Pós-Graduação em Ciência da Computação

"LEDges: nova técnica de inspeção visual automática para detecção em tempo-real de irregularidades em bordas"

Por

Ygo Neto Batista

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/∼posgraduacao

Recife–PE, Agosto/2011

Ygo Neto Batista

LEDges: nova técnica de inspeção visual automática para detecção em tempo-real de irregularidades em bordas

Dissertação apresentada ao Centro de Informática - CIn, da Universidade Federal de Pernambuco - UFPE, como exigência parcial para obtenção do título de mestre em ciências da computação.

Orientador:
Prof. Dr. Cristiano Coelho de Araújo

Co-orientador:
Prof. Dr. Abel Guilhermino da Silva Filho

Universidade Federal de Pernambuco - UFPE
Centro de Informática - CIn

Recife–PE

Agosto/2011

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Batista, Ygo Neto
LEDges: nova técnica de inspeção visual automática para detecção em tempo-real de irregularidades em bordas / Ygo Neto Batista - Recife: O Autor, 2011.
177 folhas : il., fig., tab.

Orientador: Cristiano Coelho de Araújo.
Dissertação (mestrado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2011.

Inclui bibliografia e apêndice.

1. Engenharia da computação. 2. Sistemas embarcados. I. Araújo, Cristiano Coelho de (orientador). II. Título.

621.39 CDD (22. ed.) MEI2011 – 115

Que ninguém se engane:
só se consegue a simplicidade através de muito trabalho.
—CLARICE LINSPECTOR (A paixão segundo G.H.)

Dissertação de mestrado sob o título *"LEDges: nova técnica de inspeção visual automática para detecção em tempo-real de irregularidades em bordas"*, defendida por Ygo Neto Batista e aprovada em 30 de agosto de 2011, em Recife, Estado de Pernambuco, pela banca examinadora constituída pelos doutores:

Prof. Dr. Cristiano Coelho de Araújo
Orientador

Prof. Dr. Adriano Augusto de M. Sarmento
Universidade Federal de Pernambuco — UFPE

Prof. Dr. Isnaldo José de Souza Coêlho
Universidade Federal do Vale do São Francisco — UNIVASF

Agradecimentos

Primeiramente agradeço a Deus por ter me dado a oportunidade de elaborar esta dissertação. Aos meus pais, Carlos e Dalva, e à minha esposa, Suzana, pelo amor, carinho, compreensão e respeito, antes e durante a elaboração deste trabalho. Aos meus orientadores, professores Cristiano de Araújo e Abel Guilhermino, que me acompanharam pacientemente e, de fato, me orientaram nessa caminhada. Ao professor Isnaldo Coêlho, pela presteza e interesse em conhecer a minha dissertação. Aos amigos Rafael Macieira e Ricardo Lins, pelas valiosas contribuições no desenvolvimento inicial, e ao primo e amigo Rafael Neto, no desenvolvimeno final do trabalho. À empresa Optanica Soluções Fotônicas Ltda., por acreditar na idéia proposta e investir recursos na infraestrutura para seu desenvolvimento. Aos amigos Antonio Vaz e Roberta Souza, pelo apoio incondicional.

Resumo

Este trabalho apresenta uma nova técnica de inspeção visual automática para detecção, em tempo real, de deformações em bordas (LEDges), uma aplicação e duas implementações desta técnica: uma baseada em microcontrolador e outra, em FPGA. A principal inovação da LEDges é um novo modelo de iluminação estruturada que torna mais fácil a segmentação pela limiarização da imagem do objeto sob inspeção. Esta técnica, além de reduzir significantemente o esforço computacional para executar a segmentação, representação e descrição da imagem, também reduz o uso de recursos computacionais, tais como processador e memória. Adicionalmente, a LEDges pode ser implementado em diversas arquiteturas. Esta flexibilidade permite a construção de sistemas de inspeção visual automáticos satisfazendo a demanda crescente por desempenho, com menor tempo de resposta e menor uso de recursos computacionais quando comparados a outras soluções de mesma complexidade computacional. A LEDges foi implementada e aplicada a um problema industrial real onde defeitos foram detectados com sucesso em bordas de tubos de creme dental.

Abstract

This work presents a new automatic visual inspection technique for real time flaw detection on edges (LEDges), an application and two implementations of this technique: one based on a microcontroller and another, on a FPGA. The main innovation of the LEDges is a new structured illumination model, which makes easier the segmentation by binarization of the object under inspection image. This, on one hand, significantly reduces the computational effort to perform the image segmentation, representation and description. On the other hand reduces the use of costly architectural resources such as processor and memory. Furthermore the LEDges can be implemented in different architectures. This flexibility allows the implementation of automatic visual inspection systems satisfying the increasing demand for performance with lower response time and lower use of computational resources when compared to the other solutions with same computational complexity. The solution was implemented and applied to a real industrial problem where defects were successfully detected in the edges of toothpaste tubes.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

ASIC	*Application Specific Integrated Circuit* - Circuito integrado para aplicação específica,	p. 29
CCD	*Charge-Coupled Devices*,	p. 27
CMOS	*Complementary MetalOxideSemiconductor*,	p. 28
DSP	*Digital Signal Processor* — Processador digital de sinais,	p. 30
FFT	*Fast Fourier Transformer* — Transformada rápida de Fourier,	p. 30
FPGA	*Field-programmable gate array* — Arranjo de portas lógicas programável,	p. 15
IP	*Intellectual Property* — Propriedade Intelectual,	p. 30
IVA	Inspeção Visual Automática,	p. 15
LEDges	Técnica proposta nesta dissertação para deteção de irregularidades em bordas dos objetos sob inspeção,	p. 16
LEs	*Logic Elements* — Elementos Lógicos do FPGA.,	p. 86
PCI	Placa de Circuito Impressa,	p. 33
RISC	*Reduced Instruction Set Computer* — Computador com um Conjunto Reduzido de Instruções,	p. 30
VLWI	*Very Long Instruction Word* — Instruções com palavras muito largas,	p. 30

Sumário

1 Introdução

Apesar da maturidade dos processos industriais, falhas podem ocorrer e gerar defeitos nos produtos resultantes da produção industrial. Como exemplo, observamos a produção em série de partes mecânicas, popularizada após a segunda fase da revolução industrial, no início do século XIX. Neste caso é comum que os erros no processo produtivo impactem nas dimensões do objeto em processamento e precisem ser identificados antes da entrega deste produto para ser utilizado, seja como parte integrante de um sistema maior ou seja este já o produto final.

Como exemplo será apresentado um caso industrial real, observado na filial de uma empresa multinacional instalada no município do Cabo de Santo Agostinho, Pernambuco. O transporte inapropriado de tubos vazios de creme dental pode gerar leves amassamentos nestes tubos. E, se a irregularidade não for detectada a tempo, a injeção do creme dental no tubo amassado falhará e o creme transbordará, gerando parada imediata de todo o processo produtivo. Neste caso, a parada dura aproximadamente dez minutos para manutenção da injetora e esta falha ocorre com uma freqüência aproximada de duas vezes por dia.

Tradicionalmente, a inspeção visual é realizada por pessoas, geralmente especialistas (2). Apesar de pessoas terem a capacidade de realizar a maior parte destes trabalhos melhor que as máquinas, os humanos são mais lentos e a baixa frequência das falhas torna a inspeção uma tarefa enfadonha. Como conseqüência, pode ser necessária a mobilização de um grande número de recursos humanos para executar a inspeção. Além disto, profissionais especialistas são difíceis de encontrar no mercado e, quando contratados, de se manter na indústria. Adicionalmente, para formar bons profissionais é necessário fornecer treinamentos e as suas habilidades são desenvolvidas lentamente ao longo do tempo. Associado a estas dificuldades, existem casos onde a inspeção é tediosa ou difícil, mesmo para os melhores profissionais. Em algumas aplicações as informações devem ser rapidamente ou repetitivamente extraídas do cenário. Por exemplo, a inspeção de parafusos em trilhos de trem, com este em movimento a 200 km/h (3), e a detecção de irregularidades

em bocas de garrafas em uma linha de produção de 30.000 garrafas por hora (4) são tarefas impossíveis para um humano. Por fim, há ainda os casos onde a inspeção humana pode ser uma atividade perigosa ou o local do objeto a ser inspecionado pode ser de difícil acesso; por exemplo, em reatores nucleares, na indústria química e farmoquímica, em fornos industriais, no fundo do oceano, etc. Nestes casos, proibitivos para a inspeção humana, máquinas podem ser utilizadas para realizar as inspeções visuais necessárias.

A Inspeção Visual Automática (IVA) é uma das principais aplicações de visão de máquina [1]. Além de possibilitar inspeções complexas, em ambientes perigosos ou de difícil acesso, a inspeção visual automática também pode melhorar a produtividade e a qualidade nas linhas de produção industriais (6). Por exemplo, o sistema de IVA implementado por Fernandez et al.(7) analisa a classificação de gomos de laranjas. Foram definidos dois tipos de erros de classificação: o tipo I, gomos bons classificados como ruins, e o tipo II, gomos ruins classificados como bons. Não houve benefícios significativos na precisão da classificação relacionada aos erros tipo I. Para estes erros, a inspeção humana falha em 18% das amostras enquanto a inspeção automática falha em 14,6% das amostras (redução dos erros em 19%). O ganho significativo na precisão foi obtido em relação aos erros tipo II. Neste caso, a inspeção humana falha em 30% das amostras enquanto a inspeção automática falha em 10,4% destas (redução dos erros em 65%). Além da precisão, a velocidade da inspeção também houve ganhos significativos. O sistema automático foi implementado em duas arquiteturas: uma utiliza como unidade de processamento um Pentium III e, a segunda, utiliza um FPGA. A inspeção humana necessita de 600 ms/-gomo enquanto a inspeção automática precisa de apenas 64 ms/gomo ou 22 ms/gomo se utilizada a arquitetura com o Pentium III ou com o FPGA respectivamente. Neste caso, observa-se que as inspeções com FPGA são 27 vezes mais rápidas que a inspeção humana.

Em particular, dentre os diferentes tipos de IVA, um de grande importância e constantemente demandado pela indústria, é o tipo dimensional (6). Por exemplo, o sistema de IVA implementado por Duan et al.(4) identifica o diâmetro e centro das garrafas produzidas. Outros exemplos são a análise da qualidade da soldagem em PCB (8), a medição de dimensões de cones de auto-falantes (9) e identificação de irregularidades nas bordas de tubos. Este último será implementado neste trabalho de mestrado. Como pode ser observado, existe uma ampla gama de aplicações para a IVA, sendo que cada aplicação apresenta diferentes requisitos, principalmente de desempenho, taxa de acertos da classificação para os objetos inspecionados, portabilidade, eficiência energética e custo da solução.

1. Ramo da engenharia que usa visão computacional no contexto da manufatura. (5)

Apesar da relevância da IVA para a automação industrial, ainda existem alguns desafios que precisam ser resolvidos. Sistemas de IVA são, geralmente, baseados em soluções complexas, como por exemplo, redes neurais (4, 8) e transformadas de Hough (10, 11). Estas soluções requerem uma ampla capacidade de processamento e de armazenamento, sendo implementadas, normalmente, em computadores industriais (4, 8, 9).

Outro aspecto importante que influencia significantemente na complexidade do sistema, sobretudo no algoritmo necessário para realizar o processamento da imagem, são as técnicas de aquisição da imagem, incluindo o tipo de iluminação utilizado. Geralmente são utilizados iluminação com laser (9, 10), projetores de padrões (12, 13) e multiplexação na iluminação difusa (8, 14). Esta complexidade impacta, principalmente, no custo e no desempenho das soluções existentes no mercado, tornando estas proibitivas para certas aplicações onde é necessário obter elevadas taxas de inspeção ou reduzir custos do equipamento de inspeção.

Neste trabalho é descrita uma nova técnica para inspeção visual automática dimensional para aplicações de tempo real, LEDges. As contribuições desta técnica são:

1. Uso de uma nova estratégia para iluminação, baseada em LEDs de alta potência, que torna mais fácil a identificação das bordas de interesse;
2. Algoritmo muito simples e eficiente para realizar a segmentação, representação e descrição da imagem;
3. Uso reduzido de memória para realizar todo o processamento da imagem;
4. Eliminação da necessidade de capturar o quadro inteiro antes de realizar a inspeção nas bordas de interesse.

Também são avaliadas neste trabalho duas possibilidades de arquitetura para implementar a LEDges em sistema embarcado de baixa capacidade computacional, particularmente em termos de memória e processador. Estas possibilidades foram pesquisadas e pensadas a fim de resolver um problema industrial real de IVA onde é necessário detectar amassamentos em bordas de tubos de creme dental, em linha de produção industrial.

Pelas pesquisas realizadas, não existe uma solução de IVA disponível no mercado que atenda a todos estes requisitos. Com o objetivo de preencher esta lacuna tecnológica e validar a técnica proposta, após análise das possíveis arquiteturas, a LEDges foi implementada em duas configurações, baseadas em:

1. Microcontrolador e câmera analógica;

2. FPGA e câmera digital.

Foram medidos alguns parâmetros e comparados com outros trabalhos relacionados que utilizam diferentes técnicas e infra-estruturas em relação àquelas desenvolvidas neste trabalho. Estas comparações não são triviais uma vez que cada trabalho tem seus requisitos específicos. Porém, pode-se observar que, de um modo geral, as implementações realizadas neste trabalho obtiveram melhores taxas de inspeção em relação às obtidas pelos outros trabalhos relacionados com infra-estrutura semelhante.

Esta dissertação está estruturada conforme descrito a seguir: No capítulo 2 é contextualizada a IVA dimensional. Os trabalhos relacionados são analisados no capítulo 3. A técnica proposta, LEDges, é apresentada no capítulo 4. Uma avaliação das possíveis arquiteturas para implementação da técnica proposta pode ser vista no capítulo 5. Nos capítulos 6 e 7 são apresentadas duas implementações da técnica: a primeira baseada em microcontrolador e câmera analógica e, a segunda, baseada em FPGA e câmera digital. Por fim, os resultados são apresentados no capítulo 8 e as conclusões no capítulo 9.

2 A Inspeção Visual Automática

Neste capítulo serão abordados os tipos de inspeção visual automática e os requisitos para implementação de um sistema de IVA. Além disto, serão apresentadas as etapas do processamento da imagem em um sistema de IVA típico, além da arquitetura básica utilizada em sistemas deste tipo. Este capítulo se encerra com alguns exemplos de IVA dimensional.

2.1 Classificação das IVAs

Em sistemas de visão industrial, é possível classificar as inspeções em quatro tipos básicos: superficial, estrutural, operacional e dimensional (6). Na inspeção da qualidade superficial são analisados texturas, arranhões, rachaduras, grau de rugosidade, continuidade, etc. Um exemplo típico de aplicação é a análise da qualidade de carne de aves baseada na cor da amostra (15). O segundo tipo, a inspeção da qualidade estrutural, analisa a montagem de partes mecânicas em um produto maior, verificando o posicionamento de furos, cortes, rebites, parafusos e a presença de objetos estranhos. Por exemplo, na verificação da montagem de portas e janelas em automóveis na linha de produção (16). O terceiro tipo de inspeção, a de qualidade operacional, analisa a incompatibilidade das operações aos padrões e especificações técnicas. Um exemplo desta inspeção é a verificação do processo de soldagem a *laser* (17). Por fim, a inspeção da qualidade dimensional está relacionada com a inspeção de dimensões, formas, posição, orientação, alinhamento. Este tipo de inspeção é o objeto principal desta dissertação.

Checar se as dimensões de um objeto estão dentro das tolerâncias especificadas ou se os objetos tem a forma correta, são tarefas comuns nas indústrias (6). Tais tarefas envolvem a inspeção de características geométricas do objeto, em duas ou três dimensões, e são denominados inspeção da qualidade dimensional, ou, simplesmente, inspeção dimensional. Várias indústrias estão envolvidas no desenvolvimento de sistemas de visão para automatizar a medição da qualidade dimensional. As tarefas variam desde verificar o nível

de um líquido na garrafa em indústrias de embalagens, até verificar e transmitir pela internet em tempo real a concentricidade e o diâmetro de bombas de *airbags* em carros. Um exemplo de IVA dimensional em tempo real pode ser visto em (18). Nesta aplicação são comparados produtos impressos em alta resolução, com alto grau de similaridade, onde a detecção de diferenças mínimas torna este trabalho bastante difícil para os humanos, principalmente quando este deve ser realizado em tempo real.

2.2 Requisitos para sistemas de inspeção visual automática

Existe uma ampla gama de aplicações para a IVA. Para cada domínio de aplicações existem diferentes requisitos, tais como a taxa de inspeção, a informação relevante na cena que deve ser analisada, a precisão, a portabilidade, o consumo energético, os recursos computacionais necessários, o tipo do sinal de saída do sistema, etc. As diferenças nos requisitos existem uma vez que características do ambiente e da resposta esperada pelo sistema também são diferentes, tais como a iluminação local, objeto a ser inspecionado, velocidade da produção, características que precisam se analisadas no objeto (tipo de inspeção), infra-estrutura pré-existente para realizar a interface com o usuário (ex.: redes e controladores industriais), a ação desejada como resposta, etc. Por exemplo, em algumas aplicações, o sistema deve ser capaz de diferenciar objetos com variações aceitáveis nas medidas dimensionais daqueles com variações inaceitáveis, enquanto em outras aplicações, o sistema deve ser capaz de identificar problemas de alinhamento em montagem de partes mecânicas. Logo, não existem sistemas de visão industrial capazes de abranger todas as tarefas, em todos os campos de aplicação.

O primeiro problema a resolver para automatizar as tarefas de inspeção visual é identificar que tipo de informações a máquina busca e como traduzir estas informações em medições ou características extraídas das imagens. Por exemplo, é importante especificar o que "defeito" significa em termos de medições e regras. Somente então é possível implementar estas atividades em *software* ou *hardware* (6). São comuns nas indústrias aplicações que visam simplesmente classificar o objeto sob inspeção entre "bom" e "ruim" [1].

Para que o sistema seja confiável, este deve reduzir o máximo possível os *"escape rates"*, que são os casos inaceitáveis detectados como aceitáveis, e os *"false alarms"*, que

1. Na IVA dimensional, objeto "bom" é aquele que tem valores de seus parâmetros dimensionais de acordo com os padrões estabelecidos pela indústria. Analogamente, objeto "ruim" é aquele que tem seus parâmetros desconformes com os padrões estabelecidos.

são os casos aceitáveis detectados como inaceitáveis. É responsabilidade das unidades de processamento manter o sistema confiável, sendo que a eficiência da classificação depende também da qualidade da imagem adquirida. Além de confiável, o sistema de visão industrial também deve ser robusto. Isto é, deve ser capaz de se adaptar a variações na iluminação, a marcas no objeto e realizar a inspeção mesmo com o objeto em posições incertas, sem perder a precisão e o desempenho. Além disto, o sistema de IVA deve ser capaz de identificar o objeto sob inspeção, mesmo que o plano de fundo seja composto por outros objetos com diversas formas, texturas e iluminação. Altos níveis de robustez são muito difíceis de obter ou, pelo menos, requerem um uso muito intenso de recursos computacionais. Logo, é desejável que o processo industrial possua o mínimo de incertezas nos parâmetros relacionados com o ambiente da inspeção.

Um sistema de visão industrial deve ser rápido e eficiente no custo (6) e, inclusive, possibilitar rápidas taxas de inspeção mesmo sem uso de câmeras de alta velocidade. Finalmente, um sistema de IVA deve permitir a sua instalação em espaço físico restrito. Nem sempre é possível realizar a montagem da nova solução de IVA na infra-estrutura de produção pré-existente.

Desta forma, existe uma demanda crescente por sistemas de IVA embarcados visando atender às aplicações que necessitam melhores índices de desempenho e otimizar o uso de recursos computacionais, como observado em (3, 7, 19).

2.3 Etapas de processamento da imagem em um sistema de IVA típico

A Fig. 1 mostra as etapas do processamento de imagens em um sistema de IVA típico. Estas etapas podem ser divididas em três níveis distintos de processamento. O primeiro nível, processamento de baixo nível, trata da aquisição da imagem e o melhoramento da qualidade desta. O nível intermediário resulta na descrição da imagem em um conjunto numérico digital capaz de ser processado por um sistema computacional. Por fim, o processamento de alto nível é responsável pelo reconhecimento da imagem e, finalmente, sua interpretação indicando se o objeto sob inspeção está, ou não, dentro dos padrões aceitáveis pela indústria.

A "base de conhecimento" contém uma base de dados e conhecimentos acumulados sobre a inspeção, pelo homem ou pela máquina. A interação com a base de conhecimento em todas as etapas é essencial para tornar mais precisas as tomadas de decisão, e deve

ser vista como parte integrante do processamento da imagem (20).

Parte dos conhecimentos acumulados podem ser experiências da equipe em relação a uma inspeção manual que será automatizada, onde já são conhecidos os resultados relacionados aos problemas e suas soluções. Por exemplo, na seleção manual de uma região da imagem aonde está o objeto que será inspecionado. Outro exemplo de banco de conhecimento é um banco de dados contendo todas as especificações de uma indústria para a fabricação de um produto. Estes dados serão consultados pelo computador e, se os parâmetros adquiridos durante a IVA não estiverem de acordo com os valores registrados no banco de dados, o objeto sob inspeção será classificado como "ruim".

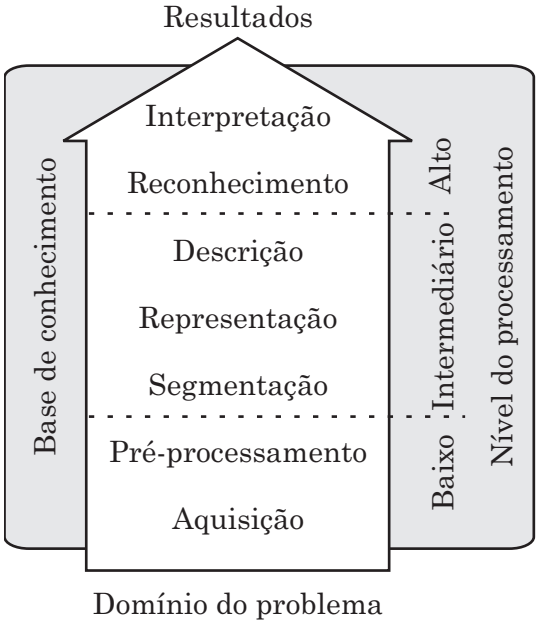


Figura 1: Etapas de processamento de um sistema de IVA típico.

2.3.1 Processamento de baixo nível

O primeiro nível de processamento, o "baixo nível", é dividido em duas etapas: (i) aquisição da imagem e (ii) pré-processamento.

Aquisição da imagem

A etapa de aquisição da imagem consiste em capturar e transferir a imagem de uma câmera, *scanner* ou arquivo salvo em uma unidade de armazenamento para a unidade de processamento. No caso da IVA em tempo real, os dispositivos de captura da imagem são uma ou mais câmeras, sendo que algumas destas câmeras podem ser configuradas para capturar apenas a região de interesse e para ajustar o tempo de exposição.

É possível aumentar a velocidade da inspeção a partir da escolha da região de interesse,

quando áreas periféricas da imagem, sem informações relevantes para a inspeção, são excluídas. Isto se dá uma vez que o algoritmo de processamento poderá ser executado em um menor número de pixels. Além disto, é possível melhorar a taxa de acertos da inspeção se uma área de difícil segmentação puder ser excluída nesta etapa do processamento. A segmentação da imagem será tratada na seção 2.3.2.

Quanto ao tempo de exposição, este é fundamental na determinação do tempo total da IVA. Quanto mais rápida for a exposição, mais rápida será a aquisição da imagem, porém, menos luz será captada pelo sensor. Logo, para aumentar a velocidade da aquisição através da redução do tempo de exposição, é necessário iluminar vigorosamente o objeto sob inspeção. Adicionalmente, se a iluminação for estruturada de modo a iluminar somente a região relevante do objeto sob inspeção, é possível ainda que a redução no tempo de exposição reduza o esforço computacional para realizar a segmentação da imagem uma vez que as informações irrelevantes do objeto e do plano de fundo não serão captadas pelo sensor.

Pré-processamento

A segunda etapa do processamento de baixo nível é o pré-processamento da imagem. Esta etapa visa melhorar a qualidade da imagem através da redução de ruídos e distorções, além do ajuste de brilho e contraste. Com isto é possível reduzir os esforços computacionais para extrair as características relevantes da imagem para a IVA. Por exemplo, o pré-processamento pode ser realizado através de um filtro mediano para reduzir os ruídos impulsivos, implementado em FPGA (21). As tarefas do pré-processamento podem ser realizadas na própria câmera, em um *hardware* específico para processamento de imagens, ou no sistema computacional principal. O *hardware* no qual será implementado o pré-processamento depende dos requisitos de desempenho do sistema.

Quando não é exigido um elevado desempenho, o pré-processamento poderá ser realizado somente no sistema computacional principal, que é para uso geral, como realizado na inspeção de garrafas apresentado em (4). Porém, quando é exigido um elevado desempenho, sobretudo em aplicações em tempo real, o pré-processamento deve ser realizado na câmera, para ajuste de brilho e contraste ou em um *hardware* específico, para a redução de ruídos e distorções. Um exemplo desta arquitetura está no sistema de IVA implementado por Fernandez et al.(7) que analisa a classificação de gomos de laranjas.

2.3.2 Processamento de nível intermediário

O nível intermediário de processamento é composto pelas etapas de: (i) segmentação, (ii) representação e (iii) descrição da imagem.

Segmentação

A segmentação, que identifica as regiões de interesse, pode ser uma das atividades mais complexas do processamento da imagem. Por um lado, um procedimento de segmentação robusto favorece substancialmente a solução bem sucedida de um problema de processamento de imagens. Por outro lado, algoritmos de segmentação inadequados quase sempre geram falha no processamento.

Tipicamente o algoritmo de segmentação visa localizar regiões e formas específicas nas imagens. O resultado é um conjunto de regiões ou um conjunto de contornos (bordas) extraídos da imagem. Por exemplo, no caso de reconhecimento de caracteres, o papel básico da segmentação é extrair caracteres individuais e palavras do fundo da imagem. A segmentação pode ser baseada em formatos, tal como na detecção de descontinuidades, pontos, linhas e bordas, ou pode ser baseada nas características dos pixels, tal como cores e intensidades.

Vários algoritmos e técnicas específicas de segmentação foram desenvolvidos, não havendo, porém, uma solução geral para o problema de segmentação de imagens. Muitas vezes para a resolução de um problema de segmentação de imagem é necessário a combinação de técnicas visando sua adaptação ao domínio do problema.

Representação

Em seguida, a imagem é representada em termos de suas características internas (pixels que compõem a região) ou externas (bordas). De um modo geral, a representação pela borda é adequada quando o interesse se concentra nas características da forma externa, tais como cantos ou pontos de inflexão. A representação por região é adequada quando o interesse se concentra em propriedades internas, tais como textura ou a forma do esqueleto. Por exemplo, a representação baseadas nos pixels que compõe a região pode ser observada em (22), inclusive a utilização de um limiar para "binarização" da imagem pode ser observada em (10, 23). A representação pelas bordas é mostrado em (4, 8).

Descrição

A próxima etapa consiste em descrever a região de interesse com base na representação escolhida. Por exemplo, a região pode ser representada por suas bordas e descrita pelo

tamanho ou número de concavidades nesta borda. Assim como nas etapas anteriores, o principal objetivo é fazer com que as características de interesse sejam enfatizadas.

Um exemplo de descrição é o conjunto de assinaturas gerado em (9), inspeção de cones de alto-falantes, quando a altura do cone é descrita por um conjunto de medidas de distância entre a base da mesa e o topo do cone.

O processamento de nível intermediário é realizado no sistema computacional principal, de uso geral, ou, quando necessário um desempenho superior, em um *hardware* específico. No caso de utilizar um *hardware* específico, é importante pesquisar os modelos que disponibilizam um pacote de *software* integrado ao hardware. Por exemplo, o Sherlock, da Teledyne Dalsa [2], é um pacote *software* baseado em Windows que se integra à plataforma MVTools. Estas ferramentas possuem 204 funções de visão computacional. Alguns algoritmos disponíveis são de geração de padrão, convolução, detecção de bordas, transformada de Fourier, transformações geométricas, entre outras.

2.3.3 Processamento de alto nível

O processamento de alto nível é composto pelo reconhecimento e interpretação da imagem.

Reconhecimento

O reconhecimento de padrões é a primeira etapa do processamento em alto nível. O reconhecimento consiste basicamente na classificação da descrição do objeto. Por exemplo, na leitura automática de um texto impresso, os padrões de interesse são os caracteres alfanuméricos. A meta é atingir uma dada precisão de reconhecimento de caracteres que sejam a mais próxima possível à excelente capacidade exibida por seres humanos na realização de tais tarefas.

Interpretação

A última etapa da IVA é a interpretação da imagem, que consiste em atribuir um significado aos diversos elementos reconhecidos na imagem.

Ao final do processamento de alto nível o sistema será capaz de tomar decisões, como controlar um processo produtivo (por exemplo, guiando um braço robótico), propagar características para outras etapas do processo (por exemplo, separação de tipos diferentes de peças) e detectar defeitos ou falhas (por exemplo, descartando peças quebradas).

2. http://www.teledynedalsa.com/

Em geral, os algoritmos para reconhecimento e interpretação da imagem são executados nos sistemas computacionais principais, de uso geral. Porém, nos casos em que é necessário maior desempenho, podem ser utilizados *hardwares* específicos que descrevem a rede neural ou lógica fuzzy, como o W.A.R.P., da SGS-Thomson [3]. Este é projetado para acelerar aplicações fuzzy. Para desenvolvimento de redes neurais, por exemplo, há o Trajan 6.0, da Trajan software [4]. Este software permite a construção de redes híbridas, *perceptrons* multicamadas, Kohonen, pseudo-Newton, entre outros. A saída deste software gera como resultado um código fonte em C ou um API de C/C++, Visual Basic, Excel, etc. É importante sempre verificar se a plataforma de *hardware* que será adquirida tem pacotes de *software* que permitem uma descrição em alto nível da rede neural ou lógica fuzzy para geração de um código otimizado.

2.4 Arquitetura típica de um sistema de IVA

Esta seção apresenta uma visão geral da arquitetura típica de um sistema de IVA. A Fig. 2 mostra uma arquitetura típica como parte integrante de um processo produtivo industrial, composta por:

1. Sistema de iluminação contendo um ou mais emissores de luz, naturais ou artificiais;
2. Conjunto óptico-eletrônico para captura da imagem, composto por uma ou mais câmeras;
3. *Hardware* específico para processamento de imagens;
4. Sistema computacional principal;
5. Sistema de controle do processo produtivo.

2.4.1 Iluminação

O primeiro item da arquitetura, o sistema de iluminação, mostrado na Fig. 2 (1), tem como propósito destacar as informações de interesse no cenário ou minimizar o efeito de outras fontes de luz sobre o objeto em inspeção. Nos dois casos o foco é melhorar a qualidade da imagem capturada e, com isto, reduzir o esforço computacional necessário para extrair as características relevantes da imagem associadas com as não conformidades em relação às características esperadas para o objeto. A iluminação pode ser bastante

3. http://www.st.com
4. http://www.trajan-software.demon.co.uk

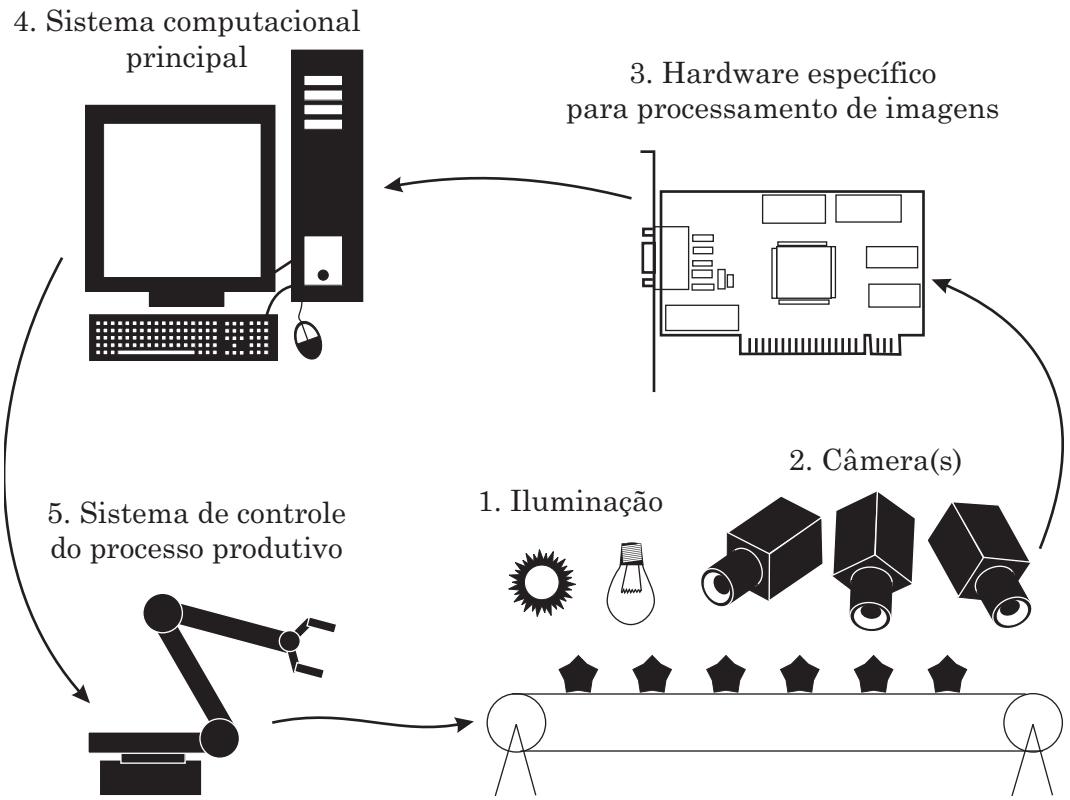


Figura 2: Arquitetura típica de um sistema de IVA.

simples, como a utilizada na inspeção de ferrovias (3). Neste caso a iluminação foi projetada para reduzir os efeitos da variação natural da luz nos trilhos dos trens. Para tal, foram utilizadas seis lâmpadas halógenas de 100 W com refletor tipo *"spot"*, sem controle de intensidade.

O sistema de iluminação também pode ser bastante complexo, como na medição de formas tridimensionais, apresentado em (13). Neste caso a iluminação é realizada através de um projetor. Durante a medição do objeto 3D, um conjunto de padrões de franjas é projetado sobre a superfície do objeto de interesse e a característica de profundidade do objeto naturalmente irá distorcer estes padrões de franjas. Neste momento, o objeto e os padrões de franjas distorcidos são capturados pela câmera sob forma de imagem digital para ser processada e gerar um modelo 3D do objeto. Outro exemplo de uso de projetores de padrões é o trabalho apresentado em (12). Este trabalho utiliza o mesmo princípio básico da triangulação entre o projetor, o objeto sob inspeção e a câmera, para determinar o formato do objeto. O diferencial deste trabalho foi o aprimoramento para aplicação em clínicas, no armazenamento digital dos modelos ortodônticos.

A iluminação também pode ser realizada utilizando *laser*. Em (9), inspeção de cones de alto-falantes, o *laser* atua com um princípio de triangulação para determinar a distância

entre o objeto e a câmera, semelhante aos sistemas que utilizam iluminação por projetores. Já em (10), inspeção de frutas em árvores, o módulo *laser* disponibiliza para a unidade de processamento as informações de distância entre o *laser* e o objeto, bem como a atenuação causada pela absorção da luz na superfície do objeto.

O último modelo que será apresentado é a iluminação multiplexada. Em (8), inspeção de PCI, são utilizadas três lâmpadas circulares com LEDs, todas com diâmetros diferentes, localizadas no topo do sistema e centralizadas com a PCI sob inspeção. Estas lâmpadas são multiplexadas no tempo, ou seja, acendem seqüencialmente, uma por vez. Logo, é necessário que a câmera capture três imagens, uma para cada lâmpada acesa, para que o processador realize a inspeção das soldas. De acordo com a quantidade de luz refletida na solda, nos três ângulos de incidência da luz, o processador será capaz de decidir se a solda está dentro dos padrões, ou não. A iluminação também pode ser multiplexada no comprimento de onda (cor da luz). Em (14) foram instalados três emissores de luz em ângulos diferentes em relação a superfície sob inspeção. Um vermelho, um verde e um azul. A luz refletida na superfície pelos três emissores é capturada por uma câmera RGB. Este tipo de câmera tem três canais de saídas independentes: um vermelho, um verde e um azul. É mostrado no trabalho (14) que, a partir da análise dessa imagem composta RGB, é possível determinar a textura da superfície sob inspeção.

2.4.2 Câmeras

O segundo item da arquitetura básica é o conjunto óptico-eletrônico para captura da imagem [5]. A função das câmeras é implementar a primeira etapa do processamento da imagem: a aquisição, conforme apresentado na seção 2.3.1. Este item da arquitetura é composto, geralmente, por uma ou mais câmeras em posições fixas, conforme apresentado na Fig. 2 (2).

Em relação ao tipo do sensor de luz utilizado na câmera, existem dois tipos básicos: CCD e CMOS. Os sensores CCD (*Charge-Coupled Devices*) são os mais utilizados em aplicações de visão computacional. A câmera CCD tem uma matriz de elementos sensíveis a luz, chamados de pixels. Logo, todos os pixels da imagem são capturados simultaneamente em valores contínuos. A digitalização de todos os pixels da imagem é realizada em uma etapa posterior. Esses sensores também têm a característica de possuir uma relação linear entre a intensidade de luz da entrada e o sinal elétrico da saída. Esta característica

5. A captura da imagem se refere ao processo de conversão da luz refletida ou emitida pelo objeto observado em um sinal elétrico. Este sinal pode ser analógico ou digital.

torna essas câmeras adequadas para aplicações de metrologia.

Já os sensores CMOS (*Complementary Metal-Oxide-Semiconductor*) convertem a imagem pixel por pixel, já no formato digital, seqüencialmente, do topo até a base da imagem. Esta característica pode ser indesejada para aplicações onde a iluminação é realizada por lâmpadas fluorescentes ou com *flash* de luz. Nestes casos a imagem pode ser adquirida com uma parte clara e outra escura. Algumas vantagens das câmeras CMOS são o baixo consumo energético, alta resolução da câmera e a alta velocidade de captura da imagem.

Quanto ao sinal de saída das câmeras, este pode ser classificado em analógico ou digital. Os padrões analógicos mais difundidos são os compostos (NTSC/PAL, EIA/CCIR), Y-C (S-Video) e RGB. Destes, os sinais NTSC e EIA são os mais comuns para a maioria das aplicações de IVA. Já os sinais Y-C e RGB fornecem imagem em qualidade superior, com as cores em canais separados. A utilização de câmeras analógicas resultam normalmente em baixa resolução da imagem, baixa taxa de aquisição (normalmente 30 quadros por segundo), sinal com mais ruídos e saída entrelaçada [6]. Além disto, antes de processar a imagem em computador é necessário que os sinais destas câmeras sejam digitalizados a partir de um hardware específico de aquisição de imagens.

Quanto ao formato digital, estas têm algumas vantagens sobre as analógicas. Geralmente câmeras digitais são configuráveis, podendo ser capturada a imagem com maior velocidade ou maior resolução. Além disto, há pouca perda de sinal, baixo nível de ruído e a varredura da imagem é realizada de maneira não-entrelaçada [7]. Os padrões mais comuns são CameraLink™ (CL), IEEE-1394 (*Firewire*), *Gigabit Ethernet* (GigE), RS-422 e RS-644. As escolhas imediatas são GigE e CL. A escolha depende da velocidade, sincronização e geografia. Somente a GigE permite longas distâncias em cabos comuns. Já a CL é mais rápida que qualquer outra interface. Além disto, somente a CL inclui facilidades para *real-time triggering*.

Quanto às cores, a câmera pode ser monocromática, colorida com um sensor ou colorida com três sensores. As câmeras monocromáticas têm um sensor único que fornece o nível de cinza. Tem a vantagem de a resolução ser 10% superior às câmeras coloridas de um sensor. Além disto, possui melhor relação sinal/ruído, melhor contraste e maior sensibilidade para baixa iluminação.

Já as câmeras coloridas de um sensor usam, geralmente, filtro de cor Bayer RGB. Esta

6. Um quadro é dividido em dois campos (pares e ímpares). Varre-se as linhas ímpares (1,3,5) e então as pares (2,4,6). Estes campos são então integrados para produzir um quadro completo

7. *Progressive Scan* - Técnica que "varre" a tela inteira em uma única passada, transmitindo todas as linhas da imagem a cada atualização

tecnologia é de baixo custo e se integra facilmente aos sistemas de IVA. Porém a resolução é inferior às câmeras coloridas de três sensores. Esta última tecnologia, com três sensores, utiliza um prisma para dividir a luz branca em três componentes. Como vantagem, estes sensores tem melhor resolução de cor. Por outro lado, esta tecnologia é mais cara e possui sensibilidade mais baixa.

Por fim, existem as câmeras inteligentes (*SmartCams*). Estas câmeras integram o *hardware* e o *software* dentro da própria câmera. Fazem parte da arquitetura destas câmeras, além do sensor, um processador, *frame grabber*, interface serial, ethernet ou USB, entrada para *triggers* de alta velocidade, E/S para CLPs, memória RAM e cartões de memória SD.

2.4.3 *Hardware* específico para processamento de imagem

O terceiro item da arquitetura, mostrado na Fig. 2 (3), é o *hardware* específico para executar parte do processamento da imagem. De fato, este item é normalmente utilizado para implementar o pré-processamento da imagem, conforme apresentado na seção 2.3.1. Porém, sobretudo nos casos que necessitam desempenho superior, este *hardware* específico também é utilizado para implementar as três etapas do processamento de nível intermediário: segmentação, representação e descrição da imagem, conforme apresentado na seção 2.3.2

Nos sistemas de IVA, a implementação de todas as etapas do processamento da imagem exclusivamente em *software*, geralmente, não é suficiente para atingir os requisitos de desempenho para a aplicação. Visando solucionar este gargalo, cada vez mais está sendo implementada parte do algoritmo de processamento da imagem em *hardware*. A seguir serão vistas algumas características da implementação em ASIC, DSP e FPGA. A escolha da tecnologia a ser utilizada deverá ter base no tamanho do chip, potência dissipada, custo e desempenho. Além disto, é importante observar a flexibilidade do uso, ambiente de programação e pacotes de *softwares* disponíveis. Estas últimas características são de grande relevância para os desenvolvedores do sistema de IVA.

ASIC

A primeira tecnologia para realizar o processamento da imagem em *hardware* é construir um circuito integrado (CI) exclusivo para a aplicação de IVA utilizando um ASIC (*Application Specific Integrated Circuit* - Circuito integrado para aplicação específica).

Atualmente a forma mais eficiente de desenvolver um ASIC é a utilização de núcleos IP [8] (*Intellectual Property* — Propriedade Intelectual) disponíveis no mercado. O tempo de desenvolvimento é reduzido por que as células adquiridas prontas já foram testadas em diferentes tecnologias de fabricação. De maneira simplificada, o design de um ASIC consiste na montagem de blocos IP, inserindo alguma lógica entre estes blocos para realizar a interface. A principal desvantagem do ASIC é que o circuito geralmente tem sua utilização limitada em uma aplicação específica.

Para reduzir a inflexibilidade, alguma programabilidade pode ser incorporada no ASIC. Uma opção é utilizar núcleos IP de processadores. Algumas empresas, como a ARM e a Synopsys, disponibilizam núcleos DSP RISC, já projetados para aplicações embarcadas. Um produto interessante oferecido pela Synopsys é o núcleo ARC [9]. Este é um processador de 32 bits, muito pequeno (em área ocupada) e de alta velocidade (com *clock* acima de 100 MHz). Além disto, a Synopsys disponibiliza uma ferramenta que permite a configuração do processador ARC de acordo com os requisitos da aplicação. Por exemplo, dependendo da aplicação, a ferramenta pode configurar o processador com maior *cache* de instruções e disponibilizar mais memória RAM ou registradores para uso geral. O código VHDL é então gerado e poderá ser sintetizado.

Outra forma de reduzir a inflexibilidade do ASIC é implementar em um único chip diversos algoritmos de processamento de imagens comuns para um domínio de aplicações. Por exemplo, um CI pode ser desenvolvido para realizar a transformada rápida de Fourier (*Fast Fourier Transformer - FFT*), detecção de bordas, transformada de Hough, entre outros. Com isto, o ASIC poderá ser reutilizado; para tal sendo necessário apenas modificar alguns sinais de entrada, parâmetros informados pelo sistema computacional principal ao chip.

DSP

Uma alternativa de menor custo é utilizar um DSP ao invés de confeccionar um ASIC. O uso de placas com DSP tem sido extensivamente utilizado em aplicações de visão industrial com requisitos de tempo real. Uma família de placas com DSP muito popular é a "Connected Media Processor", da NXP Semiconductors [10] (antiga Philips Semiconductors). Estas placas tem interface PCI e usa processadores da série TriMedia, que tem tecnologia VLIW (*Very Long Instruction Word*) voltada para aplicações multimídia, com

8. No design de semicondutores, *IP Core*, ou núcleo IP, é um bloco reutilizável de lógica, células ou layout de chip. Um núcleo IP pode ser desenvolvido e licenciado para terceiros.
9. http://www.synopsys.com/IP/ConfigurableCores/ARCProcessors/Pages/default.aspx
10. http://www.nxp.com

arquitetura Harvard e que permite processar eficientemente fluxos de dados de vídeo e de áudio. O *clock* para este processador pode ser superior à 500 MHz.

FPGA

FPGAs são competitivos em relação aos ASICs em termo de capacidade e desempenho. Uma vantagem do FPGA é permitir, rapidamente, testar o sistema em condições reais. Outra vantagem é que, quando comparado ao ASIC, o FPGA é eletricamente reprogramável. Ou seja, o FPGA pode ser reusado no caso de modificações nos requisitos do sistema ou simplesmente quando ocorrer algum erro no processo de design do FPGA. Adicionalmente, é possível inserir um núcleo IP no FPGA assim como realizado no ASIC (por exemplo, um ARM ou um PowerPC com memória RAM e periféricos).

A Acromag [11], por exemplo, desenvolve uma ampla linha de placas de processamento de imagens para uso industrial baseado em FPGA, tanto Xilinx®, quanto Altera®. A Acromag também disponibiliza para *download* diversos códigos com algoritmos para processamento de imagem. Outro exemplo é a placa utilizada neste trabalho de mestrado, a "Altera Digital Camera Development Platform", da Terasic [12]. Esta é uma placa desenvolvida para fins educacionais que utiliza um FPGA Cyclone II, da Altera. Além do FPGA, a placa disponibiliza uma câmera digital de 5 Mpixels, um display LCD de 4,3", interface USB, SD Card, memória RAM, entre outros periféricos.

2.4.4 Sistema computacional principal

O sistema computacional principal tem como função implementar, sobretudo, as etapas de processamento de alto nível: reconhecimento e interpretação da imagem, conforme apresentado na seção 2.3.3.

O sistema computacional principal, mostrado na Fig. 2 (4), geralmente contém um processador de uso geral, por exemplo um Pentium®, da Intel®, ou um Athlon®, da AMD®, que está instalado em um PC. Na realidade, desde o lançamento das instruções MMX com a tecnologia SIMD no Pentium, em 1997, processadores de uso geral estão executando mais etapas do processamento da imagem. De fato, em alguns casos, principalmente aqueles que não necessitam de processamento em tempo real, o processamento da imagem é totalmente realizado por um processador de uso geral. Por exemplo, na inspeção de garrafas (4), de frutas em árvores (10) e de soldas em PCI (8).

11. http://www.acromag.com/
12. http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=53&No=296

A principal vantagem do processador de uso geral é o ambiente de programação, que permite que o programa seja desenvolvido sem nenhum conhecimento prévio da arquitetura do computador. Por outro lado, o processador de uso geral consome mais energia que as placas específicas para processamento da imagem e, geralmente, são proibitivos para a aplicações embarcadas.

2.4.5 Sistema de controle, sensores e atuadores do processo produtivo

O último item da arquitetura básica da IVA é mostrado na Fig. 2 (5). Este sistema é composto por um controlador industrial, geralmente um CLP (Controlador Lógico Programável), sensores e atuadores. Este conjunto de elementos tem a função básica de identificar o posicionamento do objeto antes da inspeção e atuar mecanicamente no objeto visando mudar a sua posição ou forma após a inspeção.

Segundo a Associação Brasileira de Normas Técnicas (ABNT), um CLP é um equipamento eletrônico digital com *hardware* e *software* compatíveis com aplicações industriais. Adicionalmente, é um dispositivo com funcionalidades semelhante as de um microcontrolador, com seus periféricos, porém adaptado para ambientes industriais.

Os elementos sensores são dispositivos que recebem um sinal, seja óptico, sonoro, térmico ou proveniente de qualquer outra forma de energia, convertem para um sinal elétrico e condiciona este sinal para uso pelo CLP ou pelo sistema computacional principal do sistema de IVA. Os principais sensores utilizados em IVA são destinados a detectar a presença do objeto para iniciar a inspeção e, geralmente, estes sensores têm natureza capacitiva, indutiva ou óptica.

Por fim, os elementos atuadores completam a IVA permitindo mudanças na posição ou forma do objeto durante ou após a inspeção. Os atuadores mais comuns em IVA são os de cilindro pneumático, cilindro hidráulico e motores. Ainda são exemplos de atuadores: garras, pulverizadores de tinta e eletroímãs.

2.5 Exemplos de IVA dimensional

O primeiro trabalho a ser apresentado visa melhoramentos em uma aplicação popular e muito demandada: a inspeção e classificação de soldas em placas de circuito impressa (PCI). Um sistema típico para esta aplicação consiste em um sistema de iluminação apro-

priado e uma câmera no topo do sistema de inspeção (8). O processamento da imagem da PCI consiste em duas etapas principais. Na primeira, um pré-processamento é realizado para remover os ruídos da imagem e tornar a identificação das soldas mais fácil. Então, na segunda etapa, as soldas são classificadas de acordo com o tipo de defeito. A classificação normalmente é realizada a partir da quantidade de solda presente na placa. Quatro classes foram definidas: (I) solda na quantidade normal, nominada como boa, (II) excesso de solda, (III) solda insuficiente e (IV) sem solda. Resultados de simulação nos modelos geométricos das soldas têm mostrado que a classificação eficiente pode ser atingida somente após uma seleção de características ótimas, isto para que as classes não se sobreponham (24). Pesquisas tem mostrado que técnicas baseadas no histograma (25) tem desempenho melhor que técnicas baseadas na extração de características baseada na imagem em duas e três dimensões. Muitas inspeções de PCI utilizam redes neurais no projeto dos classificadores.

Uma técnica para o problema de cortar partes com formas irregulares (inclusive formas côncavas) em folhas bidimensionais (papel, couro, tecido, etc.) é visto em (26). Um sistema de visão de máquina é utilizado para adquirir as imagens das partes irregulares que serão reproduzidas industrialmente em maior escala posteriormente. Após a captura da imagem, é realizada a detecção e poligonarização das bordas. Ou seja, a visão de máquina converte um pedaço de folha cortado irregularmente em um polígono no computador. A partir desta etapa, um algoritmo genético é usado para gerar diversas combinações de posicionamento das peças irregulares dentro da folha inteira, visando minimizar as perdas de material no processo de corte destes pedaços irregulares. Esta técnica é particularmente útil para as indústrias de couro e vestuário, onde cortes irregulares são normalmente utilizados para a confecção do produto final.

Um sistema de IVA para localização de frutas esféricas em árvores, sob condições naturais, é apresentado em (10). O sistema utiliza um *laser* que fornece informações de distância entre o *laser* e o objeto sob inspeção, e de atenuação da luz, que expressa a quantidade de luz absorvida pela superfície do objeto. A partir destas informações, um algoritmo é utilizado para identificar formas esféricas e suas posições na árvore. Resultados experimentais mostram que aproximadamente 85% das frutas visíveis e 55% das parcialmente visíveis foram corretamente detectadas. A tecnologia foi embarcada no sistema robótico "AGRIBOT" visando a coleta automática de frutas.

Uma atividade muito comum em IVA é detectar linhas e círculos em imagens. A técnica apresentada em (27) se concentra na detecção de bordas e identificação de par-

tes circulares, inclusive círculos irregulares. Para tal, uma transformada de Hough em dois estágios é utilizada. Outro exemplo de detecção de formas primárias pode ser visto em (28). Neste artigo são detectadas as elipses da imagem, inclusive para aplicação de reconhecimento facial. Para tal, assim como em (27), é necessário utilizar um algoritmo de detecção de bordas associado a uma transformada de Hough.

Um exemplo de IVA dimensional em tempo real pode ser visto em (18). Nesta aplicação são comparados produtos impressos em alta resolução, com alto grau de similaridade, onde a identificação de diferenças mínimas é bastante difícil para humanos, principalmente quando este deve ser realizado em tempo real. Um algoritmo original, baseado em operações morfológicas, facilita a detecção de imperfeições em uma resolução capaz de identificar irregularidades do tamanho de um pixel. O sistema pode ser aplicado para a inspeção de cédulas, cheques e outros documentos bancários, considerando os requisitos para validação das impressões produzidas.

O último trabalho que será apresentado como exemplo de IVA dimensional nesta seção é a inspeção de roscas de parafusos (29). Neste trabalho é verificado se a rosca atende aos padrões do fabricante. Um algoritmo para detecção de bordas, baseado em interpolação linear para obter uma resolução sub-pixel, é aplicado para detectar as bordas na região de interesse. Em seguida, é verificado se a borda está inteiramente compreendida dentro da faixa de tolerância, definida pelo fabricante. O sistema foi testado em linha de produção e obteve desempenho melhor que outros métodos competitivos, inclusive a medição manual.

2.6 Análise do capítulo

Várias indústrias estão envolvidas no desenvolvimento de sistemas de visão para automatizar a inspeção das características geométricas de objetos. Isto se dá por que existe uma ampla gama de aplicações para a IVA dimensional e, para cada domínio de aplicações, existem diferentes requisitos provenientes das características ambientais desiguais entre aplicações e das diversas formas de respostas esperadas do sistema de IVA. Logo, não existem sistemas de visão industrial capazes de abranger todas as tarefas, em todos os campos de aplicação.

De um modo geral, é necessário que os sistemas de IVA identifiquem que informações são importantes nas imagens, extraí-las e utilizá-las para classificar os objetos. Vários trabalhos analisados utilizaram estas informações para classificar o objeto sob inspeção entre “bom” e “ruim”, com a confiabilidade (taxa de acerto) adequada para a aplicação.

É de responsabilidade do sistema computacional manter o sistema confiável, porém, o sistema de iluminação e de aquisição da imagem influencia significativamente o uso de recursos computacionais necessários para manter a confiabilidade.

O sistema de IVA também deve ser robusto e uma das principais dificuldades para obtenção desta característica é a composição do plano de fundo por outros objetos com diversas formas, texturas e iluminação. A segmentação pode ser uma das atividades mais complexas do processamento da imagem e, se utilizado um algoritmo inadequado, quase sempre resulta em um comprometimento completo do sistema de IVA.

Por fim, um sistema de visão industrial deve ser rápido e compacto. Desta forma, existe uma demanda crescente por sistemas de IVA embarcados com *hardware* específico para processamento de imagens.

3 *Trabalhos relacionados*

Neste capítulo serão analisados trabalhos relacionados com a IVA dimensional em tempo real para detecção de irregularidades em bordas de objetos. O foco será explorar técnicas, contidas em trabalhos no "estado da arte", para as etapas de aquisição, pré-processamento e segmentação da imagem, e que possam ser adaptadas para a inspeção de bordas em objetos no primeiro plano da imagem.

3.1 Técnicas de aquisição da imagem visando segmentação dos objetos no primeiro plano e identificação de suas bordas

Nesta seção são analisadas seis técnicas de aquisição que reduzem o esforço para identificação dos objetos no primeiro plano, bem como de suas fronteiras, as quais denominamos "bordas externas".

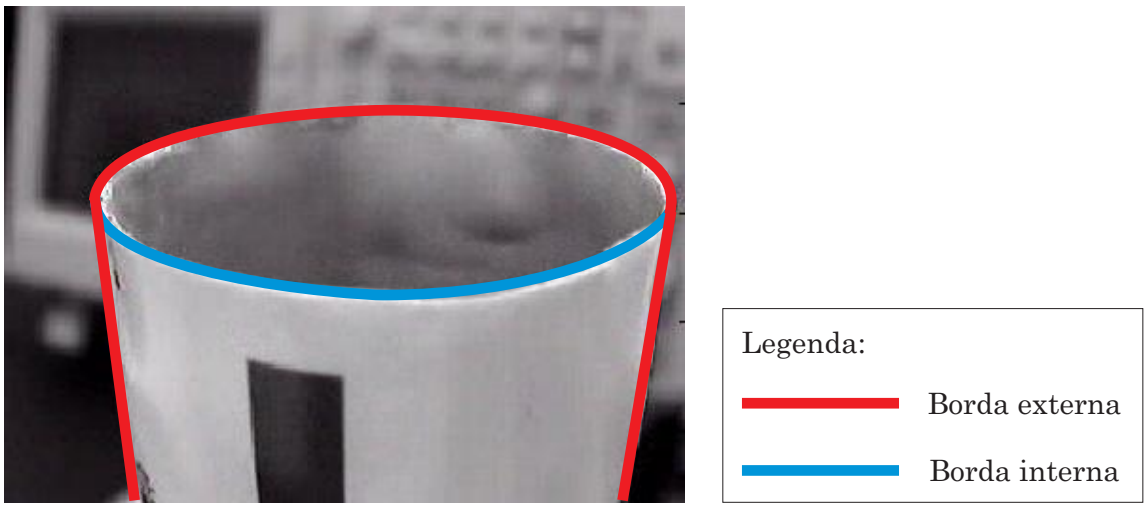


Figura 3: Bordas externas e interna de um tubo de creme dental.

Além das fronteiras dos objetos na imagem adquirida, também foi observada a capacidade de inspecionar as demais bordas dos objetos, inclusive àquelas que não estão na

fronteira, as quais denominamos “bordas internas”. As bordas externas e interna em um tubo de creme dental são apresentadas na Fig. 3.

Além da capacidade para detectar todas as bordas dos objetos no primeiro plano da imagem, também são observadas nos trabalhos: (i) desempenho e (ii) uso de recursos computacionais.

3.1.1 Extração do primeiro plano a partir de pares de imagens, com e sem iluminação específica no objeto

Esta técnica desempenha a segmentação da imagem, identificando os objetos no primeiro plano, a partir de uma única câmera. Para tal, foi observado que os objetos do primeiro plano ficam significativamente mais iluminados que o plano de fundo quando um *flash* é ativado, considerando que o primeiro plano está próximo da fonte de luz e o plano de fundo, distante. Deste modo, ao adquirir e comparar duas imagens de uma cena, com e sem *flash*, é possível identificar os objetos presentes no primeiro plano.

Esta técnica foi relatada em uma patente no ano de 1994 por Bolle et al.(30) para aplicações industriais e desperta interesse em novas pesquisas devido a sua arquitetura simples e elevado desempenho na segmentação da imagem. Por exemplo, no trabalho de Sun et al.(31) publicado em 2006 e em (32), publicado em 2008 com o mesmo autor principal, esta técnica foi aprimorada para utilização com câmeras fotográficas comerciais.

No trabalho apresentado em (32), além das mudanças de intensidade de luz produzidas pelo *flash*, o sistema também considera informações de movimento na seqüência de imagens e de cores nos pixels. Com isto é possível segmentar a imagem mesmo se houver pequenos deslocamentos na câmera ou movimento no plano de fundo entre as duas imagens adquiridas seqüencialmente, tornando a segmentação de imagens viável para implementação em qualquer câmera digital com *flash*. Para validar o trabalho, os autores realizaram uma série de testes modificando o plano de fundo original por uma imagem qualquer. Um dos testes pode ser visto na Fig. 4.

Apesar da eficiência desta técnica para segmentação de imagens, esta pode ser inviável para inspeções de bordas que exijam elevados desempenhos uma vez que é necessário adquirir duas imagens para cada inspeção. É importante ressaltar que a etapa da aquisição de imagem geralmente impacta significativamente no tempo total do processamento da imagem. Além disto, é necessário armazenar duas imagens para cada inspeção, resultando em uso intenso de memória.

(a) Cena original sem *flash*

(b) Cena original com *flash*

(c) Substituição do plano de fundo na cena (a)

(d) Substituição do plano de fundo na cena (b)

Figura 4: Imagens adquiridas seqüencialmente sem (a) e com (b) *flash*. Resultado da aplicação da técnica (c,d) posicionando o primeiro plano da cena original em novos planos de fundo

3.1.2 Iluminação com luz infra-vermelha para identificação do primeiro plano

No trabalho descrito por Wu, Boulanger e Bischof(33), publicado em 2008, foi utilizada uma técnica de segmentação similar a apresentada na seção 3.1.1. Porém, são utilizadas duas câmeras. A primeira é utilizada para adquirir a imagem colorida (RGB). A segunda é utilizada apenas para segmentar a imagem, capturando somente a luz infra-vermelha (850 nm $\pm$ 25 nm) proveniente da cena. A arquitetura do sistema de iluminação é apresentada na Fig. 5.

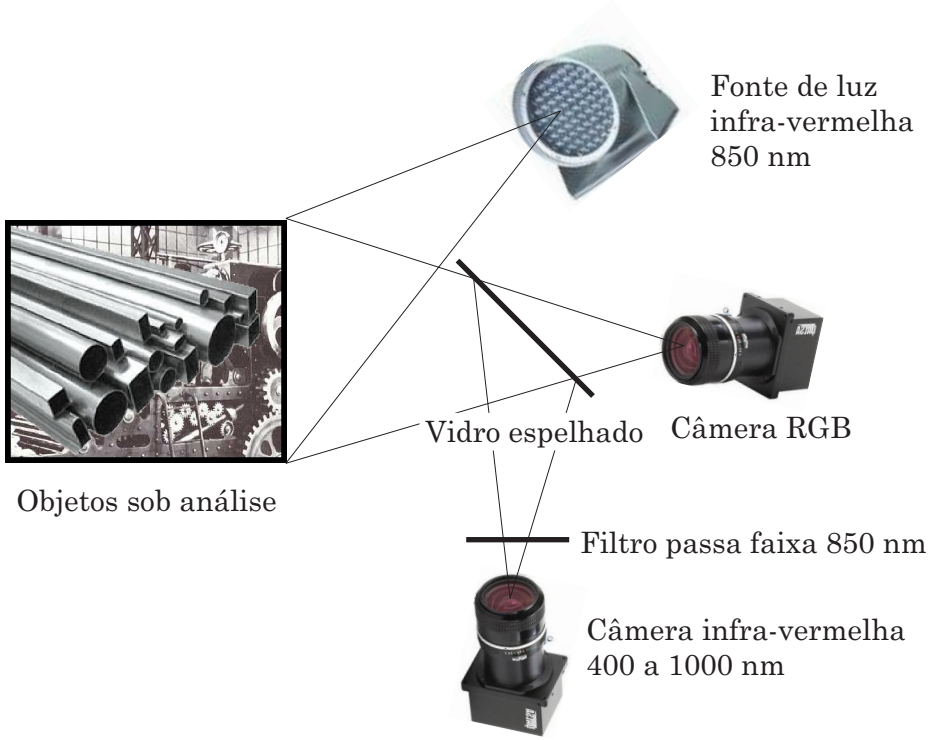


Figura 5: Arquitetura do sistema de aquisição das imagens

O diferencial desta técnica é a câmera infra-vermelha que captura com intensidade

significantemente maior os objetos do primeiro plano, sem necessidade de nenhum processamento adicional, devido uma fonte de luz infra-vermelha (850 nm) de alta intensidade que ilumina os objetos do primeiro plano. Observar na Fig. 6(b) que o plano de fundo não foi captado pela câmera infra-vermelha, porém os limites (bordas externas) do objeto do primeiro plano não estão bem definidos na imagem.

(a) Imagem colorida (b) Imagem infra-vermelha (c) Substituição do plano de fundo

Figura 6: Um exemplo de segmentação. Em (a) e (b) as imagens adquiridas. Em (c) o resultado da segmentação da imagem com substituição do plano de fundo original

Para chegar ao resultado apresentado em Fig. 6(c) é necessário um algoritmo de segmentação adicional que irá definir com precisão os limites do objeto no primeiro plano e suavizar estas bordas externas, visando uma melhor aparência do objeto quando sobreposto a um novo plano de fundo.

O mesmo autor principal publicou também o trabalho (34), em 2008. Neste, é apresentado que a mesma técnica pode ser aplicada em um sistema com iluminação infra-vermelha exclusivamente no plano de fundo, conforme apresentado na Fig. 7. Neste caso, os limites do primeiro plano podem ser detectados com muito mais nitidez do que apresentado no caso anterior.

(a) Imagem colorida (b) Imagem infra-vermelha

Figura 7: Imagens adquiridas com iluminação infra-vermelha no plano de fundo.

Visando a IVA em bordas de objetos em ambiente industrial, as técnicas de iluminação apresentadas nos trabalhos (33, 34) podem ser muito úteis devido ao seu elevado desem-

penho na detecção das bordas externas. Contudo, estas técnicas não são adequadas para identificação de bordas internas, e sim, somente para a identificação dos limites extremos do objeto no primeiro plano. Além disto, é necessário um processamento adicional para realizar a segmentação da imagem. Este processamento, no MATLAB, durou 100ms em um computador com processador Pentium 2 GHz e 1 GB de memória RAM.

3.1.3 Utilização de câmeras com registro de profundidade

No trabalho apresentado por Wang et al.(35), a segmentação do primeiro plano é realizada através de uma câmera que, além dos componentes RGB, informa também as distâncias entre os objetos e a câmera. O dispositivo utilizado foi a ZCam, da 3DV (Fig. 8(b)). O resultado da técnica é semelhante aos observados nas seções 3.1.1 e 3.1.2 e pode ser observado na Fig. 8.

(a) Cena original (b) ZCam (c) Informações de distância (d) Resultado da segmentação

Figura 8: Utilização da informação de profundidade para segmentação do primeiro plano.

Estas câmeras possuem um obturador especial, em frente ao sensor de imagem, que abre e fecha na mesma velocidade em que um pulso de luz é ativado. Devido ao tempo de viagem da luz, parte desta luz é bloqueada ao retornar (ao encontrar o obturador já fechado). Logo, a quantidade de luz do pulso sentida pelo sensor será proporcional à distância entre a câmera e o objeto sob inspeção. Além da ZCam, da 3DV, a Canesta também comercializa uma câmera com esta tecnologia, chamada de CanestaVision. A Microsoft comprou as duas empresas em 2009 e 2010, respectivamente.

Esta câmera se mostra uma boa opção para detecção de irregularidades em bordas. Como desvantagem, este sistema apresentou um baixo desempenho, 15 quadros por segundo, o que pode inviabilizar sua utilização para IVA. Além disto, a comercialização destas câmeras ainda está restrita.

3.1.4 Detecção do primeiro plano em imagens com plano de fundo estático

Os trabalhos apresentados nas seções 3.1.1, 3.1.2 e 3.1.3 são boas opções para extração do plano de fundo, mesmo quando este é dinâmico. Porém, quando o plano de fundo é estático, é possível obter uma implementação mais simples, utilizando menos recursos computacionais.

No trabalho elaborado por Casares, Velipasalar e Pinto(36) e publicado em 2010, por exemplo, um algoritmo eficiente para extração do plano de fundo foi desenvolvido visando a implementação em sistema embarcado. Nesta técnica, um modelo do plano de fundo é continuamente atualizado na memória do sistema, visando adaptar-se às lentas mudanças ambientais.

Em seguida, com o modelo de plano de fundo já construído, qualquer objeto estranho na imagem será identificado.

Visando a IVA dimensional com detecção de bordas, assim como no trabalho apresentado na seção 3.1.2, este também não identifica as bordas internas da imagem, e sim, somente identifica os limites extremos do objeto no primeiro plano. Logo, não é adequado para detecção de imperfeições em bordas de objetos.

3.1.5 Utilização de iluminação a *laser*

O trabalho descrito por Liao et al.(9), publicado em 2008, descreve a IVA em cones de alto-falantes. Este inspeciona a profundidade do cone baseado em iluminação estruturada com *laser*. Triangulando a câmera, a fonte de luz (*laser*) e o objeto (cone), conforme geometria apresentada na Fig. 9, é possível identificar através da trigonometria a distância entre o objeto e a câmera.

O cone é varrido linha por linha pelo *laser* e a imagem desta varredura é captada pela câmera, que está localizada no topo do sistema de IVA. Neste caso, o *laser* projeta a luz com um grau de liberdade, enquanto o cone é deslocado, através de um suporte mecânico móvel, na direção perpendicular a projeção do *laser*. Para cada linha varrida pelo *laser* (Fig. 10(a)), é gerada uma assinatura que descreve um perfil de profundidade do cone (Fig. 10(b)).

O foco do trabalho proposto em (9) é a precisão [1] do sistema. O desvio padrão obtido

1. Grau de variação dos resultados de uma medição.

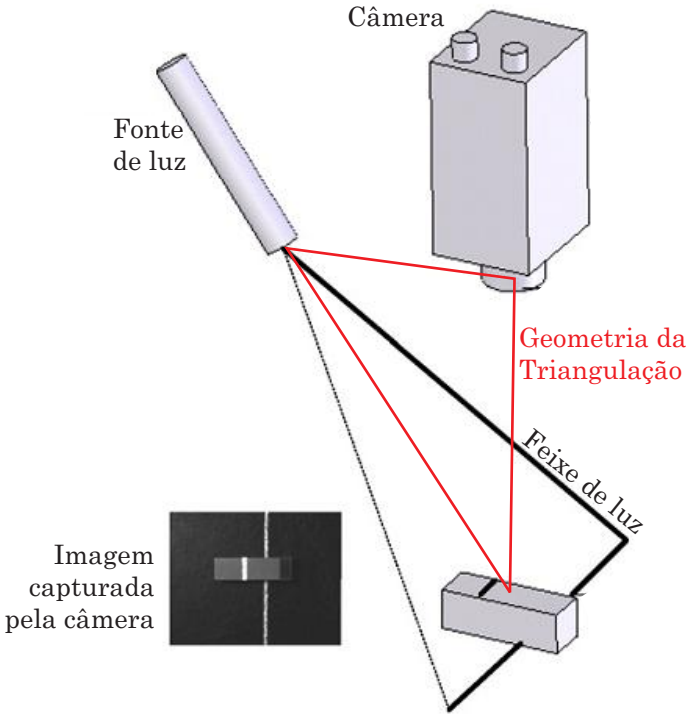
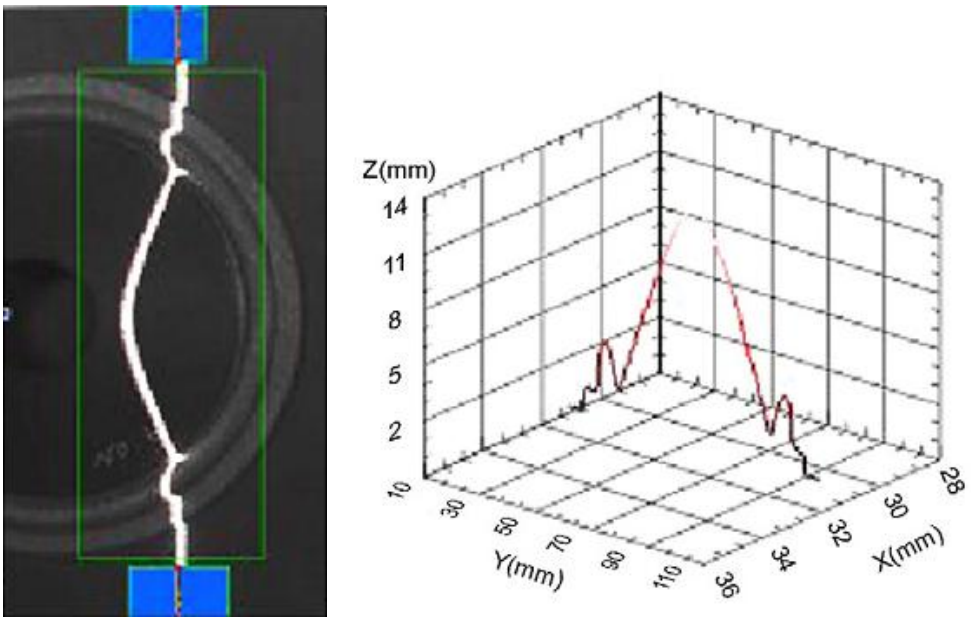


Figura 9: Geometria da triangulação.



(a) Varredura a *laser* (b) Assinatura gerada

Figura 10: Geração de uma assinatura.

no trabalho foi de 8 μm, tendo então uma excelente precisão para inspeção de bordas. Porém, a aplicação desta técnica é inviável em IVA que demande resposta rápida. O tempo total da inspeção do cone neste trabalho é de 3s. Este tempo é melhor que os 8s obtidos na inspeção manual de cones, porém excessivo para aplicações de alto desempenho.

3.1.6 Inspeção de objetos baseado em iluminação estruturada a partir de projetores de padrão

Em uma técnica semelhante à apresentada na seção 3.1.5, seguindo o mesmo princípio da triangulação entre a câmera, fonte de luz e objeto sob inspeção, é possível substituir o *laser* por um projetor de padrões. A deformação nas linhas projetadas sobre os objetos sob inspeção contém informações de profundidade da imagem.

Alguns trabalhos que utilizam esta técnica são o de o de Wang et al.(13) e Chang e Park(12), ambos publicados em 2009. Em (12), por exemplo, foi construído um aparato específico para adquirir imagens de moldes dentários. Neste caso, a câmera e o projetor são fixos enquanto o objeto sob análise rotaciona com dois graus de liberdade, possibilitando a modelagem em 3D da peça de gesso.

Esta técnica, utilizando projetores, tem melhor desempenho em relação a iluminação com *laser* (seção 3.1.5) uma vez que todas as linhas são projetadas simultaneamente. No trabalho (13), o tempo total para a medição de uma cena foi de 0,5s. Apesar de apresentar melhor desempenho, ainda é insuficiente para diversas aplicações de IVA dimensional. Além disto, esta técnica é de difícil implementação embarcada e utiliza uma fonte de luz estruturada de alto custo e delicada (o projetor de padrões), não sendo adequada para algumas aplicações industriais.

3.2 Técnicas para segmentação da imagem a partir da descontinuidade

Todas as técnicas apresentadas anteriormente neste capítulo visam ressaltar, na imagem adquirida, a região aonde estão os objetos do primeiro plano. Porém, a segmentação efetiva somente é realizada em uma etapa posterior.

A abordagem é particionar a imagem baseando-se nas mudanças bruscas nos níveis de cinza. As principais áreas de interesse nessa categoria são, em ordem crescente de complexidade, a detecção de pontos isolados, de linhas e de bordas na imagem(37).

Um ponto isolado é detectado se, ao varrer a imagem, for detectado que o nível de cinza deste ponto é completamente diferente do nível de cinza de seus vizinhos. O próximo nível de complexidade envolve a detecção de linhas em uma imagem. De maneira semelhante a detecção de pontos isolados, uma linha é detectada se um conjunto de pixels alinhados tiverem o nível de cinza completamente diferente de seus vizinhos.

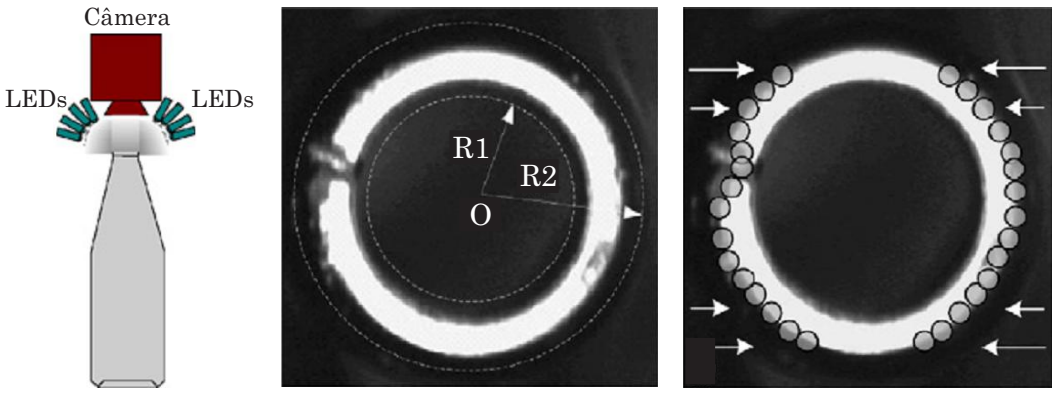
Porém, a detecção de bordas é, de longe, a abordagem mais comum para a detecção de descontinuidades significantes nos níveis de cinza(37). Existem diversos algoritmos para detecção de bordas, tais como operadores de gradiente, laplaciano e detecção combinada. Quando a etapa de aquisição disponibiliza uma imagem com bordas nítidas e com alto contraste, a utilização de um operador gradiente é a técnica mais simples e eficiente para detecção das bordas. Constantemente utilizada pelas indústrias, esta técnica consiste basicamente em identificar as variações bruscas de intensidade na imagem. A seguir é apresentado um trabalho relevante que utiliza esta técnica.

3.2.1 Detecção da borda a partir do gradiente da imagem

No trabalho descrito por Duan et al.(4) e publicado em 2007, é apresentado um sistema de visão de máquina que inspeciona o acabamento final da boca da garrafa. Neste trabalho é utilizada uma rede neural em dois níveis para identificar quebras ou rachaduras no vidro. Porém, antes é necessário determinar a área de inspeção na imagem. A segmentação é realizada a partir de parâmetros informados manualmente, R1 e R2 apresentados na Fig. 11(b), somados a um parâmetro identificado automaticamente pelo sistema de inspeção: o centro da boca da garrafa, representado pelo ponto “O” na Fig. 11(b).

O algoritmo para detecção do centro da garrafa deve ser preciso, rápido e capaz de filtrar grandes variações na imagem, uma vez que a imagem da garrafa, adquirida para aplicação em tempo real, contém muitas incertezas e pode ser distorcida em uma grande área. A transformada de Hough convencional foi descartada pelo autor por ser muito lenta e não ser útil para aplicações de alto desempenho. Já o algoritmo que utiliza o centro de gravidade da imagem pode produzir grandes erros quando a imagem tem uma região distorcida.

A solução desenvolvida pelo autor consiste em detectar as bordas externas da boca da garrafa através de um algoritmo simples e rápido, baseado na variação da intensidade dos pixels na imagem e, em seguida, identificar o ponto médio entre esses pontos. A primeira etapa, detecção das bordas, é realizada ao varrer a imagem, linha por linha. Quando houver uma brusca variação da intensidade, a borda é identificada. Só é necessário armazenar os dados da primeira borda detectada da esquerda para a direita e da primeira detectada da direita para a esquerda, conforme Fig. 11(c). A partir destes pontos, pode-se facilmente calcular o ponto médio, que representará o centro da boca da garrafa. Esta técnica se mostrou muito eficiente e eficaz para aplicações de IVA.



(a) Posicionamento da câmera, LEDs e garrafa

(b) Área de inspeção

(c) Detecção de bordas

Figura 11: Aquisição da imagem, área de inspeção e detecção das bordas da boca da garrafa.

A iluminação foi realizada com LEDs devido à qualidade da luz produzida (uniformidade e espectro) e ao longo tempo de vida útil destes dispositivos (superior a 50.000 horas) e a aquisição da imagem foi realizada a partir de uma câmera industrial CCD. O posicionamento dos LEDs, garrafa e câmera pode ser observado na Fig. 11(a). Todos os acabamentos com quebras foram inspecionados corretamente.

Apesar da eficiente técnica de segmentação da imagem, as demais etapas desta IVA são complexas, principalmente a etapa de reconhecimento, que é realizada através de duas redes neurais. Este processamento é realizado em um computador industrial com processador Pentium IV 2.4 GHz, exigindo adaptações na infraestrutura pré-existente quando a linha de produção estiver com espaço restrito.

A taxa de inspeção, de 56ms por inspeção, foi melhor que a taxa obtida pelas demais técnicas já apresentadas e adequada para a linha de produção de garrafas. Porém, esta taxa ainda é insuficiente para a aplicação de detecção de amassados em tubos de creme dental, por exemplo.

3.3 Implementação da IVA em sistemas embarcados

Na detecção de parafusos em ferrovias, proposto por Marino et al.(3) e publicado em 2007, foi utilizado um sistema embarcado baseado em FPGA e rede neural. O sistema de IVA completo é instalado no trem, visando inspecionar a ferrovia em tempo-real. O tempo de processamento de uma janela de 24 x 100 pixels, tamanho necessário para inspecionar parafusos, é de 8,09μs.

O sistema de IVA adquire a imagem da ferrovia através de uma câmera de alta velocidade, CCD de varredura com duas linhas. A janela de 24 x 100 pixels é adquirida em 15 ms. Logo, é possível observar que o tempo total da inspeção é limitado pelo tempo da aquisição da imagem. Quanto à precisão, este trabalho tem taxa de acerto entre 95% e 99,6%, dependendo do tempo da inspeção realizada.

Já no trabalho de Vega-Rodríguez, Sánchez-Pérez e Gómez-Pulido(19), inspeção de rolhas, foram desenvolvidos 16 módulos de processamento, executados em *pipeline*, e implementados em FPGA e em um computador pessoal com um Pentium-II 350 MHz e 64 Mb de memória RAM. O resultado indicou que a implementação em *hardware* processou a imagem nove vezes mais rápida que a implementação em *software*.

Por fim, no trabalho realizado por Fernandez et al.(7), é realizada a inspeção de gomos de laranjas. Neste caso, o processamento no FPGA é realizado em 22ms/gomo, enquanto o processamento em um Pentium-III foi de 64ms/gomo.

O que foi observado é que a implementação do processamento de nível intermediário e de alto nível em FPGA tende a reduzir o tempo total de processamento da IVA.

3.4 Análise do capítulo

Quanto as técnicas de aquisição da imagem visando segmentação dos objetos no primeiro plano e identificação de suas bordas, foi observado que os algoritmos mais simples e com menor complexidade de *hardware* somente detectam as bordas mais externas do objeto, seja utilizando uma vigorosa iluminação no objeto ou no plano de fundo (seções 3.1.1 e 3.1.2), seja criando um modelo estático para o plano de fundo (seção 3.1.4).

Outros trabalhos, baseados em câmera de profundidade (seção 3.1.3), *laser* (seção 3.1.5) e projetores de padrão (seção 3.1.6) possibilitam a detecção das bordas internas de objetos com uma boa precisão, exatamente conforme desejado. Porém, o tempo total de processamento, o uso intenso de recursos computacionais e o uso de sistemas de aquisição de imagens complexos destas técnicas podem inviabilizar seus usos na IVA de bordas, embarcados e em tempo real. A tabela 1 compara as técnicas analisadas.

É necessário também executar um algoritmo de detecção de bordas. Foi observado que, para aplicações industriais, com objetos de bordas nítidas (ressaltadas com a iluminação e nas etapas de aquisição e pré-processamento), a utilização do gradiente se mostrou muito eficiente e eficaz (seção 3.2.1).

Tabela 1: Comparação entre os trabalhos relacionados com foco nas técnicas de aquisição da imagem.

Técnica	Observações	Tempo da inspeção	Detecta todas as bordas	Embarcado / Uso de recursos computacionais
Extração do primeiro plano a partir de pares de imagens, com e sem iluminação específica no objeto	Muito eficiente para segmentação de imagens	N/D (1)	Não	Não (2) / Moderado
Iluminação com luz infra-vermelha para identificação do primeiro plano	Desempenho elevado na segmentação	100 ms	Não	Não / Baixo
Utilização de câmeras com registro de profundidade	Utiliza câmeras específicas, de comercialização restrita	66 ms	Sim	Não / Baixo
Detecção do primeiro plano em imagens com plano de fundo estático	Algoritmo simples	N/D	Não	Não (2) / Moderado
Utilização de iluminação a laser	Elevada precisão; Iluminação complexa	3000 ms	Sim	Não / Moderado
Inspeção de objetos baseado em iluminação estruturada a partir de projetores de padrão	Elevada precisão; Iluminação complexa	500 ms	Sim	Não / Elevado
N/D - Não Definido (1) Necessita adquirir duas imagens (2) Técnica desenvolvida para uso em sistemas embarcados				

Adicionalmente, alguns trabalhos analisados contribuiram também com boas técnicas para as etapas de reconhecimento e interpretação da imagem, visando a aplicação de inspeção de tubos. Por exemplo, o trabalho apresentado na seção 3.1.5, inspeção de cones de alto-falantes, extrai parâmetros da imagem, linha por linha, na medida em que a imagem é adquirida, gerando um conjunto de assinaturas. Este conjunto pode ser então simplesmente comparado com um banco de assinaturas de objetos bons (conformes com as especificações) previamente construído.

Quanto à arquitetura, a implementação do processamento de nível intermediário e de alto nível em FPGA reduziu o tempo total de processamento da IVA (seção 3.3) quando comparado à implementação em processador de uso geral. Além disto, somente com a solução embarcada foi possível construir alguns sistemas, como o de inspeção em ferrovias.

A intensa utilização da memória foi comum entre os trabalhos analisados, uma vez que todos armazenam as imagens adquiridas para executar o processamento. Com isto, eleva-se o custo da implementação destas soluções em sistemas embarcados, além de aumentar o tempo de processamento devido aos repetitivos ciclos de leitura e escrita na memória.

Por fim, nenhum dos trabalhos analisados atende a todos os requisitos para a aplicação de IVA dimensional com inspeção de irregularidades em bordas e alto desempenho para

implementação em sistemas embarcados de tempo-real.

4 *LEDges: uma técnica de IVA baseada em limiarização da imagem e geração de assinaturas*

No capítulo anterior foi mostrado que nenhum dos trabalhos analisados atende a todos os requisitos para a aplicação de IVA dimensional com inspeção de irregularidades em bordas e alto desempenho para implementação em sistemas embarcados de tempo-real. Visando atender a estes requisitos, neste capítulo será apresentada uma nova técnica para IVA, intitulada "LEDges".

A principal inovação desta técnica é um novo modelo de iluminação estruturada que permite a fácil segmentação da imagem adquirida através de uma simples limiarização. Esta inovação é baseada na simplicidade do algoritmo e baixa complexidade de *hardware* apresentados em (30–34), porém com o aperfeiçoamento de detectar todas as bordas do objeto nas imagens adquiridas pela câmera, e não somente as bordas mais externas. Antes, esta característica só foi possível nos trabalhos baseados em câmeras de profundidade, *laser* e projetores de padrões (9, 12, 13, 35), que têm elevado tempo total de processamento, uso intenso de recursos computacionais e o de dispositivos de captura de imagens delicados e complexos, inviabilizando seus usos em IVA embarcada e em tempo real.

Adicionalmente, a imagem resultante da etapa de limiarização viabiliza o uso de técnicas extremamente simples para implementação das etapas de descrição e reconhecimento da imagem. Logo, a LEDges é capaz de detectar todas as bordas do objeto com alto desempenho através de um sistema embarcado e com uso reduzido de recursos computacionais.

O processamento da LEDges pode ser analisado em quatro etapas, conforme pode ser observado na Fig. 12: aquisição da imagem, limiarização (segmentação), geração da assinatura (descrição) e comparação (reconhecimento).

Inicialmente a imagem do objeto é adquirida em cooperação com o sistema de iluminação. O resultado desta etapa é uma imagem contendo as áreas iluminadas dos objetos em primeiro plano em altas intensidades, enquanto o plano de fundo e as sombras são exibidos em baixas intensidades.

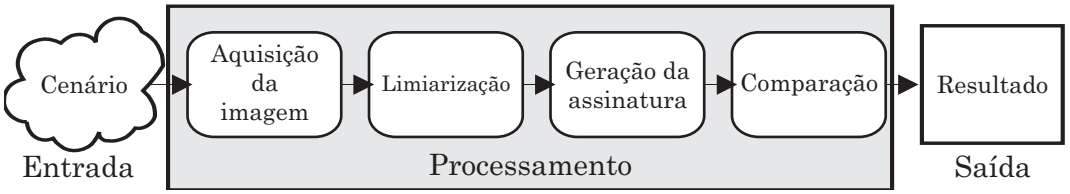


Figura 12: Etapas do processamento da LEDges.

Em seguida, na etapa de limiarização, a imagem adquirida é facilmente segmentada em dois níveis de intensidade: (I) o plano de fundo e as sombras são indicados através de regiões com intensidade zero, cor preta; (II) e áreas iluminadas nos objetos do primeiro plano são indicadas através da intensidade máxima, cor branca.

Na terceira etapa, geração da assinatura, a imagem limiarizada é representada em termos de suas bordas e descrita como assinatura, assim como realizado eficientemente em (9). Cada borda é descrita numericamente por um conjunto de distâncias medidas entre o início das linhas do quadro de imagem e as transições relevantes para a detecção da irregularidade no objeto sob inspeção, seja esta transição do branco para o preto, seja do preto para o branco. Observar que, de um modo geral, a transformada de Hough (28, 38) não é utilizada para a detecção das linhas, círculos e elipses em aplicações que exigem elevados desempenhos.

Por fim, a última etapa da LEDges é a comparação. Nesta etapa é realizado o reconhecimento da assinatura gerada através de simples comparação entre esta e a assinatura de um objeto padrão "bom". O resultado desta etapa é um sinal binário indicando se o objeto sob inspeção está de acordo com as especificações da indústria, ou não.

As seções 4.1, 4.2, 4.3 e 4.4 detalham estas quatro etapas do processamento da LEDges.

Apesar da simplicidade, a técnica proposta foi estruturada para possibilitar a sua implementação em diversas arquiteturas embarcadas, satisfazendo a demanda crescente por desempenho (tempo de resposta e precisão) e pelo uso mínimo de recursos disponíveis (financeiro, humano, físico e energético) (39). Por exemplo, a técnica pode adquirir a imagem através de uma câmera analógica ou digital, de alta ou baixa resolução e em diversas taxas de amostragem. Além disto, é possível implementar as demais etapas do processamento em diversos dispositivos: desde um microcontrolador (μc) até um circuito

integrado de aplicação específica (ASIC), a depender dos requisitos de desempenho da aplicação.

A LEDges possibilita a execução paralela das quatro etapas em um modelo de *pipeline*[1]. A Fig. 13 apresenta o *pipeline* a nível de *pixel*. Observar que a base de tempo é o *"pixel clock"*, que é a temporização gerada pela câmera, fornecendo um pulso a cada pixel adquirido.

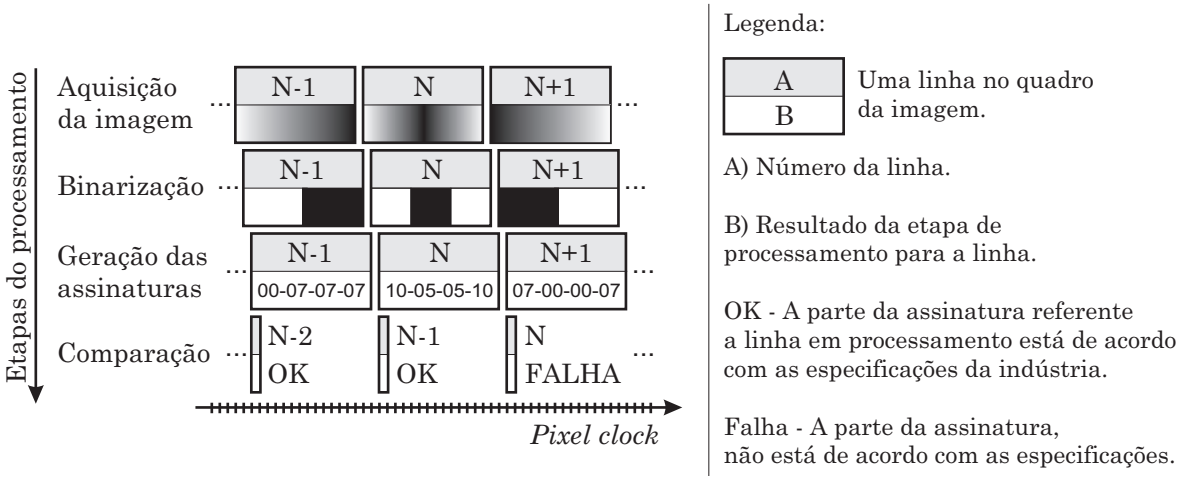


Figura 13: Fluxo de informações entre as etapas do processamento da LEDges.

Por fim, conforme a Fig. 13, é possível realizar toda a IVA sem armazenar o quadro inteiro da imagem adquirida, diferenciando dos demais sistemas de visão de máquina analisados no capítulo 3 que utilizam intensamente a memória. Logo, com a redução no uso de memória é possível reduzir o custo da implementação da solução e o tempo total de processamento devido aos reduzidos ciclos de leitura e escrita na memória.

4.1 Aquisição da imagem

Na etapa de aquisição, a imagem da região de interesse do objeto sob inspeção é capturada. A principal inovação está no uso de um *LED* de alta potência para a iluminação desta região de interesse. O resultado desta etapa é uma imagem contendo as áreas iluminadas em primeiro plano em altas intensidades, enquanto o plano de fundo e as sombras são exibidos em baixas intensidades.

Para obter o resultado desejado, o *LED*, a câmera e o objeto sob inspeção são posi-

1. Em computação, o *pipeline* é um conjunto de etapas de processamento conectados em série, tal que a saída de uma etapa é a entrada da próxima. As etapas de um *pipeline* são geralmente executadas em paralelo.

cionados em uma geometria específica, tal que a intensidade da luz emitida pelo *LED* é significantemente maior sobre a região de interesse no primeiro plano do que no plano de fundo, conforme pode ser observado na região em degradê da Fig. 14, aonde a área mais escura do degradê representa maior intensidade de luz. De fato, em casos onde a fonte de luz pode ser considerada pontual [2], a irradiância [3] é reduzida com o quadrado inverso da distância (40).

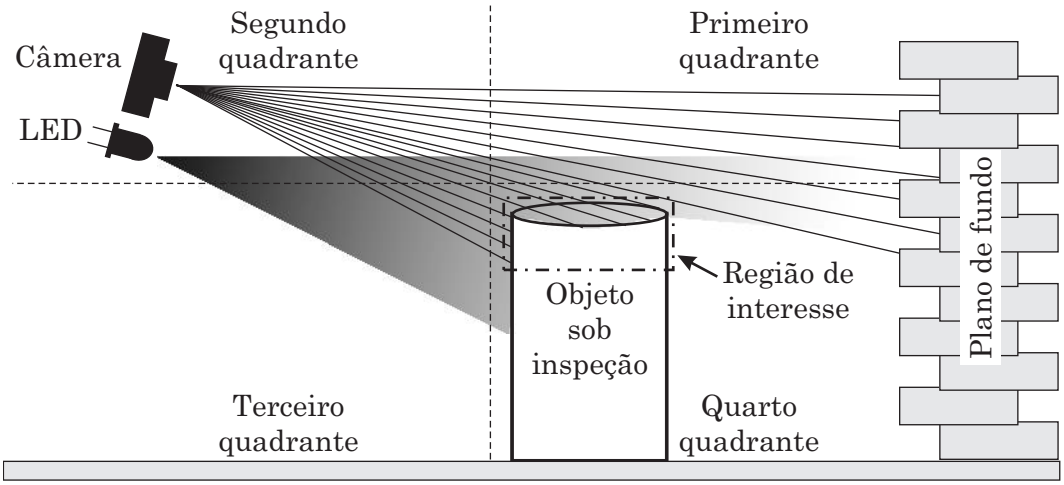


Figura 14: Posicionamento da câmera, *LED*, objeto sob inspeção e cenário.

A respeito do posicionamento relativo do *LED*, este deve estar fixado próximo à câmera. O grupo óptico, formado pelo *LED* e pela câmera, deve ser posicionado no quadrante oposto ao da região de interesse do objeto sob inspeção, de tal forma que a câmera capture uma imagem contendo todas as bordas da região de interesse iluminadas pelo *LED*. Além disso, é necessário que o posicionamento do *LED* crie, na imagem capturada, sombras entre as bordas na região de interesse.

Por exemplo, na Fig. 14 a região de interesse está posicionada no quarto quadrante e o grupo óptico no segundo. Quanto à formação das sombras, as bordas de interesse estão na face superior do objeto, logo, o *LED* deve estar fixado imediatamente abaixo da câmera. Neste posicionamento o *LED* ilumina todas as bordas e cria sombras entre as bordas na imagem capturada.

Esta técnica só é viável com a utilização do *LED* como emissor de luz. Devido a suas dimensões "pontuais", o *LED* é capaz de criar sombras nítidas nas cavidades dos objetos. Outros benefícios da utilização do *LED* são: (I) Longo tempo de vida útil. Este pode ser mais de 50.000 horas, aproximadamente 20x maior que de uma lâmpada incandescente;

2. Tem dimensões muito menores que a distância entre esta fonte e o objeto iluminado.
3. Razão entre potência luminosa útil e área.

(II) O *LED* possui elevada eficiência energética, principalmente em aplicações onde a luz monocromática é utilizada; (III) Os *LEDs* são robustos, inerentemente resistentes a vibrações, e podem permanecer na maioria das condições ambientais extremas.

Três parâmetros são fundamentais para a escolha do *LED* que será utilizado: a potência, o ângulo de visão [4] e o espectro de emissão. A potência é definida de acordo com a iluminação pré-existente no ambiente da inspeção. Quanto mais iluminado, mais potente deverá ser o *LED*. Resultados experimentais mostram que a utilização de *LEDs* de 1 W é o suficiente para a maioria das aplicações. Quanto ao ângulo de abertura, este deve ser o menor possível, suficiente apenas para iluminar a área de interesse. A depender da distância (d) e do ângulo de abertura (a), o *LED* ilumina uma área com diâmetro (D) de acordo com a fórmula a seguir:

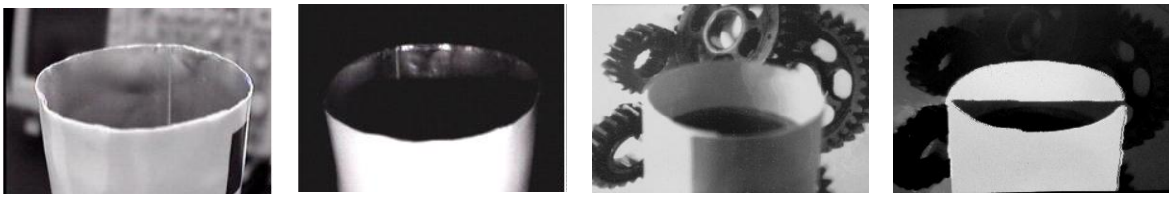
$$D = 2.d.tg\left(\frac{a}{2}\right) \tag{4.1}$$

Quanto ao espectro, o *LED* deve emitir luz no mesmo comprimento de onda que a câmera é sensível na captura da imagem. Além do espectro de sensibilidade da câmera, o tempo de exposição e o ajuste de brilho e contraste também são importantes na configuração da captura da imagem. Visando ressaltar a aquisição das bordas de interesse no primeiro plano em tons claros e todos os objetos no plano de fundo e as sombras em tons escuros, câmeras digitais permitem configuração para ajustar o tempo de exposição. Quanto mais rápida for a exposição, menos luz será captada pelo sensor. Logo, para compensar a curta janela de tempo para entrada de luz, é necessário iluminar vigorosamente o objeto sob inspeção. Já o plano de fundo, que não é iluminado fortemente, não terá sua imagem capturada com alta intensidade.

Para as câmeras analógicas, a ênfase às áreas iluminadas no primeiro plano é realizada através de um pré-processamento implementado no *hardware* interno da câmera, com o ajuste analógico automático de brilho e contraste. Esta funcionalidade foi identificada em todos os modelos pesquisados de câmeras comerciais.

Na Fig. 15 são apresentados alguns resultados para esta etapa. As imagens foram adquiridas através de câmeras analógicas e digitais (Fig. 15(a) e 15(c)). Nos dois casos é possível observar claramente que as áreas referentes ao plano de fundo e às cavidades têm menores intensidades enquanto as regiões das bordas são ressaltadas (Fig. 15(b) e 15(d)).

4. Metade do ângulo de abertura de um cone imaginário que concentra 70% de toda a energia óptica emitida pelo *LED*.



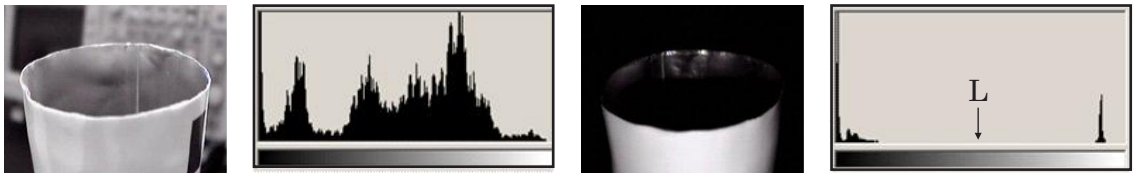
(a) Câmera analógica, *LED* desligado. (b) Câmera analógica, *LED* ligado. (c) Câmera digital, *LED* desligado. (d) Câmera digital, *LED* ligado.

Figura 15: Imagens adquiridas.

4.2 Limiarização

Na etapa de limiarização, que é uma das mais importantes abordagens para a segmentação de imagens (37), a imagem adquirida é segmentada.

Seja um histograma de nível de cinza, conforme a Fig. 16(d), correspondente a uma imagem composta por objetos iluminados sobre um fundo com luz em menor intensidade, de modo que os pixels (I) das áreas iluminadas nos objetos e (II) do plano de fundo e das sombras tenham seus níveis de cinza em dois grupos dominantes, é possível extrair as áreas iluminadas através de um limiar L que separe os dois grupos. Os pixels com intensidade maior que L pertencem ao objeto e são representados na imagem limiarizada na cor branca. Caso contrário, os pixels pertencem ao plano de fundo ou às sombras e serão representados na cor preta.



(a) Imagem adquirida com o *LED* desligado. (b) Histograma da imagem (a). (c) Imagem adquirida com o *LED* ligado. (d) Histograma da imagem (c).

Figura 16: Imagens adquiridas e seus histogramas.

As técnicas de iluminação, aquisição e pré-processamento da imagem têm um papel fundamental na limiarização. Na Fig. 16 são mostradas duas imagens: uma do objeto sem iluminação específica (Fig. 16(a)) e outra do objeto com iluminação específica e ajuste de contraste (Fig. 16(c)), conforme visto na seção 4.1. Observar que é impossível determinar dois grupos dominantes no histograma referente a imagem sem iluminação específica (Fig. 16(b)).

A limiarização mais simples é a do tipo global. Nesta, o limiar L é estabelecido manualmente observando os grupos dominantes no histograma e, a partir desta definição, a

segmentação é efetuada varrendo-se a imagem inteira, pixel por pixel, e rotulando cada pixel como sendo (I) da área iluminada ou (II) do plano de fundo ou das sombras, dependendo se o nível de cinza daquele pixel for maior ou menor que L.

O nível constante L utilizado como referência pode ser ajustado em, aproximadamente, o valor médio entre os picos pretos e brancos do histograma. No caso de um sinal digital de 10 bits, o nível será próximo a 512. Se for um sinal NTSC [5], o valor de L será próximo de 0,65 V.

Observar que, após o estabelecimento do parâmetro L, é possível realizar a limiarização sem armazenar a imagem original. Ou seja, esta etapa pode ser realizada em paralelo com a aquisição da imagem, limiarizando pixel a pixel, na medida em que estes estão sendo adquiridos, em *pipeline*. São necessários, por exemplo, 4,8 MB para armazenar uma imagem de 2592 x 1944 pixels [6], na escala de 256 tons de cinza. Nas demais técnicas estudadas, é necessário utilizar esta memória uma vez que todas armazenam a imagem adquirida. Porém, com a LEDges, não é necessário utilizar esta memória.

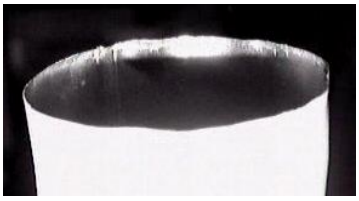
Se uma câmera analógica for utilizada, a limiarização poderá ser realizada a partir de um simples comparador de tensão analógico, onde o sinal analógico é comparado com o nível L. Ou seja, não é necessário utilizar um conversor analógico-digital. A partir da redução no uso de memória e conversores analógicos-digitais, é possível reduzir o custo da implementação da solução e o tempo total de processamento.

Na Fig. 17 é apresentada a imagem adquirida com seu histograma. A imagem está em 256 níveis de tons de cinza. Foram realizados três testes de limiarização para esta imagem. O primeiro resultado pode ser visto na Fig. 17(c) e foi obtido configurando o parâmetro L=64, apresentado como L1 na Fig. 17(b). O segundo resultado, apresentado na Fig 17(c), foi obtido a partir de L=128, representado por L2. Por fim, no terceiro teste escolhemos L=192, apresentado como L3, e obtemos o resultado apresentado na Fig. 17(d).

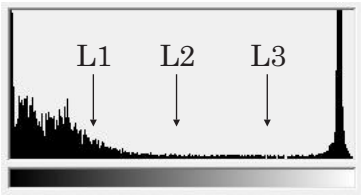
Logo, é observado que a escolha inadequada do limiar L pode impossibilitar a detecção de algumas bordas. Para o exemplo da Fig. 17, um limiar muito baixo prejudica a detecção das bordas inferiores do objeto. Enquanto que um limiar alto não permite a correta detecção das bordas superiores.

5. Neste padrão a tensão elétrica varia entre 0,3 V para a tonalidade mais escura, até 1 V para a tonalidade mais clara.

6. Resolução máxima da câmera D5M, amplamente associado ao kit didático FPGA DE2-70.



(a) Imagem adquirida.



(b) Histograma da imagem (a).

(c) Resultado para L=L1.

(d) Resultado para L=L2.

(e) Resultado para L=L3.

Figura 17: Entrada e saídas da etapa de limiarização.

4.3 Geração da assinatura

Na terceira etapa da LEDges, a imagem limiarizada é representada em termos de suas bordas e descrita como uma assinatura.

Cada borda é descrita numericamente por um conjunto de distâncias medidas entre o início das linhas do quadro de imagem e as transições relevantes para a detecção da irregularidade no objeto sob inspeção, seja esta transição do branco para o preto, seja do preto para o branco. Essas distâncias podem ser medidas através do tempo de varredura da câmera analógica ou do número de pixels nas digitais. Este conjunto de medidas compõe a assinatura do objeto sob inspeção.

Nem todas as transições na imagem limiarizada são relevantes para a detecção da irregularidade no objeto sob inspeção, principalmente por que nem todas as transições representam uma borda do objeto. Por exemplo, uma marca impressa na superfície do objeto não deve ter suas transições consideradas. Logo, é necessário que seja informado ao sistema de IVA quais bordas são relevantes. Visando facilitar essa atividade, a imagem do objeto é dividida em regiões de interesse e, para cada uma dessas regiões, devem ser definidas que transições são relevantes.

As regiões de interesse e as transições relevantes devem ser definidas pelo profissional que irá instalar o equipamento na indústria, uma vez que estes parâmetros variam de acordo com o objeto sob inspeção. Por exemplo, no caso de um tubo, a imagem foi dividida em três regiões de interesse: borda superior, inferior e corpo do tubo, conforme Fig. 18. O termo “superior” se refere à posição desta região de interesse em relação à

imagem bidimensional adquirida. O mesmo conceito é válido para o termo “inferior”.

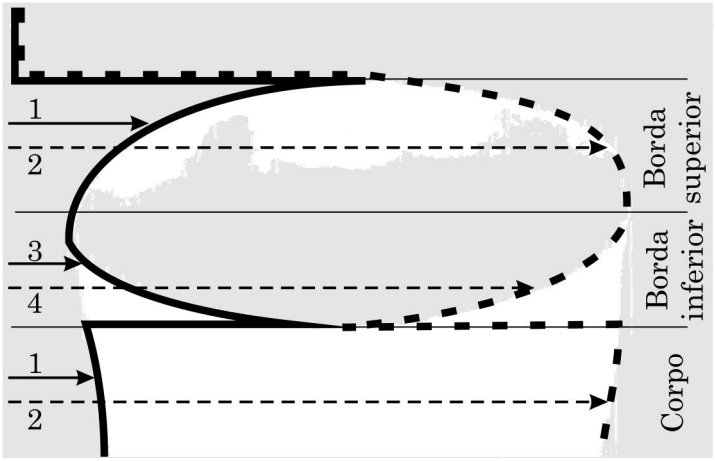


Figura 18: Transições relevantes da imagem e que formarão a assinatura do tubo.

Considerando que a câmera varre a imagem da esquerda para a direita e de cima para baixo, a borda superior foi definida como as primeiras linhas adquiridas após a identificação do primeiro pixel branco. Já a borda inferior foi definida pelas últimas linhas adquiridas enquanto houver mais de duas transições detectadas para cada linha. Por fim, o corpo foi definido como as primeiras linhas da imagem imediatamente após a região da borda inferior.

Na borda superior, para cada linha, as transições relevantes são a primeira do preto para o branco (seta 1 na Fig. 18) e a última do branco para o preto (seta 2). Na borda inferior, as transições relevantes são a primeira do branco para o preto (seta 3) e a última transição do preto para o branco (seta 4 na Fig. 18). Por fim, para o corpo do tubo, as transições relevantes são iguais às definidas para a região da borda superior. A assinatura final do objeto é então formada por todos os pontos das transições relevantes, em todas as regiões de interesse.

Observa-se ainda que, para o objeto do exemplo, são necessários dois conjuntos de transições para uma descrição completa: a assinatura esquerda, em linha grossa e contínua na Fig. 18, e a assinatura direita, em linha grossa e tracejada. O termo “esquerda” se refere a posição das bordas referentes a esta assinatura em relação a imagem bidimensional adquirida. O mesmo conceito é válido para o termo “direita”. A soma destas duas assinaturas compõem a assinatura final do objeto.

A geração da assinatura é detalhada a seguir. A Fig. 19 mostra a construção no tempo da borda superior. A Fig. 20, a inferior. A Fig. 21 mostra a formação da assinatura do corpo e a Fig. 22 apresenta o processo completo. A temporização utilizada nestas figuras

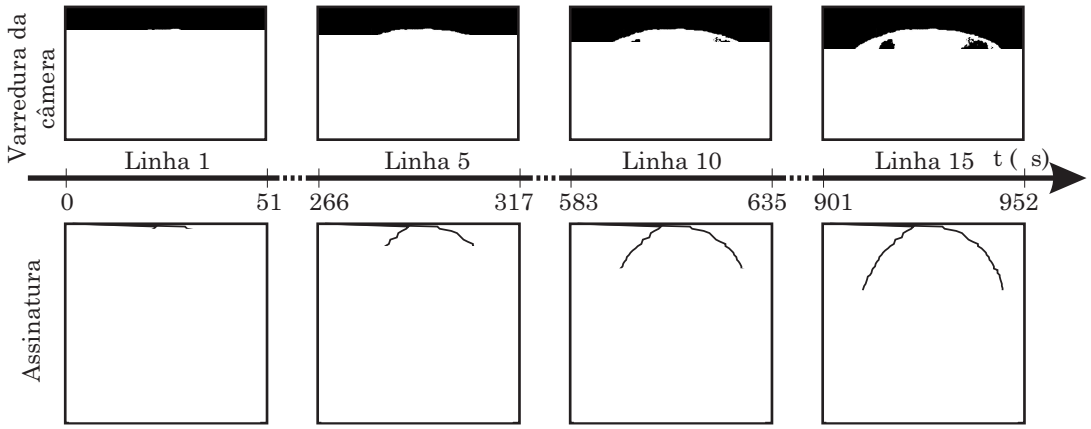


Figura 19: Geração das assinaturas da borda superior do tubo.

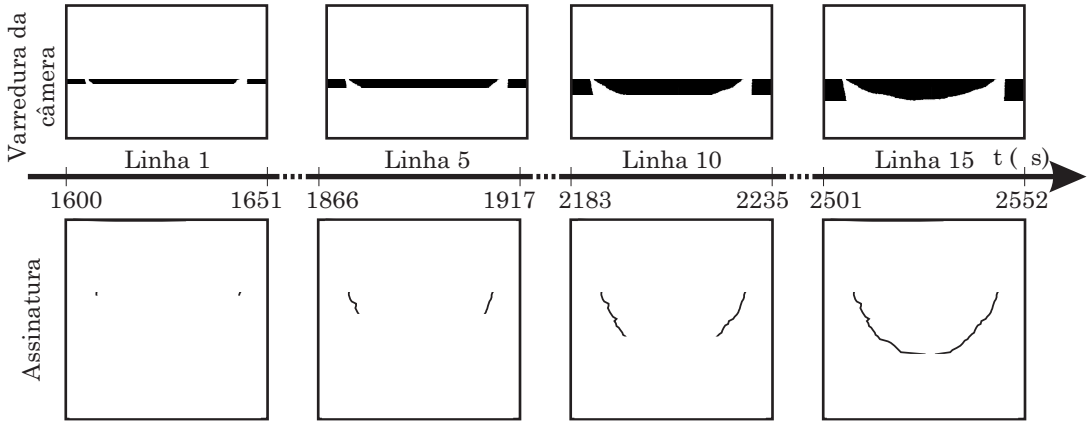


Figura 20: Geração das assinaturas da borda inferior do tubo.

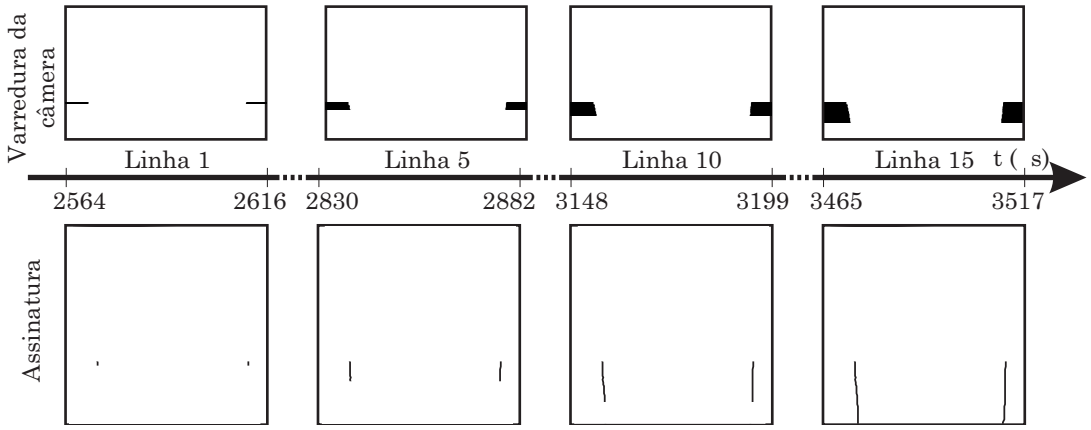


Figura 21: Geração das assinaturas do corpo do tubo.

foi referente à aquisição realizada por uma câmera NTSC padrão de vigilância. Neste exemplo, a assinatura foi completamente gerada em 3,5ms, apesar do baixo desempenho da câmera. Observe que, para um tubo, não foi necessário finalizar a verredura do quadro pela câmera para gerar a assinatura completa.

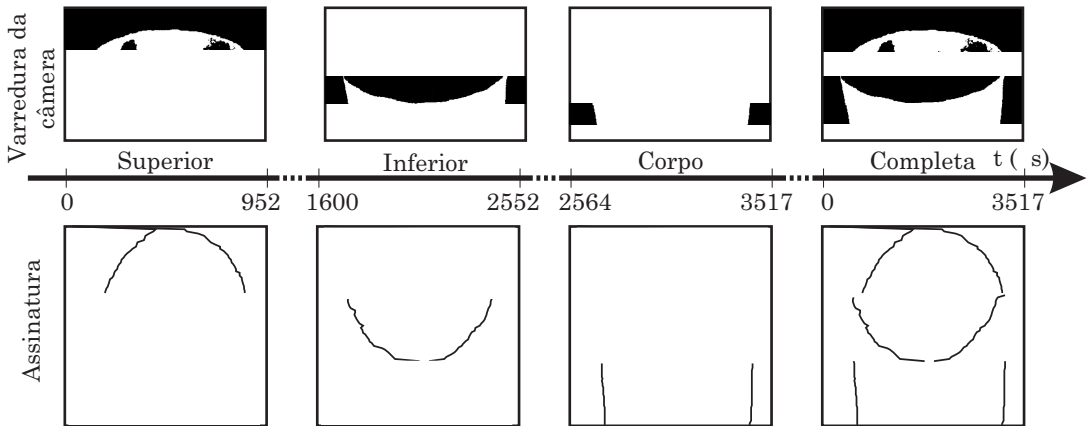


Figura 22: Composição das assinaturas completas do tubo.

A Fig. 23 mostra o resultado desta etapa, desta vez apresentando a assinatura de um tubo amassado.

(a) Imagem adquirida. (b) Resultado da limiarização. (c) Assinatura gerada.

Figura 23: Resultado da etapa de geração da assinatura para um tubo amassado.

Assim como na etapa de limiarização, a geração da assinatura não requer o armazenamento de imagens na memória. As transições relevantes podem ser identificadas na medida em que a imagem adquirida é limiarizada, pixel a pixel. Ou seja, em paralelo com as etapas anteriores. Com isto, mais uma vez, é possível reduzir o custo da implementação da solução e o tempo total de processamento.

4.4 Comparação

Por fim, a última etapa da LEDges é a comparação. Nesta etapa é realizado o reconhecimento da assinatura gerada através de simples comparações entre esta, e as assinaturas de um objeto padrão "bom".

Foi utilizado um método estatístico extremamente simples com a finalidade de com-

provar a viabilidade técnica das etapas anteriores. O método estatístico utilizado consiste em calcular a média das assinaturas de dez objetos "bons" e utilizar estes valores como referência. De fato, só é necessário armazenar duas assinaturas: (I) média + tolerância e (II) média – tolerância. O valor da tolerância pode ser baseado no desvio padrão das dez assinaturas de objetos "bons" ou pode ser um valor fixo, definido experimentalmente pelo operador. O primeiro caso, tolerância baseada no desvio padrão, exige maior processamento para obter as assinaturas de referência, porém permite a detecção de erros dimensionais menores. Estas assinaturas de referência podem ser vistas na Fig. 24. A média está representada na assinatura em linhas contínuas e, as assinaturas descritas em (I) e (II), estão em linhas tracejadas. Se a assinatura do objeto sob inspeção estiver completamente contida entre as assinaturas (I) e (II), área destacada com preenchimento na Fig. 24, o objeto sob inspeção é classificado como "bom". Se existir qualquer ponto das assinaturas fora da faixa definida, o objeto é classificado como "ruim". Ou seja, o resultado desta etapa é um sinal binário indicando se o objeto sob inspeção está de acordo com as especificações da indústria, ou não.

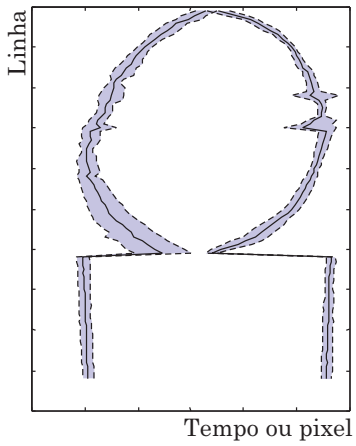


Figura 24: Média de assinaturas de dez objetos "bons" ± tolerância baseada no desvio padrão.

A comparação, também pode ser realizada em paralelo com as etapas anteriores. É possível comparar a assinatura em construção, linha a linha, com a faixa de tolerância previamente definida. Inclusive, assim que um ponto da assinatura em geração for detectado fora desta faixa, o sistema de IVA já pode finalizar a análise uma vez que, neste momento, o objeto já é classificado como "ruim". Ou seja, a aquisição da imagem pode ser finalizada antes da varredura completa do quadro.

Apesar dos bons resultados obtidos após implementação desta etapa, conforme será apresentado nos próximos capítulos, o número de objetos utilizados para validação desta etapa é insuficiente. Logo, a etapa de comparação ainda não está completamente definida,

podendo ser modificada em trabalhos futuros com a utilização de uma análise estatística mais adequada, de redes neurais ou de lógica fuzzy.

4.5 Restrições para aplicação da LEDges

Para a aquisição das imagens, a LEDges tem as mesmas restrições apresentadas nos trabalhos similares (30–34), que dependem da reflexão difusa da luz no objeto sob inspeção para realizar corretamente o processamento. A LEDges depende do posicionamento relativo do objeto sob inspeção, da câmera, da fonte de luz e do plano de fundo. Este último deve estar distante do primeiro plano em, pelo menos, a mesma distância entre o primeiro plano e a câmera. A cor e o material do objeto também podem interferir nos resultados, uma vez que a câmera precisa capturar a luz do *LED* refletida pelo objeto. (I) A cor preta absorve a luz emitida pelo *LED*, (II) materiais com alta refletividade podem não refletir a luz na direção da câmera e (III) objetos transparentes deixam passar toda luz do *LED*.

Além disto, bordas muito finas alinhadas radialmente com a câmera podem não ser detectadas. Por exemplo, na região entre a borda superior e inferior da Fig. 18. Caso seja necessária a detecção dessas bordas, é possível implementar dois sistemas independentes com a LEDges, capturando a imagem do objeto em dois ângulos distintos, capturando com boa definição todas as bordas do objeto.

Quanto ao espectro, o *LED* deve emitir luz no mesmo comprimento de onda em que a câmera é sensível na captura da imagem. Uma boa solução foi implementada em (33). Nesta, foram utilizados *LEDs* que emitem em 850 nm e é capturada somente a luz infravermelha (850 nm $\pm$ 25 nm) proveniente da cena. Para tal, uma câmera infravermelha com um filtro óptico passa faixa adicional foram utilizados. Com isto, a luz proveniente do plano de fundo que seja diferente de 850 nm, inclusive todo o espectro visível, não é capturada pela câmera. Porém, na indisponibilidade destes componentes, a técnica também se mostrou eficiente mesmo com a utilização de câmeras de amplo espectro: em todo o visível e infravermelho próximo. Nesta categoria se incluem as câmeras analógicas monocromáticas de baixo custo utilizadas em vigilância. No caso do uso destas câmeras de amplo espectro, é necessário apenas aumentar a potência do *LED* para que a câmera capture a região de interesse significantemente mais iluminada que o plano de fundo.

Sobre a definição do limiar L na etapa de limiarização, há também outras técnicas para definição deste valor que poderiam ser utilizadas na LEDges, como a limiarização ótima, seleção de limiar baseada nas características de fronteiras e limiares baseados em diversas

variáveis (37). Para ambientes não-controlados, estas são mais robustas à variação da iluminação, podem detectar objetos pequenos na imagem e com bordas mais sutis. Porém, consomem mais recursos computacionais, requerem maior tempo para processamento e necessitam que a imagem inteira seja armazenada na memória para posterior cálculo de L. Logo, enquanto o ambiente da aplicação da LEDges for controlado, é possível utilizar a técnica de limiarização global sem prejuízos para a taxa de acertos da IVA.

A respeito da identificação das transições relevantes e regiões de interesse, será pesquisada e desenvolvida uma ferramenta computacional que determine automaticamente os parâmetros para a LEDges, tais como a quantidade e posições das regiões nas quais a imagem deve ser dividida, as regras para detecção destas regiões pelo sistema computacional e o número de assinaturas necessárias para descrever o objeto. No momento, esta se apresenta como a principal dificuldade para a aplicação da LEDges em objetos complexos, com muitos níveis de profundidade e bordas irregulares.

Por fim, a técnica de comparação proposta resulta em uma análise com uma faixa de tolerância bastante restrita para variações nas assinaturas. Inclusive mínimos deslocamentos na assinatura, provenientes do posicionamento do objeto em relação à câmera, resultam em uma detecção falha de irregularidades nas bordas do objeto. Logo, esta técnica deve ser aplicada apenas em ambientes que posicionem precisamente o objeto sob inspeção em relação à câmera. Caso necessário, a análise de assinaturas é uma área amplamente estudada e é possível aprimorar esta etapa para que o sistema seja invariante a deslocamentos do objeto em relação à câmera, além de possibilitar o ajuste da tolerância.

Apesar destas restrições, a LEDges está bem ajustada para aplicações em ambientes industriais onde, normalmente, tem-se o ambiente controlado.

4.6 Análise do capítulo

Foram observadas algumas vantagens competitivas da LEDges em relação aos sistemas de IVA já desenvolvidos. A LEDges é capaz de detectar todas as bordas do objeto com alto desempenho através de um sistema embarcado e com uso reduzido de recursos computacionais. Estes diferenciais viabilizam a utilização de um sistema de IVA em algumas aplicações industriais que não tinham seus requisitos atendidos anteriormente, sobretudo em inspeções com requisitos de alto desempenho, compacto e de baixo custo.

Dentre os diferenciais da LEDges, a possibilidade de execução paralela das quatro etapas em um modelo de *pipeline* a nível de pixel se destaca. Diretamente relacionado a isto,

o uso da memória é significantemente reduzido, uma vez que não é necessário armazenar a imagem para processamento. Com isto é possível reduzir o custo da implementação da solução e o tempo total de processamento.

Outro diferencial, é a possibilidade de ser implementada em diversas arquiteturas embarcadas, satisfazendo à demanda crescente por desempenho (tempo de resposta e precisão) e pelo uso mínimo de recursos disponíveis (financeiro, humano, físico e energético). Sendo possível, inclusive, a utilização de câmeras analógicas. Neste caso, além da aquisição, a limiarização também pode ser realizada analogicamente, através de um simples comparador de tensão. Ou seja, não é necessária a utilização de um conversor analógico-digital. Com isto, é possível aumentar os mesmos benefícios já obtidos na redução do uso de memória: redução no custo da implementação da solução e no tempo total de processamento.

Algumas restrições foram apresentadas. Comuns a todos os sistemas de IVA que se baseiam na reflexão difusa da luz, as principais restrições estão relacionadas a cor, material e posicionamento do objeto em relação à câmera e ao plano de fundo. Apesar destas, a técnica se mostrou adequada para aplicações industriais.

5 *Avaliação de arquiteturas para implementação*

A definição da arquitetura a ser utilizada para implementação da LEDges é baseada nos recursos de *hardware* necessários para executar o processamento da imagem e na organização destes recursos, sempre visando atender aos requisitos da indústria quanto a inspeção.

O objetivo principal é atender corretamente a funcionalidade especificada. A segunda prioridade é atingir a velocidade de inspeção e taxa de acerto necessárias. O terceiro objetivo, este de natureza econômica, é reduzir os custos de produção. A flexibilidade para atualização dos parâmetros da inspeção é outra qualidade desejável. O esforço da equipe de engenharia para desenvolvimento do sistema de IVA e o tempo para a disponibilização deste no mercado também diferem entre arquiteturas. Por fim, há a preocupação com a eficiência energética. Porém, como o equipamento não será alimentado por baterias, e sim pela rede de energia industrial, o consumo do sistema embarcado normalmente será inferior ao limite aceitável, definido pela indústria.

O procedimento para definição das arquiteturas que serão utilizadas nas implementações da LEDges pode ser observada na Fig. 25.

Cada etapa do procedimento, apresentado na Fig. 25, será detalhada nas próximas seções. Na seção 5.1, análise da técnica, é realizada uma análise preliminar do algoritmo e a identificação dos controladores necessários. Na seção 5.2 é apresentada a modelagem da LEDges, realizada em computador, enquanto a seção 5.3, otimização, identifica os gargalos de desempenho no processamento dos dados e verifica a largura dos dados realmente necessários. Por fim, na seção 5.4, é elaborada uma lista com as características e restrições para as diferentes arquiteturas, bem como é verificada a adequação da técnica com as arquiteturas.

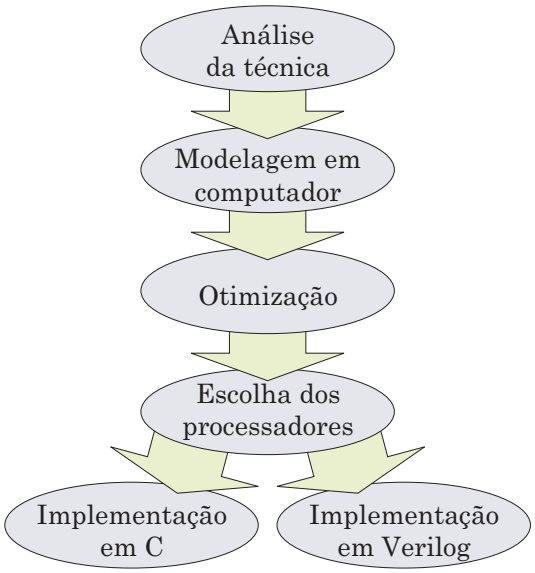


Figura 25: Fluxo para definição das arquiteturas.

5.1 Análise da técnica

A primeira etapa para definição da arquitetura é realizar uma análise da técnica. Foi realizada uma análise simples, dividida em duas atividades: (i) uma análise preliminar do algoritmo e (ii) a identificação dos controladores necessários.

Na análise preliminar do algoritmo, foram observados:

- Diagrama em blocos;
- Larguras dos barramentos requeridas para transferência das informações entre os principais blocos funcionais;
- Taxa de transferência dos dados estimada entre os principais blocos;
- Utilização da memória;
- Características de processamento das principais operações aritméticas;

Um diagrama em blocos foi elaborado e é apresentado na fig. 26. Quando utilizada uma câmera digital, é necessário adicionar dois blocos. O primeiro, configura a câmera, ajustando parâmetros tais como: tempo de exposição, sensibilidade dos sensores e área da imagem a ser adquirida. O segundo, recebe os dados da câmera e disponibiliza estes para a etapa de limiarização. Neste segundo bloco pode ser realizada uma conversão serial-paralela das informações, pode ser convertido o padrão da imagem (por exemplo, de RGGB para RGB) e pode ser filtrada a componente de cor de interesse (por exemplo, disponibilizar somente a componente R da imagem RGB).

Foi verificada também a necessidade de uma etapa de pré-processamento. Apesar do

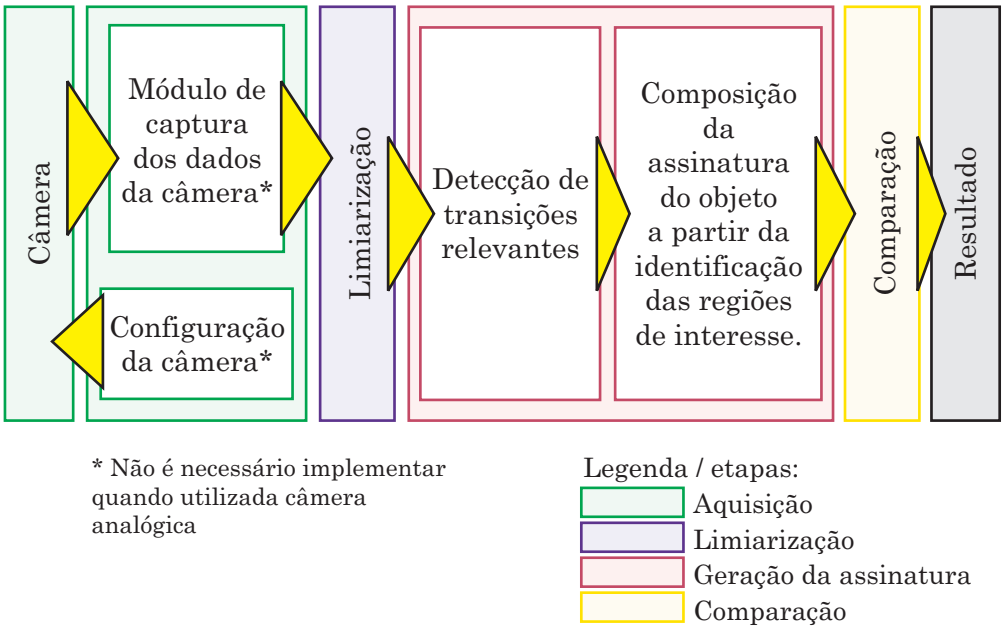


Figura 26: Diagrama em blocos para avaliação das arquiteturas.

ajuste do contraste facilitar a definição do nível do limiar, esta etapa foi descartada uma vez que é necessário armazenar a imagem na memória para realizar o pré-processamento.

Quanto a largura dos dados, foi verificado que a câmera digital testada disponibiliza uma imagem de 48 bits por pixel no padrão RGGB (12 bits por componente). Além disto, os dados são disponibilizados serialmente pela câmera, no máximo, na metade da frequência de *clock* fornecida para a câmera.

Sobre o processamento das operações aritméticas, foi identificada uma computação não-linear no cálculo do desvio padrão, na etapa de comparação. Deve-se então testar se a utilização de uma faixa de tolerância fixa, definida pelo operador, atende aos requisitos da IVA.

Quanto aos controladores, foram identificados três controladores principais: na configuração da câmera, na composição da assinatura do objeto a partir da identificação das regiões de interesse e na etapa de comparação. Todos os três controladores são simples o suficiente para implementação em máquinas de estados finitos.

5.2 Modelagem da LEDges em computador

A primeira etapa da LEDges, a aquisição da imagem, foi realizada a partir de uma câmera de baixo custo, resolução de 240 x 320 pixels, saída NTSC, e a transferência da imagem para o computador através de uma placa de captura de vídeo. A escolha da câmera

foi visando comprovar a viabilidade técnica da LEDges mesmo ao utilizar os componentes de menor desempenho, resolução e custo encontrados no mercado local.

As imagens de tubos foram adquiridas pela câmera com o LED ligado. Uma imagem de tubo "bom" é apresentada na Fig. 27. Observar que o ajuste automático de brilho e contraste foi realizado pelo *hardware* interno da câmera, destacando as bordas relevantes no primeiro plano.

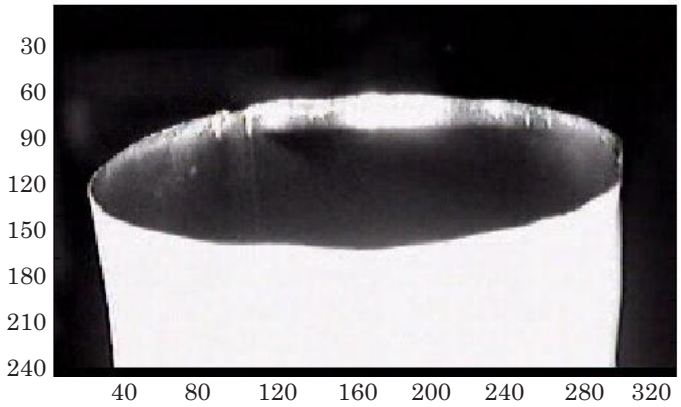


Figura 27: Imagem adquirida e armazenada no computador.

Além da imagem mostrada na Fig. 27, adicionalmente foram adquiridas e armazenadas no computador 15 imagens de tubos bons e 40 de tubos amassados.

As três etapas seguintes da LEDges foram modeladas em três programas independentes através de um *software* específico para aplicações científicas e em engenharia, o FreeMat v4.0. A uma tela do FreeMat é exibida na Fig. 28. Este é similar aos *softwares* comerciais MATLAB, da Mathworks, e IDL, da Research Systems, com o diferencial de ser livre e ter código aberto.

Na limiarização, a imagem adquirida é varrida da esquerda para a direita, de cima para baixo, exatamente na ordem em que os pixels foram enviados para o computador. Pixel por pixel, as intensidades destes são comparadas com um limiar fixo, estabelecido manualmente. Como a escala das imagens adquiridas é 256 níveis de cinza, o limiar foi definido com o valor 128. Se a intensidade do pixel for maior que o limiar, o pixel correspondente na imagem limiarizada terá valor 1. Caso contrário, o pixel terá valor 0. Esta nova matriz, a imagem limiarizada, é criada na medida em que a imagem adquirida é varrida. O resultado da limiarização realizada no FreeMat pode ser observada na Fig. 29.

A geração de assinaturas também foi implementada. Assim como na etapa anterior, a matriz da imagem limiarizada é varrida na mesma seqüência na qual a imagem foi adquirida. Para cada linha da imagem, são armazenadas as transições relevantes. O número

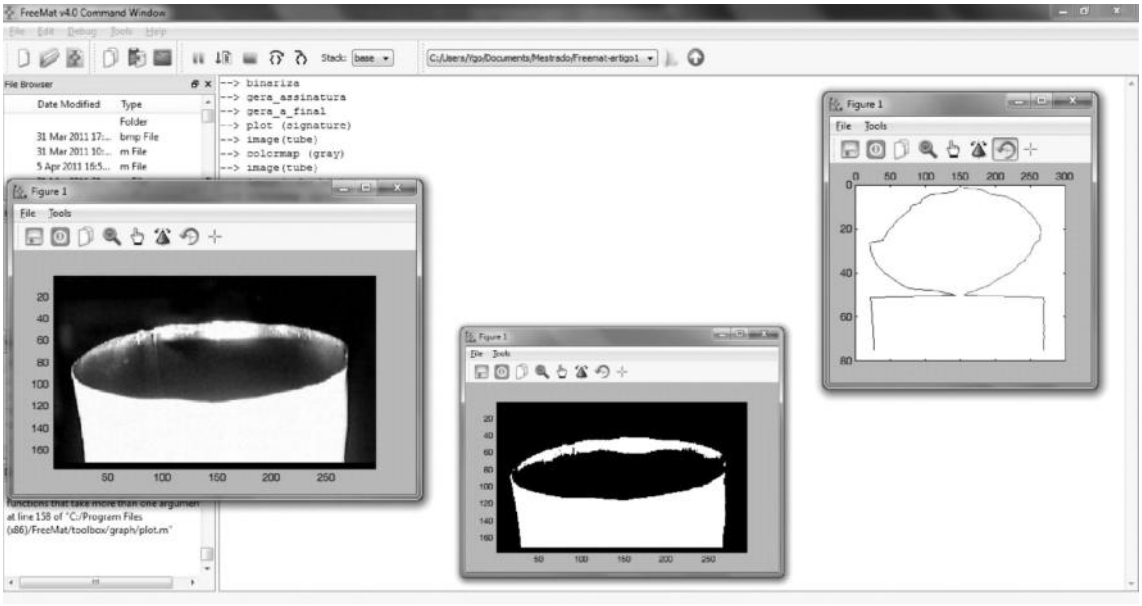


Figura 28: Tela do FreeMat.

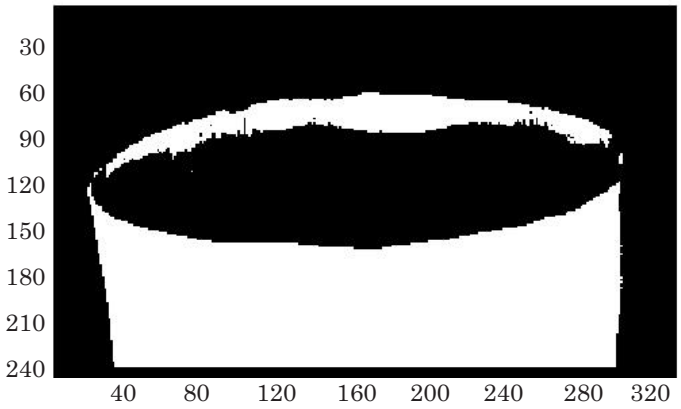
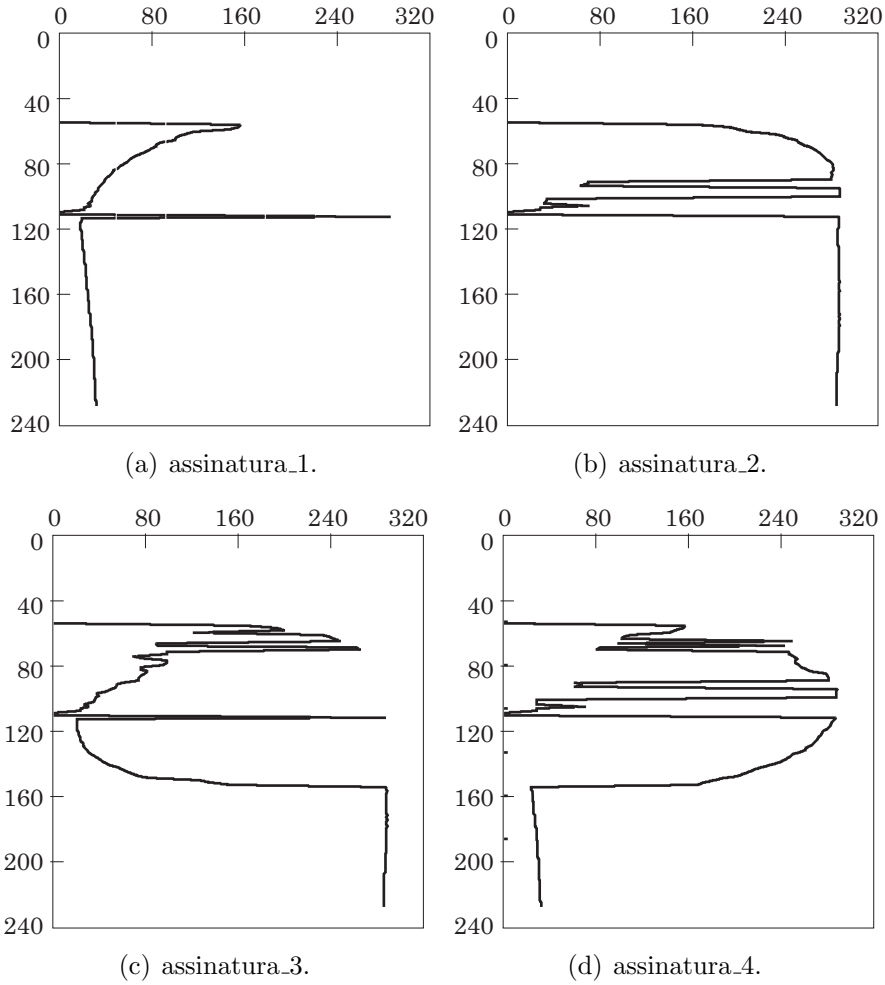


Figura 29: Imagem limiarizada.

de pixels entre o início de cada linha e a primeira transição de preto para branco é armazenada em um vetor, chamado assinatura_1 (fazendo referência à seta 1 na Fig. 18). Este vetor é criado na medida em que a imagem é varrida e o valor máximo armazenado no vetor é 320, que é a distância máxima possível entre o início da linha e uma transição. Similarmente, o vetor assinatura_2 armazena a última transição do branco para o preto. A assinatura_3, a primeira do branco para o preto e o vetor assinatura_4 armazena a última transição do preto para o branco. Os quatro vetores armazenados são apresentados na Fig. 30.

A técnica necessita que a imagem seja dividida em regiões de interesse. No caso de um tubo, sua imagem é dividida em três regiões: borda superior, borda inferior e corpo do tubo. Através da Fig. 30 e dos valores armazenados nos vetores é possível observar o número de linhas com informações consistentes para cada região de interesse. Para as



(a) assinatura_1. (b) assinatura_2.

(c) assinatura_3. (d) assinatura_4.

Figura 30: Vetores armazenados plotados.

imagens adquiridas pela câmera de baixa resolução, o número de linhas para cada região foi 24. Logo, as três regiões ocupam 72 linhas da imagem.

A assinatura final é descrita em forma de uma matriz com duas colunas e 72 linhas. A primeira coluna armazena os valores referentes à assinatura esquerda, que é composta por valores das assinaturas_1, nas regiões de borda superior e corpo do tubo, e assinatura_3, na região de borda inferior. Similarmente, a segunda coluna armazena a assinatura direita, composta por parte dos vetores assinatura_2 e assinatura_4. Logo, a assinatura completa de um tubo é composta por 144 valores, cada valor variando entre 0 e 320. A matriz da assinatura final é plotada na Fig. 31.

Cada posição da matriz deve ter comprimento mínimo de 9 bits para armazenar valores entre 0 e 320. Logo, são necessários 162 bytes para armazenar a matriz. Porém, visando reduzir a complexidade das operações no computador, foram reservados dois bytes para cada posição na matriz. Neste caso, são necessários 288 bytes de memória para armazenar a descrição do tubo.

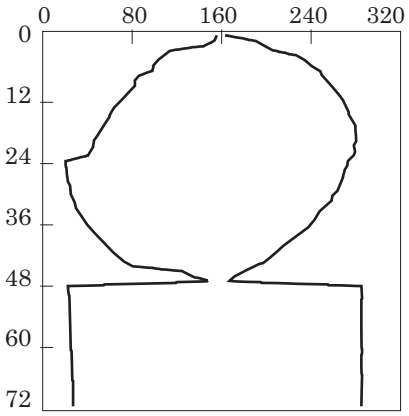


Figura 31: Matriz da assinatura final plotada.

A última etapa da LEDges, a comparação, foi implementada a partir do cálculo da média e da tolerância baseada no desvio padrão de dez assinaturas referentes a imagens de tubos "bons". Este cálculo foi realizado no FreeMat utilizando funções específicas. Os resultados foram armazenados em duas assinaturas de referência: "média + tolerância" e a "média – tolerância". Se a assinatura do objeto sob inspeção estiver completamente contida entre as assinaturas de referência, o objeto sob inspeção é classificado como "bom". Se existir qualquer ponto das assinaturas fora da faixa definida, o objeto é classificado como "ruim". O resultado foi uma taxa de acerto de 100% na classificação das 55 imagens adquiridas.

5.3 Otimização

Nesta etapa foram identificados os gargalos de desempenho no processamento dos dados. A principal modificação no algoritmo está relacionada com a adequação da geração da assinatura da borda inferior para operação em *pipeline*.

Quais linhas da imagem limiarizada compõem a borda inferior só podem ser definidas após a varredura completa desta borda. Ou seja, a assinatura da borda inferior só pode ser gerada após a varredura completa pela câmera da região da borda inferior. Uma solução neste caso seria armazenar a imagem limiarizada na medida em que a câmera adquire e, ao detectar o fim da borda inferior, parar este armazenamento e fazer uma varredura desta imagem armazenada, lendo as linhas e gerando as assinaturas de baixo para cima nesta região da borda inferior.

Para não ter que armazenar a imagem e perder a característica de execução em *pipeline* sem uso intenso de memória, foi desenvolvido um algoritmo que armazena to-

das as transições relevantes para a borda inferior, em todas as linhas da imagem, em um *buffer* cíclico com tamanho igual ao número de linhas da borda inferior, definido a partir da modelagem em computador. Ou seja, ao detectar o fim da borda inferior, o processador simplesmente copia os dados gravados no *buffer* cíclico para a memória que compõe a assinatura do tubo sob inspeção. Como esta cópia ocupa pouco o processador, esta atividade pode ser realizada durante a varredura das linhas do corpo do tubo, sem prejuízos para o desempenho da IVA.

Na otimização também foram verificadas as larguras dos dados realmente necessárias. Por exemplo, foi verificado que uma imagem monocromática de 8 bits por pixel, ao invés dos 48 bits RGB fornecidos pela câmera, é suficiente para obter uma imagem limiarizada adequada para a detecção das bordas.

5.4 Escolha do processador

A escolha é entre processador de uso geral, uso específico ou os dois associados. Para tal, a LEDges e sua modelagem foram analisadas, tendo sido identificadas características favoráveis para a implementação em FPGA ou ASIC, tais como:

1. O processamento é dividido em blocos (aquisição, limiarização, geração da assinatura e comparação) que interagem de forma simples e imutável. O modelo de execução se baseia em cada bloco processar o dado de entrada e disponibilizar o resultado para a próxima etapa;
2. A computação é invariante no tempo [1] e não é recursiva. Com isto, associado ao item anterior, é possível executar o processamento dos blocos em paralelo em um modelo de *pipeline*;
3. Um dos principais parâmetros de desempenho da IVA é a taxa de inspeção, ou seja, indicador baseado no fluxo de saída dos dados. Como esta também é a principal preocupação do processamento em *pipeline*, e considerando os dois itens anteriores, reforça-se a possibilidade de implementação em *pipeline* da LEDges;
4. O controle de fluxo dos dados é simples, ou seja, o fluxo não depende dos valores dos pixels adquiridos. Além disto, não há diversos modos de operação, formatos de dados ou parâmetros de configuração, uma vez que o ambiente da IVA é controlado, ou seja,

1. Seja um sistema relaxado que, quando sujeito a uma entrada qualquer u(t), produz uma saída y(t). O sistema é dito ser invariante no tempo se, e somente se, a resposta do sistema à entrada u(t+r) é igual a y(t+r).

este é configurado inicialmente e ocorrem poucas mudanças durante as inspeções. Para controlar o fluxo é possível implementar um simples contador associado a uma máquina de estados finitos, que é uma tecnologia rápida, ocupa pouco espaço no *chip*, eficiente no uso de energia e muito simples para se verificar e estimar os recursos necessários para implementação;

5. O algoritmo é fixo e previamente conhecido. Analisando este, observa-se um fluxo de dados regular, sem computação intensiva. Com isto é possível reduzir a área do chip e, ao transferir dados por conexões fixas, os *delays* destas conexões entre os blocos funcionais. Além disto, a regularidade facilita o compartilhamento de recursos de *hardware*, o reuso, e reduz o esforço para o desenvolvimento do sistema e sua verificação. De fato, a computação mais irregular é no cálculo do desvio padrão, na construção das assinaturas de referência. Como, esta atividade não necessita de alto desempenho, pode-se executar uma seqüência de computação irregular. Porém, se necessário aumentar o desempenho, é possível ainda construir uma *look-up table*[2] seguido de interpolação;
6. Quanto a capacidade de armazenamento, sabe-se que as memórias ocupam uma extensa área no *chip*, sendo um componente que deve ser utilizado com eficiência. Além disto, o uso intensivo da memória reduz o desempenho da IVA ou aumenta significantemente o uso de recursos computacionais. A LEDges necessita de uma memória mínima. Não é necessário armazenar a imagem adquirida nem a limiarizada, sendo apenas necessário menos de 1 kB para armazenar as duas assinaturas de referência;
7. A LEDges é insensível aos efeitos da precisão aritmética finita, ou seja, não necessita da aritmética de ponto flutuante. Também não necessita de largos barramentos de dados nem endereços. Por fim, não são realizadas muitas multiplicações ou divisões utilizando dados largos. Estes três fatores impactam positivamente na LEDges quanto ao tamanho do circuito, *delays* dos blocos lógicos, comprimento das interconexões, capacitâncias parasitas e dissipação de energia.

Além das características que sugerem a implementação da LEDges em processadores de uso específico, também foi identificada uma oportunidade de sua implementação em processadores de uso geral, mais especificamente em microprocessadores de baixo desempenho. O principal gargalo para a implementação nestes dispositivos é a necessidade de utilizar um *hardware* independente para executar cada etapa do pipeline da LEDges.

2. Tabela de valores previamente construída contendo algumas entradas e os resultados dos processamentos destas entradas ao efetuar uma determinada operação.

A seguir são relacionadas as etapas da LEDges com o *hardware* da arquitetura baseada em microcontrolador que deve ser utilizado para seu processamento.

1. Aquisição dos dados: Câmera analógica ou digital;
2. Limiarização: Comparador de tensão, se utilizado câmera analógica, ou comparador digital, se utilizado câmera digital;
3. Geração de assinaturas: Temporizador — Periférico comum em microcontroladores;
4. Comparação: Unidade de processamento principal.

Como vantagens, em relação à implementação com FPGA, a arquitetura baseada em microcontrolador:

1. É muito mais simples e rápida para implementar a modelagem realizada no FreeMat, sobretudo se programado na linguagem "C";
2. É mais flexível para a passagem dos parâmetros da IVA;
3. Em geral, tem menor custo que a baseada em FPGAs, tanto no *hardware* quanto no desenvolvimento do sistema de IVA. O menor custo é obtido nas aplicações que não exigem elevados desempenhos uma vez que é possível utilizar uma câmera analógica de baixo custo na arquitetura baseada em microcontrolador de baixo desempenho.

Por outro lado, a arquitetura baseada em microprocessador tem menor grau de integração na PCI [3] e, em geral, menor desempenho que a baseada em FPGA.

A última possibilidade é a integração dos dois processadores. Por exemplo, as etapas de limiarização e geração de assinatura, que têm maiores fluxos de dados, podem ser realizadas em processador de uso específico enquanto a comparação, que tem fluxo irregular e necessita de maior flexibilidade, em processador de uso geral. Apesar de ser uma solução bastante atraente, a utilização de dois processadores para implementação da LEDges não se mostrou necessária para atender os requisitos atuais de fluxo de dados e flexibilidade estabelecidos neste trabalho.

Foi decidido então implementar a LEDges em duas arquiteturas:

1. Baseada em FPGA para aplicações de elevado desempenho;
2. Baseada em microcontrolador para aplicações mais flexíveis e de menor custo.

3. Em geral, necessitará de um circuito externo ao microcontrolador ou computador para realizar a limiarização (um circuito comparador) e, se utilizada a câmera analógica, um segundo circuito externo para identificar os pulsos de sincronismo do sinal NTSC. Estes dois circuitos externos não são necessários quando utilizada a arquitetura baseada em FPGA.

Na Fig. 32 é apresentado um diagrama relacionando características de desempenho, esforço computacional e custo para a implementação da LEDges. A faixa de esforço computacional da LEDges foi definida a partir de uma análise da modelagem computacional. Já o fluxo de dados, é estabelecido de acordo com a câmera utilizada. Foram consideradas duas faixas de resolução de imagem. A primeira é aplicada a uma imagem adquirida através de uma câmera analógica com resolução 640 x 480 pixels (VGA) e taxa de 30 quadros por segundo. A segunda é aplicada a uma imagem adquirida através de uma câmera digital e está relacionada a uma imagem 1280 x 1024 pixels (XGA) a 60 quadros por segundo. Ao realizar a intersecção entre as características de eforço computacional e fluxo de dados, obtemos duas regiões no gráfico aonde podem ser implementadas a LEDges.

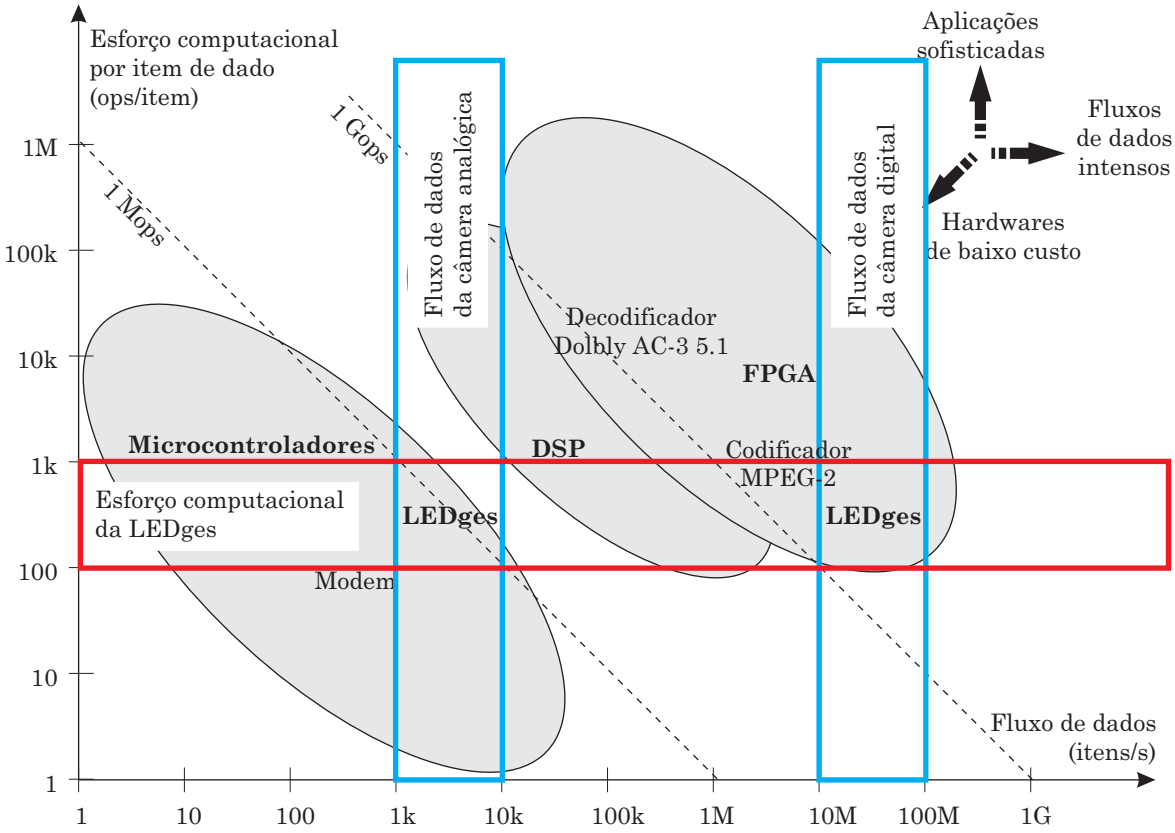


Figura 32: Relação entre fluxo de dados, esfoço computacional e custo das soluções para a implementação da LEDges. Adaptado de Kaeslin(1).

5.5 Análise do capítulo

A modelagem em computador comprova a viabilidade técnica da LEDges e permite uma análise do algoritmo e das conexões necessárias entre os quatro blocos funcionais, que implementam as etapas da LEDges, visando a escolha da arquitetura mais adequada

para a implementação embarcada desta técnica.

Como resultado desta análise, a funcionalidade especificada pode ser atendida adequadamente, tanto na arquitetura baseada em microprocessadores, quanto na baseada em FPGA. Este é o objetivo principal na escolha da arquitetura.

A segunda prioridade é alcançar a velocidade de inspeção necessária mantendo a taxa de acerto obtida na modelagem, de 100%. Neste item as duas arquiteturas analisadas diferem significantemente. No FPGA, a LEDges pode ser executada em imagens com maior resolução, detectando erros dimensionais menores, e em maior velocidade.

Enquanto a implementação em microprocessador apresenta vantagens para atender os três próximos objetivos: reduzir os custos de produção, flexibilizar a atualização dos parâmetros da inspeção e reduzir o esforço da equipe de engenharia para desenvolvimento do sistema de IVA.

Logo, a LEDges é flexível quanto à escolha da arquitetura, devendo esta ser definida apenas após uma análise criteriosa do ambiente da inspeção, sobretudo do objeto a ser inspecionado, e dos requisitos de desempenho, flexibilidade, custo e tempo de desenvolvimento apresentados pela indústria.

6 *Arquitetura baseada em microcontrolador e câmera analógica*

A LEDges, em sistema embarcado, foi validada em duas arquiteturas. A primeira, apresentada neste capítulo, baseada em microcontrolador. A segunda, assunto do próximo capítulo, em FPGA.

As principais vantagens da arquitetura baseada em microcontrolador são: (I) a flexibilidade para atualização dos parâmetros da inspeção, (II) o esforço reduzido da equipe de engenharia para o desenvolvimento da IVA e (III) os custos reduzidos para produção. Inclusive, visando a redução de custos, a câmera escolhida foi a analógica de baixo desempenho, padrão para aplicações de circuito fechado de TV (CFTV).

Como principal desvantagem, a arquitetura baseada em microcontrolador apresenta baixa velocidade de inspeção em relação à implementação com FPGA.

Neste capítulo será apresentado um diagrama em blocos da arquitetura e as etapas da LEDges relacionadas com os blocos funcionais.

6.1 Diagrama em blocos

A arquitetura baseada em microcontrolador é composta pelos elementos de *hardware* enumerados a seguir:

1. *LED* — Tem como propósito destacar as bordas do objeto sob inspeção e minimizar o efeito de outras fontes de luz sobre este objeto;
2. Câmera analógica — Adquire a imagem cujas áreas iluminadas do objeto em primeiro plano tem altas intensidades, enquanto o plano de fundo e as sombras são exibidos em baixas intensidades;

3. Comparador de tensão analógico — Executa a segmentação da imagem através da limiarização do sinal proveniente da câmera em dois níveis de intensidade: (I) o plano de fundo e as sombras são indicados através de regiões com intensidade zero, cor preta; (II) e áreas iluminadas nos objetos do primeiro plano são indicadas através da intensidade máxima, cor branca;
4. Detector de sincronismo NTSC — Identifica o início de cada quadro e cada linha da imagem adquirida. Este sinal é fundamental para a etapa de geração da assinatura;
5. Microcontrolador — Executa a geração da assinatura e comparação desta com as assinaturas de referência.

O diagrama em blocos da arquitetura é apresentado na Fig. 33. As cores dos blocos indicam a etapa da LEDges que o *hardware* implementa, conforme legenda. Cada etapa terá sua implementação detalhada nas seções seguintes.

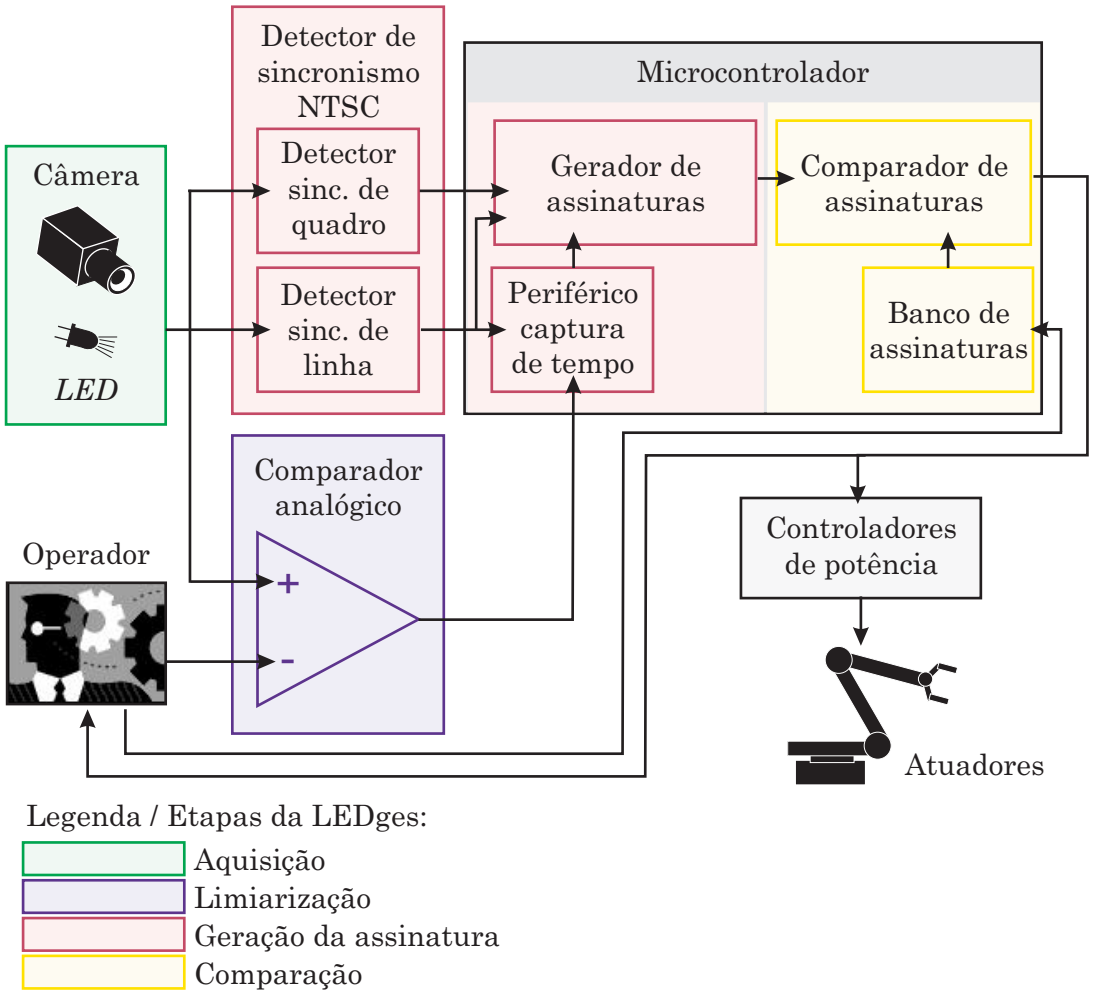


Figura 33: Diagrama em blocos.

Observar que, apesar da câmera analógica, não há o conversor analógico-digital para digitalizar a imagem adquirida. Além disto, não há módulos de memória externos, sendo

a memória interna do microcontrolador suficiente para armazenar as assinaturas de referência.

6.2 Estudo de caso: detecção de amassados em tubos de creme dental

A LEDges foi implementada e aplicada a um problema industrial real, onde defeitos foram detectados em bordas de tubos de creme dental, deste modo, validando a viabilidade técnica da arquitetura baseada em microcontrolador e câmera analógica.

Nas próximas seções serão descritos os componentes de *hardware* utilizados para a implementação da LEDges nesta arquitetura.

6.2.1 Aquisição da imagem

Para a iluminação estruturada é utilizado um *LED* vermelho, comprimento de onda dominante de 627 nm, e potência elétrica de 1 W. A cor vermelha foi escolhida por que é detectada eficientemente pelo sensor da câmera CCD e está no espectro visível. O padrão de radiação é o *lambertian* com ângulo de visão de 120°. Este padrão emite maior intensidade luminosa na direção normal ao *LED* e decresce continuamente com o ângulo em relação a normal. Com isto, as áreas localizadas no centro da imagem são mais iluminadas que as na periferia da imagem.

A captura da imagem é realizada através de uma câmera monocromática CCD, padrão NTSC, com 240 linhas entrelaçadas por quadro, 30 quadros completos por segundo. Esta é uma câmera de baixo custo, padrão em aplicações de CFTV. Comum em todas as câmeras deste tipo, o dispositivo utilizado realiza o ajuste automático de brilho e contraste, facilitando significantemente a etapa de limiarização.

A Fig. 34 mostra o aparato utilizado para a aquisição da imagem. Observar que o *LED* deve ser posicionado mais distante que a câmera em relação ao objeto sob inspeção. Com isto a luz do *LED* não incide diretamente na lente da câmera.

A escolha da câmera definiu o tempo total da inspeção uma vez que esta é a etapa mais lenta da LEDges nesta arquitetura, e, as demais etapas (limiarização, geração da assinatura e comparação), são realizadas em *pipeline*. Como a imagem é entrelaçada, a câmera varre 30 quadros pares e 30 quadros ímpares por segundo. Se é possível realizar a inspeção com metade das linhas da imagem, ou seja, somente com um quadro par ou

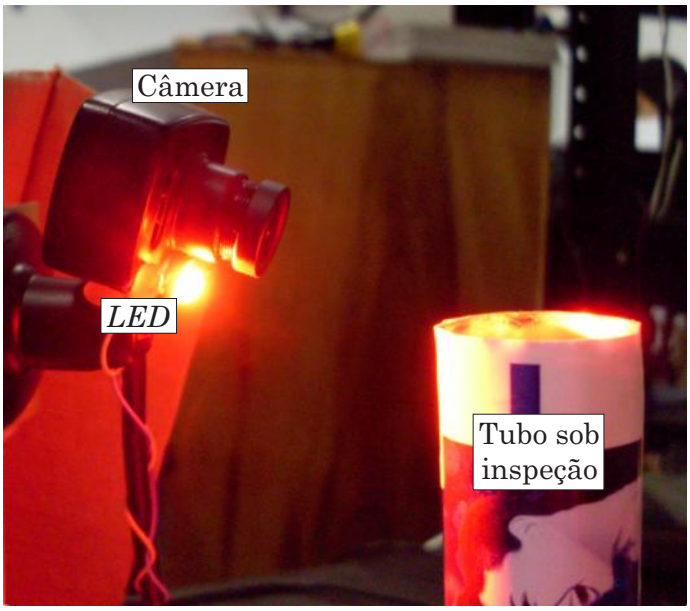


Figura 34: *Hardware* para aquisição da imagem.

somente com um quadro ímpar, teremos 60 quadros por segundo adquiridos pela câmera, sendo cada quadro com 120 linhas. Logo, o tempo para aquisição de um quadro (par ou ímpar) é de 16,66 ms, que é aproximadamente o tempo total da inspeção.

Enquanto a resolução vertical da câmera é baixa, 120 linhas por quadro (par ou ímpar), a resolução horizontal da imagem adquirida é melhor. Sabendo que o clock do processador é 60 MHz e que a câmera varre uma linha da imagem em 53 μs, a resolução horizontal é de 3179 divisões por linha.

6.2.2 Limiarização

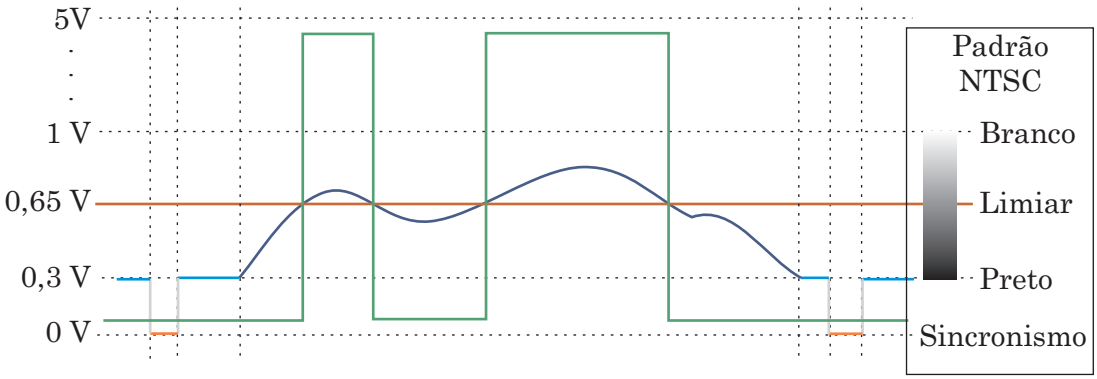
Na segmentação da imagem a partir da limiarização, foi utilizado um comparador analógico de tensão, o circuito integrado LM393, da National Semiconductor. Este CI é de baixa potência, baixo custo, alta precisão e tem o *slew rate*[1] apropriado para a limiarização.

Na etapa de limiarização o sinal elétrico (padrão NTSC) da câmera é comparado com um nível de tensão constante de referência (limiar), ajustado no início da IVA manualmente através de um resistor variável de precisão. A saída do comparador é um sinal em dois níveis, conforme apresentado na Fig. 35. Se a tensão do sinal NTSC é maior que a tensão de referência, a saída é o nível lógico "1". Caso contrário, a saída é "0".

O nível de tensão constante do limiar, utilizado como referência pelo comparador

1. Velocidade de resposta do amplificador instrumental a uma variação de tensão na entrada.

analógico, é ajustado, aproximadamente, no valor médio entre o preto (0,3 V) e o branco (1 V) do sinal NTSC, ou seja, 0,65 V. O ajuste do limiar pode ser realizado com maior precisão se a imagem limiarizada puder ser visualizada pelo operador. Para tal, foi construído um *hardware* adicional que gera um novo sinal NTSC a partir da soma ponderada do sinal limiarizado e pulsos de sincronismo. Logo, a imagem limiarizada pode ser observada em qualquer aparelho de televisão. Em laboratório, foi utilizada uma placa de captura de vídeo para visualizar esta imagem no computador, conforme apresentado na Fig. 36.



Legenda:

- Regiões não visíveis do sinal NTSC: pré-scan (5,9 µs) e post-scan (1,4 µs).
- Pulso de sincronismo de linha do sinal NTSC (4,7 µs).
- Região visível do sinal NTSC (51,5 µs)
- Tensão de limiar ajustado pelo operador
- Saída do comparador de tensão

Figura 35: Sinal adquirido padrão NTSC e saída limiarizada.

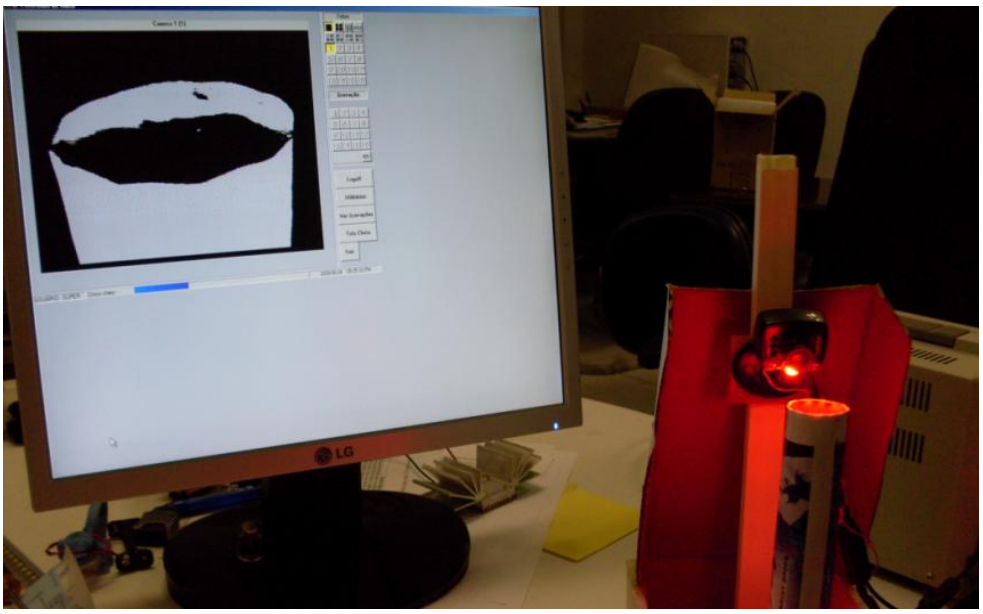


Figura 36: Imagem limiarizada observada em um PC.

6.2.3 Geração da assinatura

A geração da assinatura foi implementada no microcontrolador LPC2148, da NXP Semiconductors, baseado no processador ARM7 TDMI. A linguagem de programação utilizada foi a "C" e o código fonte consistiu em uma tradução do código elaborado na modelagem computacional executada no FreeMat. A memória de programa ocupada foi 42 KB, que representa 8,2% da memória total disponível.

O LPC2148 possui periférico de captura de tempo, fundamental para a representação da imagem que é baseada na distância entre o início de cada linha e as transições entre o preto e o branco nestas linhas. Quando é utilizada uma câmera analógica, esta distância pode ser descrita a partir do tempo de varredura da câmera entre o pulso de sincronismo de linha e a transição relevante no sinal limiarizado.

Quanto aos pulsos de sincronismo, o sinal NTSC possui três tipos: sincronismo de quadro, de linha e de cor. Para extrair esses pulsos, é utilizado um circuito integrado de baixo custo, o LM1881, da National Semiconductor. Este CI tem uma entrada, o sinal NTSC, e quatro saídas: pulso de sincronismo de quadro, sincronismo de linha, tipo do quadro (par ou ímpar) e sincronismo de cor. Estas duas últimas saídas não são utilizadas neste trabalho.

O pulso de sincronismo de quadro indica o início de um novo quadro, tanto par quanto ímpar. Em outras palavras, indica o início de uma nova inspeção, uma nova assinatura. O segundo, sincronismo de linha, indica o início de uma nova linha.

No LPC2148 foram utilizados 4 canais de captura de tempo: CAP0, CAP1, CAP2 e CAP3, todos associados com o TIMER0, de 32 bits. Os valores capturados nesses canais são armazenados nos registradores T1, T2, T3 e T4, respectivamente, fazendo referência à numeração das setas na Fig. 18. Na inicialização do programa, esses canais são configurados conforme descrito a seguir:

- Os canais CAP0 e CAP3 armazenam o valor do TIMER0 em T1 e T4, respectivamente, a cada transição do preto para o branco (borda de subida do sinal limiarizado).
- Os canais CAP1 e CAP2 armazenam o valor do TIMER0 em T2 e T3, respectivamente, a cada transição do branco para o preto (borda de descida do sinal limiarizado).

A partir de então, os quatro canais são ativados no início de cada linha. Contudo, os canais CAP0 e CAP2 são desativados assim que estes detectam as primeiras transições.

Ou seja, T1 e T3 armazenam apenas as primeiras transições do preto para o branco e do branco para o preto, respectivamente. Enquanto CAP1 e CAP3 nunca são desativados, armazenando ao final da linha, em T2 e T4, as últimas transições do preto para o branco e do branco para o preto, respectivamente.

Assim que o início de uma nova linha é detectado, é verificado pelo bloco "gerador de assinaturas", conforme Fig. 33, se os valores armazenados em T1, T2, T3 e T4 representam transições relevantes. Para LEDges aplicada a tubos, na borda superior são relevantes T1 e T2. Na borda inferior, T3 e T4 e, no corpo, T1 e T2. Se transições relevantes forem identificadas, estas são disponibilizada para a próxima etapa da LEDges: comparação. Além desta verificação, ao detectar o pulso de sincronismo de linha, os valores armazenados em T1, T2, T3 e T4 devem ser zerados para que o periférico realize novas medições de distâncias, referentes à linha atual. Observar que a verificação das regiões de interesse e transições relevantes ocorre em paralelo com a medição de tempos, realizado pelo periférico de captura de tempos.

A partir da modelagem realizada e apresentada na seção 5.2, foi observado que é possível extrair a assinatura a partir de 72 linhas da imagem, sendo 24 linhas para cada região de interesse. Porém, como a imagem é entrelaçada, o quadro enviado para o microcontrolador (par ou ímpar) tem metade das linhas. Logo, serão consideradas apenas 12 linhas para cada região de interesse da imagem: borda superior, inferior e corpo do tubo, totalizando 36 linhas para construir a assinatura. Esta pequena quantidade de linhas para formar a assinatura resulta em uma baixa resolução vertical na detecção de irregularidades nas bordas. Porém, suficiente para a aplicação de detecção de amassados em tubos de creme dental. Ao final da identificação da 36ª linha com transições relevantes, ou seja, ao final da região de corpo do tubo, a inspeção pode ser finalizada, mesmo que a câmera ainda não tenha varrido a imagem inteira.

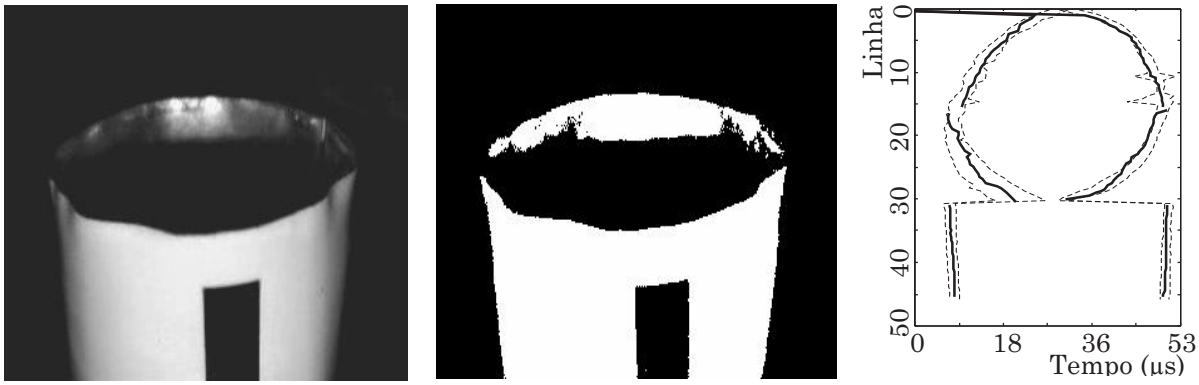
6.2.4 Comparação

O método estatístico utilizado consiste em calcular a média e o desvio padrão das assinaturas de dez objetos "bons" e utilizar estes valores como referência. A LEDges indica que devem ser armazenadas duas assinaturas na memória interna do microcontrolador: (I) média + tolerância e (II) média – tolerância, sendo a tolerância igual ao dobro do desvio padrão. Logo, como cada assinatura ocupa 72 posições de memória, são necessárias 144 posições da memória interna do microcontrolador, dentre os 8 kB de memória RAM que o LPC2148 possui. Ou seja, as assinaturas de referência ocupam apenas 7% da memória

RAM disponível no LPC2148.

Como esta etapa não é foco do trabalho, as assinaturas de referência foram calculadas no FreeMat pelo operador, enviadas para o LPC2148 via RS-232 e armazenadas no "banco de assinaturas", conforme diagrama em blocos (Fig. 33). Uma vez que as assinaturas de referência são armazenadas no microcontrolador, este pode executar a IVA de forma autônoma, sem intervenção do PC.

A etapa de comparação conseguiu classificar 100% das 55 imagens adquiridas, sendo 15 de tubos "bons", com dimensões segundo as especificações da indústria, e 40 de tubos "ruins". Algumas das imagens adquiridas e inspecionadas são apresentadas entre as Figs. 37 e 42. Além das imagens originalmente adquiridas, são apresentadas também estas limiarizadas, suas assinaturas (em linha contínua) e as assinaturas de referência (em linhas tracejadas). As assinaturas geradas foram transferidas do LPC2148 para o PC via RS-232.

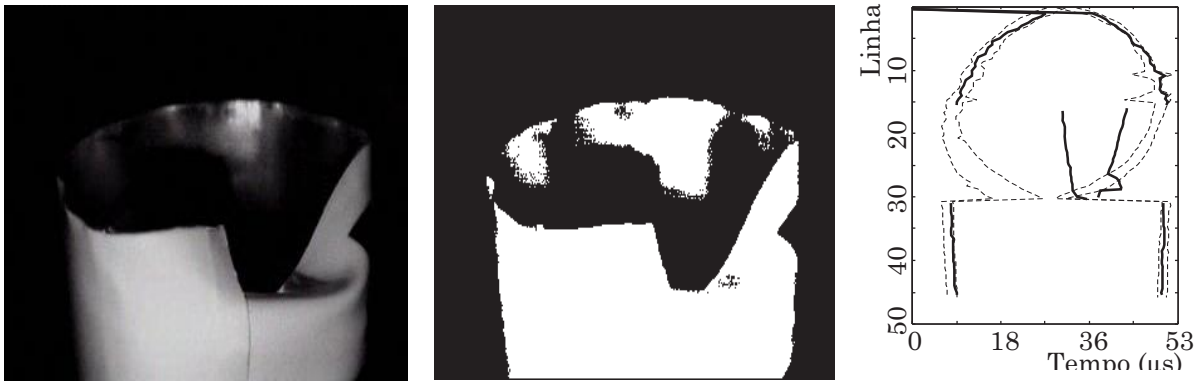


(a) Imagem adquirida. (b) Imagem limiarizada. (c) Assinatura gerada.

Figura 37: Tubo "bom" 1.

(a) Imagem adquirida. (b) Imagem limiarizada. (c) Assinatura gerada.

Figura 38: Tubo "bom" 2.

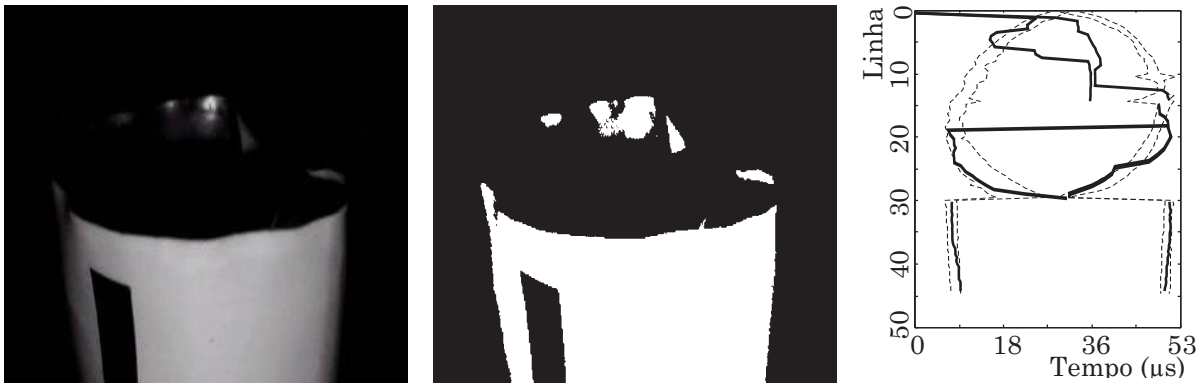


(a) Imagem adquirida. (b) Imagem limiarizada. (c) Assinatura gerada.

Figura 39: Tubo amassado 1. Imperfeição detectada na região da assinatura inferior.

(a) Imagem adquirida. (b) Imagem limiarizada. (c) Assinatura gerada.

Figura 40: Tubo amassado 2. O microcontrolador não detectou a região da borda inferior devido a impressão no tubo.



(a) Imagem adquirida. (b) Imagem limiarizada. (c) Assinatura gerada.

Figura 41: Tubo amassado 3. Observar que a impressão não modificou a assinatura da região do corpo do tubo.

(a) Imagem adquirida. (b) Imagem limiarizada. (c) Assinatura gerada.

Figura 42: Tubo amassado 4. A pequena imperfeição modifica significantemente a assinatura.

6.3 Análise do capítulo

Apesar dos componentes de baixo desempenho para aplicações de processamento de imagem, a implementação da LEDges na arquitetura baseada em câmera analógica e microcontrolador resultou em uma taxa de acerto de 100% na classificação dos tubos, atendendo a funcionalidade requerida.

O desempenho alcançado foi de 16,66 ms por inspeção, suficiente para a aplicação de detecção de irregularidades em tubos de creme dental. Quanto à resolução, a implementação apresentou baixa resolução na detecção das irregularidades, porém igualmente suficiente para a aplicação. Tanto a velocidade, quanto a resolução, são limitadas pela câmera.

A flexibilidade para atualização dos parâmetros da inspeção foi comprovada pela atualização das assinaturas de referência via RS-232, pelo ajuste do limiar pelo usuário e pela definição das áreas de interesse a partir da modelagem realizado no FreeMat. Quanto ao esforço reduzido da equipe de engenharia para o desenvolvimento da IVA, este foi comprovado pela simplicidade da tradução do programa modelado no FreeMat para a linguagem “C” utilizada no ARM7TDMI, que, inclusive, ocupou apenas 8,2% da memória disponível. Por fim, os custos reduzidos para produção foram comprovados com o uso de componentes de baixo desempenho, principalmente da câmera analógica e do microcontrolador.

7 *Arquitetura baseada em FPGA e câmera digital*

A implementação da LEDges em uma arquitetura baseada em FPGA e câmera digital visa um maior desempenho. Em comparação com o microcontrolador, a LEDges pode ser executada em imagens com maior resolução, detectando erros dimensionais menores, e em maior velocidade.

Na seção 7.1 é apresentado o *kit* FPGA utilizado. Os blocos funcionais desenvolvidos para a implementação da LEDges podem ser vistos na seção 7.2 e o estudo de caso, detecção de amassados em tubos de creme dental, na seção 7.3.

7.1 *Kit* Altera DE2-70

O *hardware* disponível no CIn / UFPE e utilizado é o *kit* de desenvolvimento DE2-70, com câmera (placa D5M) e *display* de LCD (placa LTM), da empresa taiwanesa TERASIC, conforme Fig. 43. Este *kit* está equipado com um FPGA Cyclone II 2C70, da ALTERA, que possui aproximadamente 70.000 elementos lógicos (LEs) e uma ampla memória, sendo adequado para aplicações multimídia. Adicionalmente, a Altera disponibiliza uma vasta documentação para a placa DE2-70, incluindo exemplos de uso da câmera e do *display* que foram utilizados como ponto de partida para a implementação da LEDges.

O *kit* é composto pelos componentes de *hardware* listados a seguir, sendo os destacados em negrito utilizados na implementação da LEDges.

- **FPGA Altera Cyclone® II 2C70**;
- **Osciladores de 50 MHz e 28,63 MHz para fonte de *clock***;
- **Câmera Digital RGGB, CMOS, de 5 Mpixels (placa D5M)**;
- ***Display* de LCD RGB, touchscreen, de 4,3 polegadas (placa LTM)**.

- **Dispositvo para programação USB com JTAG — USB Blaster**;
- **Botões do tipo "pushbutton", 4 unidades — KEY0 a KEY3**;
- **Chaves do tipo "toggle", 18 unidades — SW0 a SW17**;
- ***LEDs* vermelhos, 18 unidades — LEDR0 a LEDR17**;
- ***LEDs* verdes, 8 unidades — LEDG0 a LEDG7**;
- ***Displays* de 7 segmentos, 8 unidades**;
- Dispositivo para programação serial RS-232 — EPCS16;
- Memória SSRAM com 2 MB;
- Memória SDRAM com 64 MB;
- Memória Flash com 8 MB;
- Entrada para cartão de memória SD;
- CODEC de áudio com 24 bits e conectores para entrada e saída do sinal;
- Conversor analógico-digital para aplicações de vídeo, VGA;
- Decodificador de TV (NTSC/PAL/SECAM) e conector para entrada do sinal de vídeo;
- Controlador para Ethernet com conector;
- Controlador USB com conectores;
- *Driver* RS-232 com conector DB-9;
- Conector para teclado ou *mouse* PS/2;
- *Driver* para IrDA, transmissão óptica via infravermelho;
- Conector SMA;

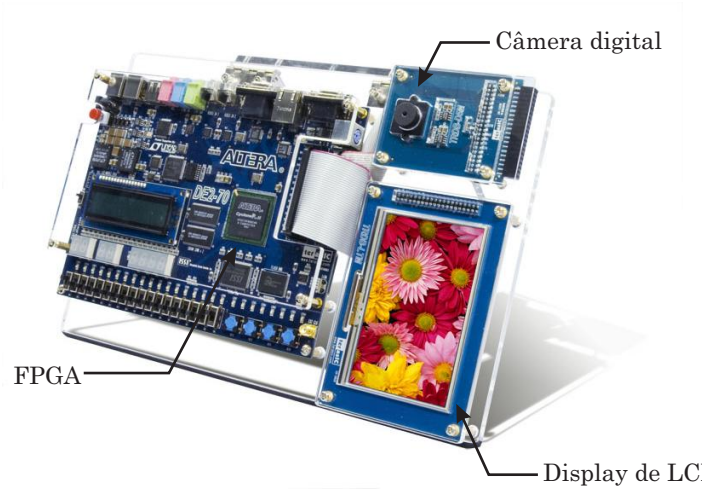


Figura 43: *Kit* de desenvolvimento Altera DE2-70 para processamento de imagens.

Quanto ao FPGA, que é o sistema computacional principal, o Cyclone II EP2C70F896C6 possui as características listadas a seguir.

- 68.416 LEs;
- 1.152.000 bits de memória RAM;
- 50 multiplicadores embarcados;
- 4 PLLs;
- 622 pinos de E/S (I/O) disponíveis para o usuário.

Além dos componentes de *hardware* disponíveis no *kit*, foi necessário apenas incluir um *LED* para executar a iluminação necessária da LEDges.

7.1.1 *Design* inicial para aplicações com câmera e *display* de LCD

Visando reduzir o tempo de desenvolvimento do sistema de IVA, foi utilizado um *design* fornecido com o *kit* da Terasic. Devido a restrições no tempo de desenvolvimento, a demonstração utiliza recursos que não serão utilizados na validação da LEDges, como a transferência da imagem capturada via conector VGA e para o computador via USB.

O sistema de aquisição e visualização de imagens, fornecido pela Terasic, possui as configurações e funcionalidades listadas a seguir, das quais somente os dois primeiros itens, em negrito, foram mantidos na implementação da LEDges.

- **Pressionando o botão KEY0 da placa DE2-70, reinicializa-se o circuito**;
- **A chave SW0 e o botão KEY1 ajustam o tempo de exposição da câmera. Quando SW[0] está desligada, o tempo de exposição é aumentado a cada acionamento de KEY1. De modo contrário, se SW0 está ligada, o tempo de exposição é reduzido a cada acionamento de KEY1**;
- Ao pressionar o botão KEY3 o sistema exibe no *display* de LCD, em constante atualização, as imagens capturadas pela câmera;
- Ao pressionar o botão KEY2 o sistema exibe no *display* de LCD a última imagem capturada pela câmera, ou seja, tira uma foto;
- A chave SW16 liga e desliga o *zoom*. Se modificada esta chave, deve-se pressionar seqüencialmente os botões KEY0 e KEY3 para que a modificação tenha efeito.
- Os *displays* de 7 segmentos apresentam a contagem de quadros capturados;
- Os LEDs verdes indicam uma contagem binária das linhas varridas pela câmera, reiniciando a contagem no início de cada quadro.

Para implementar as funcionalidades descritas, a Terasic implementou 6 módulos funcionais principais conforme enumerados a seguir, dos quais os três primeiros, em negrito,

são modificados na implementação da LEDges.

1. **Módulo de captura dos dados da câmera (placa D5M). Além das intensidades dos *pixels*, recebe os sinais de quadro válido, linha válida e *pixel clock*. Este módulo também fornece para a câmera o sinal de *clock* principal**;
2. **Módulo de configuração do sensor via protocolo de comunicação I^2C, composto pelo *driver* I^2C e uma lista contendo os endereços e valores dos registrados que devem ser atualizados na reinicialização do sistema**;
3. **Conversor do padrão Bayer fornecido pela câmera, também conhecido como RGGB, de 48 bits, para o padrão RGB, de 30 bits**;
4. Controlador da memória SDRAM. Este módulo armazena as imagens adquiridas na memória SDRAM e recupera estas para exibição no *display* de LCD;
5. *Driver* do *display* LCD, que envia o sinal de *clock* e os valores dos *pixels* para a placa LTM;
6. Módulo de configuração do *display* de LCD via o protocolo de comunicação SPI.

Um detalhamento destes módulos será apresentado na próxima seção.

O diagrama em blocos simplificado do *design* pode ser observado na Fig. 44.

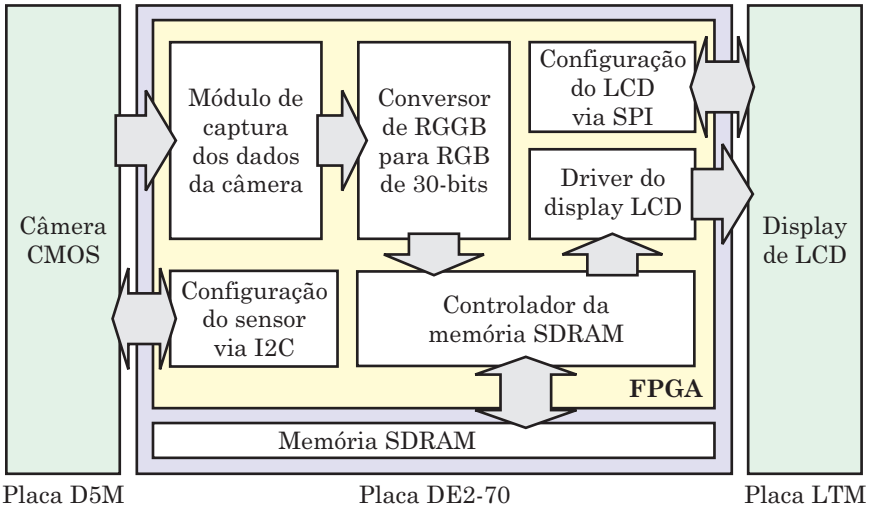


Figura 44: Diagrama em blocos do exemplo.

7.2 Blocos funcionais da implementação da LEDges em FPGA

O sistema de IVA foi segmentado em 6 grandes blocos funcionais, conforme apresentado a seguir, os quais são compostos pelos módulos descritos na seção anterior, com algumas modificações, e por módulos novos, visando atender a LEDges.

1. Aquisição da imagem;
2. Limiarização;
3. Geração da assinatura;
4. Comparação;
5. Visualização no LCD;
6. Módulo de *reset*.

7.2.1 Aquisição da imagem

Este bloco é composto por três módulos:

1. Captura dos dados da câmera;
2. Conversor de RGGB para RGB 30 bits;
3. Configuração do sensor via I^2C.

Captura dos dados da câmera

Este módulo é responsável por realizar a interface de dados com o sensor CMOS localizado na placa D5M. A câmera disponibiliza os *pixels* no padrão Bayer, que consiste em 4 cores: vermelho (R), verde1 (G1), verde2 (G2), azul (B). Este padrão também é conhecido por RGGB. O padrão Bayer imita a resposta ao espectro visível do olho humano ao ser mais sensível à cor verde em relação às cores azul e vermelho. A intensidade de cada cor é disponibilizada em 12 bits, totalizando 48 bits por *pixel*, e estes bits são enviados em 4 pulsos de *clock* principal através de um barramento de 12 bits. O módulo de captura dos dados da câmera recebe estes 4 conjuntos de bits e disponibiliza em paralelo todos os 48 bits recebidos para a próxima etapa do processamento.

Além dos valores de R, G1, G2 e B, a câmera envia também os sinais de quadro válido (fval) e de linha válida (lval). Estes são fundamentais uma vez que a câmera digital, assim

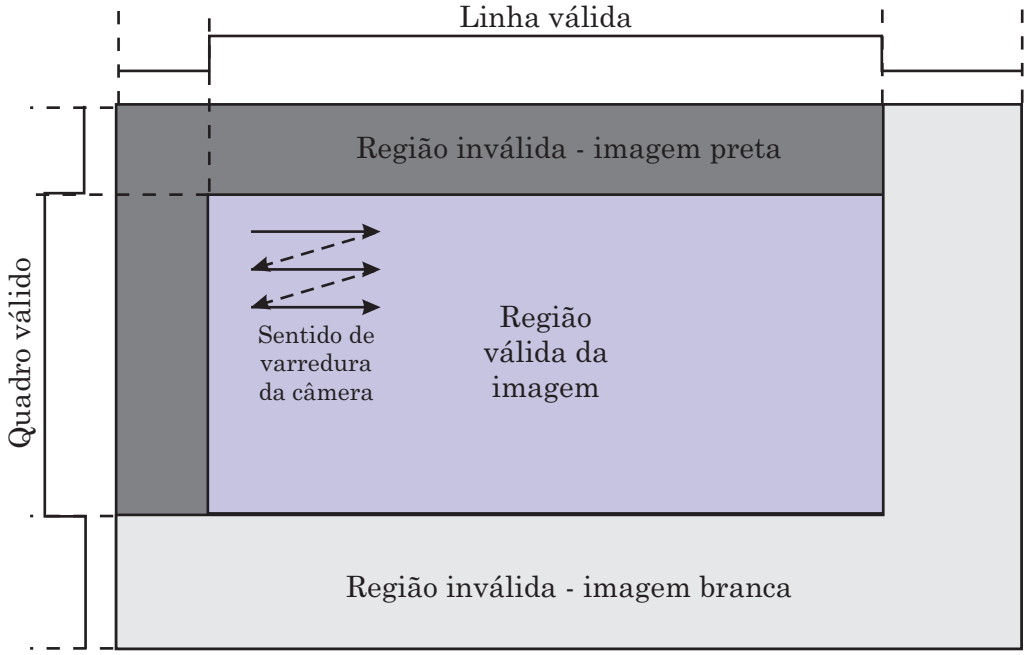


Figura 45: Região válida da imagem e sinais de linha e quadro válidos.

como a analógica, varre regiões em que não há imagem válida, conforme apresentado na Fig. 45.

A partir de uma operação lógica "E" entre quadro válido e linha válida, é obtido e disponibilizado para a próxima etapa o sinal dado válido (dval), conforme Fig. 46. O sinal dado válido indica os momentos nos quais o *pixel* disponibilizado na saída do módulo de captura contém informações da imagem válida.

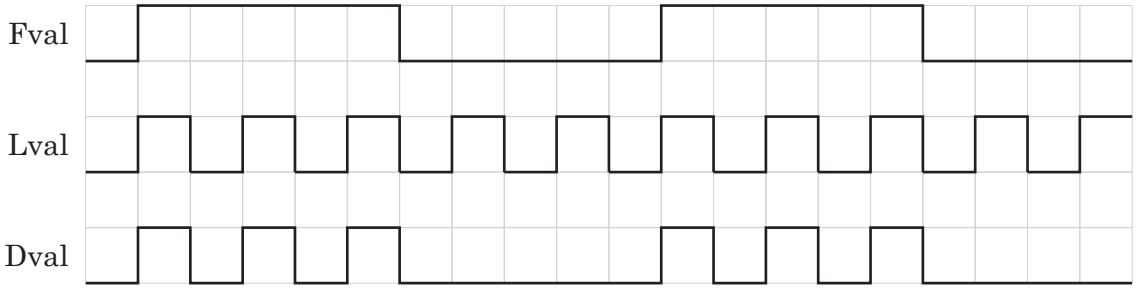


Figura 46: Sinais fval, lval e dval.

Por fim, a câmera disponibiliza o sinal *pixel clock* que alterna de estado sempre que um *pixel* é disponibilizado. Este sinal é a base de tempo utilizada em todas as demais etapas de processamento dos *pixels*.

Conversor de RGGB para RGB 30 bits

Este módulo converte o padrão Bayer, RGGB, para o padrão RGB, que é utilizado pelo *display* de LCD para exibição das imagens. Neste módulo também é reduzida a largura

de bits necessária para representar um *pixel*, de 48 bits disponibilizados pelo módulo de captura para 30 bits, sendo 10 bits para representar a cor vermelha (Componente_R), 10 para a cor verde (Componente_G) e 10 para o azul (Componente_B). A escolha de 30 bits otimiza o uso da memória SDRAM para armazenar a imagem uma vez que cada posição da memória tem 32 bits.

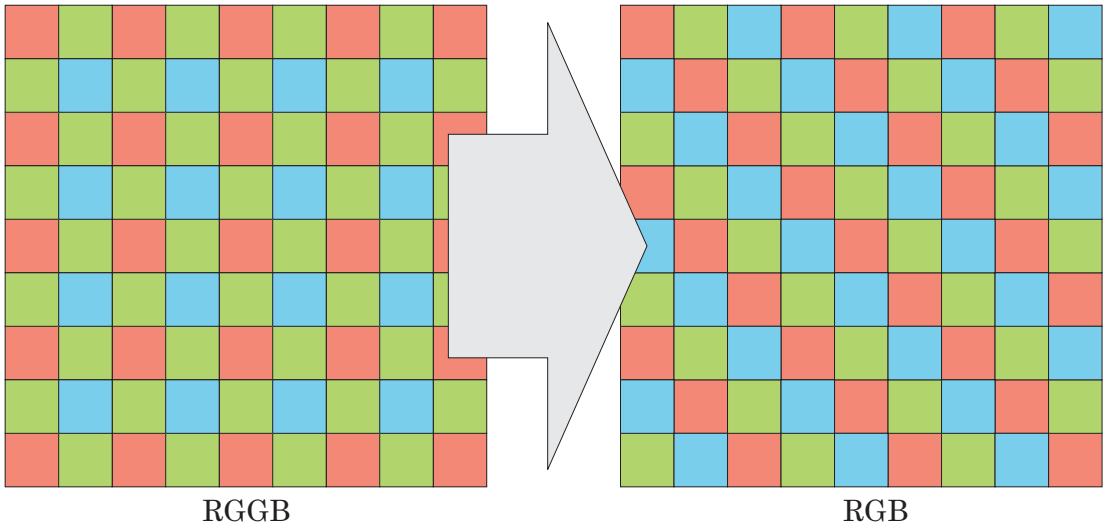


Figura 47: Conversão do padrão Bayer (RGGB) para RGB.

Além dos 30 bits em paralelo, este módulo ainda disponibiliza a contagem de linhas (XCont) e de colunas (YCont). Na configuração realizada, resolução de 640 x 480 pixels, XCont varia de 0 à 639 e YCont, de 0 à 479. Inclusive os bits YCont[8:1], bits mais significativos na contagem das linhas, estão associados aos *LEDs* verdes LEDG[7:0].

Configuração do sensor via I^2C

O módulo de configuração do sensor CMOS é composto por dois sub-módulos:

1. Armazenamento dos valores de referência para configuração dos registradores do sensor;
2. Interface de comunicação I^2C.

O primeiro sub-módulo é responsável pelo armazenamento de todos os valores que serão transferidos para os registradores do sensor CMOS. No total são 42 registradores que precisam ser configurados na inicialização do sistema de IVA. Dentre estes, alguns podem ser alterados pelo usuário.

A chave SW0 e o botão KEY1 ajustam o tempo de exposição da câmera, ou seja, altera o valor do registrador 0x00C. Quando SW0 está desligada, o tempo de exposição

é aumentado a cada acionamento de KEY1. De modo contrário, se SW0 está ligada, o tempo de exposição é reduzido a cada acionamento de KEY1.

A chave SW3 habilita a geração de padrões pela câmera. Foi configurado através dos registradores de 0x0A0 a 0x0A4 que, ao ligar a chave SW3, a câmera transmite um padrão de barras verticais em dois níveis de cinza, conforme Fig. 48. Na figura, as barras estão na horizontal por que a placa LTM, na qual está o *display*, é rotacionada em 90°em relação à base da placa DE2-70. Este modo de operação foi especialmente útil no início da implementação da LEDges no teste da etapa de limiarização e na detecção das transições relevantes.

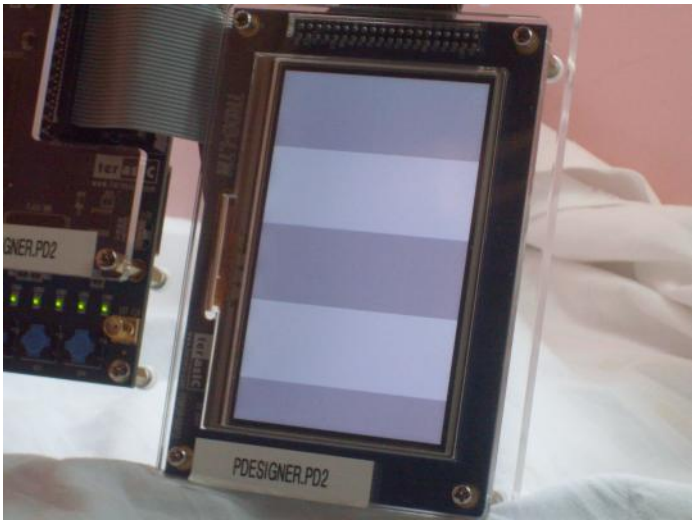


Figura 48: Padrão de barras verticais gerado pela câmera e exibido no *display* de LCD do *kit*.

O segundo sub-módulo, a interface de comunicação I^2C, é responsável pela transmissão dos dados entre o FPGA e o sensor CMOS, considerando todas as especificações do protocolo de comunicação I^2C. Este módulo tem uma porta de saída e uma porta bidirecional para a câmera: o pino de *clock* (SCLK) e o de dados (SDAT), respectivamente.

7.2.2 Limiarização

Este módulo é um simples comparador digital, sincrono com o *pixel clock*, com duas entradas de 8 bits. Uma entrada recebe os 8 bits mais significativos do sinal Componente_R e a outra, o limiar. Somente a componente vermelha do sinal RGB é utilizada na limiarização por que esta é a que tem melhor sensibilidade para a detecção da reflexão, pelo objeto sob inspeção, da luz vermelha, emitida pelo *LED*.

O valor do limiar utilizado para a comparação pode ser simplesmente o valor ajustado

pelo usuário em SW[17:10], se SW9 estiver ligada, ou pode ser definido automaticamente pelo FPGA. Se SW9 estiver desligada, o FPGA tentará, através do ajuste automático do limiar, manter o número de *pixels* brancos igual ao valor ajustado em SW[17:10]. Na prática, se o ambiente da IVA ficar "mais claro" ou "mais escuro", o limiar é ajustado para que a imagem limiarizada permaneça inalterada.

Os valores de limiar podem ser observados no *display* de 7 segmentos da placa DE2-70. Nos *displays* 7-seg[7:4] é apresentado o valor ajustado nas chaves SW[17:10]. Nos *displays* 7-seg[3:0] é apresentado o valor definido automaticamente pelo FPGA como limiar. Observar que este valor só terá sentido se o modo automático estiver habilitado.

A saída deste módulo é um sinal binário de 1 bit, o *pixel* limiarizado. Se o sinal Componente_R for maior que o limiar, o *pixel* limiarizado é "1". Caso contrário, "0".

7.2.3 Geração da assinatura

Este bloco é composto por dois módulos:

1. Detecção de transições relevantes;
2. Composição da assinatura do objeto a partir da identificação das regiões de interesse.

Este bloco é o único em que não há a interface direta com o usuário.

Detecção de transições relevantes

Este módulo tem apenas três entradas: o *pixel* limiarizado, *o pixel clock* e o linha válida (lval). Quando o sinal linha válida é habilitado, o FPGA começa a contar o número de "*pixel clock*" que ocorre entre o início da linha, marcado pela borda de subida de lval, e as transições relevantes. Logo, para todas as linhas válidas, são gerados 4 dados:

- Assinatura_1: Número de "*pixel clock*" entre o início da linha e a primeira transição do preto para o branco;
- Assinatura_2: Número de "*pixel clock*" entre o início da linha e a última transição do branco para o preto;
- Assinatura_3: Número de "*pixel clock*" entre o início da linha e a primeira transição do branco para o preto;
- Assinatura_4: Número de "*pixel clock*" entre o início da linha e a última transição do preto para o branco.

Como o valor máximo de uma assinatura é 479, devido à largura da imagem em *pixels*, cada assinatura necessita de um registrador com largura de 9 bits para ser armazenada.

Como saída, este módulo disponibiliza em paralelo as 4 assinaturas geradas e duas saídas independentes que indicam com um pulso quando ocorre uma transição positiva (do preto para o branco) ou uma transição negativa (do branco para o preto). Estes pulsos são utilizados somente para exibir no LCD os pontos em que o FPGA detectou as transições.

Composição da assinatura do objeto a partir da identificação das regiões de interesse

Este módulo tem como entradas os sinais de assinatura_1, assinatura_2, assinatura_3 e assinatura_4, lval, YCont e *pixel clock*.

De acordo com a técnica proposta, a assinatura final é composta pelas assinaturas esquerda e direita. E estas duas são formadas no final da varredura das linhas, na borda negativa de lval, e são compostas pelas assinaturas 1 e 2, para a borda superior e corpo do tubo, e assinaturas 3 e 4 para a borda inferior.

Como primeira atividade do módulo, é necessário detectar as regiões de interesse: borda superior, borda inferior e corpo do tubo. Exatamente como descrito pela técnica, a borda superior é definida como as primeiras linhas adquiridas após a identificação do primeiro *pixel* branco. A borda inferior, foi definida pelas últimas linhas adquiridas enquanto houver mais de duas transições detectadas para cada linha. Por fim, o corpo foi definido como as primeiras linhas da imagem imediatamente após a região da borda inferior. A contagem destas linhas é auxiliada por YCont.

O resultado deste módulo são dois sinais de assinaturas: a direita e a esquerda, e como esperado, ambas com 9 bits. Além da assinatura do objeto referente a uma linha, é disponibilizado um sinal binário de assinatura válida (sigval). É enviado um pulso por esta via ao final das linhas pertencentes às regiões de interesse, ou seja, quando novas assinaturas forem disponibilizadas.

7.2.4 Comparação

O módulo de comparação é responsável por comparar a assinatura gerada com as assinaturas de referência, armazenadas na memória do FPGA. Como esta etapa não é o foco da LEDges, o método de comparação foi simplificado. As assinaturas de referência são armazenadas quando a chave SW4 está ligada. Posiciona-se um objeto “bom” para

ter sua imagem adquirida e liga-se a chave SW4. A partir deste momento, desliga-se a chave SW4, e todas as demais assinaturas geradas serão comparadas com as de referência, geradas a partir de um objeto "bom".

Com somente uma assinatura como base não é possível obter o desvio padrão. Logo, foi definido um valor fixo como margem de tolerância, ajustado a partir de observações experimentais. Os valores das assinaturas de referência armazenadas foram: (I) assinatura base + tolerância e (II) assinatura base – tolerância. Se a assinatura do objeto sob inspeção estiver completamente contida entre as assinaturas (I) e (II), o objeto sob inspeção é classificado como "bom". Se existir qualquer ponto das assinaturas fora da faixa definida, o objeto é classificado como "ruim".

Este módulo tem somente uma saída: o resultado final da inspeção. Este está conectado ao *LED* verde LEDG8. Se o *LED* acender durante a IVA significa que o objeto sob inspeção está de acordo com as especificações da indústria.

7.2.5 Visualização no LCD

O bloco de visualização no LCD é composto por 5 módulos:

1. Seletor;
2. Controlador da memória SDRAM_1;
3. Controlador da memória SDRAM_2;
4. Interface de dados com o LCD;
5. Configuração do LCD via SPI.

A implementação da LEDges somente alterou os valores que são armazenados na memória SDRAM em relação ao *design* original fornecido pela Terasic. Para tal, foi incluido o módulo seletor e nenhuma linha de código contida nos quatro módulos seguintes foi modificada.

Na placa DE2-70 existem dois *chips* de memória SDRAM. Cada *chip* tem uma largura de 16 bits. Logo, um banco de memória com os dois em paralelo tem 32 bits de largura, que é suficiente para armazenar os 30 bits de um *pixel* por endereço. O módulo SDRAM_1 armazena os 10 bits da componente azul e 5 bits da componente verde. O módulo SDRAM_2 armazena os 5 bits restantes da componente verde e 10 bits da componente vermelha.

A implementação da LEDges incluiu o módulo seletor que possui duas chaves de interface com o usuário, a SW1 e SW2, que permitem modificar as informações exibidas

no *display*, a partir do chaveamento dos dados que são enviados para o banco de memória SDRAM, de acordo com a tabela 2.

Tabela 2: Modos de exibição no *display* de LCD.

SW1	**SW2**	**Informação exibida**
Desligado	Desligado	Imagem original com as 3 componentes de cores
Ligado	Desligado	Imagem original, apenas com a componente vermelha de cor, sobreposto a identificação dos pontos de transição entre o branco e o preto.
Ligado	Ligado	Imagem limiarizada exibida na cor vermelha, sobreposto a identificação dos pontos de transição entre o branco e o preto.

Os três modos de exibição são exemplificados na Fig. 49.

(a) Imagem original com as 3 componentes de cores.

(b) Imagem original, apenas com a componente vermelha exibida no LCD.

(c) Imagem limiarizada na cor vermelha e identificação dos pontos de transição.

Figura 49: Modos de exibição no LCD. Exemplo de exibição de um tubo com o *LED* desligado.

7.2.6 Módulo de *reset*

Este módulo também não foi modificado na implementação da LEDges. O módulo de *reset* é responsável por reiniciar os demais módulos na seqüência adequada. Ao pressionar o botão KEY0, são gerados três pulsos de *reset* seqüênciais:

- DLY_RST_0 — Reinicia os módulos de SDRAM;
- DLY_RST_1 — Reinicia os módulos conversor de RGGB para RGB 30 bits e configuração do sensor via I^2C;
- DLY_RST_2 — Reinicia o módulo de captura dos dados da câmera.

7.2.7 Interface com o usuário

Na tabela 3 é apresentado um resumo da interface entre a LEDges implementada em FPGA e o usuário.

Tabela 3: Interface da LEDges com o usuário.

Chave	**Funcionalidade**
Aquisição da imagem	
KEY1	Incrementa ou decrementa o tempo de exposição
SW0	Seleciona entre incrementar ou decrementar o tempo
SW3	Gera um padrão para testes
LEDG[7:0]	Apresenta contagem das linhas varridas
Limiarização	
SW[17:10]	Define o valor do limiar
SW9	Habilita a definição automática do limiar
Disp. 7-seg [3:0]	Exibe o valor do limiar ajustado manualmente
Disp. 7-seg [7:4]	Exibe o valor do limiar ajustado automaticamente
Comparação	
SW4	Armazenar a assinatura padrão
LEDG8	Indica o resultado final da IVA
Visualização no LCD	
SW1	Seleciona entre apresentar as intensidades adquiridas para as componentes verde e azul da imagem ou apresentar os pontos aonde foram detectadas transições entre o preto e o branco na imagem
SW2	Seleciona entre apresentar a intensidade adquirida para a componente vermelha da imagem ou apresentar a imagem limiarizada na cor vermelha
Módulo de *reset*	
KEY0	Reinicia a IVA

E na Fig. 50, a imagem da placa DE2-70 com o posicionamento dos botões, chaves, LEDs e *displays*.

7.2.8 Diagrama em blocos

Na Fig. 51 é apresentado um diagrama em blocos. As cores de preenchimento nas caixas de texto se referem à etapa de processamento da LEDges, conforme legenda.

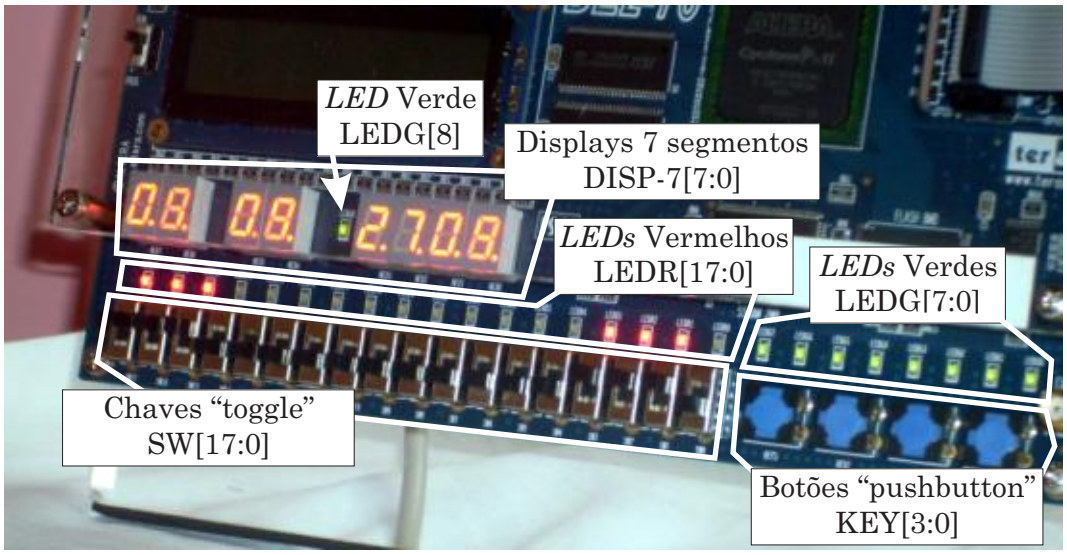


Figura 50: Interface com o usuário da placa DE2-70.

7.3 Estudo de caso: detecção de amassados em tubos de creme dental

Assim como na arquitetura baseada em microcontrolador e câmera analógica, a LEDges foi implementada em FPGA e câmera digital, e aplicada ao mesmo problema industrial real, onde defeitos foram detectados em bordas de tubos de creme dental.

Nas próximas seções, esta aplicação será descrita, bem como os componentes de *hardware* utilizados para a implementação da LEDges nesta arquitetura.

7.3.1 Aquisição da imagem

Os parâmetros ajustados e testados na etapa de aquisição da imagem foram:

1. Iluminação com o *LED*: ligado e desligado;
2. Componentes de cores a serem adquiridas: RGB e somente R;
3. Intensidade da iluminação ambiente: 500 lux e 150 lux;
4. Tempo de exposição da câmera: curto e longo.

O primeiro item está relacionado com a iluminação estruturada proposta pela LEDges. Na Fig. 52 são apresentadas as imagem adquiridas, com o *LED* desligado e ligado. O LED é o mesmo utilizado na arquitetura com microcontrolador: *LED* vermelho 1W. Ambas as imagens foram adquiridas com as componentes RGB, iluminação ambiente de 750 lux e longo tempo de exposição da câmera.

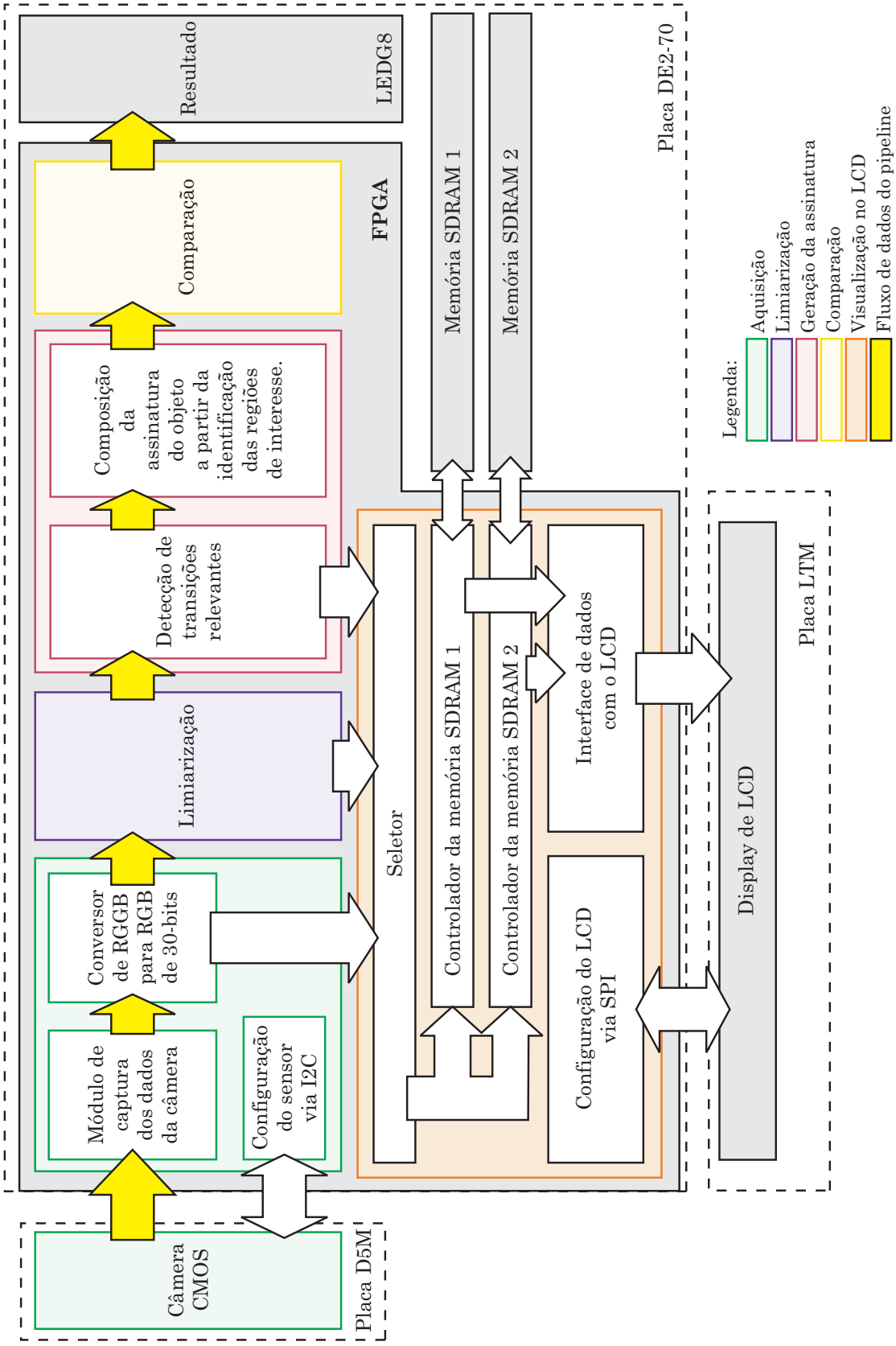


Figura 51: Diagrama em blocos.

(a) *LED* desligado. (b) *LED* ligado.

Figura 52: Influência da iluminação com *LED* proposta na LEDges na aquisição da imagem.

Observar que o objeto em primeiro plano se destaca em relação ao plano de fundo quando o *LED* está acionado. Porém, ao contrário da câmera analógica utilizada na arquitetura baseada em microcontrolador, a câmera digital não tem ajuste automático de brilho e contraste, ficando ainda presente na imagem os elementos iluminados no plano de fundo.

O segundo item a ser ajustado é a componente de cor que será adquirida. Conforme descrito pela LEDges, é desejável que somente a luz do *LED* refletida no objeto sob inspeção seja capturada pela câmera, ou seja, para que a aquisição da imagem seja realizada somente no comprimento de onda (cor) do *LED*. Com isto é possível reduzir a intensidade do plano de fundo e sombras em relação às áreas iluminadas com o *LED* no primeiro plano. Na arquitetura baseada na câmera analógica, a seleção da componente de cor para a aquisição não é possível por que a câmera é monocromática. Porém, com a câmera digital colorida é possível adquirir somente uma componente de cor da imagem. Logo, se o LED utilizado é vermelho, a imagem adquirida deverá conter somente a componente de cor vermelha (R). Na Fig. 53 é apresentado um tubo nas componentes RGB e somente com a componente R. Ambas as imagens foram adquiridas com o LED ligado, iluminação ambiente de 750 lux e longo tempo de exposição da câmera.

Observar que o tubo está com sua imagem completamente saturada na componente R. Esta saturação facilita significantemente a segmentação a partir da limiarização do histograma.

O terceiro parâmetro testado foi a iluminação ambiente. Para tal, foi utilizado um luxímetro profissional e observada a norma NBR5413. Foram testados dois casos. O primeiro, realizar a IVA em um ambiente com iluminação apropriada para tarefas industriais,

(a) Componentes RGB (b) Somente componente R

Figura 53: Aquisição da imagem com as componentes RGB e somente R.

trabalho de maquinaria médio, ambientes de escritório e tarefas com requisitos visuais normais. A intensidade luminosa média no ambiente nestes casos é de 750 lux. O segundo caso, em ambentes de trabalho bruto de maquinaria. A intensidade luminosa média neste caso é de 300 lux. A Fig. 54 apresenta duas aquisições da imagem de um tubo nas duas condições de iluminação citadas. Ambas as imagens foram adquiridas com o LED ligado, somente com a componente R e longo tempo de exposição da câmera.

(a) Iluminação ambiente de 750 lux. (b) Iluminação ambiente de 300 lux.

Figura 54: Influência da iluminação ambiente na imagem adquirida.

Observar que os objetos do plano de fundo são capturados pela câmera com maior intensidade no ambiente mais iluminado.

Por fim, foi analisada a influência do tempo de exposição na imagem adquirida. Este parâmetro influencia significantemente o tempo total da IVA uma vez que este define o "*pixel clock*". Quanto mais rápida for a exposição, menos luz será captada pelo sensor. Na Fig. 55 são apresentadas duas imagens adquiridas com diferente tempo de exposição. A primeira, adquirida com um tempo de exposição mais longo, resultando em uma taxa

de 50 aquisições por segundo. A segunda, com um tempo de exposição mais curto, resulta em uma taxa de 100 aquisições por segundo. Sendo o limite, obter uma taxa de aquisição de 150 quadros por segundo na resolução de 640 x 480 pixels. Ambas foram adquiridas com o LED ligado, somente com a componente R e iluminação ambiente de 750 lux.

(a) Longo tempo de exposição. (b) Curto tempo de exposição.

Figura 55: Influência do ajuste do tempo de exposição na aquisição da imagem.

Observar que para compensar a curta janela de tempo para entrada de luz, é necessário iluminar vigorosamente o objeto sob inspeção. Já o plano de fundo, que não é iluminado fortemente, não terá sua imagem capturada com alta intensidade.

Dois parâmetros de interesse não foram ajustados. O ajuste da resolução da imagem iria modificar o espaço necessário para armazenamento da imagem na memória SDRAM, sendo necessários ajustes no controlador de memória. Além disto, seria necessário ajustar todo o bloco de visualização no *display* de LCD. O segundo parâmetro é o foco da câmera. Neste caso, a dificuldade esteve na montagem mecânica e disponibilização das lentes necessárias para ajuste do foco. Os ajustes destes parâmetros permitiriam a identificação de irregularidades com dimensões menores e não foram realizados por restrição no tempo de implementação da LEDges e por não serem imprescindíveis para o estágio atual deste trabalho.

7.3.2 Limiarização

A limiarização foi testada em duas situações: com o plano de fundo distante e com o plano de fundo próximo. Todos os casos ocorreram com iluminação ambiente de 750 lux, limiarizando a componente R da imagem e longo tempo de exposição da câmera.

Nas Figs. 56 e 57 são apresentadas as limiarizações com o plano de fundo distante (aproximadamente a 2,5 m do tubo sob inspeção). A Fig. 56 mostra a limiarização com o

LED desligado enquanto a Fig. 57, com o *LED* ligado.

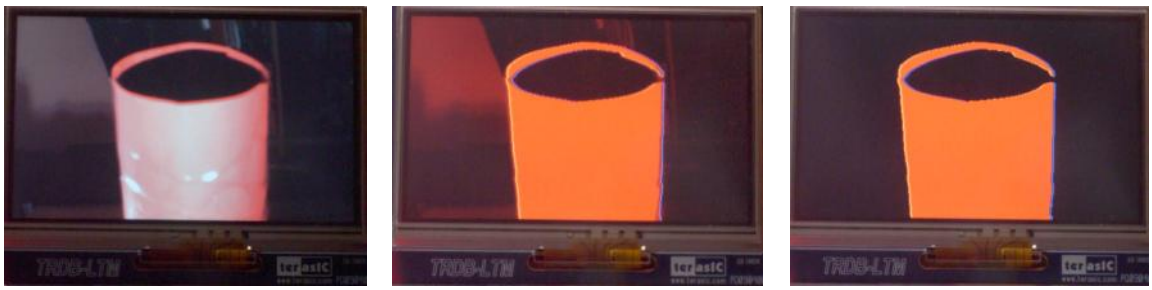
(a) Imagem adquirida — Componentes RGB.

(b) Imagem adquirida — Componente R.

(c) Imagem limiarizada.

Figura 56: Limiarização com o *LED* desligado.

As bordas ressaltadas com riscos verdes e azuis na Fig. 56(b) são devidas à chave SW1 ligada e ao limiar ajustado. Ou seja, as transições na imagem limiarizada são detectadas (apesar da imagem limiarizada não estar exibida no *display*) e, estas transições, apresentadas no *display* de LCD. Este modo de operação, onde se visualiza no LCD a componente R e as bordas detectadas, é útil para ajustar o valor do limiar.



(a) Imagem adquirida — Componentes RGB.

(b) Imagem adquirida — Componente R.

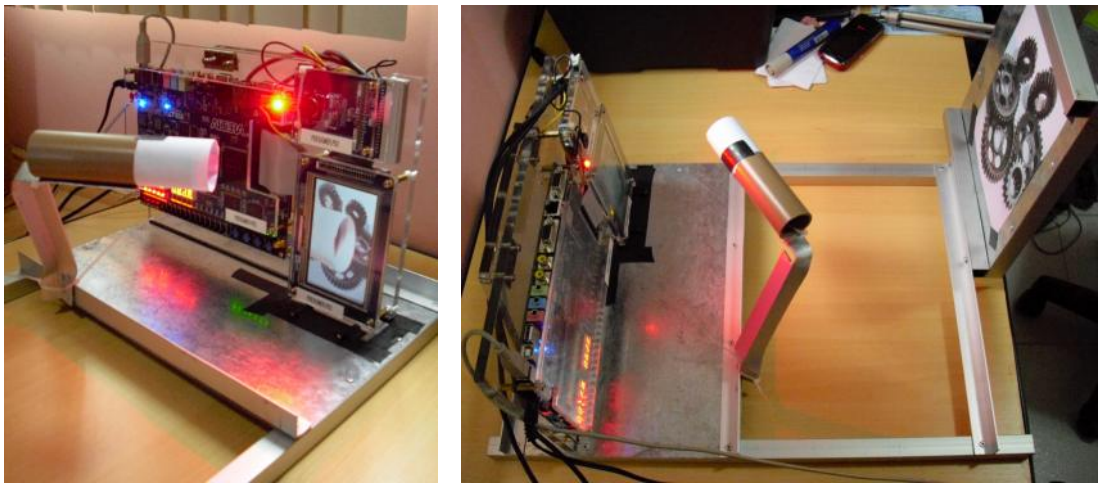
(c) Imagem limiarizada.

Figura 57: Limiarização com o *LED* ligado.

Ao comparar as imagens limiarizadas das Figs. 56(c) e 57(c) é possível constatar a importância da iluminação com *LED* para a LEDges. Não é possível separar pelo histograma o plano de fundo e sombras da área iluminada do tubo quando o *LED* está desligado.

Para realizar os testes na segunda situação, com o plano de fundo próximo, foi construído um suporte mecânico, em alumínio e zinco, para o *kit*, para o tubo e para o plano de fundo. No suporte para o plano de fundo, a aproximadamente 30 cm do tubo sob inspeção, foi fixada uma impressão de engrenagens em preto e branco, ou seja, com áreas de alta absorção (cor preta) e alta reflexão (branco do papel) da luz. Esta é uma das piores situações que pode ser encontrada na indústria. A Fig. 58 mostra o aparato construído.

Novamente, a limiarização foi testada com o LED desligado, conforme a Fig. 59, e



(a) Placa DE2-70 com a câmera e LCD.

(b) Plano de fundo impresso.

Figura 58: Aparato de aquisição para testes de limiarização.

com o LED ligado, mostrado na Fig. 60.

(a) Imagem adquirida.

(b) Exibição da componente R.

(c) Imagem limiarizada.

Figura 59: Limiarização a partir da imagem adquirida com o *LED* desligado.

(a) Imagem adquirida.

(b) Exibição da componente R.

(c) Imagem limiarizada.

Figura 60: Limiarização a partir da imagem adquirida com o *LED* ligado.

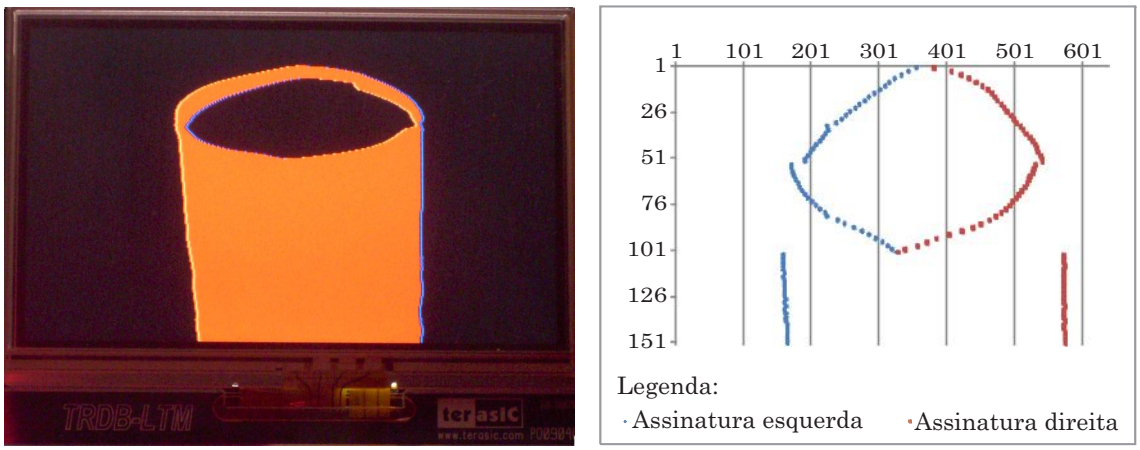
Foi observado que, com o plano de fundo próximo, é possível obter um resultado semelhante ao obtido quando o plano de fundo está distante. Porém, o ajuste do limiar tem que ser realizado com muito mais precisão uma vez que o plano de fundo é adquirido

com uma intensidade superior (observar a diferença de intensidade do plano de fundo apresentado nas Figs. 57(b) e 60(b)). Neste caso, o ajuste automático do limiar resolve o problema do ajuste preciso, inclusive mantendo o sistema invariante a modificações lentas na iluminação ambiente.

7.3.3 Geração da assinatura

A partir de observações experimentais das imagens adquiridas pela câmera CMOS da placa D5M e limiarizadas, foi definido que cada região de interesse para a imagem do tubo é composta por 50 linhas da imagem. Logo, a assinatura completa é composta por 300 pontos: 50 linhas na borda superior, 50 na borda inferior e 50 no corpo do tubo, totalizando 150 linhas da imagem. Sendo cada linha com dois pontos: um para a assinatura esquerda e um para a assinatura direita, totalizando 300 pontos.

A Fig. 61 mostra a imagem limiarizada de um tubo e sua assinatura. Os valores das assinaturas esquerdas e direitas foram transferidos para o computador a partir da interface integrada na placa DE2-70, USB Blaster, e do módulo do Quartus II, SignalTap. A tela de interface com o usuário pode ser vista na Fig. 62. Em seguida, as informações plotadas no SignalTap podem ser exportadas em arquivo de texto e abertas pelo Microsoft Excel ou qualquer outra planilha eletrônica.



(a) Imagem limiarizada. (b) Assinatura da imagem.

Figura 61: Assinatura gerada pelo FPGA.

7.3.4 Comparação

Conforme descrito na seção 7.2.4, é armazenada uma única assinatura padrão, referente a um objeto “bom” e, a partir desta, geradas as assinaturas máximas e mínimas.

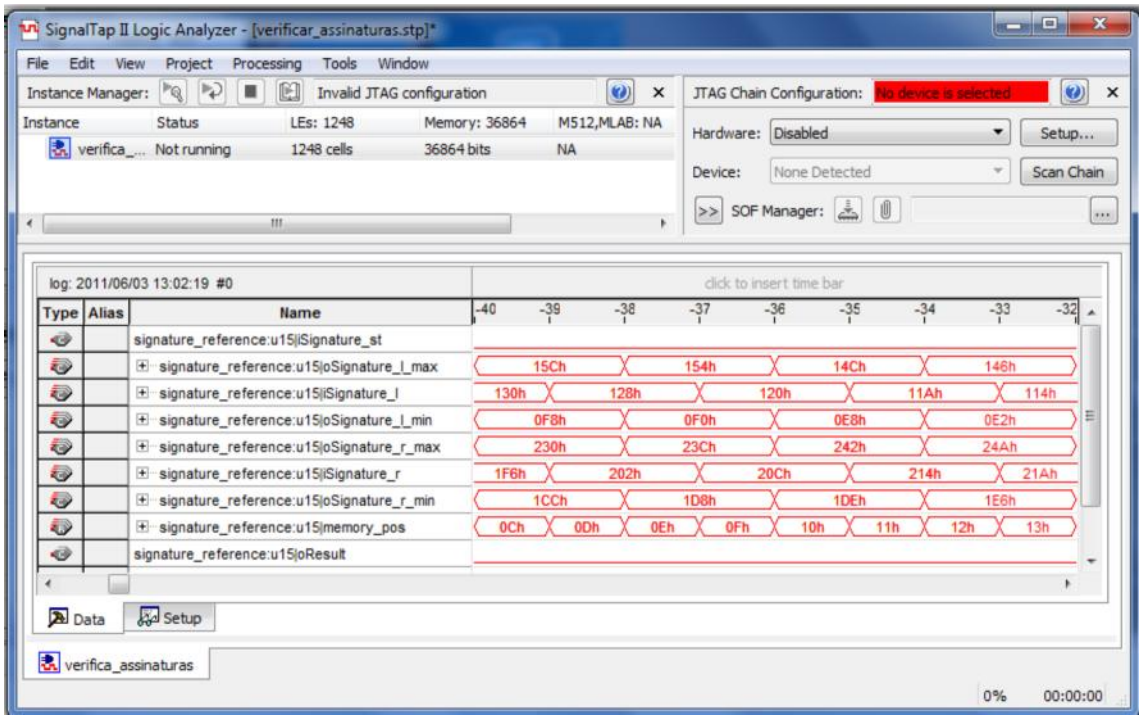


Figura 62: Visualização dos dados da assinatura no Signal Tap.

Experimentalmente, foi testado e observado que o valor de 50 como tolerância é o suficiente para identificar pequenas irregularidades nas bordas. Ou seja:

Assinatura máxima = assinatura padrão + 50

Assinatura mínima = assinatura padrão – 50

Logo, se cada assinatura tem 300 pontos e é necessário armazenar na memória interna do FPGA as assinaturas máxima e mínima, é necessário armazenar 600 pontos na memória como referência para a comparação. Para tal, são necessários 5400 bits uma vez que cada ponto ocupa 9 bits na memória.

Foram realizadas 50 inspeções de tubos, entre bons e ruins. O sistema classificou corretamente 100% das amostras. Porém, devido ao processo extremamente lento em transferir os dados da placa DE2-70 para o PC via SignalTap, os resultados foram observados no momento das inspeções através do *LED* verde LEDG8, que indica em tempo real se o tubo está, ou não, de acordo com as especificações da indústria.

7.3.5 Relatório de implementação

A seguir é apresentado um resumo do relatório gerado pelo Quartus II v10.1 ao final da síntese e compilação do design.

Dispositivo: EP2C70F896C6

Total de elementos lógicos: 3.854 / 68.416 (6%)

- Funções combinacionais: 2,738 / 68.416 (4%)

- Registradores lógicos: 2.807 / 68.416 (4%)

Total de bits de memória: 114.152 / 1.152.000 (10%)

Total de pinos: 530 / 622 (85%)

No relatório apresentado estão inclusos todos os recursos necessários para exibição da imagem no LCD, em monitor VGA, envio da imagem via USB para o PC e outras funcionalidades do design prévio fornecido pela Terasic. Já a tabela 4 apresenta apenas os recursos referentes aos blocos da LEDges.

Tabela 4: Utilização de recursos do FPGA na implementação da LEDges.

Bloco funcional	**Funções combinacionais**	**Registradores lógicos**	**Bits de memória**
Captura dos dados da câmera	55	48	0
Conversor de RGGB para RGB 30 bits	146	94	19.152
Configuração do sensor via I^2C	229	134	0
Limiarização	67	32	0
Detecção de transições relevantes	128	93	0
Composição da assinatura do objeto a partir da identificação das regiões de interesse	162	135	1.000
Comparação	203	100	12.080
Total LEDges	990 / 68.416 (1,5%)	636 / 68.416 (0,9%)	32.232 / 1.152.000 (2,8%)

7.4 Análise do capítulo

O *kit* FPGA utilizado neste trabalho, composto pelas placas DE2-70 (FPGA), D5M (câmera) e LTM (*display* de LCD), tem todas as funcionalidades necessárias para a implementação da LEDges. Inclusive o *kit* tem todos os conectores e conexões com o FPGA para futuras atualizações da implementação, como o armazenamento das imagens adqui-

ridas e assinaturas geradas em cartão de memória SD ou transmissão destas informações para o PC via USB. O *kit*, associado ao design para captura e visualização de imagens fornecido pela Terasic, reduziu significantemente o tempo de desenvolvimento da solução de IVA no FPGA.

Foram desenvolvidos todos os módulos necessários para a correta execução da LEDges. Além destes, foram desenvolvidos módulos e funcionalidades adicionais de interface com o usuário para facilitar testes e ajustes dos parâmetros. Por exemplo, o valor do limiar ajustado, tanto pelo modo manual, quanto pelo automático, pode ser visualizado no *display* de 7 segmentos. Além disto, a imagem original com as componentes RGB, somente com a componente R, a imagem limiarizada e os pontos de transição entre o branco e o preto podem ser visualizados no *display* de LCD. Uma funcionalidade que não estava prevista na LEDges é o ajuste automático do limiar. Este se mostrou útil no ajuste preciso do limiar quando o plano de fundo está bastante iluminado, seja pela iluminação ambiente ou pela influência do *LED*. A tabela 3 resume a interface entre a placa DE2-70 e o usuário, servindo como a principal referência para operação da IVA na placa. Complementarmente, o diagrama em blocos apresentado na Fig. 51 ilustra as conexões entre os módulos implementados, bem como o caminho dos dados no *pipeline* da LEDges.

Dentre os módulos desenvolvidos, os relacionados com a geração da assinatura são os mais complexos. Estes módulos são responsáveis 51% das funções combinacionais e 63% dos registradores lógicos dentre os *designs* desenvolvidos. Ressaltando que os módulos relacionados com a captura e visualização no LCD não foram desenvolvidos neste trabalho, e sim, reutilizados.

No estudo de caso, identificação de amassados em bordas de tubos de creme dental, foram testados diversos parâmetros, principalmente para as etapas de aquisição e limiarização. O melhor resultado foi obtido com o *LED* ligado, limiarizando apenas a componente R da imagem, iluminação ambiente com intensidade reduzida (testado com 150 lux) e tempo de exposição curto, resultando em taxa de 100 inspeções por segundo, sendo superior a taxa de 60 inspeções por segundo obtida com a implementação em microcontrolador. É importante ressaltar que, com a câmera utilizada e se elevada a potência do *LED*, é possível reduzir o tempo de exposição na captura da imagem sem perda de informação relevante, até o limite de 150 inspeções por segundo.

A geração da assinatura resulta em 300 pontos que são comparados com as assinaturas de referência, geradas a partir de uma assinatura de um tubo "bom". A simplificação no módulo de comparação, ao substituir o desvio padrão obtido a partir de 10 assinaturas de

tubos “bons” por um valor fixo de tolerância, não resultou em prejuízos para o sistema de IVA. Todos os tubos testados foram classificados corretamente.

Quanto ao uso de recursos, foi constatado o uso extremamente baixo de memória para executar a LEDges. São necessários somente 5400 bits para armazenar as assinaturas de referência. Em comparação com sistemas convencionais de IVA, que necessitam armazenar um quadro inteiro da imagem como referência, a LEDges utilizou somente 0,06% da memória necessária para armazenar um quadro da imagem (9216000 bits). E, em relação aos recursos totais disponíveis no FPGA, a implementação utilizou recursos computacionais mínimos. Os blocos da LEDges utilizaram apenas 1,5% das funções combinacionais, 0,9% dos registradores lógicos e 2,8% dos bits de memória disponíveis no FPGA Cyclone II EP2C70F896C6. Estes números são significativamente reduzidos, principalmente quando se sabe que este dispositivo é uma opção de baixo custo e médio desempenho.

8 Resultados

No capítulo 3 foi observado que nenhum dos trabalhos analisados atende a todos os requisitos para a aplicação de IVA dimensional com inspeção de irregularidades em bordas e alto desempenho para implementação em sistemas embarcados de tempo-real. Visando atender a estas aplicações, uma nova técnica para IVA, intitulada "LEDges", foi desenvolvida. A principal inovação da LEDges é um novo modelo de iluminação estruturada que permitiu a fácil segmentação da imagem adquirida através de uma simples limiarização. Adicionalmente, o sinal digital resultante da etapa de limiarização viabilizou o uso de técnicas extremamente simples para implementação dos algoritmos para descrição e reconhecimento da imagem.

Foram observadas algumas vantagens competitivas da LEDges em relação aos sistemas de IVA já desenvolvidos e capazes de identificar objetos no primeiro plano. Estes diferenciais viabilizam a utilização de um sistema de IVA em algumas aplicações industriais que não tinham seus requisitos atendidos anteriormente, sobretudo em inspeções com requisitos de alto desempenho, compacto e de baixo custo. Uma comparação entre as técnicas é apresentada na tabela 5.

Entre os diferenciais da LEDges, a possibilidade de execução paralela das quatro etapas de processamento (aquisição, limiarização, geração da assinatura e comparação) em um modelo de *pipeline* a nível de pixel, se destaca. Diretamente relacionado a isto, o uso da memória é significantemente reduzido, uma vez que não é necessário armazenar a imagem para processamento. Foi utilizado somente 0,06% de memória quando comparado às demais técnicas estudadas que armazenam a imagem para realizar o processamento. Com isto, é possível reduzir o custo da implementação da solução e o tempo total de processamento.

A modelagem em computador comprovou a viabilidade técnica da LEDges. Como resultado desta análise, a funcionalidade especificada, que é o principal objetivo a ser alcançado na escolha da arquitetura, pode ser atendida adequadamente em diversas ar-

Tabela 5: Tabela comparativa entre as técnicas de IVA.

Técnica	Observações	Tempo da inspeção	Detecta todas as bordas	Embarcado / Uso de recursos computacionais
Extração do primeiro plano a partir de pares de imagens, com e sem iluminação específica no objeto	Muito eficiente para segmentação de imagens	N/D (1)	Não	Não (2) / Moderado
Iluminação com luz infra-vermelha para identificação do primeiro plano	Desempenho elevado na segmentação	100 ms	Não	Não / Baixo
Utilização de câmeras com registro de profundidade	Utiliza câmeras específicas, de comercialização restrita	66 ms	Sim	Não / Baixo
Detecção do primeiro plano em imagens com plano de fundo estático	Algoritmo simples	N/D	Não	Não (2) / Moderado
Utilização de iluminação a laser	Elevada precisão; Iluminação complexa	3000 ms	Sim	Não / Moderado
Inspeção de objetos baseado em iluminação estruturada a partir de projetores de padrão	Elevada precisão; Iluminação complexa	500 ms	Sim	Não / Elevado
LEDges	**Elevado desempenho com otimização dos recursos utilizados**	**10 ms**	**Sim**	**Sim / Baixo**

N/D - Não Definido
(1) Necessita adquirir duas imagens
(2) Técnica desenvolvida para uso em sistemas embarcados

quiteturas. Foi definido que a LEDges seria implementada em microcontrolador e FPGA. Um segundo objetivo é alcançar a velocidade de inspeção necessária mantendo a taxa de acerto obtida na modelagem, de 100%. Neste item, as duas arquiteturas analisadas diferem significantemente. No FPGA, a LEDges pode ser executada em imagens com maior resolução, detectando erros dimensionais menores, e em maior velocidade. Enquanto a

implementação em microprocessador apresentou três vantagens: custos de produção reduzidos, atualização dos parâmetros da inspeção flexível e esforço da equipe de engenharia reduzido para desenvolvimento do sistema de IVA. A possibilidade de ser implementada em diversas arquiteturas embarcadas, satisfazendo a demanda crescente por desempenho (tempo de resposta e precisão) e pelo uso mínimo de recursos disponíveis (financeiro, humano, físico e energético), também destacou a LEDges em relação às demais técnicas analisadas.

Apesar dos componentes de baixo desempenho para aplicações em processamento de imagem, a implementação da LEDges na arquitetura baseada em câmera analógica e microcontrolador resultou em uma taxa de acerto de 100% na classificação dos tubos, atendendo à funcionalidade requerida. O desempenho alcançado foi de 16,66 ms por inspeção, suficiente para a aplicação de detecção de irregularidades em tubos de creme dental. Quanto à resolução, a implementação apresentou baixa resolução vertical na detecção das irregularidades, com a assinatura formada por 72 pontos da imagem, porém, igualmente suficiente para a aplicação. Tanto a velocidade, quanto a resolução, são limitadas pela câmera. A flexibilidade para atualização dos parâmetros da inspeção foi comprovada pela simples interface via RS-232 com o PC para ajuste de todos os parâmetros da LEDges. Quanto ao esforço reduzido da equipe de engenharia para o desenvolvimento da IVA, esta foi comprovada pela simplicidade da tradução do programa modelado no FreeMat para a linguagem “C” utilizada no ARM7. Por fim, os custos reduzidos para produção foram comprovados com o uso de componentes de baixo desempenho, principalmente da câmera analógica e do microcontrolador.

A respeito da implementação em FPGA com câmera digital, a resolução da imagem adquirida foi, de fato, superior em relação a arquitetura baseada em microcontrolador, tendo a assinatura formada por 300 pontos da imagem. Além disto, foi obtida uma taxa de 100 inspeções por segundo (ou 10 ms por inspeção), sendo superior a todas demais técnicas analisadas. Foi identificado ainda que é possível reduzir o tempo de exposição na captura da imagem até o limite de obter 6,66 ms por inspeção. Quanto aos recursos utilizados, em relação aos recursos totais disponíveis em um FPGA de baixo custo, foi utilizado apenas 1,5% das funções combinacionais, 0,9% dos registradores lógicos e 2,8% dos bits de memória disponíveis. Quanto ao esforço de engenharia, o kit, associado ao *design* para captura de imagens fornecido pela Terasic, reduziu significantemente o esforço para implementação da solução. Por fim, apesar da simplificação na etapa de comparação, o sistema apresentou uma taxa de acerto de 100% na classificação dos tubos.

A tabela 6 resume os resultados obtidos na implementação em microcontrolador e em FPGA.

Tabela 6: Tabela comparativa entre as implementação em microcontrolador e FPGA.

Parâmetros	**Arquitetura 1**	**Arquitetura 2**
Objetivos / Parâmetros otimizados	Taxa de inspeção por segundo; Resolução.	Custos; Esforço para implementação; Flexibilidade para atualizações.
Processador principal	FPGA Cyclone II 2C70	LPC2148 - Núcleo ARM7 TDMI
Uso do processador principal	3%	9%
Tecnologia da câmera	CMOS Digital	CCD Analógica NTSC
Taxa de aquisição = Taxa de inspeção	10 ms / inspeção	16,66 ms / inspeção
Resolução da imagem (horiz x vertical)	640 x 480 pixels	3179 x 120 pixels
Pontos gerados por assinatura final	300	72
Possibilidade de ajustar a resolução e taxa de aquisição	Sim (1)	Não
Custo do processador principal	Moderado	Baixo
Custo da câmera	Moderado	Baixo
Esforço para desenvolvimento e atualização do design	Moderado	Baixo
Flexibilidade para atualização dos parâmetros	Moderada	Alta
Taxa de acerto nas inspeções	100%	100%

(1) Ajuste de 2592 x 1944 pixels @ 66 ms/inspeção até 640 x 480 pixels @ 6,66 ms/inspeção

9 Conclusões e trabalhos futuros

Apesar de existirem diversas indústrias envolvidas no desenvolvimento de sistemas de IVA, não há uma solução capaz de abranger todas as tarefas de inspeção, em todos os campos de aplicações. A aplicação selecionada neste trabalho, inspeção de irregularidades em bordas, com alto desempenho, para implementação em sistemas embarcados de tempo-real, se mostrou demandada pelas indústrias, uma vez que checar se as dimensões de um objeto estão dentro das tolerâncias especificadas é uma tarefa comum. Diversos trabalhos que executam este tipo de inspeção foram identificados, principalmente os baseados em câmeras de profundidade, iluminação com laser e projetores de padrões. Porém, estas soluções complexas apresentaram elevado tempo total de processamento, uso intenso de recursos computacionais, dispositivos de captura de imagem delicados e complexos.

A técnica proposta, LEDges, realiza uma segmentação robusta da imagem a partir de uma simples e eficiente limiarização. Tal característica foi viabilizada a partir do desenvolvimento de uma nova estratégia para iluminação, baseada em LEDs de alta potência, que torna mais fácil a identificação das bordas de interesse. Este foi o foco do trabalho, uma vez que a segmentação pode ser uma das atividades mais complexas do processamento da imagem e, se utilizado um algoritmo inadequado, resulta em um comprometimento completo do sistema de IVA. Adicionalmente, os dados resultantes da etapa de limiarização viabilizam o uso de técnicas extremamente simples para implementação das etapas de descrição e reconhecimento da imagem. Além disto, todas as etapas foram executadas em paralelo, em *pipeline* a nível de *pixel*, sem armazenamento de nenhuma imagem na memória. Por fim, a LEDges eliminou a necessidade de capturar o quadro inteiro antes de realizar a inspeção. Estas características possibilitaram um elevado desempenho com otimização dos recursos computacionais utilizados.

Foi comprovado que a LEDges é flexível na escolha da arquitetura para implementação uma vez que a técnica foi testada em duas arquiteturas, definidas após uma análise criteriosa do ambiente da inspeção, sobretudo do objeto a ser inspecionado, e dos requisitos de desempenho, flexibilidade, custo e tempo de desenvolvimento. Ambas as arquiteturas

foram testadas com 55 imagens e obtiveram taxa de acerto nas inspeções de 100%. A viabilidade técnica foi comprovada. A primeira, baseada em microcontrolador e câmera analógica, foi otimizada nos parâmetros de custos, esforço da equipe de engenharia para implementação e flexibilidade para operação da IVA. A segunda, baseada em FPGA e câmera digital, otimizada na taxa de inspeções por segundo e resolução na detecção de irregularidades nas bordas. Sempre utilizando poucos recursos computacionais, sobretudo memória. Os sistemas de IVA foram aplicados com sucesso a um problema industrial real, onde amassados em bordas de tubos de creme dental foram detectados.

No estudo de caso citado, a implementação em FPGA resultou em uma taxa de 10 ms por inspeção, com possibilidade de ajuste até o limite mínimo de 6,66 ms por inspeção, e resolução de 640 x 480 pixels. Valores estes significativamente melhores que os apresentados pelas técnicas concorrentes, capazes de executar o mesmo tipo de inspeção. A implementação em FPGA se mostrou a melhor opção para aplicações semelhantes e que exijam elevado desempenho. Já a implementação em microcontrolador resultou em uma taxa de 16,66 ms por inspeção e uma menor resolução para detecção de amassados em relação a obtida com o FPGA. Porém, os diferenciais da implementação em microcontrolador fazem com que esta seja a melhor opção para as aplicações semelhantes que necessitem de um baixo nível de esforço para implementação a partir da modelagem realizada em computador no FreeMat e que tenham restrições no custo para implementação, uma vez que todos os componentes utilizados são de baixo custo, sobretudo o microcontrolador LPC2148 e a câmera analógica monocromática padrão NTSC, comumente utilizado em circuitos internos de TV. Logo, os resultados obtidos comprovam que a LEDges e suas implementações, não são somente viáveis, como também são as soluções mais adequadas para aplicações similares.

Trabalhos futuros

As próximas atividades que serão desenvolvidas estão listadas a seguir:

Em uma busca prévia nas bases de patentes, não foi identificada nenhuma solicitação de patente de invenção, nem de modelo de utilidade, que contenham conceitos similares aos desenvolvidos neste trabalho. Logo, será necessário:

- Realizar busca detalhada de anterioridade nas bases de dados nacionais e internacionais;
- Elaborar e solicitar um pedido de patente;

Quanto à continuidade no desenvolvimento da técnica, um dos principais gargalos é

definição prévia das transições relevantes e regiões de interesse da imagem para a inspeção de um objeto qualquer. Logo, para viabilizar o uso da técnica comercialmente, será necessário:

- Desenvolver um algoritmo capaz de identificar, a partir de um conjunto de imagens limiarizadas de objetos "bons" e "ruins", as transições relevantes e regiões de interesse;
- Desenvolver um sistema para verificação funcional do sistema;
- Aprimorar ou substituir a técnica de análise estatística na etapa de comparação a partir dos resultados obtidos na verificação funcional.

Ainda visando a inserção comercial do produto desenvolvido, será necessário:

- Realizar medições mais precisas da precisão e exatidão do sistema.

Por fim, diversos parâmetros da câmera digital são configuráveis e podem ser melhor explorados. Para automação dos testes das funcionalidades da câmera, será importante:

- Desenvolver um suplemento, provavelmente para o FreeMat, visando uma rápida e simples conversão do código utilizado na modelagem em computador para um código em Verilog no que seja referente à etapa de aquisição da imagem.

Após a execução das atividades de pesquisa e desenvolvimento descritas, o sistema de inspeção desenvolvido estará pronto para novas aplicações.

Referências

1 KAESLIN, H. *Digital integrated circuit design: from VLSI architectures to CMOS fabrication*. [S.l.]: Cambridge Univ Pr, 2008.

2 MITAL, A.; GOVINDARAJU, M.; SUBRAMANI, B. A comparison between manual and hybrid methods in parts inspection. *Integrated Manufacturing Systems*, MCB UP Ltd, v. 9, n. 6, p. 344–349, 1998.

3 MARINO, F. et al. A real-time visual inspection system for railway maintenance: automatic hexagonal-headed bolts detection. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, IEEE, v. 37, n. 3, p. 418–428, 2007.

4 DUAN, F. et al. A machine vision inspector for beer bottle. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 20, n. 7, p. 1013–1021, 2007.

5 STEGER, C.; ULRICH, M.; WIEDEMANN, C. *Machine vision algorithms and applications*. [S.l.]: Vch Verlagsgesellschaft Mbh, 2008.

6 MALAMAS, E. et al. A survey on industrial vision systems, applications and tools. *Image and Vision Computing*, Elsevier, v. 21, n. 2, p. 171–188, 2003.

7 FERNANDEZ, C. et al. Automated visual inspection system for the classification of preserved vegetables. In: IEEE. *Industrial Electronics, 2002. ISIE 2002. Proceedings of the 2002 IEEE International Symposium on*. [S.l.], 2002. v. 1, p. 265–269.

8 KIM, T. et al. Visual inspection system for the classification of solder joints. *Pattern Recognition*, Elsevier, v. 32, n. 4, p. 565–575, 1999.

9 LIAO, C. et al. Application of the structured illumination method for automated visual inspection of the loudspeaker cones. *Journal of Materials Processing Technology*, Elsevier, v. 200, n. 1-3, p. 59–70, 2008.

10 JIMENEZ, A.; CERES, R.; PONS, J. A vision system based on a laser range-finder applied to robotic fruit harvesting. *Machine Vision and Applications*, Springer, v. 11, n. 6, p. 321–329, 2000.

11 CUCCHIARA, R. Genetic algorithms for clustering in machine vision. *Machine Vision and Applications*, Springer, v. 11, n. 1, p. 1–6, 1998.

12 CHANG, M.; PARK, S. Automated scanning of dental impressions. *Computer-Aided Design*, Elsevier, v. 41, n. 6, p. 404–411, 2009.

13 WANG, Z. et al. Three-dimensional shape measurement with a fast and accurate approach. *Applied Optics*, Optical Society of America, v. 48, n. 6, p. 1052–1061, 2009.

14 BRINGIER, B.; HELBERT, D.; KHOUDEIR, M. Photometric reconstruction of a dynamic textured surface from just one color image acquisition. *JOSA A*, Optical Society of America, v. 25, n. 3, p. 566–574, 2008.

15 BARNI, M.; CAPPELLINI, V.; MECOCCI, A. Colour-based detection of defects on chicken meat. *Image and vision computing*, Elsevier, v. 15, n. 7, p. 549–556, 1997.

16 CHUNG, Y.; KIM, K. Automated visual inspection system of automobile doors and windows using the adaptive feature extraction. In: IEEE. *Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES'98. 1998 Second International Conference on*. [S.l.]. v. 3, p. 286–293.

17 JENG, J.; MAU, T.; LEU, S. Gap inspection and alignment using a vision technique for laser butt joint welding. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 16, n. 3, p. 212–216, 2000.

18 TORRES, F. et al. Automated real-time visual inspection system for high-resolution superimposed printings. *Image and vision computing*, Elsevier, v. 16, n. 12-13, p. 947–958, 1998.

19 VEGA-RODRÍGUEZ, M.; SÁNCHEZ-PÉREZ, J.; GÓMEZ-PULIDO, J. Cork stopper classification using fpgas and digital image processing techniques. In: IEEE. *Digital Systems, Design, 2001. Proceedings. Euromicro Symposium on*. [S.l.], 2001. p. 270–275.

20 BROSNAN, T.; SUN, D. Improving quality inspection of food products by computer vision–a review. *Journal of Food Engineering*, Elsevier, v. 61, n. 1, p. 3–16, 2004.

21 VEGA-RODRÍGUEZ, M.; SÁNCHEZ-PÉREZ, J.; GÓMEZ-PULIDO, J. An fpga-based implementation for median filter meeting the real-time requirements of automated visual inspection systems. In: CITESEER. *Proceedings of the 10th Mediterranean Conference on Control and Automation, Citeseer*. [S.l.], 2007.

22 HORSCH, K. et al. Automatic segmentation of breast lesions on ultrasound. *Medical Physics*, v. 28, p. 1652, 2001.

23 MAR, N.; FOOKES, C.; YARLAGADDA, P. Design of automatic vision-based inspection system for solder joint segmentation. *Journal of Achievements in Materials and Manufacturing Engineering*, International OCSCO World Press, v. 34, n. 2, p. 145–151, 2009.

24 OYELEYE, O.; LEHTIHET, E. A. A classification algorithm and optimal feature selection methodology for automated solder joint defect inspection. *Journal of manufacturing systems*, Elsevier, v. 17, n. 4, p. 251–262, 1998.

25 JAGANNATHAN, S. Automatic inspection of wave soldered joints using neural networks. *Journal of Manufacturing Systems*, Elsevier, v. 16, n. 6, p. 389–398, 1997.

26 ANAND, S. et al. An integrated machine vision based system for solving the nonconvex cutting stock problem using genetic algorithms. *Journal of Manufacturing Systems*, Citeseer, v. 18, n. 6, p. 396–414, 1999.

27 TSAI, D. A machine vision approach for detecting and inspecting circular parts. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 15, n. 3, p. 217–221, 1999.

28 ZHANG, S.; LIU, Z. A robust, real-time ellipse detector. *Pattern Recognition*, Elsevier, v. 38, n. 2, p. 273–287, 2005.

29 HUNSICKER, R. et al. Automatic vision inspection and measurement system for external screw threads. *Journal of Manufacturing Systems*, Elsevier, v. 13, n. 5, p. 370–384, 1994.

30 BOLLE, R. et al. *Object imaging system*. [S.l.]: Google Patents, maio 20 1997. US Patent 5,631,976.

31 SUN, J. et al. Flash matting. *ACM Transactions on Graphics (TOG)*, ACM, v. 25, n. 3, p. 772–778, 2006.

32 SUN, J. et al. Flash cut: Foreground extraction with flash and no-flash image pairs. In: IEEE. *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. [S.l.], 2008. p. 1–8.

33 WU, Q.; BOULANGER, P.; BISCHOF, W. Robust real-time bi-layer video segmentation using infrared video. In: IEEE. *Computer and Robot Vision, 2008. CRV'08. Canadian Conference on*. [S.l.], 2008. p. 87–94.

34 WU, Q.; BOULANGER, P.; BISCHOF, W. Bi-layer video segmentation with foreground and background infrared illumination. In: ACM. *Proceeding of the 16th ACM international conference on Multimedia*. [S.l.], 2008. p. 1025–1026.

35 WANG, L. et al. Tofcut: Towards robust real-time foreground extraction using a time-of-flight camera. In: CITESEER. [S.l.], 2010.

36 CASARES, M.; VELIPASALAR, S.; PINTO, A. Light-weight salient foreground detection for embedded smart cameras. *Computer Vision and Image Understanding*, Elsevier, v. 114, n. 11, p. 1223–1237, 2010.

37 GONZALEZ, R.; WOODS, R. *Digital image processing*. [S.l.]: Prentice Hall Upper Saddle River, NJ:, 2002.

38 CHIEN, C.; CHENG, Y.; LIN, T. Robust ellipse detection based on hierarchical image pyramid and hough transform. *JOSA A*, Optical Society of America, v. 28, n. 4, p. 581–589, 2011.

39 MACKRORY, J.; DANIELS, M. The impact of new technology in machine vision. *Sensor Review*, MCB UP Ltd, v. 15, n. 1, p. 8–11, 1995.

40 RYER, A.; LIGHT, V. Light measurement handbook. Citeseer, 1997.

APÊNDICE A – Modelagem em FreeMat

Code A.1: Aquisição da imagem

```
tube = imread ('tube.bmp');
size_tube = size (tube);
```

Code A.2: Limiarização

```
threshold = 128;
for x = 1:size_tube(1),
   for y = 1:size_tube(2),
      if tube(x,y) > threshold,
         tube_bin(x,y) = 1;
      else
         tube_bin(x,y) = 0;
      end
   end
end
```

Code A.3: Geração da assinatura - Detecção das transições relevantes

```
for x = 1:size_tube(1),
     flag_a = 0;
     flag_c = 0;
     for y = 2:size_tube(2),

       if ((tube_bin(x,y) == 1) && (tube_bin(x,y-1)== 0) && (flag_a == 0)),
          signature_a(x) = y;
          flag_a = 1;
       end

       if ((tube_bin(x,y) == 0) && tube_bin(x,y-1) == 1),
           signature_b(x) = y;
       end

       if ((tube_bin(x,y) == 0) && (tube_bin(x,y-1) == 1) && (flag_c == 0)),
```

```
16            signature_c(x) = y;
17            flag_c = 1;
18        end
19
20        if ((tube_bin(x,y) == 1) && (tube_bin(x,y-1) == 0)),
21            signature_d(x) = y;
22        end
23
24    end
25 end
```

Code A.4: Geração da assinatura - Detecção das regiões de interesse e construção das assinaturas

```
1  x = 1;
2
3  while signature_a(x) == 0,
4      x = x + 1;
5  end
6
7  x = x - 1;
8
9  for k = 1:25,
10     signature(k,1) = signature_a(x+k);
11     signature(k,2) = signature_b(x+k);
12 end
13
14 x = 100;
15
16 while signature_c(x)<signature_d(x),
17     x = x + 1;
18 end
19
20 x = x - 1;
21
22 for k = 1:25,
23     signature(k+25,1) = signature_c(x+k-25);
24     signature(k+25,2) = signature_d(x+k-25);
25 end
26
```

```
27 for k = 1:25,
28     signature(k+50,1) = signature_a (x+k);
29     signature(k+50,2) = signature_b (x+k);
30 end
```

Code A.5: Plota as assinaturas

```
1 signature (1:25,1) = signature_a (42:66);
2 signature (1:25,2) = signature_b (42:66);
3 signature (26:50,1) = signature_c (91:115);
4 signature (26:50,2) = signature_d (91:115);
5 signature (51:75,1) = signature_a (116:140);
6 signature (51:75,2) = signature_b (116:140);
7 hold off;
8 plot (signature);
```

APÊNDICE B – Código em "C" para ARM7

Code B.1: Código para LPC2148

```
1  \\ Código escrito por Rafael M. Macieira
2
3  /*
       ************************************************************************
4   *   einttest.c:  main C entry file for Philips LPC214x Family
        Microprocessors
5   *
6   *   Copyright(C) 2006, Philips Semiconductor
7   *   All rights reserved.
8   *
9   *   History
10  *   2005.10.01  ver 1.00    Prelimnary version, first Release
11  *
12  ************************************************************************
       */
13 #include <lpc214x.H>                        /* LPC21xx definitions */
14 //#include "type.h"
15 #include "types.h"
16 #include "irq.h"
17 #include "extint.h"
18 #include "lcd.h"
19
20 const int LIMITE = 10000;
21 unsigned int signature[15][2],//, signatureBottom[15][2],
22                          defaultSignature[15][2],//, defaultSignatureBottom
                             [15][2],
23                          lineCount = 0,
24                          defaultLastIndex = 0,
25                          signatureCalcTemp = 0;
```

```
26 char s[100], i, k;
27 byte start, calib, fim = 0;
28
29
30
31 void delay(){
32         unsigned char n;
33         unsigned long i;
34         for ( n=0x00;n<10;n++ ) {
35                 for ( i=0x00;i<100000;i++ ) ;
36         }
37 }
38
39 /*
40         void delay()
41 */
42 void delay2(){
43         unsigned char n;
44         unsigned long i;
45         for ( n=0x00;n<5;n++ ) {
46                 for ( i=0x00;i<100000;i++ ) ;
47         }
48 }
49
50
51 void intLineSync(void) __irq {
52         int diffDefault, diffTemp;
53         lcd_puts("-1");
54         if (lineCount < 15) {
55                 lcd_puts("0");
56                 signature[lineCount][0] = T0CR1 - T0CR0;
57                 signature[lineCount][1] = T0CR3 - T0CR0;
58                 lcd_puts("1");
59         }else{
60                 lcd_puts("2");
61                 if(T0CR2!=0 && T0CR3!=0){
62                         lcd_puts("3");
63                         signature[lineCount][0] = T0CR3 - T0CR0;
64                         signature[lineCount][1] = T0CR2 - T0CR1;
65                 }else{
66         lcd_puts("4");
67         //desabilita IRQ de linha no fim do quadro para nao contar com os
               pixels somente pretos
```

```
68          VICIntEnClr = 1 << 16; //Desabilita IRQ de Linha
69          //CHECAGEM DE ASSINATURA OU CALIBRAR!
70
71          if (calib == 1){
72                  defaultLastIndex = lineCount - 1;
73                  for (k=0;k<30;k++) {
74                          defaultSignature[k][0] = signature[k][0];
75                          defaultSignature[k][1] = signature[k][1];
76                  }
77          }else{
78                  diffDefault = defaultLastIndex;
79                  diffTemp = lineCount - 1;
80                  for (k=0;k<14;k++){
81                          signatureCalcTemp += (defaultSignature[k][0] -
                               signature[k][0])^2;
82                          signatureCalcTemp += (defaultSignature[diffDefault
                               ][0] - signature[diffTemp][0])^2;
83                          if (diffDefault == 29) {
84                                  diffDefault = 15;
85                          }else{
86                                  diffDefault++;
87                          }
88                          if (diffTemp == 29) {
89                                  diffTemp = 15;
90                          }else{
91                                  diffTemp++;
92                          }
93                  }
94                  if (signatureCalcTemp < LIMITE){
95                          IOCLR0 |= BIT31;
96                  }else{
97                          IOSET0 |= BIT31;
98                  }
99
100         }
101         fim = 1;
102         VICIntEnClr = 1 << 14;//Frame Syc IRQ
103         lcd_puts("Fim!");
104 }
105         }
106
107         if (lineCount == 29) {
108                 lineCount = 15;
```

```
109         }else{
110                 lineCount++;
111         }
112         T0CCR|=(BIT1 | BIT3);
113         T0CCR&=(~BIT7 & ~BIT9);
114         EXTINT|= BIT2;
115         VICVectAddr = 0;
116 }
117
118 void Timer0_intBinarySignal(void) __irq {
119 switch (T0IR){
120         case BIT4:
121                 lcd_puts("B0");
122                 if (lineCount == 0) {
123                         VICIntEnable = 1 << 16; //Line Sync IRQ
124                 }
125                 T0CCR&=(~BIT1);  //Desabilita o Timer Cap0.0
126                 T0CCR|=(BIT7);  //Habilita o Timer Cap0.2
127                 T0IR|= BIT4;
128                 break;
129         case BIT5:
130                 lcd_puts("B2");
131                 T0CCR&=(~BIT3);  //Desabilita o Timer Cap0.1
132                 T0CCR|=(BIT9);  //Habilita o Timer Cap0.3;
133                 T0IR|= BIT5;
134                 break;
135         /*case BIT6:
136                 T0CCR&=(~BIT3);  //Desabilita o Timer Cap0.1
137                 T0CCR|=(BIT9);  //Habilita o Timer Cap0.3;
138                 break;
139         case BIT7:
140                 T0CCR&=(~BIT3);  //Desabilita o Timer Cap0.1
141                 T0CCR|=(BIT9);  //Habilita o Timer Cap0.3;
142                 break;*/
143 }
144 VICVectAddr = 0;
145 }
146
147 void intFrameSync(void) __irq {
148         lcd_puts("\nR1");
149         //VICIntEnClr = 1 << 14;//Frame Syc IRQ
150         //lcd_puts("\fRRRRRRRR");
151         lcd_puts("R2");
```

```
152         if (start = 1){
153                 lcd_puts("R3");
154                 calib = 1;
155                 T0CCR|=(BIT2 | BIT5);// | BIT8 | BIT11);  //Atualiza a os
                        canais de captura com interrupção
156                 T0CCR|=(BIT1 | BIT3);// | BIT7 | BIT9);
157                 start = 0;
158         }
159         lcd_puts("R4");
160         //Habilita a IRQ de Linha apenas quando começa os pixels brancos.
161         lineCount = 0;
162         VICIntEnable = 1 << 4;
163         T0TCR = 0X03;
164         T0TCR = 0X01;
165         lcd_puts("R5");
166         EXTINT|= BIT0;
167         VICVectAddr = 0;
168 }
169
170 void intStart(void) __irq {
171         VICIntEnClr = 1 << 17;
172         VICIntEnable = 1 << 14;//Frame Syc IRQ
173         start = 1;
174         EXTINT|= BIT3;
175         VICVectAddr = 0;
176 }
177
178 /*void intColorSync(void) __irq {
179         T0TCR = 0X01;
180         EXTINT|= BIT1;
181         VICVectAddr = 0;
182
183 } */
184
185 void init(){
186         //** Direção dos pinos
187         IODIR0=0x0;      // todo port0 como Entrada
188         IODIR0|=BIT31;
189         //** Fim da direção dos pinos
190
191         //** Definição dos Pinos
192         //PINSEL0&=~BIT3 & ~BIT2;
                                //Desabilita IRQ no P0.1
```

```
193         PINSEL1|=(BIT28 | BIT29);
                    //P0.30 como Timer Cap0.0
194         PINSEL0 = (PINSEL0 | BIT9) & ~BIT8;                    //P0.4 como
                Timer Cap0.1
195         PINSEL1 = (PINSEL1 | BIT25) & ~BIT24;                  //P0.28
                como Timer Cap0.2
196         PINSEL1 = (PINSEL1 | BIT27) & ~BIT26;                  //P0.29
                como Timer Cap0.3
197
198         //CPSR|=BIT7;
199         PINSEL0|=(BIT2 | BIT3);          //P0.1 como Interrupcao Externa 0
200         PINSEL0|=(BIT6 | BIT7);                  //P0.3 como Interrupcao
                Externa 1
201         PINSEL0|=(BIT14 | BIT15);        //P0.7 como Interrupcao Externa 2
202         PINSEL1 |= BIT9 | BIT8;                          //P0.20 como
                Interrupcao Externa 3
203         //** Fim Definição dos Pinos
204
205         //** Habilitando IRQ para TIMER0
206         /*VICVectCntl0 = 0x20 | 4;
207         VICVectCntl1 = 0x20 | 14;
208         VICVectAddr0 = (unsigned)intDesabCap;
209         VICVectAddr1 = (unsigned)intExt0;*/
210         //EXTPOLAR|= BIT0 | BIT1 | BIT2;// | BIT3;
211         EXTINT|= BIT0 | BIT1 | BIT2 | BIT3;
212         EXTMODE = BIT0 | BIT1 | BIT2 | BIT3;
213         EXTPOLAR = 0;//BIT0 | BIT1 | BIT2 | BIT3;
214         //VICIntEnable|=(BIT4);// | BIT14 | BIT15 | BIT16);
215         lcd_puts("\fWaiting...");
216         if ( install_irq( 4, (void *) Timer0_intBinarySignal    ) == FALSE )
                {} //Captura
217         if ( install_irq( 14, (void *)intFrameSync ) == FALSE ){} //Color
                Sync IRQ
218         if ( install_irq( 16, (void *)intLineSync   ) == FALSE ){} //Line
                Sync IRQ
219         if ( install_irq( 17, (void *) intStart    ) == FALSE ){} //Start
                Aplicação
220         VICIntEnable = 1 << 17;
221
222         //** Fim Habilitando IRQ para TIMER0
223
224         //** Configuração do Timer
```

```
225         //T0IR = 0X16;              // gerando Interrupt flag for capture
                channel 0 event.
226         T0TCR = 0X00; // Desabilitado  ************* 0X01 | 0X02
227         T0TC = 0x0000;
228         T0PR = 0x0000; //tem que definir o divisor (Prescale)
229         T0PC = 0x0000;
230         T0CCR = 0;
231         //T0CCR|=(BIT2 | BIT5);// | BIT8 | BIT11);  //Atualiza a os canais
                de captura com interrupção
232         //T0CCR|=(BIT1 | BIT3);// | BIT7 | BIT9);
233         T0CTCR = 0x00;  //Incremento a cada PCLK
234         //** Fim Configuração do Timer
235 }
236
237 /*
    ****************************************************************************

238 **   Main Function  main()
239 ****************************************************************************
    */
240 int main (void){
241         char s1[100], s2[100];
242         lcd_init();
243         //IODIR1=0xFFFFFFFF;
244     //inicializa configuração das Interrupções
245         init_VIC();
246         //inicializa configurações desta aplicação
247         init();
248
249     /* initialize GPIO pins as external interrupts */
250     //EINTInit();
251         //T0TCR = 0X01;
252         while(1){
253                 if (fim==1){
254                         for (k=0;k<30;k++){
255                                 printf(s1,"Linha %d- Valor1: %d\r\nValor2:
                                        %d",k,defaultSignature[k][0],
                                        defaultSignature[k][1]);
256                                 lcd_puts(s1);
257                                 delay();delay();
258                         }
259                 }
260                 /*while (T0TC < 10000){}
```

```
261                             T0TCR = 0X03;
262                             T0TCR = 0X01;
263                             IOSET0 |= BIT22;
264                     while (T0TC < 10000){}
265                             T0TCR = 0X03;
266                             T0TCR = 0X01;
267                             IOCLR0 |= BIT22;*/
268             }
269     return 0;
270 }
271
272 /*
    ******************************************************************************

273 **                                End Of File
274 ******************************************************************************
    */
```

APÊNDICE C – Código em Verilog para FPGA

Para não sobrecarregar este documento, foram inseridos neste apêndice apenas os arquivos modificados na implementação da LEDges, o que representa um terço do código fonte completo. Os demais arquivos podem ser obtidos no site da TerASIC.

Code C.1: Código principal

```
1  // --------------------------------------------------------------------
2  // Copyright (c) 2008 by Terasic Technologies Inc.
3  // --------------------------------------------------------------------
4  //
5  // Permission:
6  //
7  //   Terasic grants permission to use and modify this code for use
8  //   in synthesis for all Terasic Development Boards and Altera Development
9  //   Kits made by Terasic.  Other use of this code, including the selling
10 //   ,duplication, or modification of any portion is strictly prohibited.
11 //
12 // Disclaimer:
13 //
14 //   This VHDL/Verilog or C/C++ source code is intended as a design
       reference
15 //   which illustrates how these types of functions can be implemented.
16 //   It is the user's responsibility to verify their design for
17 //   consistency and functionality through the use of formal
18 //   verification methods.  Terasic provides no warranty regarding the use
19 //   or functionality of this code.
20 //
21 // --------------------------------------------------------------------
22 //
23 //                     Terasic Technologies Inc
24 //                     356 Fu-Shin E. Rd Sec. 1. JhuBei City,
25 //                     HsinChu County, Taiwan
```

```
26 //                      302
27 //
28 //                      web: http://www.terasic.com/
29 //                      email: support@terasic.com
30 //
31 // ---------------------------------------------------------------------
32 //
33 // Major Functions:      DE2_70 D5M LTM
34 //
35 // ---------------------------------------------------------------------
36 //
37 // Revision History :
38 // ---------------------------------------------------------------------
39 //   Ver  :| Author            :| Mod. Date :| Changes Made:
40 //   V1.0 :| Keith Shen        :| 08/04/16  :| Initial Revision
41 // ---------------------------------------------------------------------
42
43 module DE2_70
44         (
45                 ////////////////////    Clock Input
                        ////////////////////
46                 iCLK_28,                                        //
                        28.63636 MHz
47                 iCLK_50,                                        //
                        50 MHz
48                 iCLK_50_2,                                      //
                        50 MHz
49                 iCLK_50_3,                                      //
                        50 MHz
50                 iCLK_50_4,                                      //
                        50 MHz
51                 iEXT_CLOCK,                                     //
                        External Clock
52                 ////////////////////    Push Button
                        ////////////////////
53                 iKEY,                                           //
                        Pushbutton[3:0]
54                 ////////////////////    DPDT Switch
                        ////////////////////
55                 iSW,                                            //
                        Toggle Switch[17:0]
56                 ////////////////////    7-SEG Dispaly
                        ////////////////////
```

```
57          oHEX0_D,                                    //
                Seven Segment Digit 0
58          oHEX0_DP,                                   //
            Seven Segment Digit 0 decimal point
59          oHEX1_D,                                    //
                Seven Segment Digit 1
60          oHEX1_DP,                                   //
            Seven Segment Digit 1 decimal point
61          oHEX2_D,                                    //
                Seven Segment Digit 2
62          oHEX2_DP,                                   //
            Seven Segment Digit 2 decimal point
63          oHEX3_D,                                    //
                Seven Segment Digit 3
64          oHEX3_DP,                                   //
            Seven Segment Digit 3 decimal point
65          oHEX4_D,                                    //
                Seven Segment Digit 4
66          oHEX4_DP,                                   //
            Seven Segment Digit 4 decimal point
67          oHEX5_D,                                    //
                Seven Segment Digit 5
68          oHEX5_DP,                                   //
            Seven Segment Digit 5 decimal point
69          oHEX6_D,                                    //
                Seven Segment Digit 6
70          oHEX6_DP,                                   //
            Seven Segment Digit 6 decimal point
71          oHEX7_D,                                    //
                Seven Segment Digit 7
72          oHEX7_DP,                                   //
            Seven Segment Digit 7 decimal point
73          //////////////////////      LED
               //////////////////////
74          oLEDG,                                      //
                LED Green[8:0]
75          oLEDR,                                      //
                LED Red[17:0]
76          //////////////////////      UART
               //////////////////////
77          oUART_TXD,                                  //
                UART Transmitter
78          iUART_RXD,                                  //
```

```
                UART Receiver
79              oUART_CTS,                          //      UART Clear
                    To Send
80              iUART_RTS,                          //      UART Requst
                     To Send
81              ////////////////////////        IRDA
                    ////////////////////////
82              oIRDA_TXD,                                  //
                UART Transmitter
83              iIRDA_RXD,                                  //
                UART Receiver
84              ////////////////////    SDRAM Interface
                    ////////////////
85              DRAM_DQ,                                    //
                SDRAM Data bus 32 Bits
86              oDRAM0_A,                                   //
                SDRAM0 Address bus 13 Bits
87              oDRAM1_A,                                   //
                SDRAM1 Address bus 13 Bits
88              oDRAM0_LDQM0,                           //
                    SDRAM0 Low-byte Data Mask
89              oDRAM1_LDQM0,                           //
                    SDRAM1 Low-byte Data Mask
90              oDRAM0_UDQM1,                           //
                    SDRAM0 High-byte Data Mask
91              oDRAM1_UDQM1,                           //
                    SDRAM1 High-byte Data Mask
92              oDRAM0_WE_N,                            //
                    SDRAM0 Write Enable
93              oDRAM1_WE_N,                            //
                    SDRAM1 Write Enable
94              oDRAM0_CAS_N,                           //
                    SDRAM0 Column Address Strobe
95              oDRAM1_CAS_N,                           //
                    SDRAM1 Column Address Strobe
96              oDRAM0_RAS_N,                           //
                    SDRAM0 Row Address Strobe
97              oDRAM1_RAS_N,                           //
                    SDRAM1 Row Address Strobe
98              oDRAM0_CS_N,                            //
                    SDRAM0 Chip Select
99              oDRAM1_CS_N,                            //
                    SDRAM1 Chip Select
```

```
100         oDRAM0_BA,                                          //
                SDRAM0 Bank Address
101         oDRAM1_BA,                                          //
                SDRAM1 Bank Address
102         oDRAM0_CLK,                                         //
                SDRAM0 Clock
103         oDRAM1_CLK,                                         //
                SDRAM1 Clock
104         oDRAM0_CKE,                                         //
                SDRAM0 Clock Enable
105         oDRAM1_CKE,                                         //
                SDRAM1 Clock Enable
106         ////////////////////    Flash Interface
                ////////////////
107         FLASH_DQ,                                           //
                FLASH Data bus 15 Bits (0 to 14)
108         FLASH_DQ15_AM1,                                 //  FLASH
                Data bus Bit 15 or Address A-1
109         oFLASH_A,                                           //
                FLASH Address bus 26 Bits
110         oFLASH_WE_N,                                    //
                FLASH Write Enable
111         oFLASH_RST_N,                                   //
                FLASH Reset
112         oFLASH_WP_N,                                    //
                FLASH Write Protect /Programming Acceleration
113         iFLASH_RY_N,                                    //
                FLASH Ready/Busy output
114         oFLASH_BYTE_N,                                  //
                FLASH Byte/Word Mode Configuration
115         oFLASH_OE_N,                                    //
                FLASH Output Enable
116         oFLASH_CE_N,                                    //
                FLASH Chip Enable
117         ////////////////////    SRAM Interface
                ////////////////
118         SRAM_DQ,                                            //
                SRAM Data Bus 32 Bits
119         SRAM_DPA,                                           //
                 SRAM Parity Data Bus
120         oSRAM_A,                                            //
                SRAM Address bus 22 Bits
121         oSRAM_ADSC_N,                               //      SRAM
```

```
    Controller Address Status
122 oSRAM_ADSP_N,                   //      SRAM Processor
    Address Status
123 oSRAM_ADV_N,                    //      SRAM Burst Address
    Advance
124 oSRAM_BE_N,                     //      SRAM Byte Write
    Enable
125 oSRAM_CE1_N,                            //      SRAM Chip
    Enable
126 oSRAM_CE2,                              //      SRAM Chip
    Enable
127 oSRAM_CE3_N,                            //      SRAM Chip
    Enable
128 oSRAM_CLK,                      //      SRAM Clock
129 oSRAM_GW_N,                             //  SRAM Global
    Write Enable
130 oSRAM_OE_N,                             //      SRAM Output
     Enable
131 oSRAM_WE_N,                             //      SRAM Write
    Enable
132 ////////////////////    ISP1362 Interface
    ////////////////
133 OTG_D,                                                   //
          ISP1362 Data bus 16 Bits
134 oOTG_A,                                                  //
          ISP1362 Address 2 Bits
135 oOTG_CS_N,                                              //
          ISP1362 Chip Select
136 oOTG_OE_N,                                              //
          ISP1362 Read
137 oOTG_WE_N,                                              //
          ISP1362 Write
138 oOTG_RESET_N,                                          //
    ISP1362 Reset
139 OTG_FSPEED,                                             //
          USB Full Speed, 0 = Enable, Z = Disable
140 OTG_LSPEED,                                             //
          USB Low Speed,  0 = Enable, Z = Disable
141 iOTG_INT0,                                              //
          ISP1362 Interrupt 0
142 iOTG_INT1,                                              //
          ISP1362 Interrupt 1
143 iOTG_DREQ0,                                             //
```

```
                ISP1362 DMA Request 0
144             iOTG_DREQ1,                                     //
                ISP1362 DMA Request 1
145             oOTG_DACK0_N,                                   //
                ISP1362 DMA Acknowledge 0
146             oOTG_DACK1_N,                                   //
                ISP1362 DMA Acknowledge 1
147             ////////////////////    LCD Module 16X2
                ///////////////
148             oLCD_ON,                                        //
                LCD Power ON/OFF
149             oLCD_BLON,                                      //
                LCD Back Light ON/OFF
150             oLCD_RW,                                        //
                LCD Read/Write Select, 0 = Write, 1 = Read
151             oLCD_EN,                                        //
                LCD Enable
152             oLCD_RS,                                        //
                LCD Command/Data Select, 0 = Command, 1 = Data
153             LCD_D,                                          //      LCD
                 Data bus 8 bits
154             ////////////////////    SD_Card Interface
                ///////////////
155             SD_DAT,                                         //
                SD Card Data
156             SD_DAT3,                                        //
                SD Card Data 3
157             SD_CMD,                                         //
                SD Card Command Signal
158             oSD_CLK,                                        //
                SD Card Clock
159             ////////////////////    I2C
                ////////////////////////////
160             I2C_SDAT,                                       //
                I2C Data
161             oI2C_SCLK,                                      //
                I2C Clock
162             ////////////////////    PS2
                ////////////////////////////
163             PS2_KBDAT,                                      //
                PS2 Keyboard Data
164             PS2_KBCLK,                                      //
                PS2 Keyboard Clock
```

```
165         PS2_MSDAT,                                          //
                PS2 Mouse Data
166         PS2_MSCLK,                                          //
                PS2 Mouse Clock
167         ////////////////////    VGA
                ////////////////////////////
168         oVGA_CLOCK,                                 //      VGA
                Clock
169         oVGA_HS,                                            //
                VGA H_SYNC
170         oVGA_VS,                                            //
                VGA V_SYNC
171         oVGA_BLANK_N,                               //      VGA
                BLANK
172         oVGA_SYNC_N,                                //      VGA
                SYNC
173         oVGA_R,                                             //
                VGA Red[9:0]
174         oVGA_G,                                             //
                VGA Green[9:0]
175         oVGA_B,                                             //
                VGA Blue[9:0]
176         ////////////    Ethernet Interface
                ////////////////////////
177         ENET_D,                                     //
                DM9000A DATA bus 16Bits
178         oENET_CMD,                                          //
                DM9000A Command/Data Select, 0 = Command, 1 = Data
179         oENET_CS_N,                                         //
                DM9000A Chip Select
180         oENET_IOW_N,                                //
                DM9000A Write
181         oENET_IOR_N,                                //
                DM9000A Read
182         oENET_RESET_N,                              //
                DM9000A Reset
183         iENET_INT,                                          //
                DM9000A Interrupt
184         oENET_CLK,                                          //
                DM9000A Clock 25 MHz
185         ////////////////        Audio CODEC
                ////////////////////////
186         AUD_ADCLRCK,                                //
```

```
                Audio CODEC ADC LR Clock
187             iAUD_ADCDAT,                                    //
                Audio CODEC ADC Data
188             AUD_DACLRCK,                                    //
                Audio CODEC DAC LR Clock
189             oAUD_DACDAT,                                    //
                Audio CODEC DAC Data
190             AUD_BCLK,                                               //
                        Audio CODEC Bit-Stream Clock
191             oAUD_XCK,                                               //
                        Audio CODEC Chip Clock
192             ///////////////       TV Decoder
                ////////////////////////
193             iTD1_CLK27,                                             //
                        TV Decoder1 Line_Lock Output Clock
194             iTD1_D,                                         // TV
                Decoder1 Data bus 8 bits
195             iTD1_HS,                                                //
                        TV Decoder1 H_SYNC
196             iTD1_VS,                                                //
                        TV Decoder1 V_SYNC
197             oTD1_RESET_N,                                   //      TV
                Decoder1 Reset
198             iTD2_CLK27,                                             //
                        TV Decoder2 Line_Lock Output Clock
199             iTD2_D,                                         // TV
                Decoder2 Data bus 8 bits
200             iTD2_HS,                                                //
                        TV Decoder2 H_SYNC
201             iTD2_VS,                                                //
                        TV Decoder2 V_SYNC
202             oTD2_RESET_N,                                   //      TV
                Decoder2 Reset
203             ////////////////////    GPIO
                ////////////////////////////
204             GPIO_0,                                                 //
                        GPIO Connection 0 I/O
205             GPIO_CLKIN_N0,                                  //
                GPIO Connection 0 Clock Input 0
206             GPIO_CLKIN_P0,                          //      GPIO
                Connection 0 Clock Input 1
207             GPIO_CLKOUT_N0,                         //      GPIO
                Connection 0 Clock Output 0
```

```
208                 GPIO_CLKOUT_P0,                 //      GPIO Connection 0
                        Clock Output 1
209                 GPIO_1,                                             //
                            GPIO Connection 1 I/O
210                 GPIO_CLKIN_N1,                  //      GPIO Connection 1
                        Clock Input 0
211                 GPIO_CLKIN_P1,                  //      GPIO Connection 1
                        Clock Input 1
212                 GPIO_CLKOUT_N1,                 //      GPIO Connection 1
                        Clock Output 0
213                 GPIO_CLKOUT_P1                  //      GPIO Connection 1
                        Clock Output 1
214
215         );
216
217 //
    =============================================================================

218 // PARAMETER declarations
219 //
    =============================================================================


220
221
222 //
    =============================================================================

223 // PORT declarations
224 //
    =============================================================================


225 ////////////////////////        Clock Input
    ////////////////////////
226 input                   iCLK_28;                                    //
    28.63636 MHz
227 input                   iCLK_50;                                    //      50
    MHz
228 input                   iCLK_50_2;                                  //      50
    MHz
229 input           iCLK_50_3;                              //      50 MHz
230 input           iCLK_50_4;                              //      50 MHz
231 input           iEXT_CLOCK;                             //      External
    Clock
```

```
232 ////////////////////////     Push Button
        ////////////////////////
233 input    [3:0]    iKEY;                                  //      Pushbutton
        [3:0]
234 ////////////////////////     DPDT Switch
        ////////////////////////
235 input    [17:0]   iSW;                                   //      Toggle
        Switch[17:0]
236 ////////////////////////     7-SEG Dispaly   ////////////////////////
237 output   [6:0]    oHEX0_D;                               //      Seven
        Segment Digit 0
238 output                oHEX0_DP;                         //  Seven
        Segment Digit 0 decimal point
239 output   [6:0]    oHEX1_D;                               //      Seven
        Segment Digit 1
240 output                oHEX1_DP;                         //  Seven
        Segment Digit 1 decimal point
241 output   [6:0]    oHEX2_D;                               //      Seven
        Segment Digit 2
242 output                oHEX2_DP;                         //  Seven
        Segment Digit 2 decimal point
243 output   [6:0]    oHEX3_D;                               //      Seven
        Segment Digit 3
244 output                oHEX3_DP;                         //  Seven
        Segment Digit 3 decimal point
245 output   [6:0]    oHEX4_D;                               //      Seven
        Segment Digit 4
246 output                oHEX4_DP;                         //  Seven
        Segment Digit 4 decimal point
247 output   [6:0]    oHEX5_D;                               //      Seven
        Segment Digit 5
248 output                oHEX5_DP;                         //  Seven
        Segment Digit 5 decimal point
249 output   [6:0]    oHEX6_D;                               //      Seven
        Segment Digit 6
250 output                oHEX6_DP;                         //  Seven
        Segment Digit 6 decimal point
251 output   [6:0]    oHEX7_D;                               //      Seven
        Segment Digit 7
252 output                oHEX7_DP;                         //  Seven
        Segment Digit 7 decimal point
253 ////////////////////////////    LED
        ////////////////////////////
```

```
254 output  [8:0]  oLEDG;                                   //  LED Green
    [8:0]
255 output  [17:0] oLEDR;                                   //  LED Red
    [17:0]
256 ////////////////////////////  UART  ////////////////////////////
257 output                     oUART_TXD;                   //
    UART Transmitter
258 input                      iUART_RXD;                   //
    UART Receiver
259 output                     oUART_CTS;          //  UART Clear To Send
260 input                      iUART_RTS;          //  UART Requst To Send
261 ////////////////////////////  IRDA  ////////////////////////////
262 output                     oIRDA_TXD;                   //
    IRDA Transmitter
263 input                      iIRDA_RXD;                   //
    IRDA Receiver
264 ////////////////////////  SDRAM Interface ////////////////////////
265 inout   [31:0] DRAM_DQ;                                 //  SDRAM Data
    bus 32 Bits
266 output  [12:0] oDRAM0_A;                                //  SDRAM0
    Address bus 13 Bits
267 output  [12:0] oDRAM1_A;                                //  SDRAM1
    Address bus 13 Bits
268 output                     oDRAM0_LDQM0;                //  SDRAM0 Low-
    byte Data Mask
269 output                     oDRAM1_LDQM0;                //  SDRAM1 Low-
    byte Data Mask
270 output                     oDRAM0_UDQM1;                //  SDRAM0 High
    -byte Data Mask
271 output                     oDRAM1_UDQM1;                //  SDRAM1 High
    -byte Data Mask
272 output                     oDRAM0_WE_N;                 //  SDRAM0
    Write Enable
273 output                     oDRAM1_WE_N;                 //  SDRAM1
    Write Enable
274 output                     oDRAM0_CAS_N;                //  SDRAM0
    Column Address Strobe
275 output                     oDRAM1_CAS_N;                //  SDRAM1
    Column Address Strobe
276 output                     oDRAM0_RAS_N;                //  SDRAM0 Row
    Address Strobe
277 output                     oDRAM1_RAS_N;                //  SDRAM1 Row
    Address Strobe
```

```
278 output          oDRAM0_CS_N;                    //  SDRAM0 Chip Select
279 output          oDRAM1_CS_N;                    //  SDRAM1 Chip Select
280 output  [1:0]   oDRAM0_BA;                      //  SDRAM0 Bank Address
281 output  [1:0]   oDRAM1_BA;                      //  SDRAM1 Bank Address
282 output          oDRAM0_CLK;                     //  SDRAM0 Clock
283 output          oDRAM1_CLK;                     //  SDRAM1 Clock
284 output          oDRAM0_CKE;                     //  SDRAM0 Clock Enable
285 output          oDRAM1_CKE;                     //  SDRAM1 Clock Enable
286 ////////////////////////    Flash Interface    ////////////////////////
287 inout   [14:0]  FLASH_DQ;                       //  FLASH Data bus 15 Bits (0 to 14)
288 inout           FLASH_DQ15_AM1;                 //  FLASH Data bus Bit 15 or Address A-1
289 output  [21:0]  oFLASH_A;                       //  FLASH Address bus 26 Bits
290 output          oFLASH_WE_N;                    //  FLASH Write Enable
291 output          oFLASH_RST_N;                   //  FLASH Reset
292 output          oFLASH_WP_N;                    //  FLASH Write Protect /Programming Acceleration
293 input           iFLASH_RY_N;                    //  FLASH Ready /Busy output
294 output          oFLASH_BYTE_N;                  //  FLASH Byte/ Word Mode Configuration
295 output          oFLASH_OE_N;                    //  FLASH Output Enable
296 output          oFLASH_CE_N;                    //  FLASH Chip Enable
297 ////////////////////////    SRAM Interface    ////////////////////////
298 inout   [31:0]  SRAM_DQ;                        //  SRAM Data Bus 32 Bits
299 inout   [3:0]   SRAM_DPA;                       //  SRAM Parity Data Bus
300 output  [18:0]  oSRAM_A;                        //  SRAM Address bus 21 Bits
```

```
301 output                  oSRAM_ADSC_N;          //      SRAM Controller
    Address Status
302 output                  oSRAM_ADSP_N;          //      SRAM Processor
    Address Status
303 output                  oSRAM_ADV_N;           //      SRAM Burst Address
    Advance
304 output  [3:0]   oSRAM_BE_N;            //      SRAM Byte Write Enable
305 output                  oSRAM_CE1_N;           //      SRAM Chip Enable
306 output                  oSRAM_CE2;             //      SRAM Chip Enable
307 output                  oSRAM_CE3_N;           //      SRAM Chip Enable
308 output                  oSRAM_CLK;             //      SRAM Clock
309 output                  oSRAM_GW_N;            //  SRAM Global Write
    Enable
310 output                  oSRAM_OE_N;            //      SRAM Output Enable
311 output                  oSRAM_WE_N;            //      SRAM Write Enable
312 ////////////////////    ISP1362 Interface      ////////////////////////
313 inout   [15:0]  OTG_D;                                 //      ISP1362
    Data bus 16 Bits
314 output  [1:0]   oOTG_A;                               //      ISP1362
    Address 2 Bits
315 output                  oOTG_CS_N;                          //
    ISP1362 Chip Select
316 output                  oOTG_OE_N;                          //
    ISP1362 Read
317 output                  oOTG_WE_N;                          //
    ISP1362 Write
318 output                  oOTG_RESET_N;                       //      ISP1362
    Reset
319 inout                   OTG_FSPEED;                         //      USB
     Full Speed, 0 = Enable, Z = Disable
320 inout                   OTG_LSPEED;                         //      USB
     Low Speed,  0 = Enable, Z = Disable
321 input                   iOTG_INT0;                          //
    ISP1362 Interrupt 0
322 input                   iOTG_INT1;                          //
    ISP1362 Interrupt 1
323 input                   iOTG_DREQ0;                         //
    ISP1362 DMA Request 0
324 input                   iOTG_DREQ1;                         //
    ISP1362 DMA Request 1
325 output                  oOTG_DACK0_N;                       //      ISP1362 DMA
     Acknowledge 0
326 output                  oOTG_DACK1_N;                       //      ISP1362 DMA
```

```
        Acknowledge 1
327 ////////////////////    LCD Module 16X2 ////////////////////////////
328 inout    [7:0]    LCD_D;                                    //      LCD Data
        bus 8 bits
329 output                oLCD_ON;                                  //      LCD
        Power ON/OFF
330 output                oLCD_BLON;                                //      LCD
        Back Light ON/OFF
331 output                oLCD_RW;                                  //      LCD
        Read/Write Select, 0 = Write, 1 = Read
332 output                oLCD_EN;                                  //      LCD
        Enable
333 output                oLCD_RS;                                  //      LCD
        Command/Data Select, 0 = Command, 1 = Data
334 ////////////////////    SD Card Interface       ////////////////////////
335 inout                 SD_DAT;                                   //      SD
        Card Data
336 inout                 SD_DAT3;                                  //      SD
        Card Data 3
337 inout                 SD_CMD;                                   //      SD
        Card Command Signal
338 output                oSD_CLK;                                  //      SD
        Card Clock
339 ////////////////////////        I2C
        ////////////////////////////////
340 inout                 I2C_SDAT;                                 //      I2C
        Data
341 output                oI2C_SCLK;                                //      I2C
        Clock
342 ////////////////////////        PS2
        ////////////////////////////////
343 inout                 PS2_KBDAT;                                //      PS2
        Keyboard Data
344 inout                 PS2_KBCLK;                                //      PS2
        Keyboard Clock
345 inout                 PS2_MSDAT;                                //      PS2
        Mouse Data
346 inout                 PS2_MSCLK;                                //      PS2
        Mouse Clock
347 ////////////////////////        VGA
        ////////////////////////////
348 output                oVGA_CLOCK;                       //      VGA Clock
349 output                oVGA_HS;                                  //      VGA
```

```
    H_SYNC
350 output                  oVGA_VS;                        //      VGA
    V_SYNC
351 output                  oVGA_BLANK_N;                   //      VGA BLANK
352 output                  oVGA_SYNC_N;                    //      VGA SYNC
353 output  [9:0]   oVGA_R;                                 //      VGA Red
    [9:0]
354 output  [9:0]   oVGA_G;                                 //      VGA Green
    [9:0]
355 output  [9:0]   oVGA_B;                                 //      VGA Blue
    [9:0]
356 ////////////////        Ethernet Interface
    ////////////////////////////
357 inout   [15:0]  ENET_D;                                 //      DM9000A
    DATA bus 16Bits
358 output                  oENET_CMD;                      //
    DM9000A Command/Data Select, 0 = Command, 1 = Data
359 output                  oENET_CS_N;                     //
    DM9000A Chip Select
360 output                  oENET_IOW_N;                    //      DM9000A
    Write
361 output                  oENET_IOR_N;                    //      DM9000A
    Read
362 output                  oENET_RESET_N;                  //      DM9000A
    Reset
363 input                   iENET_INT;                      //
    DM9000A Interrupt
364 output                  oENET_CLK;                      //
    DM9000A Clock 25 MHz
365 ////////////////////    Audio CODEC
    ////////////////////////////
366 inout                   AUD_ADCLRCK;                    //      Audio CODEC
    ADC LR Clock
367 input                   iAUD_ADCDAT;                    //      Audio CODEC
    ADC Data
368 inout                   AUD_DACLRCK;                    //      Audio CODEC
    DAC LR Clock
369 output                  oAUD_DACDAT;                    //      Audio CODEC
    DAC Data
370 inout                   AUD_BCLK;                       //
    Audio CODEC Bit-Stream Clock
371 output                  oAUD_XCK;                       //
    Audio CODEC Chip Clock
```

```
372 ////////////////////    TV Devoder
        ////////////////////////////
373 input                   iTD1_CLK27;                         //      TV
        Decoder1 Line_Lock Output Clock
374 input    [7:0]    iTD1_D;                                   //      TV Decoder1
         Data bus 8 bits
375 input                   iTD1_HS;                            //      TV
        Decoder1 H_SYNC
376 input                   iTD1_VS;                            //      TV
        Decoder1 V_SYNC
377 output                  oTD1_RESET_N;                       //      TV Decoder1
         Reset
378 input                   iTD2_CLK27;                         //      TV
        Decoder2 Line_Lock Output Clock
379 input    [7:0]    iTD2_D;                                   //      TV Decoder2
         Data bus 8 bits
380 input                   iTD2_HS;                            //      TV
        Decoder2 H_SYNC
381 input                   iTD2_VS;                            //      TV
        Decoder2 V_SYNC
382 output                  oTD2_RESET_N;                       //      TV Decoder2
         Reset
383
384 ////////////////////////        GPIO    ////////////////////////////////
385 inout    [31:0]   GPIO_0;                                   //      GPIO
        Connection 0 I/O
386 input                   GPIO_CLKIN_N0;                      //      GPIO
        Connection 0 Clock Input 0
387 input                   GPIO_CLKIN_P0;          //      GPIO Connection 0
        Clock Input 1
388 inout                   GPIO_CLKOUT_N0;         //      GPIO Connection 0
        Clock Output 0
389 inout                   GPIO_CLKOUT_P0;         //      GPIO Connection 0
        Clock Output 1
390 inout    [31:0]   GPIO_1;                                   //      GPIO
        Connection 1 I/O
391 input                   GPIO_CLKIN_N1;          //      GPIO Connection 1
        Clock Input 0
392 input                   GPIO_CLKIN_P1;          //      GPIO Connection 1
        Clock Input 1
393 inout                   GPIO_CLKOUT_N1;         //      GPIO Connection 1
        Clock Output 0
394 inout                   GPIO_CLKOUT_P1;         //      GPIO Connection 1
```

```
    Clock Output 1
////////////////////////////////////////////////////////////////////
//
    ============================================================================

// REG/WIRE declarations
//
    ============================================================================


//      CCD
wire    [11:0]  CCD_DATA;
wire            CCD_SDAT;
wire            CCD_SCLK;
wire            CCD_FLASH;
wire            CCD_FVAL;
wire            CCD_LVAL;
wire            CCD_PIXCLK;
wire            CCD_MCLK;       //      CCD Master Clock

wire    [15:0]  Read_DATA1;
wire    [15:0]  Read_DATA2;
wire            VGA_CTRL_CLK;
wire    [11:0]  mCCD_DATA;
wire            mCCD_DVAL;
wire            mCCD_DVAL_d;
wire    [15:0]  X_Cont;
wire    [15:0]  Y_Cont;
wire    [9:0]   X_ADDR;
wire    [31:0]  Frame_Cont;
wire            DLY_RST_0;
wire            DLY_RST_1;
wire            DLY_RST_2;
wire            Read;
reg             [11:0]  rCCD_DATA;
reg                     rCCD_LVAL;
reg                     rCCD_FVAL;
wire    [11:0]  sCCD_R;
wire    [11:0]  sCCD_G;
wire    [11:0]  sCCD_B;

////////////////////////////////////////////////////////////////////YGO
    ////////////////////////
```

```
wire     [11:0]  sCCD_G_SDRAM; //Ygo
wire     [11:0]  sCCD_B_SDRAM; //Ygo
wire     [11:0]  sCCD_R_SDRAM; //Ygo
wire                          pix_bin; //Ygo
wire     [31:0]  signatures;
wire     [9:0]           signature_a;
wire     [9:0]           signature_b;
wire     [9:0]           signature_c;
wire     [9:0]           signature_d;
wire     [9:0]           signature_l;
wire     [9:0]           signature_r;
wire     [9:0]           signature_val;
wire                          pix_posedge;
wire                          pix_negedge;
wire     [11:0]  AutoThreshold;
wire     [9:0]           signature_l_min;
wire     [9:0]           signature_l_max;
wire     [9:0]           signature_r_min;
wire     [9:0]           signature_r_max;
wire                          result;
//wire                        CCD_START;
//wire                        CCD_END;
assign sCCD_G_SDRAM = iSW[1] ? {pix_posedge,pix_posedge,pix_posedge,
    pix_posedge,pix_posedge,pix_posedge,pix_posedge,pix_posedge,pix_posedge,
    pix_posedge,pix_posedge,pix_posedge} : sCCD_G; //Ygo  12'd0 : sCCD_G
assign sCCD_B_SDRAM = iSW[1] ? {pix_negedge,pix_negedge,pix_negedge,
    pix_negedge,pix_negedge,pix_negedge,pix_negedge,pix_negedge,pix_negedge,
    pix_negedge,pix_negedge,pix_negedge} : sCCD_B; //Ygo  12'd0 : sCCD_B
assign sCCD_R_SDRAM = iSW[2] ? {pix_bin,pix_bin,pix_bin,pix_bin,pix_bin,
    pix_bin,pix_bin,pix_bin,pix_bin,pix_bin,pix_bin,pix_bin} : sCCD_R; //Ygo
assign signatures[7:0] = signature_d[9:4];
assign signatures[15:8] = signature_c[9:4];
assign signatures[23:16] = signature_b[9:4];
assign signatures[31:24] = signature_a[9:4];
//
    ////////////////////////////////////////////////////////////////////////////


wire                    sCCD_DVAL;
wire     [9:0]  oVGA_R;                                  //      VGA Red
    [9:0]
wire     [9:0]  oVGA_G;                                  //      VGA Green
    [9:0]
```

```
466 wire    [9:0]   oVGA_B;                                 //      VGA Blue
        [9:0]
467 reg             [1:0]   rClk;
468 wire                    sdram_ctrl_clk;
469
470 //
        ============================================================================

471 // Structural coding
472 //
        ============================================================================

473
474 assign  CCD_DATA[0]     =       GPIO_1[11];
475 assign  CCD_DATA[1]     =       GPIO_1[10];
476 assign  CCD_DATA[2]     =       GPIO_1[9];
477 assign  CCD_DATA[3]     =       GPIO_1[8];
478 assign  CCD_DATA[4]     =       GPIO_1[7];
479 assign  CCD_DATA[5]     =       GPIO_1[6];
480 assign  CCD_DATA[6]     =       GPIO_1[5];
481 assign  CCD_DATA[7]     =       GPIO_1[4];
482 assign  CCD_DATA[8]     =       GPIO_1[3];
483 assign  CCD_DATA[9]     =       GPIO_1[2];
484 assign  CCD_DATA[10]=   GPIO_1[1];
485 assign  CCD_DATA[11]=   GPIO_1[0];
486 assign  GPIO_CLKOUT_N1  =       CCD_MCLK;
487 assign  CCD_FVAL        =       GPIO_1[18];
488 assign  CCD_LVAL        =       GPIO_1[17];
489 assign  CCD_PIXCLK      =       GPIO_CLKIN_N1;
490 assign  GPIO_1[15]      =       1'b1;   // Trigger
491 assign  GPIO_1[14]      =       DLY_RST_1;
492
493 assign  oLEDR           =       iSW;
494 assign  oLEDG[7:0]              =       Y_Cont[8:1];    // Ygo
495 assign  oLEDG[8]        = result;       //Ygo
496
497 assign  oTD1_RESET_N = 1'b1;
498 assign  oVGA_CLOCK      =       ~VGA_CTRL_CLK;
499
500 always@(posedge iCLK_50)        rClk    <=      rClk+1;
501
502 always@(posedge CCD_PIXCLK)
503 begin
```

```
504         rCCD_DATA       <=      CCD_DATA;
505         rCCD_LVAL       <=      CCD_LVAL;
506         rCCD_FVAL       <=      CCD_FVAL;
507 end
508
509
510 Reset_Delay                     u1      (       .iCLK(iCLK_50),
511                 .iRST(iKEY[0]),
512                 .oRST_0(DLY_RST_0),
513                 .oRST_1(DLY_RST_1),
514                 .oRST_2(DLY_RST_2)
515         );
516
517
518 CCD_Capture                     u2      (       .oDATA(mCCD_DATA),
519                 .oDVAL(mCCD_DVAL),
520                 .oX_Cont(X_Cont),
521                 .oY_Cont(Y_Cont),           // at 01DFh
522                 .oFrame_Cont(Frame_Cont),
523                 .iDATA(rCCD_DATA),
524                 .iFVAL(rCCD_FVAL),
525                 .iLVAL(rCCD_LVAL),
526                 .iSTART(!iKEY[3]),
527                 .iEND(!iKEY[2]),
528                 .iCLK(CCD_PIXCLK),
529                 .iRST(DLY_RST_2)
530         );
531
532 RAW2RGB                         u3      (  .iCLK(CCD_PIXCLK),
533                 .iRST_n(DLY_RST_1),
534                 .iData(mCCD_DATA),
535                 .iDval(mCCD_DVAL),
536                 .oRed(sCCD_R),
537                 .oGreen(sCCD_G),
538                 .oBlue(sCCD_B),
539                 .oDval(sCCD_DVAL),
540                 .iMIRROR(1'b1),   //Ygo   iSW[17]
541                 .iX_Cont(X_Cont),
542                 .iY_Cont(Y_Cont)
543         );
544
545 comparador                      u12 (    .iRed(sCCD_R[11:0]),   //Ygo
546                 .ithreshold(iSW[17:10]),   //Ygo
```

```
547                 .CCD_PIXCLK(CCD_PIXCLK),
548                 .iAutoThreshold (iSW[09]),
549                 .iFval (rCCD_FVAL),
550                 .AutoThreshold (AutoThreshold),
551                 .opix_bin(pix_bin)  //Ygo
552         ); //Ygo
553
554 signature_generation u13 (
555         .CCD_PIXCLK(CCD_PIXCLK),
556         .ipix_bin(pix_bin),
557         .iDval(sCCD_DVAL),
558         .signature_a(signature_a),
559         .signature_b(signature_b),
560         .signature_c(signature_c),
561         .signature_d(signature_d),
562         .pix_posedge(pix_posedge),
563         .pix_negedge(pix_negedge)
564         );
565
566 signature_filter                u14 (.iSignature_a(signature_a),
567          .iSignature_b(signature_b),
568          .iSignature_c(signature_c),
569          .iSignature_d(signature_d),
570          .iY_Cont(Y_Cont),
571          .iDval (sCCD_DVAL),
572          .iCCD_PIXCLK (CCD_PIXCLK),
573          .oSignature_l (signature_l),
574          .oSignature_r (signature_r),
575          .oSignature_val (signature_val)
576          );
577
578 signature_reference     u15 (
579          .iSignature_st (iSW[4]),
580          .iSignature_l (signature_l),
581          .iSignature_r (signature_r),
582          .iSignature_val (signature_val),
583          .iFval (rCCD_FVAL),
584          .oSignature_l_min (signature_l_min),
585          .oSignature_l_max (signature_l_max),
586          .oSignature_r_min (signature_r_min),
587          .oSignature_r_max (signature_r_max),
588          .oResult (result)
589          );
```

```
590
591 SEG7_LUT_8                    u4      (       .oSEG0(oHEX0_D),.oSEG1(
        oHEX1_D),
592         .oSEG2(oHEX2_D),.oSEG3(oHEX3_D),
593         .oSEG4(oHEX4_D),.oSEG5(oHEX5_D),
594         .oSEG6(oHEX6_D),.oSEG7(oHEX7_D),
595         .iDIG({8'b0,iSW[17:10],4'b0,AutoThreshold[11:0]}) ); //Frame_Cont
              [31:0] //Ygo
596
597 vga_pll                        u5      (       .inclk0(iCLK_50_2),
598                 .c0(ltm_nclk),
599                 .c1()
600         );
601
602
603 sdram_pll                      u6      (       .inclk0(iCLK_50_3),
604                 .c0(sdram_ctrl_clk),
605                 .c1(oDRAM0_CLK),
606                 .c2(oDRAM1_CLK)
607         );
608
609 assign CCD_MCLK = rClk[0];
610
611 Sdram_Control_4Port     u7      (       //      HOST Side
612                  .REF_CLK(iCLK_50),    //iCLK_50
613                  .RESET_N(1'b1),
614                 .CLK(sdram_ctrl_clk),
615
616                 //      FIFO Write Side 1
617                  .WR1_DATA({sCCD_G_SDRAM[11:7],  sCCD_B_SDRAM[11:2]}), //
                        Ygo
618                 .WR1(sCCD_DVAL),
619                 .WR1_ADDR(0),
620                 .WR1_MAX_ADDR(800*480),
621                 .WR1_LENGTH(9'h100),
622                 .WR1_LOAD(!DLY_RST_0),
623                 .WR1_CLK(CCD_PIXCLK),
624
625                 //      FIFO Read Side 1
626                  .RD1_DATA(Read_DATA1),
627                 .RD1(Read),
628                 .RD1_ADDR(0),
629                 .RD1_MAX_ADDR(800*480),
```

```
630                     .RD1_LENGTH(9'h100),
631                     .RD1_LOAD(!DLY_RST_0),
632                     .RD1_CLK(~ltm_nclk),
633
634                     //      SDRAM Side
635                       .SA(oDRAM0_A[11:0]),
636                       .BA(oDRAM0_BA),
637                     .CS_N(oDRAM0_CS_N),
638                     .CKE(oDRAM0_CKE),
639                     .RAS_N(oDRAM0_RAS_N),
640                     .CAS_N(oDRAM0_CAS_N),
641                     .WE_N(oDRAM0_WE_N),
642                     .DQ(DRAM_DQ[15:0]),
643                     .DQM({oDRAM0_UDQM1,oDRAM0_LDQM0})
644             );
645
646 Sdram_Control_4Port     u8      (       //      HOST Side
647                       .REF_CLK(iCLK_50), //iCLK_50
648                       .RESET_N(1'b1),
649                     .CLK(sdram_ctrl_clk),
650
651                     //      FIFO Write Side 1
652                       .WR1_DATA({sCCD_G_SDRAM[6:2], sCCD_R_SDRAM[11:2]}), //Ygo
653                     .WR1(sCCD_DVAL),
654                     .WR1_ADDR(0),
655                     .WR1_MAX_ADDR(800*480),
656                     .WR1_LENGTH(9'h100),
657                     .WR1_LOAD(!DLY_RST_0),
658                     .WR1_CLK(CCD_PIXCLK),
659
660                     //      FIFO Read Side 1
661                       .RD1_DATA(Read_DATA2),
662                     .RD1(Read),
663                     .RD1_ADDR(0),
664                     .RD1_MAX_ADDR(800*480),
665                     .RD1_LENGTH(9'h100),
666                     .RD1_LOAD(!DLY_RST_0),
667                     .RD1_CLK(~ltm_nclk),
668
669                     //      SDRAM Side
670                       .SA(oDRAM1_A[11:0]),
671                       .BA(oDRAM1_BA),
672                     .CS_N(oDRAM1_CS_N),
```

```
673                     .CKE(oDRAM1_CKE),
674                     .RAS_N(oDRAM1_RAS_N),
675                     .CAS_N(oDRAM1_CAS_N),
676                     .WE_N(oDRAM1_WE_N),
677                     .DQ(DRAM_DQ[31:16]),
678                     .DQM({oDRAM1_UDQM1,oDRAM1_LDQM0})
679             );
680
681     assign  oUART_TXD = iUART_RXD;
682
683     I2C_CCD_Config          u9      (       //      Host Side
684                     .iCLK(iCLK_50),
685                     .iRST_N(DLY_RST_1),
686                     .iEXPOSURE_ADJ(iKEY[1]),
687                     .iEXPOSURE_DEC_p(iSW[0]),
688                     .iMIRROR_SW(1'b1), //Ygo  iSW[17]
689                     .ipattern_SW (iSW[3]), // Ygo
690                     //      I2C Side
691                     .I2C_SCLK(GPIO_1[20]),
692                     .I2C_SDAT(GPIO_1[19])
693
694             );
695
696     touch_tcon                      u10     ( .iCLK(ltm_nclk),
697             .iRST_n(DLY_RST_2),
698             // sdram side
699             .iREAD_DATA1(Read_DATA1),
700             .iREAD_DATA2(Read_DATA2),
701             .oREAD_SDRAM_EN(Read),
702             // lcd side
703             .oLCD_R(ltm_r),
704             .oLCD_G(ltm_g),
705             .oLCD_B(ltm_b),
706             .oHD(ltm_hd),
707             .oVD(ltm_vd),
708             .oDEN(ltm_den)
709             );
710
711     lcd_3wire_config        u11     (
712                                                             // Host Side
713             .iCLK(iCLK_50),                 //iCLK_50
714             .iRST_n(DLY_RST_0),
715             // 3 wire Side
```

```
716             .o3WIRE_SCLK(ltm_sclk),
717             .io3WIRE_SDAT(ltm_sda),
718             .o3WIRE_SCEN(ltm_scen),
719             .o3WIRE_BUSY_n(ltm_3wirebusy_n)
720             );
721
722     assign  adc_penirq_n  =GPIO_CLKIN_N0;
723     assign  adc_dout      =GPIO_0[0];
724     assign  adc_busy      =GPIO_CLKIN_P0;
725     assign  GPIO_0[1]         =adc_din;
726     assign  GPIO_0[2]         =adc_ltm_sclk;
727     assign  GPIO_0[3]         =ltm_b[3];
728     assign  GPIO_0[4]         =ltm_b[2];
729     assign  GPIO_0[5]         =ltm_b[1];
730     assign  GPIO_0[6]         =ltm_b[0];
731     assign  GPIO_0[7]         =~ltm_nclk;
732     assign  GPIO_0[8]         =ltm_den;
733     assign  GPIO_0[9]         =ltm_hd;
734     assign  GPIO_0[10]        =ltm_vd;
735     assign  GPIO_0[11]        =ltm_b[4];
736     assign  GPIO_0[12]        =ltm_b[5];
737     assign  GPIO_0[13]        =ltm_b[6];
738     assign  GPIO_CLKOUT_N0    =ltm_b[7];
739     assign  GPIO_0[14]        =ltm_g[0];
740     assign  GPIO_CLKOUT_P0    =ltm_g[1];
741     assign  GPIO_0[15]        =ltm_g[2];
742     assign  GPIO_0[16]        =ltm_g[3];
743     assign  GPIO_0[17]        =ltm_g[4];
744     assign  GPIO_0[18]        =ltm_g[5];
745     assign  GPIO_0[19]        =ltm_g[6];
746     assign  GPIO_0[20]        =ltm_g[7];
747     assign  GPIO_0[21]        =ltm_r[0];
748     assign  GPIO_0[22]        =ltm_r[1];
749     assign  GPIO_0[23]        =ltm_r[2];
750     assign  GPIO_0[24]        =ltm_r[3];
751     assign  GPIO_0[25]        =ltm_r[4];
752     assign  GPIO_0[26]        =ltm_r[5];
753     assign  GPIO_0[27]        =ltm_r[6];
754     assign  GPIO_0[28]        =ltm_r[7];
755     assign  GPIO_0[29]        =ltm_grst;
756     assign  GPIO_0[30]        =ltm_scen;
757     assign  GPIO_0[31]        =ltm_sda;
758
```

```
759 assign ltm_grst          = iKEY[0];
760 assign adc_ltm_sclk      = ltm_sclk ;
761
762 // Touch panel signal //
763 wire     [7:0]   ltm_r;          //      LTM Red Data 8 Bits
764 wire     [7:0]   ltm_g;          //      LTM Green Data 8 Bits
765 wire     [7:0]   ltm_b;          //      LTM Blue Data 8 Bits
766 wire             ltm_nclk;       //      LTM Clcok
767 wire             ltm_hd;
768 wire             ltm_vd;
769 wire             ltm_den;
770 wire             adc_dclk;
771 wire             adc_cs;
772 wire             adc_penirq_n;
773 wire             adc_busy;
774 wire             adc_din;
775 wire             adc_dout;
776 wire             adc_ltm_sclk;
777 wire             ltm_grst;
778 // LTM Config//
779 wire             ltm_sclk;
780 wire             ltm_sda;
781 wire             ltm_scen;
782 wire             ltm_3wirebusy_n;
783
784 endmodule
```

Code C.2: Configuração da câmera via I2C

```
1  // --------------------------------------------------------------------
2  // Copyright (c) 2008 by Terasic Technologies Inc.
3  // --------------------------------------------------------------------
4  //
5  // Permission:
6  //
7  //   Terasic grants permission to use and modify this code for use
8  //   in synthesis for all Terasic Development Boards and Altera Development
9  //   Kits made by Terasic.  Other use of this code, including the selling
10 //   ,duplication, or modification of any portion is strictly prohibited.
11 //
12 // Disclaimer:
13 //
14 //   This VHDL/Verilog or C/C++ source code is intended as a design
       reference
```

```
15 //   which illustrates how these types of functions can be implemented.
16 //   It is the user's responsibility to verify their design for
17 //   consistency and functionality through the use of formal
18 //   verification methods.  Terasic provides no warranty regarding the use
19 //   or functionality of this code.
20 //
21 // ------------------------------------------------------------------------
22 //
23 //                     Terasic Technologies Inc
24 //                     356 Fu-Shin E. Rd Sec. 1. JhuBei City,
25 //                     HsinChu County, Taiwan
26 //                     302
27 //
28 //                     web: http://www.terasic.com/
29 //                     email: support@terasic.com
30 //
31 // ------------------------------------------------------------------------
32 //
33 // Major Functions:       I2C_CCD_Config
34 //
35 // ------------------------------------------------------------------------
36 //
37 // Revision History :
38 // ------------------------------------------------------------------------
39 //   Ver  :| Author                    :| Mod. Date :| Changes Made:
40 //   V1.0 :| Johnny Fan                :| 08/04/16  :| Initial
       Revision
41 // ------------------------------------------------------------------------
42
43 module I2C_CCD_Config ( //      Host Side
44           iCLK,
45           iRST_N,
46           iEXPOSURE_ADJ,
47           iEXPOSURE_DEC_p,
48           iMIRROR_SW,
49           ipattern_SW,
50           //      I2C Side
51           I2C_SCLK,
52           I2C_SDAT
53           );
54
55 //      Host Side
56 input                     iCLK;
```

```
57  input                   iRST_N;
58
59  //      I2C Side
60  output                  I2C_SCLK;
61  inout                   I2C_SDAT;
62
63  //      Internal Registers/Wires
64  reg     [15:0]          mI2C_CLK_DIV;
65  reg     [31:0]          mI2C_DATA;
66  reg                             mI2C_CTRL_CLK;
67  reg                             mI2C_GO;
68  wire                    mI2C_END;
69  wire                    mI2C_ACK;
70  reg     [23:0]          LUT_DATA;
71  reg     [5:0]           LUT_INDEX;
72  reg     [3:0]           mSetup_ST;
73
74  //////////////   CMOS sensor registers setting //////////////////////
75
76  input           iEXPOSURE_ADJ;
77  input           iEXPOSURE_DEC_p;
78
79  input           iMIRROR_SW;
80  input           ipattern_SW;
81
82  parameter       default_exposure                        = 16'h0400; //Ygo
        Original 16'h0500
83  parameter       exposure_change_value           = 16'd30;
84
85  reg     [24:0]  combo_cnt;
86  wire            combo_pulse;
87
88  reg     [3:0]   iexposure_adj_delay;
89  wire            exposure_adj_set;
90  wire            exposure_adj_reset;
91  reg     [15:0]  senosr_exposure;
92
93
94  wire [23:0] sensor_start_row;
95  wire [23:0] sensor_start_column;
96  wire [23:0] sensor_row_size;
97  wire [23:0] sensor_column_size;
98  wire [23:0] sensor_row_mode;
```

```
99  wire [23:0] sensor_column_mode;
100 wire [23:0] Mirror_d;
101 wire [23:0] test_pattern_control; //Ygo
102
103 assign sensor_start_row                = 24'h010036;
104 assign sensor_start_column             = 24'h020010;
105 assign sensor_row_size                 = 24'h0303BF;
106 assign sensor_column_size              = 24'h04063F;
107 assign sensor_row_mode                 = 24'h220011;
108 assign sensor_column_mode              = 24'h230011;
109
110 assign Mirror_d                                = iMIRROR_SW ?  24'h204000
    : 24'h20c000;
111 assign test_pattern_control            = ipattern_SW ? 24'hA00000 : 24'
    hA00039; // Ygo //barras verticais = A00039 0001 1001
112
113 always@(posedge iCLK or negedge iRST_N)
114         begin
115                 if (!iRST_N)
116                         begin
117                                 iexposure_adj_delay <= 0;
118                         end
119                 else
120                         begin
121                                 iexposure_adj_delay <= {iexposure_adj_delay
                                    [2:0],iEXPOSURE_ADJ};
122                         end
123         end
124
125 assign  exposure_adj_set = ({iexposure_adj_delay[0],iEXPOSURE_ADJ}==2'b10)
    ? 1 : 0 ;
126 assign  exposure_adj_reset = ({iexposure_adj_delay[3:2]}==2'b10) ? 1 : 0 ;
127
128 always@(posedge iCLK or negedge iRST_N)
129 begin
130         if (!iRST_N)
131                 senosr_exposure <= default_exposure;
132         else if (exposure_adj_set|combo_pulse)
133 begin
134         if (iEXPOSURE_DEC_p)
135                 begin
136                         if ((senosr_exposure < exposure_change_value)||
137                                 (senosr_exposure == 16'h0))
```

```
138                                 senosr_exposure <= 0;
139                         else
140                                 senosr_exposure <= senosr_exposure -
                                        exposure_change_value;
141                 end
142         else
143                 begin
144                         if (((16'hffff -senosr_exposure) <
                                    exposure_change_value)||
145                                 (senosr_exposure == 16'hffff))
146                                 senosr_exposure <= 16'hffff;
147                         else
148                                 senosr_exposure <= senosr_exposure +
                                        exposure_change_value;
149                 end
150 end
151 end
152
153
154 always@(posedge iCLK or negedge iRST_N)
155         begin
156         if (!iRST_N)
157                 combo_cnt <= 0;
158         else if (!iexposure_adj_delay[3])
159                 combo_cnt <= combo_cnt + 1;
160         else
161                 combo_cnt <= 0;
162         end
163
164 assign combo_pulse = (combo_cnt == 25'h1fffff) ? 1 : 0;
165
166 wire    i2c_reset;
167
168 assign i2c_reset = iRST_N & ~exposure_adj_reset & ~combo_pulse ;
169
170 ////////////////////////////////////////////////////////////////////
171
172 //      Clock Setting
173 parameter       CLK_Freq        =       50000000;       //      50      MHz
174 parameter       I2C_Freq        =       20000;          //      20      KHz
175 //      LUT Data Number
176 parameter       LUT_SIZE        =       27;  //Ygo antes 25
177
```

```
178 ////////////////////   I2C Control Clock        ////////////////////////
179 always@(posedge iCLK or negedge i2c_reset)
180 begin
181         if(!i2c_reset)
182         begin
183                 mI2C_CTRL_CLK   <=      0;
184                 mI2C_CLK_DIV    <=      0;
185         end
186         else
187         begin
188                 if( mI2C_CLK_DIV        < (CLK_Freq/I2C_Freq) )
189                 mI2C_CLK_DIV    <=      mI2C_CLK_DIV+1;
190                 else
191                 begin
192                         mI2C_CLK_DIV    <=      0;
193                         mI2C_CTRL_CLK   <=      ~mI2C_CTRL_CLK;
194                 end
195         end
196 end
197 ////////////////////////////////////////////////////////////////////
198 I2C_Controller  u0      (       .CLOCK(mI2C_CTRL_CLK),          //
        Controller Work Clock
199         .I2C_SCLK(I2C_SCLK),                //      I2C CLOCK
200         .I2C_SDAT(I2C_SDAT),                //      I2C DATA
201         .I2C_DATA(mI2C_DATA),               //      DATA:[SLAVE_ADDR,SUB_ADDR,
              DATA]
202         .GO(mI2C_GO),                               //      GO transfor
203         .END(mI2C_END),                             //      END transfor
204         .ACK(mI2C_ACK),                             //      ACK
205         .RESET(i2c_reset)       );
206 ////////////////////////////////////////////////////////////////////
207 ////////////////////  Config Control  ////////////////////////////
208 //always@(posedge  or negedge iRST_N)
209 always@(posedge mI2C_CTRL_CLK or negedge i2c_reset)
210 begin
211 if(!i2c_reset)
212 begin
213         LUT_INDEX       <=      0;
214         mSetup_ST       <=      0;
215         mI2C_GO         <=      0;
216
217 end
218 else if(LUT_INDEX<LUT_SIZE)
```

```
219 begin
220 case(mSetup_ST)
221 0:        begin
222                     mI2C_DATA       <=      {8'hBA,LUT_DATA};
223                     mI2C_GO         <=      1;
224                     mSetup_ST       <=      1;
225         end
226 1:        begin
227                     if(mI2C_END)
228                     begin
229                             if(!mI2C_ACK)
230                             mSetup_ST       <=      2;
231                             else
232                             mSetup_ST       <=      0;
233                             mI2C_GO         <=      0;
234                     end
235         end
236 2:        begin
237                     LUT_INDEX       <=      LUT_INDEX+1;
238                     mSetup_ST       <=      0;
239         end
240 endcase
241 end
242 end
243 ////////////////////////////////////////////////////////////////////
244 ////////////////////    Config Data LUT    //////////////////////////
245 always
246 begin
247         case(LUT_INDEX)
248         0       :       LUT_DATA        <=      24'h000000;
249         1       :       LUT_DATA        <=      Mirror_d;
                                //      Mirror Row and Columns
250         2       :       LUT_DATA        <=      {8'h09,senosr_exposure};//
                    Exposure
251         3       :       LUT_DATA        <=      24'h050000;
                                //      H_Blanking
252         4       :       LUT_DATA        <=      24'h060019;
                                //      V_Blanking
253         5       :       LUT_DATA        <=      24'h0A8000;
                                //      change latch
254         6       :       LUT_DATA        <=      24'h2B0033;
                                //      Green 1 Gain
```

```
255         7       :       LUT_DATA        <=      24'h2C0135;
                            //      Blue Gain
256         8       :       LUT_DATA        <=      24'h2D0339;
                            //      Red Gain
257         9       :       LUT_DATA        <=      24'h2E0033;
                            //      Green 2 Gain
258         10      :       LUT_DATA        <=      24'h100051;
                            //      set up PLL power on
259         11      :       LUT_DATA        <=      24'h111804;
                            //      PLL_m_Factor<<8+PLL_n_Divider
260         12      :       LUT_DATA        <=      24'h120001;
                            //      PLL_p1_Divider
261         13      :       LUT_DATA        <=      24'h100053;
                            //      set USE PLL
262         14      :       LUT_DATA        <=      24'h980000;
                            //      disble calibration
263         15      :       LUT_DATA        <=      test_pattern_control;
                    //      Test pattern control        //Ygo 24'hA00000
    0011 1001 0x39
264         16      :       LUT_DATA        <=      24'hA10A00;
                            //      Test green pattern value //Ygo
265         17      :       LUT_DATA        <=      24'hA20000;
                            //      Test red pattern value //Ygo
266         18      :       LUT_DATA        <=      24'hA30D00;
                            //      Test blue pattern value         //Ygo
267         19      :       LUT_DATA        <=      24'hA400AF;
                            //      Test pattern bar width //Ygo
268         20      :       LUT_DATA        <=      sensor_start_row        ;
                    //      set start row
269         21      :       LUT_DATA        <=      sensor_start_column ;   //
                  set start column
270         22      :       LUT_DATA        <=      sensor_row_size;
                    //      set row size to
271         23      :       LUT_DATA        <=      sensor_column_size;
                    //      set column size to 2047
272         24      :       LUT_DATA        <=      sensor_row_mode;
                    //      set row mode in bin mode
273         25      :       LUT_DATA        <=      sensor_column_mode;
                    //      set column mode  in bin mode
274         26      :       LUT_DATA        <=      24'h4901A8;
                            //      row black target
275         default:LUT_DATA        <=      24'h000000;
276         endcase
```

```
277 end
278
279 endmodule
```

Code C.3: Etapa de limiarização

```
1  module comparador (
2          input[11:0] iRed,
3          input[7:0] ithreshold,
4          input CCD_PIXCLK,
5          input iFval,
6          input iAutoThreshold,
7          output reg [11:0] AutoThreshold,
8          output reg opix_bin
9          );
10
11 reg [18:0] ContPIX;
12
13 //assign opix_bin = (iRed > {ithreshold[9:0],2'b0}) ? 1'b1 : 1'b0;
14
15 always @ (posedge CCD_PIXCLK)
16         begin
17
18         if (iAutoThreshold == 1'b0)
19         begin
20         if (iRed > {ithreshold[7:0],4'b0})
21                 opix_bin <= 1;
22         else
23                 opix_bin <= 0;
24         end
25         else
26         begin
27         if (iRed > AutoThreshold)
28                 opix_bin <= 1;
29         else
30                 opix_bin <= 0;
31         end
32
33         if (iFval == 1'b0)
34         begin
35         ContPIX <= 0;
36         end
37         else
38         begin
```

```
39          if (opix_bin == 1'b1)
40          ContPIX <= ContPIX + 1;
41          end
42
43          end
44
45  always @ (negedge iFval)
46          if (ContPIX > {ithreshold[7:0],11'b0})
47          AutoThreshold <= AutoThreshold + 1;
48          else
49          AutoThreshold <= AutoThreshold - 1;
50
51  endmodule
```

Code C.4: Geração da assinatura - Detecção das transições relevantes

```
1   module signature_generation (
2           input CCD_PIXCLK,
3           input ipix_bin,
4           input iDval,
5           output reg [9:0] signature_a, //primeiro posedge
6           output reg [9:0] signature_b, //ultimo negedge
7           output reg [9:0] signature_c, //primeiro negedge
8           output reg [9:0] signature_d,
9           output reg pix_posedge,
10          output reg pix_negedge
11          ); //ultimo posedge
12
13          reg [9:0]cont;
14          reg [9:0]signature_a1;
15          reg [9:0]signature_b1;
16          reg [9:0]signature_c1;
17          reg [9:0]signature_d1;
18          reg ipix_bina;
19
20
21          always @ (posedge CCD_PIXCLK)
22          begin
23          if (iDval)
24                  begin
25
26                  if (signature_d1 == (cont - 1))
27                  begin
28                  if (cont != 0)
```

```
29                pix_posedge <= 1'b1;
30                end
31                else
32                pix_posedge <= 1'b0;
33
34                if (signature_b1 == (cont - 1))
35                pix_negedge <= 1'b1;
36                else
37                pix_negedge <= 1'b0;
38
39                cont <= cont + 1;
40                if (cont == 10'd0)
41                        begin
42                        signature_a <= 10'd0;
43                        signature_b <= 10'd0;
44                        signature_c <= 10'd0;
45                        signature_d <= 10'd0;
46                        end
47                        else
48                        begin
49                                if (pix_posedge)
50                                begin
51                                signature_a <= signature_a1;
52                                signature_d <= signature_d1;
53                                end
54                                if (pix_negedge)
55                                begin
56                                signature_c <= signature_c1;
57                                signature_b <= signature_b1;
58                                end
59                        end
60                end
61        else
62                begin
63                cont <= 0;
64                signature_a1 <= 10'd810;
65                signature_b1 <= 10'd810;
66                signature_c1 <= 10'd810;
67                signature_d1 <= 10'd810;
68                signature_a <= 10'd0;
69                signature_b <= 10'd0;
70                signature_c <= 10'd0;
71                signature_d <= 10'd0;
```

```
72                  end
73
74          if (!ipix_bina)
75          if (ipix_bin)
76  //always @(posedge ipix_bin)
77          begin
78          signature_d1 <= cont;
79          if (signature_a == 10'd0)
80          signature_a1 <= cont;
81          end
82
83          if (ipix_bina)
84          if (!ipix_bin)
85  //      always @(negedge ipix_bin)
86          begin
87          signature_b1 <= cont;
88          if (signature_c == 10'd0)
89          signature_c1 <= cont;
90          end
91
92          ipix_bina <= ipix_bin;
93
94          end
95
96          endmodule
```

Code C.5: Geração da assinatura - Detecção das regiões de interesse e construção das assinaturas

```
1  module signature_filter (
2          input [9:0] iSignature_a,
3          input [9:0] iSignature_b,
4          input [9:0] iSignature_c,
5          input [9:0] iSignature_d,
6          input iDval,
7          input [15:0] iY_Cont,
8          input iCCD_PIXCLK,
9          output reg [9:0] oSignature_l, //a - c
10         output reg [9:0] oSignature_r, //b - d
11         output oSignature_val
12         );
13
14 reg [7:0] cont1;
```

```
15 reg [7:0] cont2;
16 reg flag_top, flag_bottom, flag_bottom2, flag_body, flag_null, iDval_old;
17 reg PIXCLK;
18 reg [9:0] Signature_a;
19 reg [9:0] Signature_b;
20 reg [9:0] Signature_c;
21 reg [9:0] Signature_d;
22 reg oldnull;
23 reg [15:0] Y_Cont_bottom;
24 wire Sig_val_tmp1;
25 reg Sig_val_tmp2;
26 reg [9:0] Signature_ml [0:49];
27 reg [9:0] Signature_mr [0:49];
28 reg [7:0] index;
29 wire matrix_sig_val;
30
31 always @ (negedge iDval)
32         begin
33
34         if (iY_Cont == 16'd1)
35                 begin
36                 cont1 <= 0;
37                 flag_top <= 0;
38                 flag_null <= 0;
39                 flag_bottom <= 0;
40                 flag_body <= 0;
41                 end
42
43         Signature_a <= iSignature_a;
44         Signature_b <= iSignature_b;
45         Signature_c <= iSignature_c;
46         Signature_d <= iSignature_d;
47
48         if (flag_top == 0)
49                 if (cont1 == 0)
50                         if (iSignature_a != 0)
51                         flag_top <= 1;
52
53         if (flag_top == 1)
54                 begin
55                 cont1 <= cont1 + 1;
56                 if (cont1 == 8'd50)
57                         begin
```

```
58                          flag_top <= 0;
59                          flag_null <= 1;
60                          Y_Cont_bottom <= iY_Cont + 100;
61                          end
62                  end
63
64          if (flag_null == 1)
65                  begin
66
67                  if ((iSignature_a < iSignature_c) && (iSignature_c <
                        iSignature_d) && (iSignature_d < iSignature_b) && (
                        iY_Cont > Y_Cont_bottom))
68                  oldnull <= 1;
69                  else
70                  oldnull <= 0;
71
72                  if ((iSignature_c > iSignature_d) && (oldnull == 1))
73                  begin
74                  flag_null <= 0;
75                  flag_bottom <= 1;
76                  flag_body <= 1;
77                  end
78                  end
79
80          if (flag_body == 1)
81                  begin
82                  cont1 <= cont1 + 1;
83                  if (cont1 == 8'd100)
84                          begin
85                          flag_body <= 0;
86                          end
87                  end
88
89          end
90
91  always @ (posedge iCCD_PIXCLK)
92          begin
93          iDval_old <= iDval;
94          PIXCLK <= !PIXCLK;
95          end
96
97  always @ (posedge PIXCLK)
98          begin
```

```
99
100         if (flag_bottom == 0)
101                 begin
102                 flag_bottom2 <= 0;
103                 cont2 <= 0;
104                 end
105
106         if (flag_bottom == 1)
107                 begin
108                 if (cont2 < 8'd50)
109                 begin
110                 cont2 <= cont2 + 1;
111                 flag_bottom2 <= 1;
112                 end
113                 else
114                 flag_bottom2 <= 0;
115                 end
116
117         end
118
119         always
120         begin
121         Sig_val_tmp1 = ((iDval && (flag_top || flag_body)) || (!PIXCLK &&
                flag_bottom2));
122         oSignature_val = ((iDval_old && (flag_top || flag_body)) || (!
                PIXCLK && flag_bottom2));
123         matrix_sig_val = ((iDval && flag_null) || (!PIXCLK && flag_bottom2)
                );
124         end
125
126         always @ (posedge Sig_val_tmp1)
127         begin
128                         if (flag_top == 1)
129                                 begin
130                                 oSignature_l <= Signature_a;
131                                 oSignature_r <= Signature_b;
132                                 end else
133                         if (flag_bottom2 == 1)
134                                 begin
135                                 oSignature_l <= Signature_ml [index]; //
                                        COPIAR DA MATRIZ!
136                                 oSignature_r <= Signature_mr [index];
137                                 end else
```

```
138                         if (flag_body == 1)
139                                 begin
140                                 oSignature_l <= Signature_a;
141                                 oSignature_r <= Signature_b;
142                                 end
143                 end
144
145 always @ (posedge matrix_sig_val)
146         begin
147         if (flag_null == 1)
148         begin
149         Signature_ml [index] <= Signature_c; //COPIAR NA MATRIZ!
150         Signature_mr [index] <= Signature_d;
151         end
152         if (index == 8'd49)
153         index <= 8'd0;
154         else
155         index <= index + 1;
156         end
157
158
159 endmodule
```

Code C.6: Etapa de comparação

```
1  module signature_reference (iSignature_l, iSignature_r, iSignature_val,
       iFval, iSignature_st,
2          oSignature_l_min, oSignature_l_max, oSignature_r_min,
               oSignature_r_max, oResult);
3
4  input [9:0] iSignature_l;
5  input [9:0] iSignature_r;
6  input                   iSignature_val;
7  input                   iFval;
8  input                   iSignature_st;
9  output reg [9:0] oSignature_l_min;
10 output reg [9:0] oSignature_l_max;
11 output reg [9:0] oSignature_r_min;
12 output reg [9:0] oSignature_r_max;
13 output reg oResult;
14
15 reg [9:0] signature_l_min [0:150];
16 reg [9:0] signature_l_max [0:150];
17 reg [9:0] signature_r_min [0:150];
```

```
18 reg [9:0] signature_r_max [0:150];
19
20 reg [7:0] memory_pos;
21 reg         result_tmp;
22
23 always @ (posedge iSignature_val)
24         begin
25         if (iSignature_st == 1)
26         begin
27         if (iSignature_l > 50)
28                 signature_l_min [memory_pos] <= iSignature_l - 8'd50;
29                 else
30                 signature_l_min [memory_pos] <= 8'd0;
31         signature_l_max [memory_pos] <= iSignature_l + 8'd50;
32         if (iSignature_r > 50)
33                 signature_r_min [memory_pos] <= iSignature_r - 8'd50;
34                 else
35                 signature_r_min [memory_pos] <= 8'd0;
36         signature_r_max [memory_pos] <= iSignature_r + 8'd50;
37         end
38         oSignature_l_min <= signature_l_min [memory_pos];
39         oSignature_l_max <= signature_l_max [memory_pos];
40         oSignature_r_min <= signature_r_min [memory_pos];
41         oSignature_r_max <= signature_r_max [memory_pos];
42         end
43
44 always @ (negedge iFval or posedge iSignature_val)
45         begin
46         if (!iFval)
47                 memory_pos <= 8'd0;
48                 else
49                 memory_pos <= memory_pos + 8'd1;
50         end
51
52 always @ (posedge iSignature_val)
53         if ((iSignature_l < signature_l_min [memory_pos]) || (iSignature_l
              > signature_l_max [memory_pos]) || (iSignature_r <
              signature_r_min [memory_pos]) || (iSignature_r > signature_r_max
               [memory_pos]))
54                 result_tmp <= 1'b1;
55                 else
56                 result_tmp <= 1'b0;
57
```

```
58 always @ (posedge result_tmp or posedge iFval)
59         if (result_tmp)
60                 oResult <= 1'b1;
61                 else
62                 oResult <= 1'b0;
63
64         endmodule
```