para vírgula flutuante, é realizada com os seguintes procedimentos:

1. Iterativa divisão do valor a calcular por 4.0 até este se tornar menor que 4.0. O número de iterações necessárias é guardado. Este passo coloca valores superiores a 4.0 no intervalo $[1.0,\ 4.0[$.

2. Iterativa multiplicação do valor a calcular por 4.0 até este se tornar maior ou igual a 1.0. O número de iterações necessárias é também guardado. Este passo coloca valores inferiores a 1.0 no intervalo $[1.0,\ 4.0[$. Todas as seguintes operações serão realizadas sobre este valor. Estes cálculos não introduzem perda de precisão por serem realizados numa variável de vírgula flutuante.

3. Multiplicação do valor obtido por $0x4000$ e retipagem do resultado para inteiro, colocando assim a parte inteira do valor nos bits 16 e 15, e a parte decimal nos restantes 14 bits. Este passo não introduz perda de precisão nos 16 bits resultantes.

4. Realização da mesma multiplicação em vírgula flutuante e subtracção do resultado ao valor original.

5. Multiplicação do valor obtido no passo anterior por 0x100000000 e retipagem do resultado para inteiro. Este passo produz uma variável de 32 bits correspondentes aos bits [15 a 46] da parte decimal do valor a calcular.

6. Envio por FSL dos dois inteiros obtidos nos passos anteriores.

7. Leitura do barramento FSL. Os 24 bits menos significativos contêm o resultado da operação. Como foi enviado um valor no formato [2 bits inteiros, 46 decimais], o resultado obtido está no formato [1 bit inteiro, 23 decimais].

8. É então realizada uma divisão em vírgula flutuante do valor obtido por 0x800000 e o resultado guardado numa variável de vírgula flutuante.

9. O valor é multiplicado então por 2^i e dividido por 2^j, em que i e j foram o número de iterações necessários nos passos 1 e 2, respectivamente.

Outra funcionalidade que mereceu mais cuidado foi a leitura de um grupo de pixéis da memória. Um grupo de pixéis consiste num bloco de 5 x 5 pixéis. Como a memória está organizada em sectores com 4 bytes/pixéis de tamanho, a operação de ler um grupo necessita sempre de 10 operações de leitura. Assim, ao analisar-se a imagem toda, tem de se realizar $640 \cdot 480 \cdot 10 = 3.072.000$ operações de leitura da memória. Para diminuir o número de leituras existem duas soluções: analisar a imagem horizontalmente linha a linha, lendo só os 5 pixéis da direita em cada ciclo, ou verticalmente coluna a coluna, analisando os 5 pixéis inferiores. Como uma leitura dos 5 pixéis inferiores representa só duas operações de leitura, ficou assim decidido que a imagem seria analisada verticalmente, coluna por coluna.

Após a implementação da optimização descrita, para a análise integral da imagem são necessários cerca de $640 \cdot 480 \cdot 2 = 614.400$ operações de leitura.

De notar que por ser realizado varrimento vertical, este não pode ser realizado enquanto as câmaras estão a gravar imagens nas memórias, pois assim estaríamos a observar partes de um *frame* recente na parte superior e a parte inferior seria do *frame* anterior. Como o algoritmo de análise das imagens deverá correr em imagens estáticas, com as câmaras desligadas, este factor não é limitativo.

Foi também escrita uma biblioteca de funções para tratamento matricial. As funções simples de multiplicar e de inverter matrizes foram escritas de base, enquanto que a de calcular a SVD foi retirada de [7]. Esta última requer algumas funções, que foram escritas de base da forma mais simples possível para poupar espaço na memória de programas. As invocações às funções de cálculo da raiz quadrada foram substituídas pela função com o mesmo fim implementada em hardware com o algoritmo CORDIC

Todas as funções utilizadas foram alteradas para incluírem a escrita no barramento série de informação de depuração. Com o intuito de poupar memória, estas funções de escrita no barramento estão condicionadas a uma variável chamada DEBUG. Caso seja declarada como 0, todas as funcionalidades de depuração e teste não são incluídas, permitindo utilizar a memória libertada para outros fins.

A interface série permite ao utilizador controlar e visualizar em *Runtime* a execução dos algoritmos, através de um menu. Este menu recebe comandos do utilizador, executando a função correspondente e mostrando todas as mensagens de depuração e é descrito no capítulo 5. Este menu foi constantemente alterado ao longo do desenvolvimento do capítulo seguinte, de forma a permitir o teste das funções e algoritmos implementados. No fim do desenvolvimento este menu foi minimizado no intuito de poupar memória de programas.

Capítulo 4

Solução Proposta

Este capítulo descreve a implementação dos módulos e funções que fazem parte da solução proposta. Será dada uma breve descrição geral da solução e de seguida é explicado cada módulo e função detalhadamente.

4.1 Descrição Geral

A figua 4.1 mostra os módulos e barramentos principais do sistema. Os módulos a sombreado vermelho são os recursos externos à FPGA e a branco são os módulos de suporte à solução. Estes já foram abordados no capítulo anterior. A solução proposta é implementada resumidamente em duas partes distintas:

- A sombreado azul estão representadas as funções implementadas no Microblaze, que realizam a análise de duas imagens das câmaras e cálculo da matriz de transformação;
- A verde está o módulo de transformação das imagens, que aplica em tempo real a matriz calculada ao vídeo proveniente das câmaras.

O funcionamento global do sistema é comandado pelo MicroBlaze e a sequência é a descrita na figura 4.2. A azul estão as funcionalidades implementadas no MicroBlaze e a verde estão as implementadas no módulo de rectificação ou transformação.

1. O sistema inicializa as câmaras e espera um determinado número de frames para o controlo de brilho estabilizar. De seguida, as câmaras são desactivadas, estabilizando a imagem guardada na memória RAM;
2. O MicroBlaze analisa a imagem da direita e selecciona as melhores características como candidatas;
3. Para cada candidato, é realizada uma procura por correspondências na imagem da esquerda. A busca é realizada em torno da recta epipolar calculada na última iteração. Inicialmente é considerada uma recta epipolar horizontal com a mesma coordenada Y que o candidato. As melhores correspondências são guardadas;

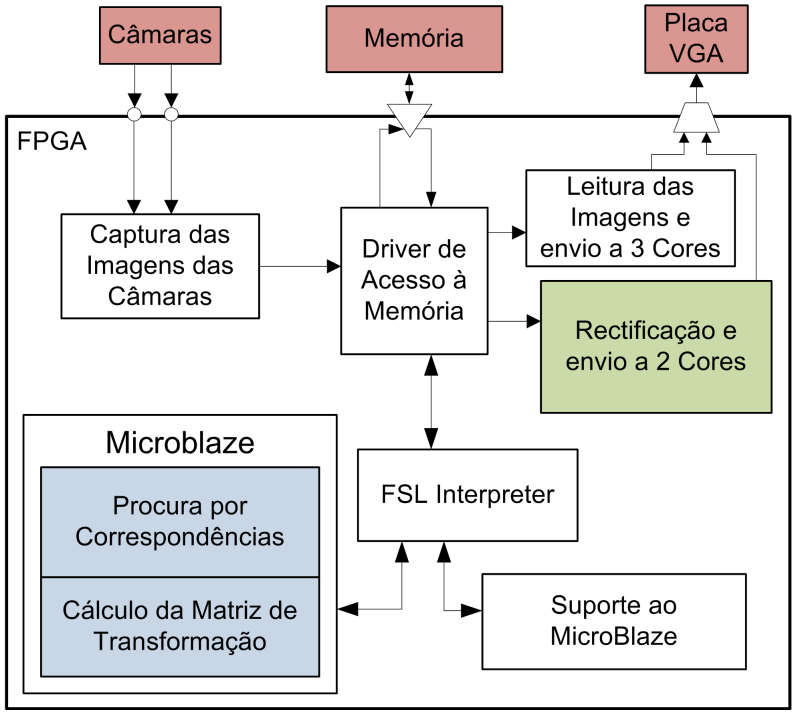


Figura 4.1: Módulos e barramentos principais

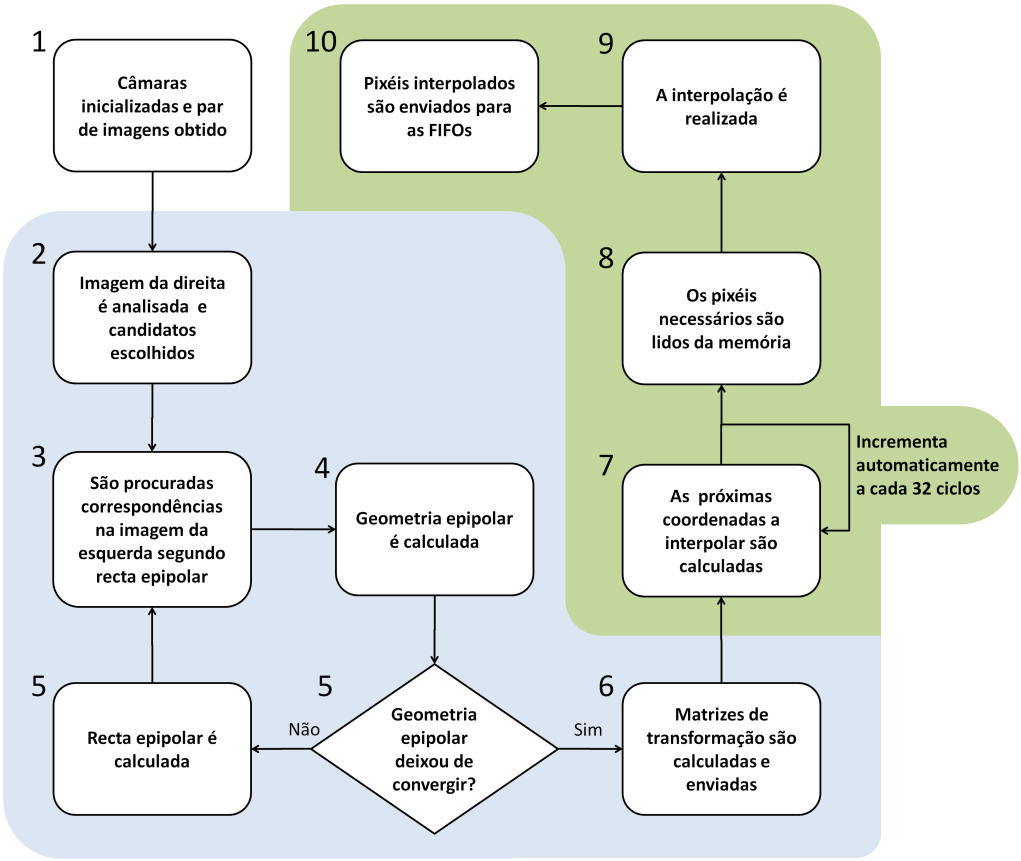


Figura 4.2: Diagrama geral da solução proposta

4. As coordenadas dos candidatos e correspondências são utilizadas para calcular a geometria epipolar;

5. Um método iterativo repete os pontos 3 e 4 até a geometria epipolar estabilizar;

6. As coordenadas e a geometria epipolar é utilizada para calcular as matrizes de transformação e estas são enviadas por FSL para registos na FPGA;

7. O módulo de transformação calcula para um grupo de 4 pixéis as coordenadas a interpolar das imagens das câmaras;

8. Os grupos de pixéis necessários para a interpolação das coordenadas calculadas são lidos da memória e guardados em registos;

9. Interpolação bi-linear é realizada em torno das coordenadas calculadas;

10. O resultado da interpolação é enviado para a placa VGA e mostrado no ecrã. A imagem da câmara direita é mostrada a vermelho e a da esquerda é mostrada a azul;

11. O ponto 7 é repetido a cada 32 ciclos para os pixéis de uma imagem toda e, novamente, no inicio de um par de imagens novas.

Estes passos são descritos com mais detalhe nas secções seguintes.

4.2 Cálculo das Matrizes de Transformação

Nesta secção será explicada a sequência de processamento executada no MicroBlaze.

Mais detalhadamente, a sequência a azul da figura 4.2 é constituída por uma série de funções e algoritmos, completamente parametrizáveis. Parte dos algoritmos foram desenhados com base noutros, propostos na literatura, e outra parte foram desenvolvidos.

Estes algoritmos podem-se dividir em duas classes distintas: os que realizam a busca de correspondências nas imagens e os que calculam as matrizes de transformação. Para além destes, existe outra funcionalidade que permite o melhoramento iterativo dos resultados.

4.2.1 Procura de Correspondências

Foi criada então uma biblioteca de funções necessárias à busca de correspondências entre as duas imagens na memória.

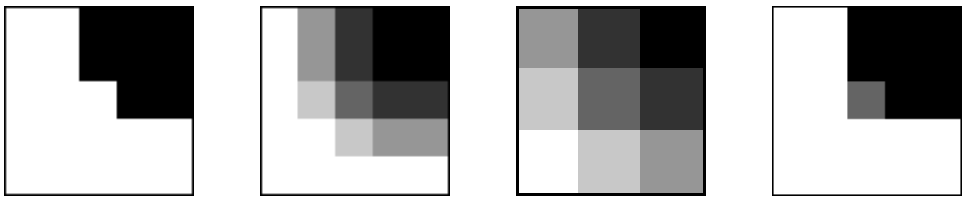
Anteriormente em 2.2 foi visto que a fiabilidade do cálculo das matrizes de transformação aumenta com a dispersão das correspondências encontradas. Para evitar, então, que todos os pares encontrados pertençam ao mesmo objecto, decidiu-se dividir a imagem em pequenos blocos de tamanho fixo. O tamanho dos blocos utilizados é de 128 x 80 pixéis devido à falta de capacidade da memória para armazenar os dados, no entanto, em testes foi concluído que a qualidade da solução

aumenta com a diminuição do tamanho destes blocos.

Após esta divisão é aplicada a seguinte sequência de processamento à imagem:

1. A imagem da direita é varrida coluna a coluna, com uma janela de 5 x 5 pixéis lidos da memória.

   Devido aos cálculos mais importantes serem realizados na imagem de referência, esta deve ser a imagem da câmara com melhor qualidade. Neste projecto a qualidade das imagens é semelhante, pois as câmaras são do mesmo modelo. Por convenção decidiu-se utilizar a imagem da direita como imagem de referência.



(a) Canto visualizado (b) Janela 5x5 (c) Janela 3x3 (d) Janela 5x5 pós processamento

Figura 4.3: Tamanho da janela e processamento realizado

   A janela de análise foi inicialmente configurada com um tamanho de 3 x 3 pixéis. Devido ao problema de Demosaicing introduzido na secção 3.1 uma janela com esta dimensão não contém informação fidedigna da imagem, pois os seus pixéis são muito influenciados pelos vizinhos, como se pode visualizar na figura 4.3. Decidiu-se aumentar a janela para 5 x 5 pixéis e assim melhorar a confiança na informação da janela.

2. A janela é submetida a um detector de características, sendo descartada se não houver um interesse mínimo. Caso seja descartada, a sequência é reiniciada do ponto 1.

   Inicialmente, a janela é submetida a um processo que avalia cada pixel da janela comparando-o com o pixel central do candidato. Este processo indica para cada pixel, se a sua luminosidade é inferior, igual, ou superior à do pixel central. É considerado igual um pixel com uma diferença de luminosidade inferior a um valor de ruído, 20 neste projecto. A figura 4.3.(d) mostra o resultado deste processo no canto analisado anteriormente. Este método tem a propriedade de não fazer distinção entre objectos com muito ou pouco contraste, melhorando a análise de imagens com objectos de variadas intensidades e contrastes.

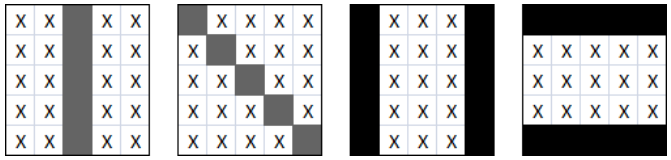


Figura 4.4: Detector de arestas e fios

A janela é, então, submetida a um detector de características não linear, e descartada caso corresponda a uma aresta ou fio, e não corresponda a um canto. A razão é que uma janela de um fio ou aresta é potencialmente parecida com outra do mesmo fio ou aresta, diminuindo a unicidade e levando a possíveis erros. Alguns padrões de janelas eliminadas são as representadas na figura 4.4.

3. O interesse (peso) da janela é calculado, e as melhores são consideradas candidatos. O interesse e as coordenadas dos melhores candidatos são guardados em memória.

   O interesse de uma janela representa a sua qualidade, e é calculado como o maior valor entre a soma dos pixéis de luminosidade inferior e a soma dos pixéis de luminosidade superior. Assim, o valor varia entre 0 e 24. Este método tem a propriedade de preferir os cantos mais apertados ou agudos e dar menor importância aos cantos arredondados. Por exemplo, a figura 4.3.(d) possui um interesse de 16, que corresponde à soma dos pixéis de intensidade menor, representados a branco. Uma janela que apresente um interesse inferior a 8 é descartada, assim como um interesse superior a 22, para descartar eventuais erros de leitura da memória ou picos luminosos derivados de ruído das câmaras.

   As intensidades e coordenadas são guardados, mas não poderá haver no mesmo bloco dois candidatos mais próximos que uma distância mínima. Assim, só são guardados os candidatos que estejam a uma distância superior a 3 pixéis afastados de outro candidato, eliminando os vícios dentro do mesmo bloco da imagem.

4. Após os candidatos de toda a imagem serem calculados, é iniciada a procura por correspondências. Para cada candidato, uma parte da imagem da esquerda é varrida com uma janela de 5 x 5. A zona a pesquisar é de tamanho parametrizável, sendo o utilizado em testes uma janela definida horizontalmente entre [-130, 50] e verticalmente entre [-60, 60], em coordenadas relativas à coordenada do candidato. Esta zona é suficiente para módulos de duas câmaras com uma disposição praticamente plana, a apontar para o mesmo sítio com um ângulo ligeiramente convergente, ou seja, quase rectificadas. Deve, portanto, ser ajustada aquando da utilização com outra disposição. O seu alargamento não reduz significativamente os resultados obtidos, mas aumenta bastante o tempo de computação necessário.

5. A janela é lida da imagem esquerda na memória e submetida ao mesmo processo do ponto 2: são detectadas as características, calculado o interesse e é descartada se não possuir um interesse mínimo. Este passo serve principalmente para aumentar a velocidade de busca, evitando cálculos desnecessários em janelas pouco interessantes.

6. A semelhança entre a janela e a janela do candidato é medida utilizando vários algoritmos com pesos diferentes. Estes algoritmos devem avaliar as janelas e avaliar com segurança a semelhança entre elas. Consistem em algoritmos tanto lineares como não lineares, para evitar que um mesmo objecto iluminado de maneira diferente ou com fundo diferente seja considerado semelhante.

As janelas de ambos os blocos são sujeitos ao algoritmo SAD[1] e o resultado é subtraído a um valor pré definido de 1000. O valor obtido representa a base da semelhança. Para refinar a semelhança é calculada a disparidade das janelas no resultado do processo representado no ponto 2 e na figura 4.3.(b,d). Esta disparidade varia entre 0 para janelas com o mesmo padrão e 46 para janelas completamente diferentes. Decidiu-se, então, multiplicar este valor por 20, aumentando a importância dada ao resultado, e subtrair à semelhança base.

Para desempatar decidiu-se incluir um peso para a disparidade da coordenada Y das janelas, dando preferência assim a janelas com a mesma coordenada vertical e também um peso ao interesse obtido no passo anterior, dando prioridade às correspondências entre janelas muito interessantes. Estes dois pesos são menores que os do parágrafo anterior, apenas desempatando situações um pouco ambíguas.

7. O resultado da semelhança e as coordenadas das correspondências mais semelhantes são guardadas em memória.

   Assim, para cada candidato da imagem direita é guardado um vector com as correspondências na imagem esquerda e respectivos pesos (semelhança). Quanto maior for o tamanho deste vector, melhores resultados se obterão nos passos seguintes. Neste projecto é utilizado um tamanho de 10, devido a restrições de memória.

8. A semelhança dos melhores pares é recalculada, agora utilizando uma janela maior em torno dos candidatos e respectivas correspondências possíveis. Este passo melhora a fiabilidade, fazendo sobressair o par correcto de entre todas as correspondências calculadas anteriormente.

   Não existe um tamanho óptimo para janela a utilizar. Quanto maior, mais informações se pode comparar. Contudo, devido à diferente perspectiva dos objectos nas câmaras, e possíveis diferenças no fundo de um mesmo objecto, o alargamento da imagem poderá diminuir erradamente o grau de confiança do par correcto. Optou-se, portanto, por se utilizar uma janela de tamanho 15 x 15.

   Para cada candidato, o vector de correspondências resultante do passo anterior contém as possíveis correspondências na outra imagem. É agora aplicado mais um refinamento à semelhança calculada. Para tal, é utilizada uma janela de 15 x 15 pixéis em torno dos pixéis a avaliar (candidato e cada correspondência possível) e é realizado o processamento apresentado nos pontos 5 e 6. A janela de 15 x 15 é avaliada como a soma das avaliações das 9 janelas de 5 x 5 que a constituem. O resultado obtido corresponde à nova semelhança entre o candidato e cada uma das correspondências.

9. É criada uma lista com os melhores pares de candidatos e correspondências.

   O peso dos pares nesta lista não é o peso calculado no ponto anterior, que indica a semelhança, mas sim um peso indicativo da unicidade do par. Ou seja, um candidato que tenha

[1] Algoritmo muito conhecido e apresentado em [23]. Consiste na soma das diferenças absolutas entre os valores de luminosidade dos pixéis de ambas as janelas. Este algoritmo é muito sensível a diferenças de luminosidade e ruído.

apenas uma correspondência com o valor elevado de semelhança é muito valorizado, e um candidato que tenha muitas possíveis correspondências com elevada semelhança é desvalorizado. Esta unicidade traduz a fiabilidade do par.

Para o cálculo da unicidade é utilizado um método simples: a unicidade é a diferença entre o valor da semelhança da correspondência mais semelhante e o da segunda mais semelhante. A lista consiste nos candidatos encontrados, a sua correspondência mais semelhante, e o peso associado, que é a unicidade aqui calculada.

Esta lista é ordenada pelos pesos e são apenas guardados os N melhores pares (40 utilizados neste projecto). Mesmo sobrando apenas os 40 melhores pares e dependendo das imagens analisadas, podem permanecer ainda alguns pares menos únicos. Assim estabeleceu-se outro critério, baseado na diferença do peso do primeiro elemento para o último. Para isso, todos os pares com peso inferior a $(MelhorPeso)/10$ são eliminados da lista.

Esta sequência produz uma lista com as coordenadas de pares de correspondência das duas imagens, com alguma segurança. Está assim concluída a procura de correspondências.

4.2.2 Melhoramento Iterativo das Correspondências

Para melhorar a fiabilidade das correspondências é ainda utilizado um método iterativo. Este utiliza as coordenadas encontradas para estimar a geometria epipolar e assim conseguir procurar de forma mais efectiva novas correspondências.

Como foi referido na secção 2.2, se for conhecida a recta epipolar de um ponto, o ponto correspondente na outra imagem encontra-se sobre a recta epipolar. Essa recta epipolar é definida como $l' = F \cdot Q$, em que F é a matriz fundamental e Q as coordenadas do candidato.

A sequência de processamento é a seguinte:

1. A lista de correspondência calculada na secção anterior é utilizada para gerar a matriz vX descrito em 2.2, e a Matriz Fundamental F é calculada por SVD.

   Como foi referido na secção 2.1.2, a SVD realiza o Lest Mean Square da solução, o que faz com que pares errados de correspondências não destruam completamente a Matriz Fundamental calculada. Se a maioria dos pares forem correctos, e for utilizado um número suficiente de pares, esta matriz dá uma boa estimativa da geometria epipolar.

2. A Matriz Fundamental é guardada e a lista dos melhores pares e todas as correspondências são eliminadas. A geometria epipolar é utilizada para repetir a procura por correspondências.

   O algoritmo é reiniciado do ponto 4 da secção anterior 4.2.1, mas utilizando a Matriz Fundamental para melhorar os resultados. A lista com os melhores candidatos da imagem da direita não é apagada, pois esta não se alterou, poupando assim algum processamento.

   A Matriz Fundamental é previamente multiplicada por uma matriz de normalização e outra de desnormalização, para poder ser utilizada com as coordenadas puras dos candidatos. Estas matrizes são explicadas na secção seguinte 4.2.3.

A busca por correspondências realizada passa agora a ser realizada em torno da recta epipolar e não de um bloco de dimensões fixas. Desta forma, para cada candidato, é multiplicada a matriz Fundamental pelas suas coordenadas, resultando numa recta $[A,\ B,\ C]$ em coordenadas homogéneas como descrito na secção 2.1.1. A busca fica então definida a um paralelogramo, definido horizontalmente e verticalmente por

$$X \in [-130\ ,\ 50],\ \ Y \in [\frac{-130 \times A + C}{-B} + dY\ ,\ \frac{50 \times A + C}{-B} + dY]$$

Corresponde a uma zona com as mesmas coordenadas horizontais do candidato, mas cujas coordenadas verticais variam entre $-dY$ e $+dY$ em torno da recta epipolar. dY é definido inicialmente a 60 pixéis e é dividido para metade em cada iteração, mas nunca diminuindo dos 3 pixéis, reduzindo portanto a área de busca até um mínimo de 3 pixéis para cada lado da recta epipolar.

Por questões de simplificação, esta definição de zona parte do princípio que as rectas epipolares são praticamente horizontais, ou seja, que as câmaras se encontram praticamente alinhadas horizontalmente. Caso se encontrassem verticalmente alinhadas ou não alinhadas, seria necessário a alteração deste cálculo.

3. O pontos 1 e 2 são repetidos até um de três acontecimentos ocorrer:

   (a) A cada iteração é calculado o epipólo da imagem referência. Este cálculo é apresentado na secção seguinte. Se ao fim da terceira iteração este se encontrar demasiado perto do centro da imagem, as câmaras são reactivadas temporariamente, um par de imagens novas é obtido e o algoritmo começa de novo no ponto 1 da secção anterior.

   (b) Não foram encontrados pontos suficientes para realizar o cálculo necessário (8 pares de coordenadas como explicado em 2.2.

   (c) Decorreram 10 iterações[2] seguidas sem ocorrer um dos acontecimentos anteriores, o que indica que um número mínimo de pares sobreviveu à busca em zonas de dimensões reduzidas e, portanto, a geometria epipolar é de confiança.

Estes passos melhoram significativamente os pares encontrados, eliminando todos os pares iniciais errados e deixando apenas os de maior confiança. É portanto absolutamente necessário a execução destes para o cálculo das Matrizes de Transformação.

4.2.3 Cálculo das Matrizes

Após um conjunto fiável de pares de coordenadas correspondentes entre as duas imagens é possível proceder-se ao cálculo das Matrizes de Transformação.

[2] O número de iteracções tem de ser suficientemente grande para que a janela seja reduzida até ao mínimo indicado anteriormente, 6 pixéis de altura neste projecto. As iterações seguintes não introduzem um grande melhoramento, podendo ser evitadas para poupar tempo de processamento.

Na secção 2.2 vimos que Hartley propõe uma Matriz de Transformação como $H = M \cdot G \cdot R \cdot T$. O método utilizado e proposto neste projecto é baseado nesse, com algumas alterações necessárias.

A fórmula geral do método a seguir descrito é

$$kH = D \cdot [C \cdot T \cdot G \cdot R\,]^{-1} \cdot N\ ,\ k = H_{3,3}^{-1}\ ,\ e$$

$$k'H' = D \cdot [C \cdot G' \cdot R'\,]^{-1} \cdot N\ ,\ k' = H_{3,3}'^{-1}$$

As matrizes enviadas para o módulo de transformação são H e H'. k e k' são constantes que colocam a 1 a posição $3,3$ da matriz correspondente, no intuito de simplificar os cálculos no módulo de transformação.

De notar que as matrizes H e H' aqui obtidas são diferentes das obtidas por Hartley [12], no sentido em que as deste fazem a transformação das coordenadas da imagem, para as do monitor. As calculadas neste projecto realizam a operação contrária e, para cada ponto do ecrã, dizem qual a coordenada correspondente na imagem. Esta diferença é obtida com uma inversão matricial, como se pode observar na fórmula apresentada e simplifica os cálculos necessários no módulo de transformação.

Serão, agora, descritas cada uma das matrizes presentes na fórmula apresentada.

N e D São matrizes de Normalização e Desnormalização, respectivamente. A matriz N transforma o sistema de coordenadas normal, $[0,\ 640][0,\ 480]$, em coordenadas normalizadas, $[-1,\ 1][-0.75,\ 0.75]$. Como foi referido em 2.1.2, o método de cálculo da SVD obterá resultados mais precisos quando as coordenadas estão normalizadas. Esta operação consiste numa translação que coloca a origem no centro da imagem e num escalamento que encolhe a imagem por um factor de 320 vezes. A matriz D realiza a operação inversa, transformando um sistema de coordenadas $[-1,\ 1][-0.75,\ 0.75]$ em $[0,\ 640][0,\ 480]$.

$$N = \begin{vmatrix} 1/320 & 0 & -1 \\ 0 & 1/320 & -0.75 \\ 0 & 0 & 1 \end{vmatrix},\ e\ D = \begin{vmatrix} 320 & 0 & 320 \\ 0 & 320 & 240 \\ 0 & 0 & 1 \end{vmatrix}$$

R e G Estas matrizes são as matrizes introduzidas por Hartley e descritas na secção 2.2. Estas são as responsáveis por colocar o epipólo de uma imagem numa recta horizontal que passa pela origem e de o colocar no infinito, respectivamente.

$$R = \begin{vmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix}\ e\ G = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{-1}{f} & 0 & 1 \end{vmatrix}$$

Para o cálculo destas matrizes precisamos de saber $cos(\theta)$, $sin(\theta)$ e f, em que θ é o ângulo formado pelo epipólo com o eixo horizontal e f é a coordenada horizontal do epipólo após

a rotação por R. Assim, precisamos simplesmente de saber as coordenadas do epipólo. Na secção 2.2 foi explicado o método proposto por Hartley para calcular o epipólo, sabendo apenas a matriz fundamental: $F \cdot e = 0$ e este pode ser resolvido por SVD resultando nas coordenadas do epipólo e. O cálculo das matrizes R e G consistem em:

1. Calcular a Matriz Fundamental utilizando a lista de pares de correspondências obtidas da secção anterior:
$$vX \cdot F = 0$$

2. Realizar a SVD da Matriz Fundamental, obtendo as coordenadas do epipólo:
$$F \cdot e = 0$$

3. Transformar as coordenadas do epipólo em coordenadas cartesianas. O epipólo da imagem da esquerda pode ser retirado da mesma SVD, como foi explicado em 2.1.2.
$$e = [\frac{e_1}{e_3}, \frac{e_2}{e_3}, 1]$$

4. Com as coordenadas de e, calcular o co-seno, seno e f é trivial. f é calculado como o módulo com sinal do epipólo. De notar que, como interessa o ângulo que e faz com a horizontal, o co-seno é sempre positivo. Caso seja negativo é necessário trocar o sinal de ambos: co-seno e seno. Este problema é resolvido ao utilizar-se o f calculado:
$$f = \begin{cases} -\|e\| & e_1 < 0 \\ \|e\| & e_1 \geq 0 \end{cases}, \; cos(\theta) = \frac{e_1}{f} \; e \; sin(\theta) = \frac{e_2}{f}$$

É necessário calcular também as matrizes R' e G'. O método a utilizar é o mesmo mas com as coordenadas do epipólo da segunda imagem, obtido também da SVD de F.

T Esta é uma matriz de translação e escala. As duas matrizes anteriores tornam as rectas epipolares em rectas perfeitamente horizontais, mas não corrigem diferenças de translação e escala que possa haver nas imagens. Esta matriz pode ser calculada através da lista de pares e das matrizes R e G calculadas anteriormente.

Hartley propõe um método de minimização da distância entre as coordenadas candidato - correspondência. Para rectificação de imagens em estereoscopia não há necessidade de minimizar a coordenada horizontal, pois esta é que contém a informação de profundidade. Assim, para não danificar esta informação, não é realizada qualquer translação na coordenada horizontal. O factor de escala aplicado em X será o mesmo que em Y, para preservar a relação largura/altura da imagem.

De notar que esta matriz T só precisa de ser aplicada a uma das matrizes. Neste projecto decidiu-se aplicar a H, matriz da imagem da direita.

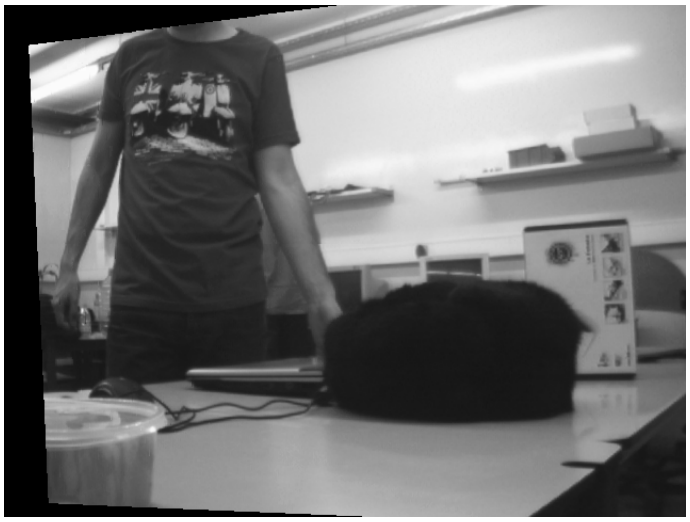


Figura 4.5: Margens em imagens rectificadas

Sendo $I = G \cdot R \cdot Q$, $I' = G' \cdot R' \cdot Q'$, em que Q e Q' são as coordenadas dos pontos candidatos e respectiva correspondência, e I e I' são as respectivas coordenadas após a aplicação das matrizes R e G, queremos encontrar S tal que as coordenadas verticais de $S \cdot I$ igualem as de I'. Como foi visto, a matriz terá um factor de escala e um de translação vertical. Então, para a criar basta encontrar dois valores, k e d tal que $Y \times k + d = Y'$, para todos os Y e Y' da lista de pares de correspondências. Esta equação é equivalente a $Y \times k - Y' + d = 0 \rightarrow [Y,\ Y',\ 1] \cdot [k,\ -1, d]^T = 0$, bastando então realizar a SVD de $v[Y,\ Y',\ 1]$. A SVD retorna o vector E como solução, e então $[k,\ -1, d] = [-\frac{E_1}{E_2},\ -1,\ -\frac{E_3}{E_2}]$.

Após este cálculo pode-se construir a matriz

$$T = \begin{vmatrix} k & 0 & 0 \\ 0 & k & d \\ 0 & 0 & 1 \end{vmatrix}$$

C Após a aplicação das matrizes de transformação, é comum obter-se uma imagem deslocada em que só partes dela é que são reconstruídas, como se pode ver na imagem 4.5. No intuito de diminuir essas margens e aumentar a área visível, decidiu-se aplicar mais uma matriz C. Como só se pode retirar informação de profundidade da área comum às duas imagens, e este género de algoritmos é normalmente utilizado com esse fim, decidiu-se maximizar a área visível da zona comum a ambas as imagens, mas mantendo a relação altura/largura. A figura 4.6 ilustra o efeito desta matriz sobre a imagem no monitor.

Para criar esta matriz, é necessário calcular as coordenadas da imagem original, no monitor. Para isto, basta multiplicar as coordenadas que se quer conhecer pelas matrizes que calculámos previamente: $L = T \cdot G \cdot R,\ eL' = G' \cdot R'$. Para maximizar a área comum da imagem, é necessário saber onde se encontra o ponto intermédio de cada lado de cada imagem. Estes pontos são representados pelas coordenadas $[-1,\ 0,\ 1]^T$, $[1,\ 0,\ 1]^T$, $[0,\ -1,\ 1]^T$, $[0,\ 1,\ 1]^T$

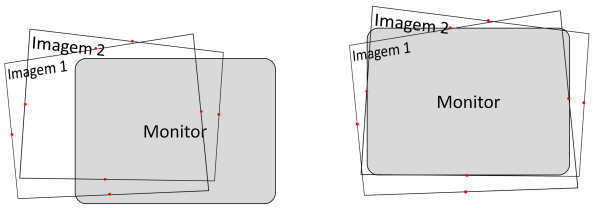
de ambas as imagens.

$$L \cdot \begin{vmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} P1x & P2x & P3x & P4x \\ P1y & P2y & P3y & P4y \\ P1w & P2w & P3w & P4w \end{vmatrix}$$

Depois de obtidas as coordenadas destes pontos, apenas as coordenadas horizontais dos pontos $P1$ e $P2$, e as verticais dos pontos $P3$ e $P4$ são guardadas.

$$P1 = \frac{P1x}{P1w},\ P2 = \frac{P2x}{P2w},\ P3 = \frac{P3y}{P3w},\ P4 = \frac{P4y}{P4w}$$

Os pontos correspondentes da outra imagem são igualmente calculados, utilizando a matriz L' em vez da L. É calculado, então, o menor valor absoluto de cada ponto, ou seja, menor entre $P1$ e $P1'$, entre $P2$ e $P2'$, etc. e esses pontos são considerados o limite da área visível. Na figura 4.6 pode-se visualizar os pontos a vermelho. Esta estimativa da área visível é precisa para correcções com pouca rotação em torno de Z, ou seja, para câmaras quase alinhadas horizontalmente que é o caso das câmaras deste projecto. Para outro género de montagens deveria de ser utilizado outro método. Para o factor de escala da matriz C decidiu-se maximizar a imagem apenas horizontalmente, ignorando os pontos verticais, mantendo contudo a relação altura/largura. Este facto pode levar a que a parte superior e inferior da imagem sejam cortadas, mas levará a que a matriz final amplie a imagem horizontalmente, evitando possíveis erros de leitura. Esta limitação é discutida na secção seguinte.



(a) Pré correcção (b) Pós correcção

Figura 4.6: Correcção da área visível

Com os pontos mínimos conhecidos, construir a matriz C é trivial. Por exemplo, caso os quatro pontos da imagem da direita ($P1$, $P2$, $P3$ e $P4$) fossem os mínimos, a matriz seria:

$$C = \begin{vmatrix} \frac{2}{P2-P1} & 0 & -\frac{P1+P2}{P2-P1} \\ 0 & \frac{2}{P2-P1} & -\frac{P3+P4}{P4-P3} \\ 0 & 0 & 1 \end{vmatrix}$$

Todo o cálculo das matrizes é efectuado a partir das coordenadas dos pares de correspondências. Após os cálculos aqui apresentados, as matrizes H e H' podem ser calculadas, e enviadas para o módulo de Transformação de Imagens.

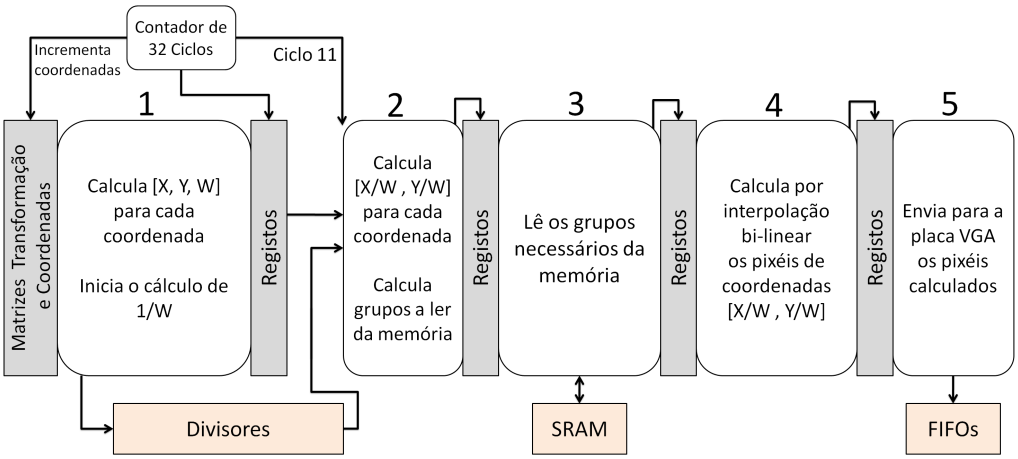


Figura 4.7: Módulo de transformação

O envio das matrizes é realizado como números de 18 bits, devido a limitações nos multiplicadores que serão introduzidas na secção seguinte. Destes 18 bits, 1 é para o sinal, 1 para a parte inteira, e 16 para a parte decimal, limitando portanto os valores da matriz ao intervalo $[-2,\ 2[$. Os valores da matriz relativos às translações têm uma organização de 1 bit para o sinal, 10 para a parte inteira e 7 para a decimal, devido à necessidade de translações na ordem das centenas de pixéis. A passagem dos números calculados em vírgula flutuante para este formato é feita sem perda de precisão da mesma forma apresentada na secção 3.6.6 para os valores enviados para o módulo de cálculo de raiz quadrada.

4.3 Transformação das Imagens

A transformação das imagens é realizada através da multiplicação das coordenadas do monitor pelas matrizes de transformação guardadas em registos. Este passo aparentemente simples implica a realização de diferentes operações, como multiplicações e divisões, leituras da memória SRAM, interpolações, envio de dados para uma placa externa. Como estas operações demoram demasiado tempo, e o sistema deverá correr em tempo real, tiveram de ser implementadas para serem executadas em paralelo.

Os requisitos de tempo real obrigam, como foi explicado no capítulo 3, a que sejam analisados 2 conjuntos de 4 pixéis a cada 32 ciclos de relógio. Assim, o tempo de execução máximo de cada fase deverá ser de 32 ciclos de relógio. Optou-se, então, por dividir o sistema em 5 fases de processamento distintas. Cada fase é implementada através de uma máquina de estados independente.

A figura 4.7 representa o funcionamento global deste módulo.

O paralelismo entre as diferentes fases foi obtido colocando registos entre cada uma das fases, possibilitando que cada uma destas processe dados diferentes ao mesmo tempo. O sincronismo entre elas é conseguido através de sinais de controlo e de um contador de ciclos externo. O controlador de ciclos é necessário devido ao módulo rígido de Divisor.

O módulo auxiliar de divisão foi implementado através do CoreGen para calcular o inverso de um número. Este possui uma latência de 39 ciclos de relógio, mas aceita dados novos na entrada a cada 4 ciclos de relógio pois é implementado em *pipeline*. Como a latência é superior a 32 ciclos de relógio, os dados têm de ser escritos e lidos por fases diferentes, sincronizadamente. Para isso decidiu-se implementar o contador de ciclos externos, que controla a inicialização das diferentes fases.

Foram implementados dois divisores independentes, um para cada imagem, pois no total são necessárias 8 divisões. Mesmo com dois divisores, entre a primeira ordem de divisão e a recepção do último valor calculado decorrem $39+3\times4=51$ ciclos de relógio, sobrando 13 ciclos de relógio para realizar outras operações antes e depois desse cálculo.

Para permitir a utilização da frequência de relógio de 100 MHz na operação deste módulo, todas as multiplicações tiveram de ser realizadas em módulos à parte e em *pipeline*. Estes módulos apresentam latências variáveis e serão mais bem descritos na descrição da fase correspondente. Os multiplicadores existentes na FPGA são de 18 bits e após alguns testes a utilização de multiplicadores maiores com a frequência de relógio de 100 MHz mostrou-se difícil. Decidiu-se, portanto, utilizar multiplicadores de no máximo 18 bits, sendo assim implementados nos multiplicadores dedicados da FPGA.

Pela mesma razão, os registos para os valores das matrizes de transformação foram limitados a 18bits. Como o maior número a ser multiplicado por estes valores será 639.0, que contém 10 bits, após a multiplicação haverá ainda uma precisão decimal de aproximadamente 6 bits.

A implementação das máquinas de estados descritas na figura 4.7 será explicada nas seguinte secções. A matriz de transformação é denominada de H, as coordenadas no monitor de $[X,\ Y]$, e as coordenadas homogéneas pós transformação de $[U,\ V,\ W]$.

4.3.1 Transformação das Coordenadas

A inicialização desta fase é controlada por um sinal externo vindo do módulo das câmaras, activando este módulo quando a linha de coordenada Y 240 das imagens das câmaras estiver a ser escrita para a memória. Esta inicialização tardia é necessária, pois com a transformação a ser realizada, várias linhas vão ter de ser lidas para preencher uma única no ecrã. Assim, ao iniciar na linha 240 é garantida a distância máxima constante entre a leitura da imagem e a actualização da mesma.

Este módulo é executado a cada 32 ciclos de relógio, uma vez para cada grupo de 4 coordenadas/pixéis das imagens, varrendo a imagem horizontalmente. Após analisar a imagem toda, a sua execução é pausada, ficando à espera do sinal externo de inicialização.

Inicialmente este módulo calcula para cada coordenada do grupo o seu correspondente valor W. Este cálculo necessita de 2 multiplicações e uma soma, e corresponde a

$$W = H_{3,1}\times X + H_{3,2}\times Y + 1$$

para cada coordenada $[X,\ Y]$ do grupo. É realizado ao mesmo tempo para ambas as imagens, e os resultados são enviados para os divisores auxiliares, um W' a cada 4 ciclos de relógio. Devido à latência dos multiplicadores, o primeiro valor só é enviado para o divisor auxiliar no 4ž ciclo de relógio.

Após este cálculo, as coordenadas U e V são calculadas como

$$U = H_{1,1} \times X + H_{1,2} \times Y + H_{1,3}\ ,\ V = H_{2,1} \times X + H_{2,2} \times Y + H_{2,3}$$

Esta fase necessita de 20 ciclos de processamento, ficando os restantes 12 à espera do contador de 32 ciclos para ser reiniciado com coordenadas novas. São necessários 4 multiplicadores para estes cálculos e estes realizam a operação $(A \times B + C \times D + E)$ com uma latência de 2 ciclos de relógio.

4.3.2 Cálculo das Coordenadas dos Pixéis a Ler

Esta fase é a responsável pelo cálculo das coordenadas dos grupos de pixéis necessários à interpolação. Nesse sentido, a execução desta é pausada até ao ciclo 11, que corresponde ao ciclo em que o resultado da primeira divisão se encontra disponível na saída do divisor. Após obter o valor de $1/W$ é realizada a transformação para coordenadas cartesianas.

$$[U,\ V] = [\frac{U}{W},\ \frac{V}{W}]$$

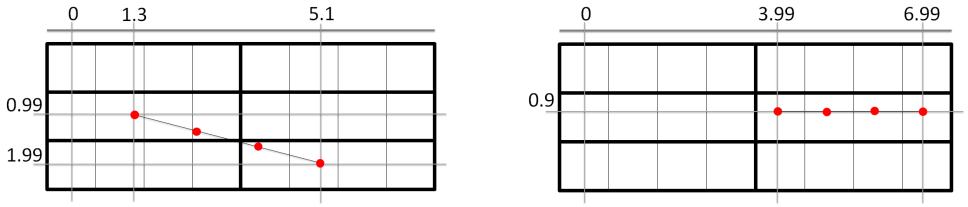
Esta operação é realizada para as 8 coordenadas, resultando nas posições finais a interpolar da imagem.

Após se obter estas coordenadas são calculados os grupos de pixéis necessários à interpolação. Como haverá apenas 32 ciclos para a leitura destes grupos, 16 ciclos por imagem, e como foi visto na secção 3.6.3 cada leitura de 4 pixéis demora 2 ciclos de relógio, poderiam ser realizadas no máximo 8 leituras de grupos. Como este módulo corre concorrentemente com o módulo de escrita nas memórias que realiza 2 operações a cada 32 ciclos, só poderão ser realizadas no máximo 7 leituras.

Decidiu-se, portanto, efectuar a leitura de 6 grupos de pixéis para cada conjunto de coordenadas, como mostra a figura 4.8. Para as coordenadas destes grupos basta calcular as do primeiro grupo e incrementar para os outros. Para o cálculo do primeiro grupo é encontrado o sinal da derivada das coordenadas. Realizando $dV = (V_1 - V_2) > 0 ?\ 1\ :\ 0$ obtemos um 1 se as coordenadas tiverem uma derivada positiva no ecrã. Analisando agora as primeiras coordenadas e subtraindo a derivada caso seja positiva, obtém-se o grupo desejado.

$$U_G = (int)U_1 >> 2\ ,\ V_G = (int)V_1 - dV$$

Esta fase é também a responsável por detectar a posição dos pixéis a ler, enviando informação até à última fase indicando se se encontram dentro da área da imagem.



(a) Limite de inclinação (b) Limite de ampliação

Figura 4.8: Limites de deformação das matrizes

Esta fase fica 11 ciclos à espera inicialmente e demora 18 ciclos a ser executada. No final do processamento é enviado um sinal de controlo para que a fase seguinte possa ser executada, como será descrito na secção seguinte.

Para não se perder precisão nas multiplicações aqui presentes, e visto que a saída do divisor é de 26 bits, foi implementado um multiplicador superior a 18 bits da seguinte forma.

$$O = A \times B[16:0] + (A \times B[25:17]) << 17$$

Esta operação foi ainda dividida em 2 ciclos, um para realizar as multiplicações individualmente e outro para somar os resultados, no intuito de permitir a utilização do relógio de 100 MHz. Esta fase necessita assim de 4 multiplicadores, dois para cada imagem.

4.3.3 Leitura das Imagens da Memória

Esta fase é a responsável pela leitura dos grupos da memória. Essa leitura ocorre concorrentemente com a escrita em memória por parte do módulo introduzido em 3.6.4, portanto decidiu-se relaxar ligeiramente o ciclo rígido de 32 ciclos de relógio. Assim, este módulo é iniciado após a activação de um sinal de controlo da fase anterior permitindo que demore esporadicamente entre 29 e 35 ciclos a executar.

Esta fase foi de simples implementação, pois só efectua operações de leitura à memória, guardando os valores lidos em registos. De notar que, como foi discutido em 3.6.3, uma operação de leitura à memória de um canal demora 4 ciclos de relógio. Assim, este módulo foi implementado utilizando 2 canais de leitura das memórias, permitindo realizar pedidos de leitura em *pipeline*: a segunda operação é pedida antes da primeira ser concluída.

O tempo de execução desta fase é de 25 ciclos, devido a 1 ciclo de *overhead* inicial antes de efectuar as leituras. Na realidade demora sempre 29 ciclos devido à utilização do canal da memória pelo módulo de escrita das imagens das câmaras.

4.3.4 Interpolação

Com as coordenadas calculadas e os dados lidos da memória, é necessário efectuar a interpolação. Esta fase é responsável pela interpolação e é iniciada quando a anterior termina.

Para simplificar os cálculos, as coordenadas das posições a interpolar são transformadas em coordenadas relativas aos grupos de pixéis lidos. Assim, todas as coordenadas a interpolar situam-se no intervalo $[0-7,\ 0-2]$ como mostra a figura 4.8.

Ao limitar as coordenadas a um intervalo, estamos a impor limitações nas matrizes de transformação. Para estes intervalos e, visto que são sempre calculadas tendo como base o primeiro pixel, a matriz de transformação tem como limitações:

- Não representar uma inversão no eixo X, ou seja, $H_{1,1}$ tem de ser positivo. Em situações normais, em que as câmaras estão orientadas de igual maneira, esta condição é sempre verdadeira;
- A imagem não poderá ser reduzida. A distância horizontal entre o primeiro e o último pixel de um grupo a ler não poderá ser superior a 3 pixéis, como mostra a figura 4.8. Como a área comum a ambas as imagens é sempre inferior à área de uma imagem, ao maximizar-se a área visível como foi explicado em 4.2.3, esta limitação não acontecerá.
- Como só se pode ler da memória uma altura equivalente a 3 pixéis, também existe uma limitação na inclinação máxima de um grupo. Caso o ponto anterior seja cumprido, a disparidade máxima horizontal é de 3 pixéis e a vertical de 1, levando a uma limitação no ângulo com a horizontal de $18,4$, como mostra a figura 4.8. Este ângulo máximo é suficiente para os casos em que as câmaras se encontram praticamente alinhadas, como no presente projecto. Por exemplo, na imagem 4.5 o ângulo máximo é de $7,5$, que corresponde à beira superior da imagem

Após esta transformação, a interpolação necessita de 3 cálculos para cada pixel a interpolar:

1. Cálculo das coordenadas dos pixéis circundantes;
2. Cálculo dos pesos relativos de cada pixel;
3. Multiplicação dos pesos pelo valor de luminosidade dos pixéis.

A operação 1 corresponde, simplesmente, a retirar a parte inteira da coordenada relativa.

Na operação 2 é necessário calcular 4 pesos distintos, 2 para cada eixo, peso superior e inferior, e da esquerda e da direita, representados na figura 4.9. Para calcular o peso da esquerda basta retirar a parte decimal da coordenada horizontal e o peso superior corresponde à parte decimal da coordenada vertical. Como os pesos simbolizam a distância ao centro do pixel, para calcular os pesos da direita e o inferior é apenas necessário subtrair a 1 o peso da esquerda e o superior, respectivamente. A figura 4.9 dá um exemplo desses pesos.

Para a fase 3 foi desenhado um multiplicador que possui 4 ciclos de latência e calcula:

$$O = l_1 \times PH_1 \times PV_1 + l_2 \times PH_2 \times PV_1 + l_3 \times PH_1 \times PV_2 + l_4 \times PH_2 \times PV_2.$$

Em que l_{1-4} são as luminosidades dos pixéis circundantes, $PH_{1,2}$ são os pesos da esquerda e da direita respectivamente, e $PV_{1,2}$ são os pesos superior e inferior respectivamente, como exemplificado na figura 4.9.

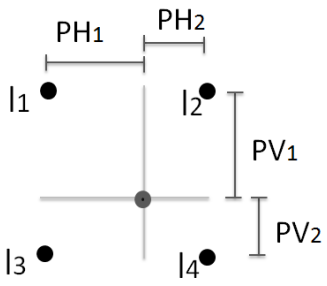


Figura 4.9: Pesos para interpolação

Os resultados das interpolações são guardados em registos e disponibilizados à máquina de estados seguinte. Esta fase utiliza 8 multiplicadores.

4.3.5 Envio das imagens rectificadas para a placa VGA

Esta máquina de estados tem por função enviar os resultados para a placa VGA. O envio é realizado a uma cadência de 25 MHz, o que faz com que esta fase demore 20 ciclos a concluir. O desfasamento do sinal de relógio de 25 MHz pode ser ajustado com os interruptores presentes na placa, mas o seu funcionamento não apresentou qualquer erro com o desfasamento de 0 ciclos.

A inicialização desta máquina de estados é efectuada com um sinal de controlo vindo do módulo anterior.

Os pixéis interpolados da imagem da direita são enviados para a FIFO responsável pelo vermelho e a da esquerda para o azul, facilitando a visualização e permitindo a utilização dos óculos 3D.

Capítulo 5

Validação e Análise de Resultados

Neste capítulo será discutido o processo de avaliação dos diferentes métodos que constituem o sistema. Todos os módulos e métodos desenvolvidos foram constantemente testados ao longo do projecto.

Para a implementação dos módulos de suporte foram criados alguns ficheiros testbench, principalmente quando os resultados não eram os esperados. Os ficheiros de teste criados foram bastante simples, muitas vezes só relacionados com a linguagem verilog, não sendo portanto importante colocá-los ou mencioná-los neste documento.

5.1 Driver de Acesso à Memória

Dos módulos de suporte, o mais difícil de testar foi o driver de acesso à memória, devido às restrições temporais. Após um estudo aprofundado dos timings necessários e planificação correcta do módulo, este apresentou os resultados esperados.

O bom funcionamento do driver é provado em todos os outros resultados deste projecto, pois todas as imagens passam pela memória pelo menos uma vez.

Após a implementação dos módulos auxiliares, a validação dos resultados foi feita através da comunicação série e do monitor.

5.2 Menu de Testes

Para facilitar esses testes foi implementado um Menu no terminal série, que permite interagir em tempo real com a FPGA. Esse menu está ilustrado na figura 5.1. Através deste menu foi possível testar, então, outras funcionalidades como se pode ver.

5.3 Módulo de Transformação

O teste ao módulo de transformação das imagens foi efectuado com outro menu.

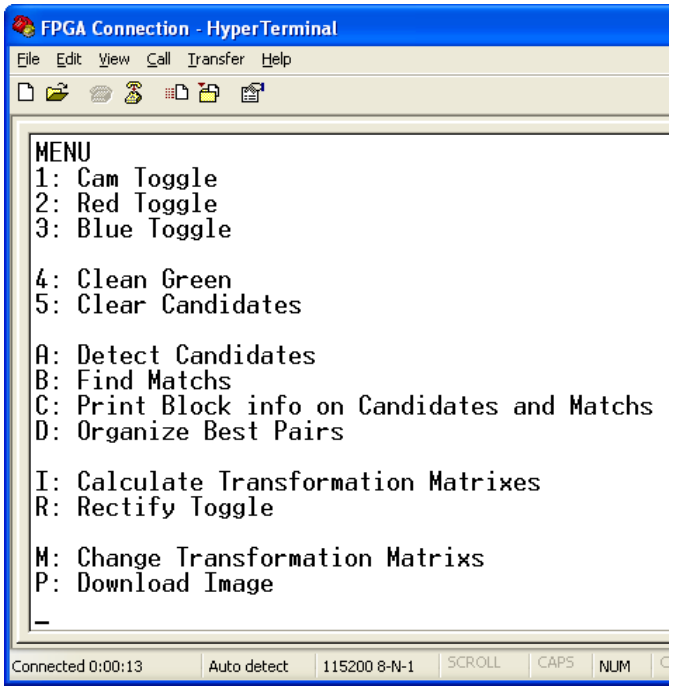
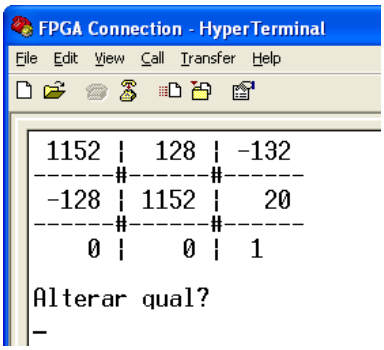


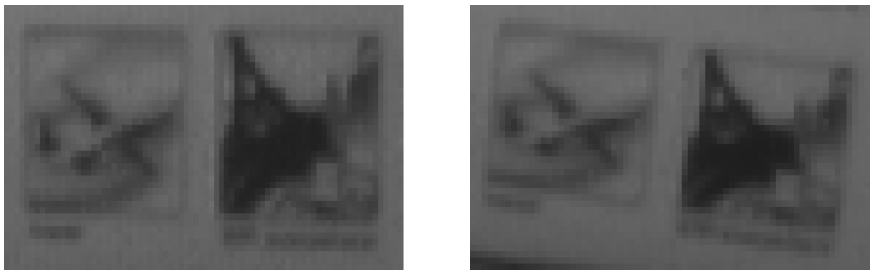
Figura 5.1: Menu principal de testes ao sistema



(a) Teste ao módulo de transformação. Valores multiplicados por 1000.

(b) Distorção correspondente

Figura 5.2: Menu de testes ao módulo de transformação



(a) Detalhe de uma imagem original (b) Detalhe de uma imagem transformada

Figura 5.3: Avaliação da interpolação bi-linear

Este menu permite incrementar e decrementar os valores das matrizes, visualizando instantaneamente a distorção correspondente das imagens no monitor.

As imagens 5.2 mostram um dos testes realizados com esse menu e respectiva imagem de uma das câmaras após transformação. Podemos visualizar que o módulo funciona correctamente e apurar os limites práticos de distorção mencionados na secção 4.3.4. Reparou-se que, principalmente o limite de ampliação, ao ser ligeiramente ultrapassado não se traduz em erros visíveis na imagem. A imagem 5.2.(b), por exemplo, está sujeita a um factor de ampliação de 0.89 e não aparenta ter nenhum erro de leitura, nem quando vista ampliada, como se pode ver em 5.3.(b).

5.4 Interpolação

A interpolação bi-linear mostrou-se menos destrutiva que o esperado. Como se pode ver nas imagens 5.3, em que se compara o mesmo detalhe de uma imagem original e de uma transformada (detalhe de um objecto da imagem 5.2.(b)) é visível apenas uma ligeira desfocagem. Esta é devida ao factor de escala de 0.89 aplicado à imagem.

Testes mais específicos realizados no intuito de testar a interpolação não apresentaram resultados concretos, em parte devido às outras questões que afectam as imagens. O foco manual das câmaras, o facto da lente das câmaras ser muito sensível ao toque e o problema de Demosaicing, mostraram ser factores mais degradantes da imagem do que a interpolação bi-linear.

O módulo de transformação, a função de envio das matrizes e a deformação provocada pela interpolação estão assim validados.

5.5 Cálculo Matricial

Os testes ao método de cálculo da matriz de transformação foram realizados em Excel. Esta ferramenta mostrou-se muito eficaz na compreensão dos passos envolvidos necessários.

Assim, foi construída uma folha de Excel que aplica uma transformação escolhida num conjunto de pontos, representada na figura 5.4. Nesta folha de Excel as coordenadas da coluna "deformados" são consideradas coordenadas na imagem da esquerda, as coordenadas "desfasadas

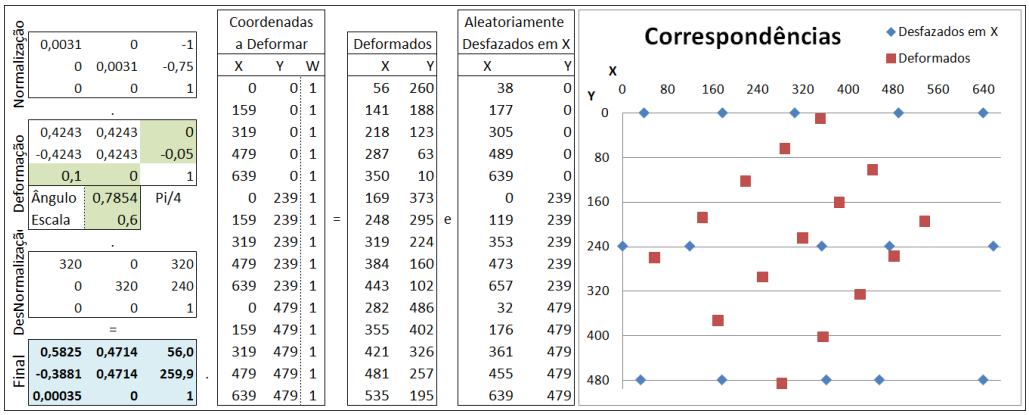


Figura 5.4: Folha de Excel para observar a deformação de uma imagem por matriz.

aleatoriamente em X" são consideradas como os pontos correspondentes na imagem da direita, e a matriz a azul é a matriz correcta de transformação que se terá de obter. No gráfico incluído é possível visualizar-se a deformação seleccionada nas imagens.

Outra folha utiliza estes valores de coordenadas e calcula as matrizes de transformação. Esta folha de cálculo foi o principal método de estudo, e foram realizados inúmeros testes em busca do algoritmo mais simples e preciso do cálculo das matrizes, visto que os algoritmos existentes em publicações não satisfazem as necessidades deste projecto.

Com os testes realizados foi detectado um fenómeno não descrito na literatura que acontece no cálculo das matrizes. Como se pode visualizar na figura 5.4, decidiu-se arredondar as coordenadas obtidas para o inteiro mais próximo, de maneira a simular melhor o que acontece nas imagens reais, em que as coordenadas dos pares são geralmente inteiros. Os testes efectuados com 9 coordenadas resultaram numa matriz final que, após aplicada, resulta em erros na posição vertical de alguns objectos de aproximadamente 10 pixéis. Visto que o objectivo deste projecto é obter um par de imagens o mais rectificado possível, este valor mostrou-se excessivamente elevado.

Na simulação, o cálculo da mesma matriz utilizando por base os valores correctos sem arredondamentos, resultaram num erro vertical inferior a 0,01 pixéis na imagem toda, o que comprova o método de cálculo da matriz.

Encontrou-se uma solução simples para o problema dos arredondamentos, que consiste em utilizar algumas dezenas de coordenadas, em vez de as 8 mínimas. Para testar a solução foram geradas folhas de Excel que aceitam um variado número de pares de coordenadas. Nesta simulação foram gerados vários pares de correspondências, de coordenadas e disparidade horizontal aleatória entre os $[-80,\ 80]$ ou os $[-20,\ 20]$ pixéis. A deformação entre as imagens foi modificada entre testes e aleatória. Após a aplicação da deformação às diversas coordenadas, estas foram arredondadas ao número inteiro mais próximo e o método de cálculo matricial foi aplicado. A matriz resultante foi testada com outros pares aleatórios deformados através da mesma matriz, com o intuito de simular imagens obtidas em momentos diferentes das mesmas câmaras. Os testes efectuados são expostos na tabela 5.1.

Na tabela é possível observar que, com a utilização de um número elevado de pares, é possível

Tabela 5.1: Precisão dos resultados com diferentes números de pares

Pares utilizados	9	13	25	100	250
Erro máximo vertical em pixéis:					
Disparidade horizontal até 80 pixéis	10 a 14	4 a 6	1.80 a 2.90	0.60 a 0.75	0.10 a 0.25
Disparidade horizontal até 20 pixéis	50 a 57	13 a 16	7 a 11	1.70 a 2.30	0.20 a 0.40
...20 pixéis, coordenadas dispersas	5 a 8	2,0 a 2,9	1,5 a 2,6	0,50 a 0,70	

eliminar os erros de arredondamentos. Esta diminuição no erro provém do cálculo da SVD, que aplica o método *Least Mean Square* para a minimização do erro da solução. Ao arredondar as coordenadas introduz-se erros aleatórios entre $[-0.5,\ 0,5]$ nos valores a utilizar na SVD. Se as coordenadas, por acaso, possuírem esses erros a tender para um padrão, esse padrão é detectado pela SVD e minimizado, resultando em erros. No caso da utilização de algumas dezenas de pontos, a distribuição dos erros fica mais homogénea, diminuindo a probabilidade de haver um padrão no erro e, portanto, reduzindo o erro do cálculo da SVD.

Na tabela é também possível visualizar que a qualidade da informação espacial é um factor importante na diminuição dos erros obtidos. A observação de uma imagem com informação espacial variada, isto é, objectos longe e perto, corresponde a pares de correspondência com disparidade horizontal mais elevada. Como é visível na tabela 5.1 este factor melhora a precisão dos resultados.

A qualidade da informação também depende da dispersão dos pares na imagem. Principalmente na utilização de poucos pares de correspondências pode acontecer que a maioria das coordenadas se encontre numa determinada área, prejudicando os resultados nas restantes.

Apesar de não ser possível aumentar a disparidade horizontal dos objectos sem danificar informação, o método implementado dá mais valor a pares de zonas da imagem diferentes e dispersas, melhorando assim a informação espacial e, consequentemente, os resultados. Esta melhoria é também retratada na tabela 5.1, e a sua implementação foi descrita no capítulo anterior.

Devido a restrições de memória da placa utilizada não é possível guardar um número de pares superior a 40, mas com a utilização dos outros métodos foi possível reduzir o erro resultante dos arredondamentos.

O algoritmo implementado no MicroBlaze para cálculo das Matrizes foi testado com as coordenadas criadas no Excel, no intuito de comparar a precisão. A diferença nos valores das Matrizes calculadas não foi significativa, comprovando a precisão do cálculo e do módulo de cálculo da raiz quadrada implementado em CORDIC. No entanto, o erro devido aos arredondamentos continua presente.

5.6 Imagens com Pouca Informação Espacial

Após a realização de mais testes e do estudo aprofundado sobre geometria epipolar chegou-se à conclusão que a informação espacial é subvalorizada na literatura existente. Ao examinar-se um par de imagens como a imagem 5.5, em que os pontos interessantes se encontram todos num plano do espaço, não é possível extrair informação epipolar suficiente para o cálculo da matriz fundamental. Apesar de os pontos possuírem uma discrepância horizontal elevada, uma vez que o objecto na imagem se encontra relativamente perto das câmaras, esta discrepância forma um "padrão" na imagem. Isto leva a que haja uma infinidade de equações para o cálculo dos epipólos e rectas epipolares. Uma imagem tirada de avião da superfície terrestre apresenta o mesmo problema, pois os objectos presentes encontram-se num mesmo plano, a superfície da terra. Mesmo se a imagem for retirada de um ângulo oblíquo, o epipólo da imagem pode-se encontrar sobre qualquer ponto pertencente a esse plano.

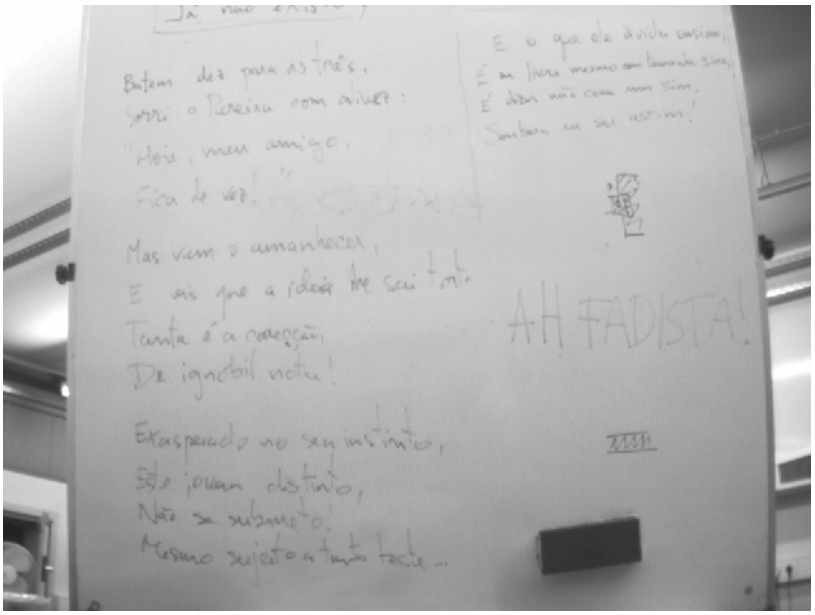


Figura 5.5: Imagem com pouca informação espacial

Para perceber esta limitação pode-se imaginar um par de imagens de um objecto completamente plano. É trivial perceber que existe um conjunto de transformações: rotações, escalamentos e translações, que coloquem os objectos de ambas as imagens nas mesmas precisas coordenadas, isto é, as coordenadas dos objectos numa imagem são iguais às coordenadas dos mesmos objectos na outra imagem. É igualmente fácil de perceber que, após estas transformações, em que se obtém literalmente a mesma imagem de duas imagens iniciais diferentes, uma qualquer transformação aplicada a ambas as imagens continuará a produzir duas imagens iguais. O facto de essa transformação poder ser agrupada com as anteriores, formando uma única, prova que existe uma infinidade de transformações possíveis de "rectificar" esse par de imagens. Isto prova também que existem diversas matrizes fundamentais desse mesmo par de imagens.

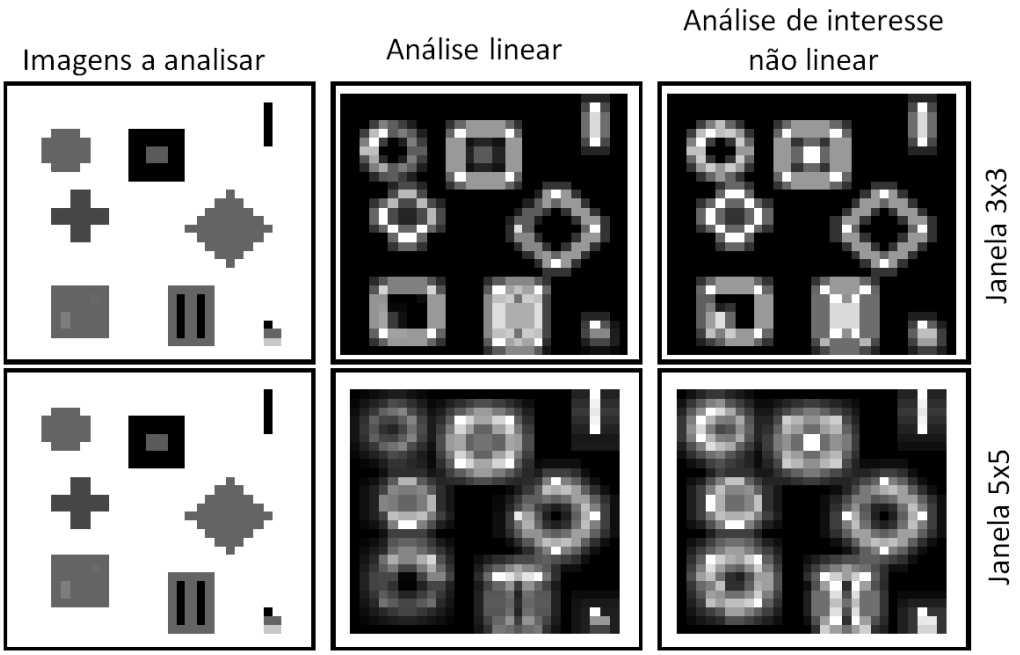


Figura 5.6: Comparação entre dois algoritmos e tamanhos da janela

Este facto torna difícil a utilização da matriz fundamental para rectificação de imagens sem informação espacial suficiente, como fotos de objectos planos ou aéreas. Para a rectificação deste tipo de imagens é necessário impor certas limitações à matriz fundamental, como rotações em Y e X nulas.

Como o objectivo deste trabalho é rectificar imagens estereográficas, partiu-se do princípio que essas imagens possuem informação espacial suficiente, não impondo portanto qualquer limitação à matriz fundamental.

5.7 Procura de Correspondências

Como foi referido no capítulo anterior, o primeiro passo na busca de correspondências consiste em detectar características de um grupo de pixéis e avaliar cada pixel da imagem segundo um interesse.

O método proposto na implementação foi desenvolvido também em Excel e vários métodos foram testados para a detecção de pontos interessantes. Foi estudada a utilização de diferentes tamanhos de janela. Na figura 5.6 é ilustrada a folha de Excel utilizada para efectuar este estudo.

Na figura é visível que a utilização da janela 5 x 5 não é muito vantajosa em relação à de 3 x 3. Este facto é devido à boa definição dos contornos dos objectos estudados. Em imagens reais com contornos menos definidos, uma janela 3 x 3 não possui informação suficiente da imagem para a poder avaliar, como é mostrado na figura 4.3 no capítulo 4.

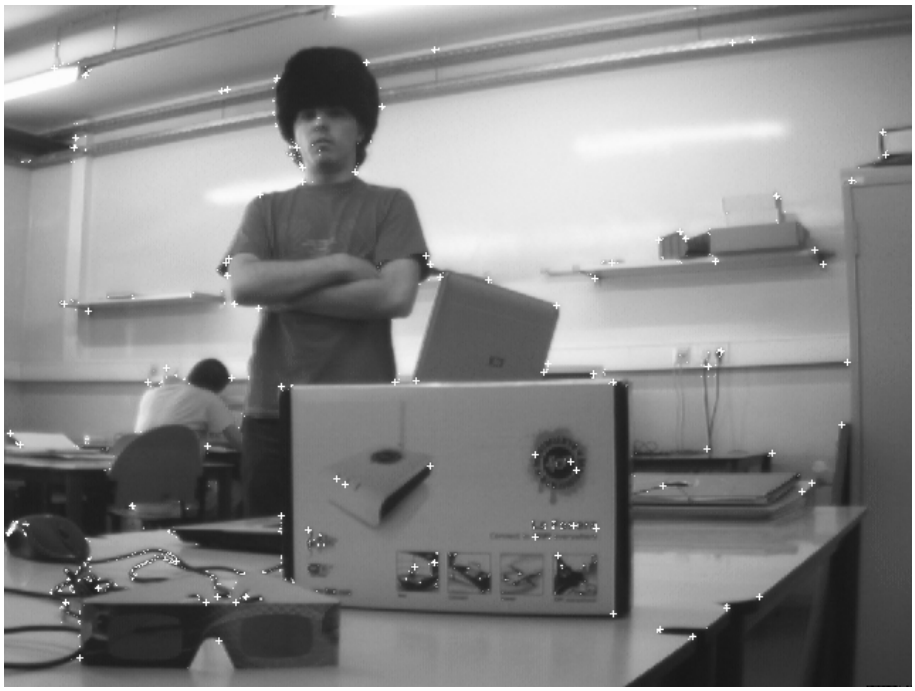


Figura 5.7: Pontos de interesse e candidatos

A imagem 5.7 mostra os pontos seleccionados pelo algoritmo como pontos de interesse, e os melhores candidatos de cada grupo estão assinalados com um $+$. Os pontos detectados encontram-se espalhados pelos objectos e pela imagem, como é desejável, e as esquinas são correctamente escolhidas como favoritas comprovando o funcionamento do método.

A imagem 5.8 mostra os resultados do algoritmo de encontrar correspondências sobre a mesma imagem 5.7. Como se pode visualizar, o algoritmo encontrou 6 correspondências possíveis para o candidato sobre a letra "a" a azul. Entre as correspondências possíveis estão duas letras "a", e um "o", mas o "a" correcto foi detectado, como se pode confirmar na imagem 5.9, devido à segunda análise efectuada com uma janela de 15 x 15 pixéis.

Na imagem 5.9 são visíveis todas as correspondências encontradas para os candidatos. Apesar de terem sido detectadas algumas correspondências erradas, assinaladas com uma seta branca, a maioria está correcta. A selecção seguinte para a criação da lista de melhores pares refinou as correspondências, resultando na imagem 5.10.

Nesta imagem podemos observar que persistem, ainda, dois pares de correspondências errados, e todos os outros anteriormente errados foram eliminados da lista.

O processo iterativo de melhoramento destes pares através da geometria epipolar removeu completamente os pares errados.

Na imagem 5.11 estão representadas as rectas epipolares de três candidatos, após a última iteração de melhoramento.

Na caixa ampliada é visível que a rolha de uma garrafa de água foi detectada, e a recta epipolar passa correctamente sobre a rolha da garrafa na outra imagem, com 1 pixel de disparidade vertical. Este valor de disparidade é aceitável e coerente com os obtidos em testes anteriores.

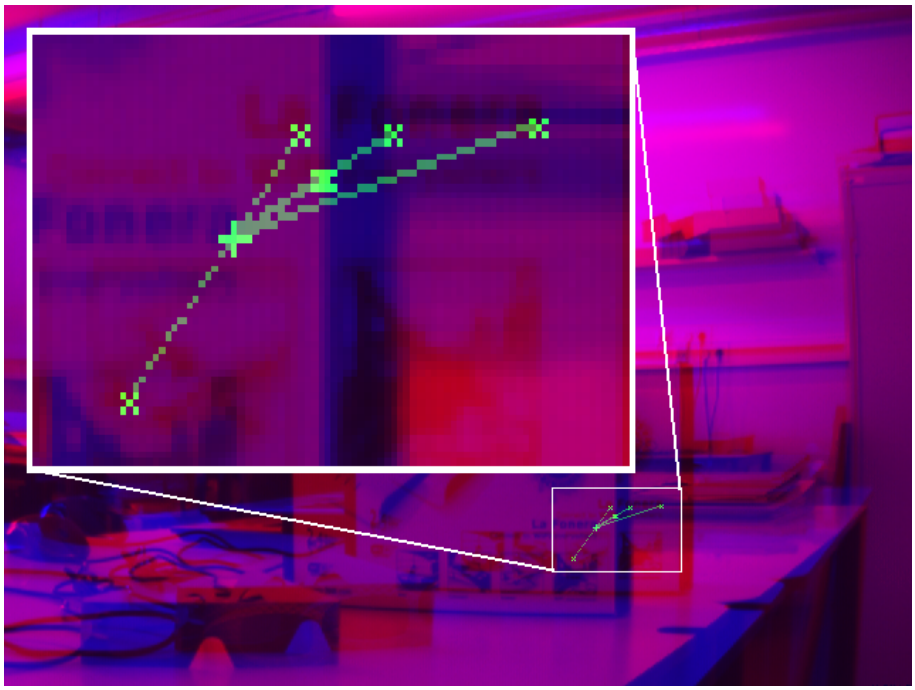


Figura 5.8: Possiveis correspondências de um candidato

Figura 5.9: Pares encontrados

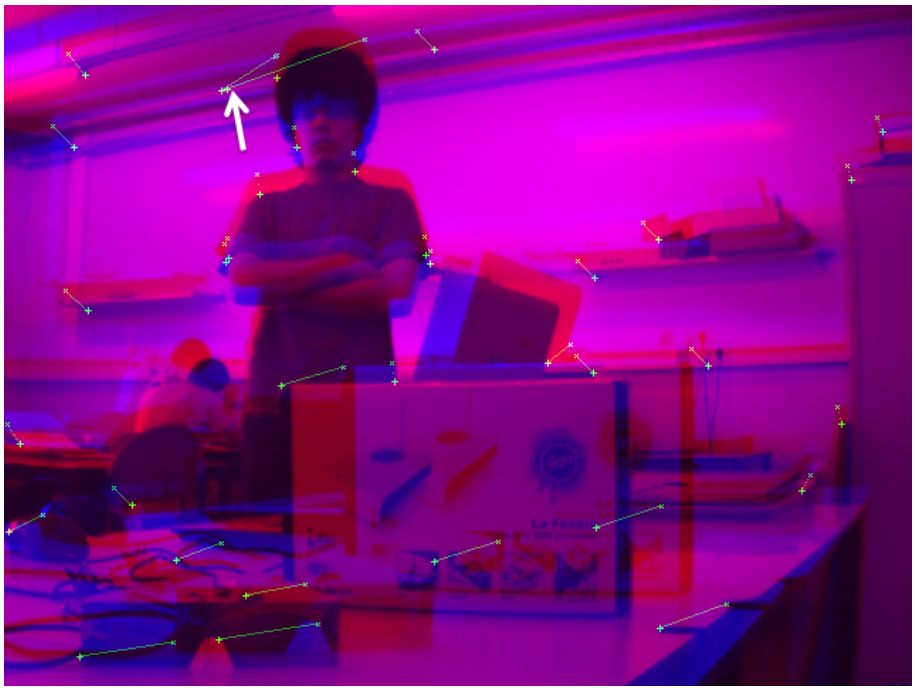


Figura 5.10: Melhores pares

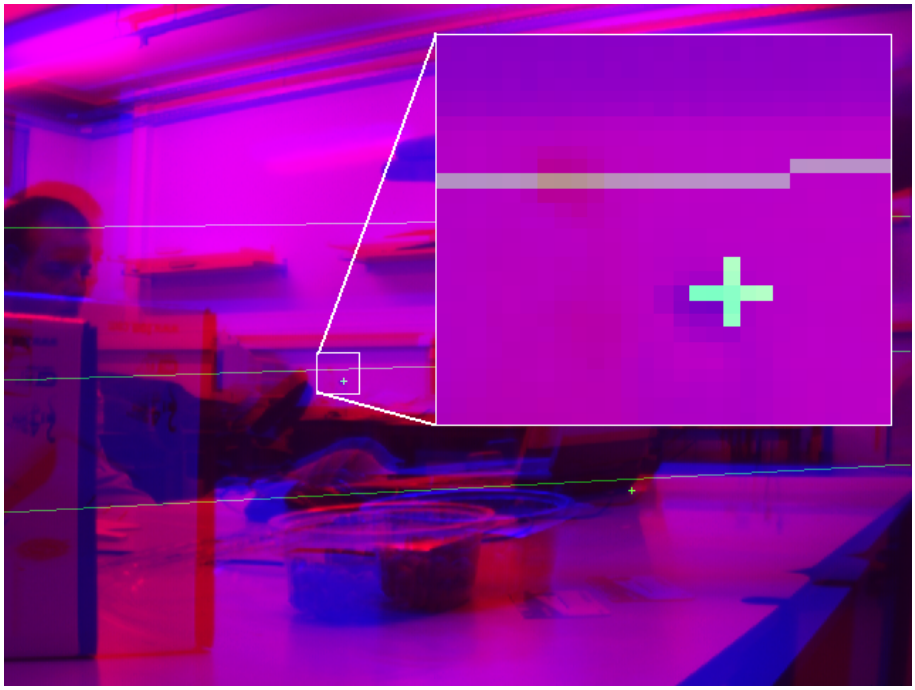


Figura 5.11: Informação epipolar de uma imagem analisada

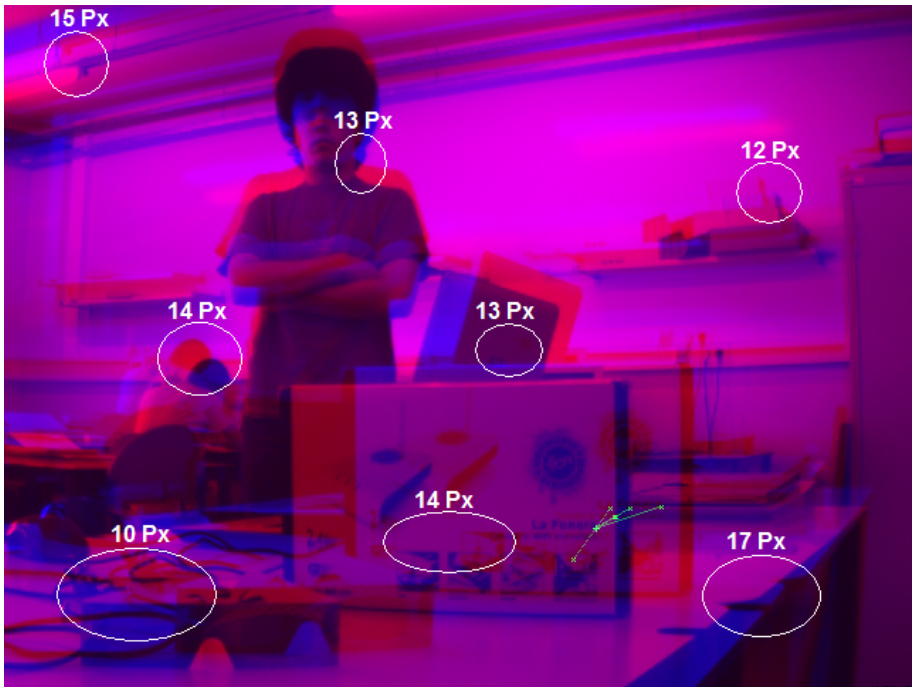


Figura 5.12: Exemplo de erros em imagens não rectificadas

5.8 Resultados Finais

Nas figuras 5.13 e 5.14 são visíveis exemplos de imagens rectificadas utilizando os métodos apresentados e alguns detalhes são analisados e com o respectivo erro vertical indicado. As duas imagens foram obtidas em instantes diferentes com cálculos de matrizes independentes.

Para comparação, na imagem 5.12 podem ser visualizadas as discrepâncias verticais em imagens não rectificadas. Apesar de as câmaras se encontrarem visualmente alinhadas, uma análise detalhada mostrou que existem erros em objectos de até 17 pixéis. Como se pode visualizar, os erros obtidos após rectificação são de no máximo 2 pixéis, coerente com as simulações realizadas.

Durante os testes efectuados foram sempre obtidos resultados satisfatórios, como os aqui apresentados. Como foi observado nas simulações, para diminuir o erro obtido, seria necessário a utilização de um maior número de pares, o que requereria um maior espaço de memória. As imagens também deveriam ser mais nítidas para se poder identificar mais pontos de interesse nas imagens.

Dependendo das imagens, o algoritmo demora entre 30 segundos e 1 minuto a calcular, podendo demorar mais, caso a imagem possua pouca informação espacial ou seja de difícil análise.

Apesar da distorção ocular estar visivelmente presente nas imagens das câmaras, esta mostrou-se pouco prejudicial aos resultados da rectificação.

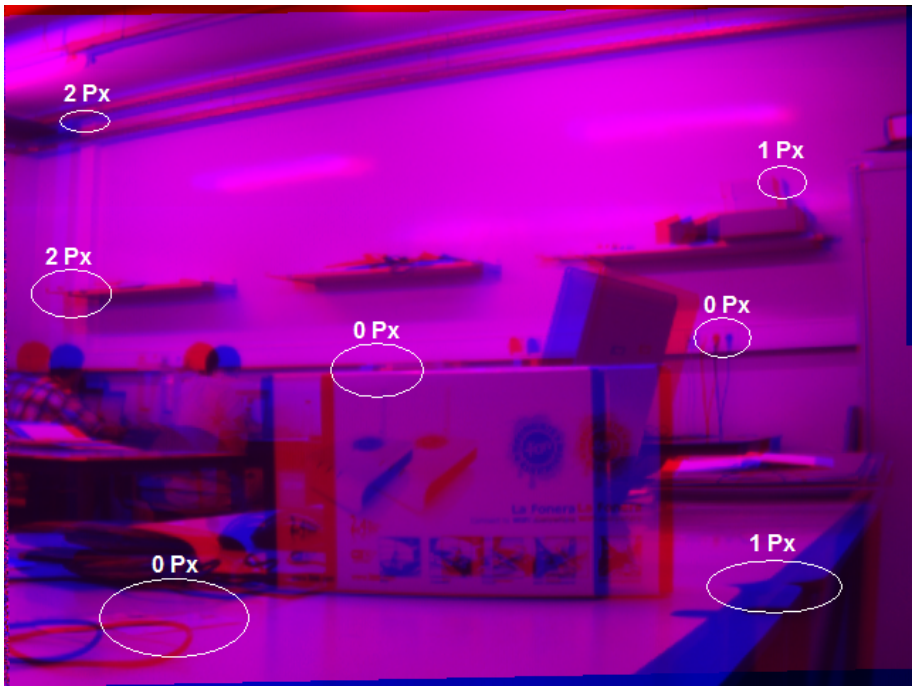


Figura 5.13: Exemplo de imagens rectificadas 1

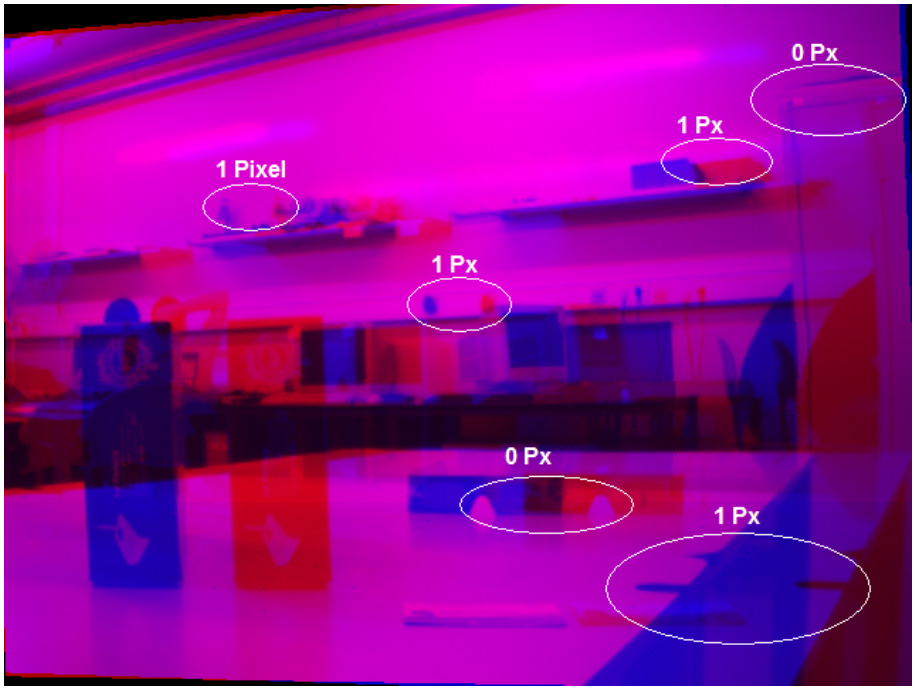


Figura 5.14: Exemplo de imagens rectificadas 2

5.9 Utilização de Recursos

A utilização de recursos do sistema na FPGA é a descrita na tabela 5.2.

Tabela 5.2: Utilização de Recursos

BRAMs	64 KB de 64 KB	100%
Slices	9,468 de 13,312	71%
LUTs	15,969 de 26,624	59%
Slice Flip Flops	10,788 de 26,624	40%
MULT18X18s	23 de 32	71%

Como se pode observar, o recurso de utilização crítica da FPGA foi as BRAMs, onde está implementada toda a primeira parte de processamento. Um sistema com mais BRAMs, ou seja, memória para mais instruções sequenciais, daria ao sistema mais elasticidade na implementação. Dos 64KB utilizados, apenas 16KB foram utilizados como memória de dados. Um sistema com 128KB de memória poderia, então, utilizar mais de 100 pares nos cálculos da matriz e utilizar funções mais eficazes para a determinação desses pares. Nos restantes recursos não houve quaisquer limitações.

Capítulo 6

Conclusão e Trabalhos Futuros

Com a concretização deste projecto foi estudada a importância da rectificação de imagens estéreo. Provou-se que, mesmo em kits de câmaras estéreo com montagens quase-rectificadas, os objectos podem apresentar uma discrepância vertical significativa entre as duas imagens. A diminuição este deslocamento é importante para a posterior análise das imagens, e provou-se que este processo é de fácil implementação em sistemas dinâmicos como as FPGA.

Com o desenvolvimento deste trabalho mostrou-se que não é necessário um sistema de rápido processamento para se obter uma matriz de transformação fiável e se realizar transformação em tempo real em imagens de tamanho aceitável. Concluiu-se, assim, que os dois passos independentes, cálculo da matriz e transformação, podem ser implementados em conjunto em sistemas dinâmicos como as FPGA, resultando assim num sistema igualmente fiável e mais útil do que se fossem implementados em sistemas diferentes.

O sistema implementado calcula, eficazmente e com uma precisão significativa, as matrizes de transformação, e aplica-as em tempo real ao vídeo estéreo vindo de duas câmaras. O sistema desenvolvido é capaz de rectificar imagens de dois vídeos, obtendo uma disparidade vertical máxima de 2 pixéis, e com velocidades até aos 25 FPS a 640 x 480 pixéis. O sistema utilizou cerca de 65% dos recursos disponíveis, sendo facilmente escalável para imagens de tamanho superior.

Concluiu-se, também, que para diminuir o erro obtido é necessário a utilização de um maior número de pares de correspondências nos cálculos. As simulações mostram que seriam necessários mais de 100 pares de correspondências ou detectar as coordenadas das correspondências com precisão sub-pixel, para a obtenção de erros inferiores a 1 pixel.

Trabalhos Futuros

Algumas melhorias ao cálculo das matrizes de transformação podem ser realizadas, principalmente caso esteja disponível mais memória de dados e de instruções. Por exemplo, o método proposto por Pilu e Lorusso [14] não pôde ser aqui implementado nem testado por falta de memória.

Apesar das correspondências erradas serem eficientemente eliminadas utilizando a série de funções simples propostas neste projecto, existem outras funções também simples que poderiam

melhorar os resultados na série. Por exemplo, após o cálculo da geometria epipolar, a segunda iteração de refinamento das correspondências poderia ser efectuada em imagens diferentes com candidatos novos, e não nas mesmas imagens com mesmos candidatos.

Se houvesse memória disponível para armazenar duas imagens completas, seria também possível executar a transformação em paralelo com o melhoramento iterativo da matriz. Este melhoramento poderia ser efectuado ao longo do tempo sobre imagens novas, utilizando como base de partida a matriz anterior. Esta proposta faz sentido, principalmente, se as câmaras estiverem num ambiente menos estável, em que é necessário recalibrar esporadicamente devido a alterações físicas nas câmaras.

No que diz respeito à transformação das imagens, a interpolação bi-linear implementada neste projecto pode prejudicar gravemente o detalhe em imagens com contraste elevado. Em câmaras de melhor qualidade é aconselhável a implementação de um método de interpolação melhor, possivelmente não linear de preservação de características. Para a implementação destas propostas é aconselhável a utilização de uma placa de desenvolvimento com um barramento de acesso às memórias mais rápido. O sistema desenvolvido é fácil de escalar para estes requisitos ou para imagens de tamanho superior, bastando para isso melhorar a velocidade do barramento de memória. Uma das hipóteses seria utilizar a memória DDR também presente para armazenar uma das imagens, duplicando a largura de banda das memórias, e permitindo assim o processamento de imagens com o dobro da área ou o dobro dos FPS. Um controlador mais inteligente de memória poderia ser implementado, optimizando as leituras necessárias e, assim, possivelmente permitindo o processamento de imagens com 1024 x 768 pixéis e 25 FPS.

Referências

[1] Sean McHugh. Digital photography tutorials, Fevereiro 2009. http://www.cambridgeincolour.com/tutorials.htm.

[2] Stephen T. Barnard e Martin A. Fischler. Computational stereo. *ACM Comput. Surv.*, 14(4):553–572, 1982.

[3] H. G. Longbotham, H. N. Kothari, e P. Yan. X-ray stereo imaging technique for disparity analysis. páginas 24–26, Junho 1995.

[4] Nick Holliman. Binocularity - knowledge about all aspects of 3d displays and their application, 2004. http://www.binocularity.org/.

[5] António João da Silva Lopes. Módulo 1 - computação gráfica: Transformações bidimensionais, 12 2001. http://www.prof2000.pt/users/ajlopes/trans2d/trans2d.htm.

[6] Dr. C.-K. Shene. Homogeneous coordinates, 8 2008. http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/geometry/homo-coor.html.

[7] William Press, Saul Teukolsky, William Vetterling, e Brian Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 2nd edição, 1992.

[8] Dan Kalman. A singularly valuable decomposition: The SVD of a matrix. *The College Mathematics Journal*, 27(1):2–23, 1996.

[9] Barak Shilo Katrina Sokolova. Experiments in stereo vision, Dezembro 2006. http://disparity.wikidot.com/.

[10] Andrea Fusiello. Epipolar rectification, Março 2000. http://profs.sci.univr.it/~fusiello/rectif_cvol/rectif_cvol.html.

[11] Roger Mohr e Bill Triggs. Projective geometry for image analysis a tutorial given at isprs, Julho 1996. http://lear.inrialpes.fr/people/triggs/pubs/isprs96/isprs96.html.

[12] Richard I. Hartley. Theory and practice of projective rectification. *International Journal of Computer Vision*, 35(2):115–127, November 1999.

[13] D. Scharstein, R. Szeliski, e R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. páginas 131–140, 2001.

[14] Maurizio Pilu e Adele Lorusso. Uncalibrated stereo correspondence by singular value decomposition, Julho 1997. http://www.bmva.ac.uk/bmvc/1997/papers/081/index.html.

[15] S. E. El-Khamy, M. M. Hadhoud, M. I. Dessouky, B. M. Salam, e F. E. A. El-Samie. A new edge preserving pixel-by-pixel (pbp) cubic image interpolation approach. Em *Proc. Twenty-First National Radio Science Conference NRSC 2004*, páginas C11–1–9, 16–18 March 2004.

[16] Yunde Jia, Xiaoxun Zhang, Mingxiang Li, e Luping An. A miniature stereo vision machine (msvm-iii) for dense disparity mapping. Em *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1*, páginas 728–731, Washington, DC, USA, 2004. IEEE Computer Society.

[17] Xinting Gao, R. Kleihorst, e B. Schueler. Implementation of auto-rectification and depth estimation of stereo video in a real-time smart camera system. Em *Computer Vision and Pattern Recognition Workshops*, páginas 1–7, Anchorage, AK,, 2008.

[18] Quasar Electronics. C3188a - 1/3"digital output colour camera module (ov7620), Junho 2009. http://www.quasarelectronics.com/c3188a-digital-output-cmos-colour-camera-module-omnivision-ov7620.htm.

[19] Xilinx. Platform studio and the embedded development kit (edk). http://www.xilinx.com/tools/platform.htm.

[20] Leonardo Volpi. Software per calcolo numerico, Março 2007. http://digilander.libero.it/foxes/index.htm.

[21] José Carlos dos Santos Alves. Projecto de sistemas digitais, 2008. http://paginas.fe.up.pt/~aja/PSDI_200708/.

[22] Cypress. Cy7c1062av33, Junho 2009. http://www.cypress.com/?rID=13131.

[23] The MathWorks. Video and image processing blockset - sad, Junho 2009. http://www.mathworks.com/access/helpdesk/help/toolbox/vipblks/ref/sad.html.