# Adaptable Architecture for the Development of Computer Vision Systems in FPGA

Ubiratan Ramos, Maurício Edgar Stivanello e Marcelo Ricardo Stemmer

***Abstract*—Computer vision systems are increasingly used in industry for inspection or process control. The more demanding requirements observed today make the implementation of this type of systems a technological challenge. Many of the computational architectures available allow us to meet the main functional requirements related to the use case and also the non-functional ones, such as processing time restrictions and connectivity. However, the requirement for adaptability so that such systems can be easily modified to meet different use cases or even accommodate environmental changes remains a challenge. This work proposes a flexible architecture for computer vision systems using FPGA. This architecture combines components of the processing flow of the vision system implemented in hardware and software to obtain advantages associated with both approaches. The proposed solution is validated against different real use cases existing in the industry. The results obtained allow us to affirm that such architecture brings an interesting advantages since it meets the operational requirements present in industrial applications, demands less development effort and can be easily adapted to new usage scenarios.**

***Index Terms*—Automated Inspection, Image Processing, Computer Vision, FPGA, NIOS.**

## I. Introdução

Sistemas de visão computacional são cada vez mais empregados na indústria em tarefas de inspeção ou controle [1], [2], [3]. Tais sistemas empregam imagens na análise de características de peças e partes de maneira automatizada [3]. A operação em ambiente industrial traz consigo uma série de requisitos com restrições quanto ao tamanho físico das soluções, requisitos de tempo, adaptabilidade das soluções para novos casos de inspeção e a necessidade de integração com outros sistemas presentes na planta industrial. Assim sendo, sua implementação representa um desafio tecnológico e exige que tanto os componentes de *software* como de *hardware* sejam criteriosamente selecionados e interconectados.

Uma questão muito importante no desenvolvimento de um sistema que atenda aos requisitos descritos refere-se à escolha da plataforma computacional a ser utilizada. Existem várias alternativas baseadas em diferentes tecnologias, como os tradicionalmente utilizados processadores de propósito geral (GPP - *General Purpose Processor*), processadores de sinais digitais (DSP - *Digital Signal Processor*) e os dispositivos reconfiguráveis (FPGA – *Field Programmable Gate Arrays*).

U. Ramos é Professor no Instituto Federal Catarinense, São Francisco do Sul (SC), Brasil, e-mail: ubiratan.ramos@ifc.edu.br

M. E. Stivanello é Professor no Instituto Federal de Santa Catarina, Florianópolis (SC), Brasil, e-mail: mauricio.stivanello@ifsc.edu.br

M. R. Stemmer é Professor da Universidade Federal de Santa Catarina, Florianópolis (SC), Brasil, e-mail: marcelo.stemmer@ufsc.br

Processadores de propósito geral presentes em computadores convencionais têm sido utilizados na implementação de diversos sistemas de visão computacional [1], [4]. Por serem amplamente empregados, encontram-se disponíveis uma variedade de linguagens, bibliotecas e mesmo código legado que permitirá alcançar grande produtividade [5]. Além disso, as soluções desenvolvidas podem ser facilmente adaptadas ou atualizadas. Porém, tais soluções podem não ser bem dimensionadas ao ambiente industrial, visto que o *software* associado ao sistema de visão executa sobre *hardware* e sistema operacional de propósito geral, que trazem consigo uma série de recursos, serviços e processos não necessários para tal tipo de aplicação.

Já os DSPs são microprocessadores de propósito específico, com arquitetura e conjunto de instruções projetados especificamente para implementar algoritmos de processamento de sinais [6]. Tal característica os torna uma escolha excelente para tarefas de processamento de imagens especializadas como codificação e decodificação de video [7]. O desenvolvimento de um sistema de visão computacional completo, porém, se torna desafiador. Para este tipo de aplicação, o DSP precisará ser constantemente reconfigurado para executar as diferentes tarefas, algumas relacionadas ao processamento digital das imagens e outras relacionadas às rotinas de controle. Nesse contexto, recursos como registradores do núcleo do processador, acesso a memória interna e externa e periféricos de entrada e saída são compartilhados por todas as tarefas, ou *threads*. A necessidade de sincronização entre tais tarefas torna o projeto do sistema complexo [8].

Por sua vez, a tecnologia FPGA permite implementar sistemas dedicados melhor dimensionados quando comparados aos implementados em GPPs, visto que apenas o fluxo de processamento relacionado ao tratamento de imagens e operações de controle são executados [9], [10]. Além disso, se comparada aos DSPs, tal tecnologia oferece uma implementação onde cada tarefa é alocada com seus próprios recursos e é executada de forma independente, simplificando o projeto do sistema [8]. Em contrapartida, a implementação em FPGA é consideravelmente mais especializada e custosa visto que todo o código legado, documentação e bibliotecas de sistemas de visão computacional existentes para plataformas convencionais não podem ser diretamente utilizados [11].

Uma alternativa para utilização de recursos legados de outras plataformas é a utilização de processadores implementados sobre a malha FPGA. Tal recurso permite combinar blocos de processamento totalmente implementados em FPGA com rotinas implementadas em linguagens convencionais, como C ou C++. Esta abordagem torna-se ainda mais interessante

devido ao fato de que é possível sintetizar múltiplos processadores em um mesmo projeto [12], [13].

No presente trabalho, diferente de outros encontrados na literatura que implementam sistemas de visão computacional em FPGA para atender a aplicações específicas, propõe-se uma arquitetura flexível e bem dimensionada. Tal arquitetura busca utilizar os recursos de sintetização de algoritmos diretamente em *hardware* e a de execução de algoritmos em um processador sintetizado, ou *soft-core processor*, para se obter vantagens associadas às duas tecnologias. As contribuições frente aos trabalhos correlatos são:

1) A descrição de uma arquitetura em FPGA que fornece maior adaptabilidade e demanda menor esforço de desenvolvimento quando comparada a abordagens que empregam a implementação do fluxo de processamento unicamente em *hardware*. Tal arquitetura atende a requisitos operacionais gerais presentes em aplicações industriais e pode ser facilmente adaptada a novos cenários de uso.
2) A validação da abordagem proposta utilizando casos reais presentes no ambiente industrial.
3) A disponibilização da implementação de tal solução à comunidade como *software* de código aberto na internet.

Na Seção II são descritos trabalhos correlatos. Na Seção III é apresentado o fluxo geral de processamento de imagens que orientou o projeto da arquitetura proposta. Na Seção IV são apresentados os detalhes da arquitetura proposta. Na seção V são apresentados detalhes de implementação e resultados experimentais obtidos pela utilização da arquitetura proposta frente a diferentes casos de inspeção da indústria. Na Seção VI são apresentadas as conclusões e trabalhos futuros.

## II. A Utilização de FPGA no Desenvolvimento de Sistemas de Visão Computacional

Nos últimos anos vários trabalhos têm sido elaborados utilizando FPGA no desenvolvimento de sistemas de processamento de imagens e de visão computacional. Tais sistemas são bastante aderentes a esta plataforma pela existência de requisitos normalmente presentes em suas aplicações, como restrições de tempo e necessidade de interfaceamento com dispositivos externos. Além disso, muitos dos algoritmos comumente utilizados, como filtragens e convoluções, são altamente paralelizáveis, possibilidade que pode ser explorada de diferentes formas em tal tecnologia.

Dentre exemplos que podem ser citados estão sistemas para compressão de imagens [14], detecção de características e objetos [15], [16], [17], [18], inspeção de peças defeituosas da indústria [19], [10], dentre outras aplicações.

Uma característica geral encontrada em tais sistemas é que a arquitetura é voltada especificamente para cada caso tratado [20]. O custo de desenvolvimento de tal abordagem é alto. A natureza refinada da tecnologia FPGA os torna difíceis de programar [11]. Diferente das plataformas que empregam processadores de propósito geral, que possuem uma arquitetura computacional fixa capaz de fornecer um alto nível de abstração, nas plataformas FPGA se faz necessário projetar não somente os algoritmos da aplicação mas também toda a arquitetura computacional. Além disso, os algoritmos são tradicionalmente implementados em *hardware* utilizando linguagens como Verilog e VHDL, que não fornecem o mesmo nível de abstração encontrados em linguagens de implementação em *software* [21]. Tais aspectos levam a um aumento significativo na complexidade de tais projetos.

Recentemente, no trabalho apresentado por Garcés-Socarrás et al. (2020), propõe-se uma biblioteca de blocos de processamento para o ambiente MATLAB que fornece certo grau de abstração para o desenvolvimento de sistemas de processamento de imagens em FPGA [22]. Tal proposta, voltada para implementação de algoritmos sintetizados em *hardware*, fornece uma alternativa de fluxo reconfigurável baseada em uma arquitetura genérica.

Neste contexto identifica-se a contribuição proposta no presente trabalho de fornecer uma arquitetura de fluxo reconfigurável que contemple ainda a utilização de processadores sintetizados, de forma a aumentar a compatibilidade com código legado, prover adaptabilidade e reduzir o custo de desenvolvimento de sistemas de visão em FPGA.

## III. Fluxo de Processamento Geral e sua Implementação

Um sistema de visão computacional empregado na indústria executará tarefas que, na grande maioria das vezes, estarão associadas a contagem, localização ou medição. Apesar da diferença existente entre tais tarefas, muitas das etapas presentes na sequência de processamento das imagens digitais necessária para que sejam realizadas são compartilhadas [3]. Na Fig. 1-a é apresentado o fluxo básico genérico de um sistema de processamento de imagens para inspeção automatizada.

Neste fluxo, quando uma dada peça a ser inspecionada é apresentada ao sistema, é efetuada a aquisição de uma imagem. Em seguida, é realizado um processamento denominado segmentação, onde o objetivo é separar os *pixels* pertencentes aos objetos de interesse dos objetos que fazem parte do plano de fundo. Na sequência, é realizada a detecção e descrição dos agrupamentos de *pixels* resultantes da segmentação. Os descritores que caracterizam os agrupamentos de pontos que representam os objetos são então classificados em função dos requisitos da aplicação. Na sequência, com base nos resultados obtidos, sinais de descarte são aplicados e/ou rotinas de registro são executadas em caso de peça fora de especificação.

Tal sequência de etapas pode ser generalizada para diferentes aplicações, onde apenas as técnicas de processamento de imagens empregadas em cada etapa varia. Assim sendo, este fluxo de processamento geral é utilizado como base para a proposição da arquitetura descrita na seção seguinte.

## IV. Arquitetura Proposta

A arquitetura proposta é voltada para aplicações de inspeção automatizada encontradas na indústria. Assim sendo, os principais requisitos que orientaram a sua concepção foram:

- Implementar o fluxo de processamento descrito na Seção III de forma flexível, de modo que possa ter suas etapas facilmente adaptadas para diferentes tipos de inspeção;
- Permitir a parametrização ou mesmo a adaptação do fluxo de processamento sem a necessidade de modificações

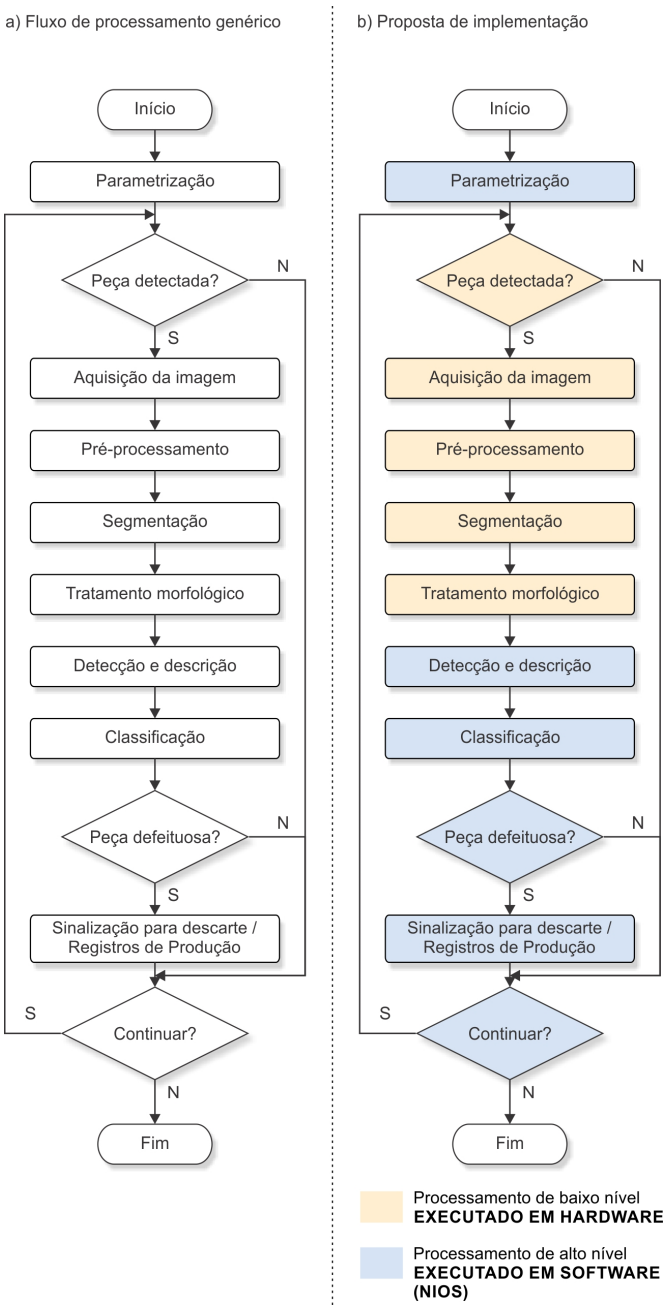



Fig. 1. Fluxos de processamento para sistemas de inspeção

em *hardware* (malha FPGA), o que representaria grande custo de desenvolvimento;
- Facilitar a utilização de código legado proveniente de sistemas originalmente desenvolvidos para arquitetura baseada em processadores de propósito geral;
- Permitir a sincronização da aquisição de imagens por gatilhos externos, assim como possibilitar a integração com outros sistemas presentes na linha de produção.

Observando os requisitos funcionais e não funcionais descritos e também as etapas que caracterizam um fluxo de processamento geral de um sistema de visão apresentadas na Seção III, foi definida uma divisão lógica das etapas de processamento, como apresentada na Fig. 1-b. O primeiro segmento de etapas, aqui classificadas como processamento de baixo nível, corresponde a algoritmos preliminares que vão da aquisição até a segmentação da imagem. Tais algoritmos, que envolvem conversão de cores, filtragens, limiarização e aplicação de operações morfológicas, compartilham a característica de que tanto o dado de entrada quanto o de saída correspondem a *pixels* das imagens. Em função do tipo de processamento que realizam, que muitas vezes são independentes para cada ponto ou que empregam núcleos de convolução baseados em vizinhança, estes algoritmos são fortes candidatos a serem implementados em *hardware*, com fluxo de processamento em paralelo de forma relativamente simplificada. A maioria destes algoritmos estão presentes em um grande número de aplicações de visão e não sofrem alteração na sua implementação em função do tipo de inspeção. Uma vez que precisem ser utilizados, basta ativá-los e parametrizá-los. Em função disso, é proposto que tal segmento de etapas seja implementado em *hardware*.

O segundo segmento de etapas, classificadas como processamento de alto nível, corresponde a algoritmos mais diretamente relacionados ao tipo de inspeção realizada ou ao domínio de aplicação. Nestas etapas a saída deixa de ser os *pixels* que compõem uma imagem, e passam a trazer dados associados às características a serem avaliadas, como agrupamentos de objetos, quantidades, descritores como área, dimensões, dentre outras. A combinação de tais algoritmos ou etapas é altamente dependente do tipo de inspeção que está sendo realizada e, desta forma, deve ser customizada para cada aplicação. Por este motivo, é proposto que sejam implementados em *software*, tendo em vista a maior flexibilidade presente nesta abordagem.

### A. Desenvolvimento do Hardware do Projeto

A arquitetura de *hardware* proposta é mostrada na Fig. 2-a. A solução é composta majoritariamente por blocos de propriedade intelectual (IP) disponíveis na ferramenta *Platform Designer* da Intel relativos à aquisição de imagens, gerência de memória, *clock*, processador, dentre outros.

Adicionalmente, foi incluído um bloco denominado **Rotinas em *Hardware***, que contém a implementação feita em Verilog das etapas de mais baixo nível do fluxo de processamento descrito na seção IV. O detalhamento deste bloco pode ser observado na Fig. 2-b. A seleção das diversas fontes de imagem e das operações morfológicas aplicadas a estas é feita através do registrador $Image\_sel$, bits 5 a 0. Isto garante flexibilidade e modularidade ao *hardware*, uma vez que existem rotas disponíveis para a implantação de novas rotinas.

Para a execução das rotinas de alto nível do fluxo de processamento foi incluído um processador *soft processor* do tipo NIOS II/f [23]. Tal processador utiliza uma arquitetura RISC de 32 bits e é altamente configurável, com destaque para o controlador de interrupções. Este processador apresenta alto grau de flexibilidade no que diz respeito à integração com dispositivos personalizados e também suporte para rotinas em C/C++. O processador foi configurado para executar a 100 MHz, com 4 kbytes de *cache* de instruções, 2 kbytes de *cache* de dados e modo de depuração JTAG habilitado.

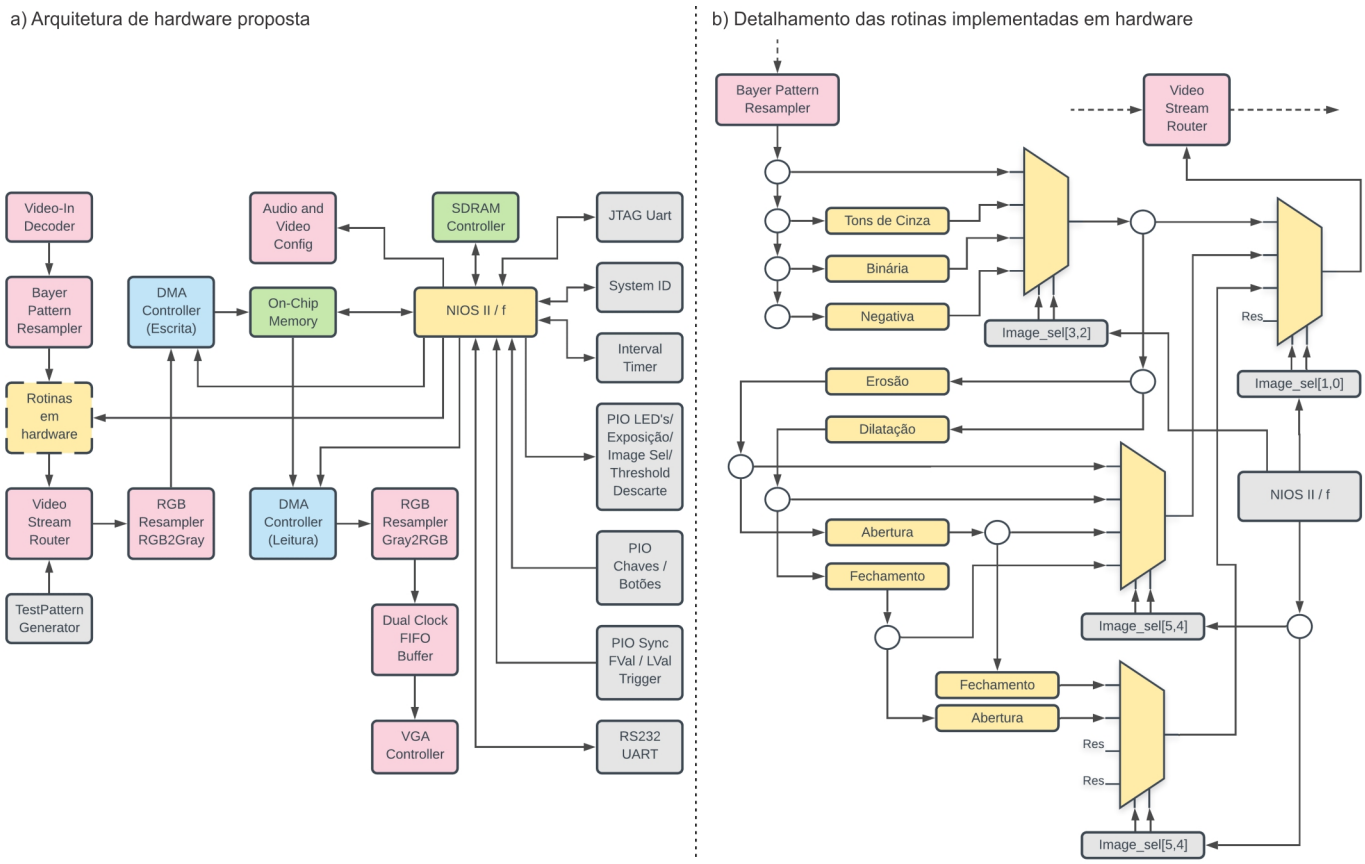



Fig. 2. Detalhamento da arquitetura de *hardware* proposta

No sistema são utilizadas memórias *on-chip* e *off-chip*. Visto que as rotinas e bibliotecas em C++ não caberiam na memória *on-chip*, optou-se por alocá-las na memória *off-chip*. Assim, utilizou-se a memória *on-chip* para a alocação do *buffer* de quadros de vídeo. Tal configuração, onde a memória de programa está separada da memória de dados, traz a vantagem de evitar a disputa de barramento por tais estruturas.

No que se refere às interfaces, além das portas paralelas tradicionais (PIO) para acesso aos LED's, chaves e botões da placa, foram criadas algumas portas para acesso do processador NIOS II a alguns sinais importantes do *hardware*. Assim, estão disponíveis ao processador a possibilidade de ajustar o tempo de exposição da câmera através de *software*, sincronizar o processamento com os sinais de início e fim de quadro (FVal) e de linha horizontal (LVal) da interface de entrada de vídeo e configuração de limiares de sensibilidade dos algoritmos de binarização e de negativação implementados em *hardware*. Também pode ser feita através de *software* a seleção da imagem de entrada (real, cinza, binarizada ou negativada) e dos algoritmos morfológicos (erosão, dilatação, abertura, fechamento, abertura seguida de fechamento e fechamento seguido de abertura) aplicados à imagem em tempo real selecionada. Ainda, foi incluído um bloco UART para fornecer uma interface serial assíncrona para comunicação com um programa de configuração, descrito na seção IV-B.

### *B. Desenvolvimento do Software do Projeto*

O *software* do sistema corresponde à implementação das etapas de mais alto nível do fluxo de processamento descrito na seção IV. Tais etapas podem variar tanto no que se refere aos algoritmos de detecção e classificação empregados quanto nos parâmetros utilizados, visto que estes são altamente dependentes do tipo de inspeção que está sendo realizada.

No contexto do presente trabalho foram implementados algoritmos utilizados em tarefas de verificação de presença/ausência, quantidade, tamanho e distância. A detecção dos objetos é baseada na definição de regiões de interesse e localização de partes através de um algoritmo de rotulação de componentes conectados. A classificação de cada imagem é baseada na avaliação de descrições derivadas das áreas analisadas e coordenadas posicionais das partes detectadas.

As rotinas de *software* foram implementadas na linguagem C++ fazendo-se uso do ambiente de desenvolvimento *NIOS II Software Build Tools*. Para a implementação dos algoritmos de processamento de imagens foi utilizada a biblioteca OpenCV [5], [18]. Tal biblioteca não é nativamente compatível com o ambiente de desenvolvimento do NIOS e, desta forma, o código contendo as definições de tipos e funções pode exigir modificações frente a possíveis incompatibilidades.

Uma interface com o usuário foi construída para que o sistema possa ser configurado para diferentes casos de uso. A configuração é realizada através de um computador conectado ao dispositivo FPGA, sendo a comunicação entre estes reali-

zada pela interface serial JTAG. Através desta interface o usuário pode selecionar os algoritmos de *hardware* e *software* que comporão o fluxo de processamento, e também parametrizá-los definindo valores de limiares e outros valores de referência. Tanto uma versão baseada em linha de comando quanto uma versão baseada em interface gráfica estão disponíveis.

## V. Resultados Experimentais

A arquitetura descrita na seção IV foi implementada utilizando um kit de desenvolvimento DE1-SoC com uma placa de expansão TRDB-D5M, que inclui uma câmera colorida de 5 MP. Tal câmera foi configurada para aquisição de imagens a uma resolução de 640 X 480 *pixels* e 45 quadros por segundo. No desenvolvimento do projeto de *hardware* e do *software* foi utilizada a ferramenta Quartus II. De forma a contribuir com trabalhos futuros, o projeto está disponibilizado para a comunidade na forma de *software* de código aberto [1].

Para a realização de testes experimentais, os componentes físicos do sistema foram afixados em um suporte de alumínio. Para se obter uma iluminação adequada dos objetos inspecionados foi afixado no suporte um anel de LEDS de alta luminosidade. Na Fig. 3 é exibida a bancada montada, que inclui um monitor ligado à placa para visualização dos resultados e uma das peças utilizadas como mensurando.

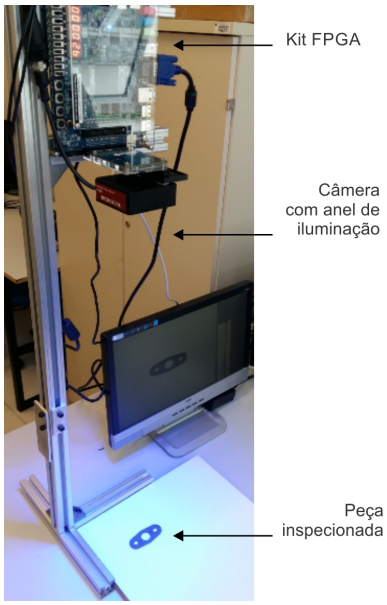



Fig. 3. Bancada utilizada nos experimentos

Para a validação da arquitetura proposta frente aos requisitos operacionais presentes na indústria foram utilizados dois casos de teste existentes no catálogo da solução comercial de inspeção por imagens Microscan AutoVISION [24]. Os casos de teste selecionados correspondem à inspeção de uma junta, apresentada na Fig. 4-a, e também de um componente eletrônico, no caso um transistor com encapsulamento TO-220, apresentado na Fig. 4-b. Além destes, utilizou-se como um terceiro caso de testes a inspeção de tampas de metal para verificação da presença de rebites, apresentada na Fig. 4-c [4].

[1]Código-fonte do projeto implementado: sites.florianopolis.ifsc.edu.br/csi

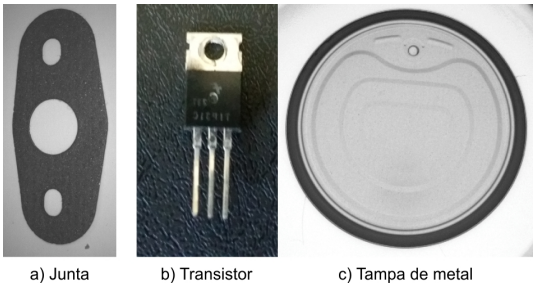



Fig. 4. Casos de inspeção empregados nos experimentos

Para cada um dos casos de uso é realizada uma adaptação do fluxo geral de inspeção implementado, efetuando-se alterações nos algoritmos implementados em *software* e modificando os parâmetros dos algoritmos implementados em *hardware* conforme necessidade, como apresentado na Fig. 5.

### A. Inspeção de Junta

O objetivo da inspeção da junta é verificar o número de furos presentes, identificar os dois furos externos e realizar a medição da distância entre estes. No caso de ausência de um ou mais furos, ou de uma distância entre os furos externos fora de um valor limite arbitrado, o sistema deve ser capaz de reconhecer e sinalizar a desconformidade.

O fluxo geral de processamento do sistema implementado foi adaptado de forma que realizasse a sequência de etapas apresentada na Fig. 5-a, cujo resultado é a aprovação ou rejeição das peças. A detecção dos furos se baseia em uma filtragem morfológica, na rotulação de componentes conexos, em uma descrição baseada em descritores de área e centroide. Por fim, a classificação é realizada pelas comparações de tamanhos, quantidades e distâncias.

Na Fig. 6 é apresentada a representação da inspeção na tela, onde os furos extremos são interligados por uma reta, cuja medida é avaliada para verificar não conformidades.

### B. Inspeção de Componente Eletrônico

O objetivo da inspeção do componente eletrônico é verificar uma possível ausência de terminais. O fluxo geral de processamento do sistema implementado foi adaptado de forma que realizasse a sequência de etapas apresentada na Fig. 5-b, cujo resultado é a aprovação ou rejeição das peças. Uma região de interesse é configurada de forma que o processamento da imagem seja limitado a um retângulo que abranja a localização esperada para os terminais. A detecção dos terminais se baseia em uma filtragem morfológica, na rotulação de componentes conexos e em uma descrição baseada em descritores de área. Por fim, a classificação é realizada pelas comparações de tamanhos e avaliação de quantidades.

Na Fig. 7-a é apresentada a imagem de um componente não defeituoso, enquanto na Fig. 7-b é apresentada a imagem de um componente defeituoso pela ausência de um dos terminais.

### C. Inspeção de Tampa de Metal

O objetivo da inspeção das tampas de metal é verificar uma possível ausência de rebite. O fluxo geral de processamento do

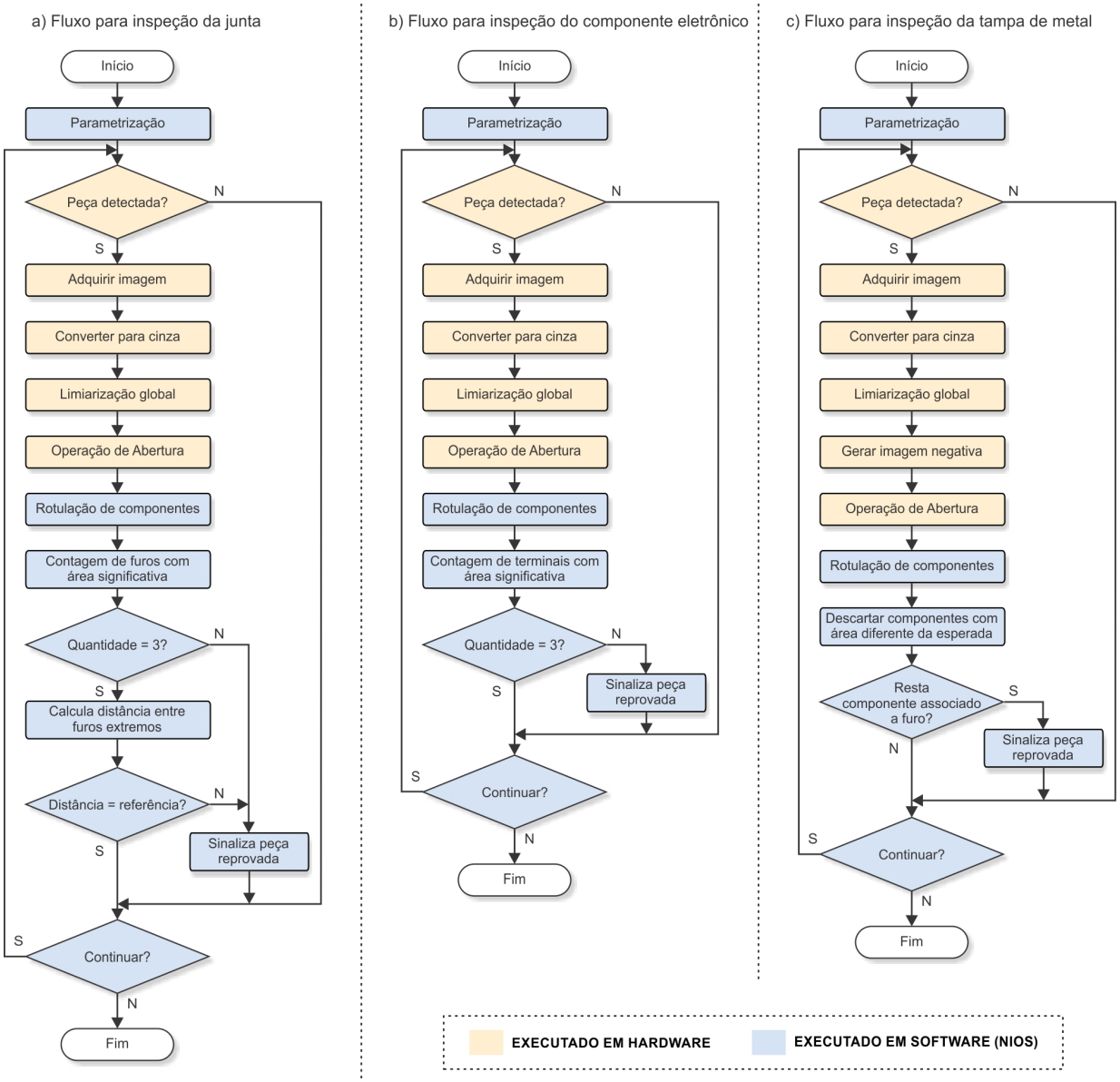



Fig. 5. Fluxos de processamento criados para cada caso de inspeção

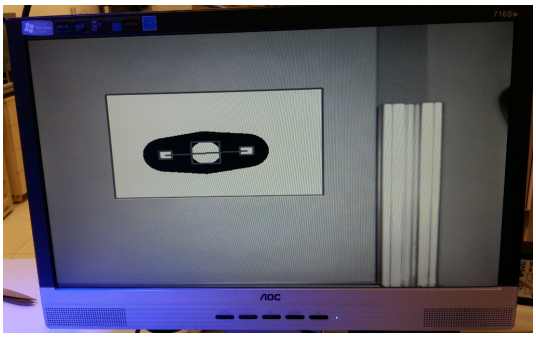
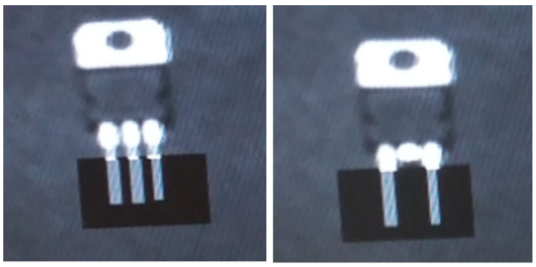


Fig. 6. Representação visual da inspeção da junta

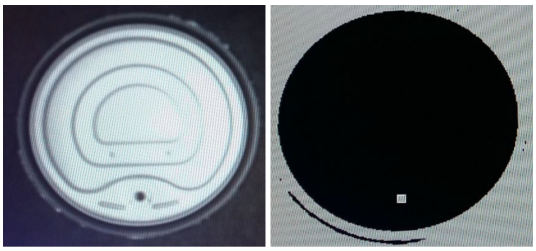


a) Componente eletrônico aprovado b) Componente eletrônico reprovado

Fig. 7. Representação visual da inspeção do componente eletrônico

sistema implementado foi adaptado de forma que realizasse a sequência de etapas apresentada na Fig. 5-c, cujo resultado é a aprovação ou rejeição da peça. A detecção do rebite se baseia em uma filtragem morfológica, na rotulação de componentes conexos e em uma descrição baseada em descritores de área. Por fim, a classificação é realizada pelas comparações de tamanhos dos componentes.

Na Fig. 8-a é exibida a imagem de uma tampa sem o rebite, onde a peça é classificada como reprovada pela presença do furo passante detectado, como apresentado na Fig. 8-b.



a) Imagem original b) Imagem processada com furo do rebite detectado

Fig. 8. Representação visual da inspeção da tampa de metal

### *D. Avaliação Geral dos Resultados*

O objetivo dos experimentos descritos para cada caso de inspeção não se concentraram no levantamento de métricas que caracterizam um sistema de visão computacional, como taxas de acurácia ou erro, por exemplo. Tal avaliação, que é influenciada por fatores como por exemplo os componentes ópticos utilizados, as condições de iluminação e outros aspectos relacionados à problemática de implantação de um sistema de visão, não pertence ao escopo do presente trabalho.

A motivação da avaliação descrita é apresentar a capacidade do sistema de representar rotinas de inspeção variadas condizentes com demandas reais presentes na indústria. Nesse sentido podemos afirmar que o sistema foi capaz de realizar todas as verificações esperadas para cada caso de inspeção. É importante enfatizar o fato de que a adequação do sistema para cada caso de inspeção exige que apenas rotinas implementadas em *software* sejam modificadas, de forma que o custo de desenvolvimento é significativamente reduzido.

No que diz respeito aos requisitos funcionais e não funcionais que orientaram o desenvolvimento da arquitetura, todos foram atendidos, visto que a mesma permite:

- Alterar o fluxo geral de processamento de forma flexível, de modo que possa ter suas etapas facilmente adaptadas para diferentes tipos de inspeção;
- Parametrizar os algoritmos sem que modificações em *hardware* (malha FPGA) sejam necessárias;
- Utilizar código legado de sistemas desenvolvidos para arquitetura baseada em processadores de propósito geral. Por exemplo, rotinas existentes em bibliotecas como a OpenCV podem ser utilizadas, desde que recompiladas com o compilador específico para o processador NIOS;
- Realizar a sincronização da aquisição de imagens ou da sinalização do resultado da inspeção através da inclusão de interfaces paralelas PIO no *hardware* do projeto.

No que se refere a uma avaliação do custo computacional das rotinas de inspeção, um componente temporizador foi integrado na implementação para servir de base de tempo para tal cálculo. Os tempos médios de processamento foram calculados para os dois grandes grupos de rotinas presentes no fluxo de processamento do sistema implementados em *hardware* e em *software*, conforme apresentado na Tabela I.

TABELA I
CUSTO COMPUTACIONAL

| Bloco de rotinas | Tempo (ms) |
|---|---|
| Rotinas implementadas em *hardware* | 0,76 |
| Rotinas implementadas em *software* | 292,12 |
| Tempo total | 292,88 |

Os tempos demonstram a eficiência da utilização das rotinas em *hardware* que, apesar do maior custo de implementação, apresentam tempos de execução muito reduzidos.

O atendimento do volume de inspeção disponibilizado pela solução frente a cenários presentes na indústria é altamente dependente da aplicação. Alguns casos de uso, como a inspeção de tecidos, demandam por análises a uma velocidade menor [1], [25]. Por exemplo, casos de inspeção de furos e rasgos podem ser atendidos por uma frequência de inspeção de 0.91 Hz. Para tais cenários o sistema desenvolvido atende tal requisito. Outras aplicações, como por exemplo a inspeção de rebites em tampas de metal, podem exigir frequências significativamente maiores [4]. Para estes casos seriam necessárias melhorias para que o custo computacional fosse reduzido.

Na Tabela II são apresentados indicadores que refletem a utilização dos recursos demandados para a sintetização da arquitetura no dispositivo FPGA fornecidas pelo relatório de compilação da ferramenta Quartus. É possível observar a existência de recursos disponíveis para a extensão das funcionalidades da arquitetura no sentido de incluir novas rotinas implementadas tanto em *hardware* como em *software*.

TABELA II
INDICADORES DO CONSUMO DE RECURSOS DO DISPOSITIVO FPGA

| Tipo | Utilizado | Disponível | Porcentagem |
|---|---|---|---|
| Unidades lógicas | 4096 | 62070 | 13% |
| Blocos de memória (bits) | 2896712 | 4065280 | 71% |
| Proc. digitais de sinais | 18 | 87 | 21% |
| Pinos | 240 | 457 | 53% |

## VI. CONCLUSÕES E TRABALHOS FUTUROS

No presente trabalho é apresentada uma arquitetura em FPGA voltada para o desenvolvimento de sistemas de visão computacional. Diferente de outros trabalhos encontrados na literatura, que procuram atender a aplicações específicas, propõe-se uma arquitetura flexível e bem dimensionada cujo custo de desenvolvimento seja reduzido quando comparado à abordagens tradicionais de implementação, buscando popularizar a tecnologia. Para isso, um fluxo de processamento geral de estrutura modificável é utilizado. Nele, rotinas de mais baixo nível são sintetizadas em *hardware* e rotinas de mais alto nível, mais fortemente dependentes da aplicação e que exigirão modificações para cada caso de uso, são executadas em um processador de propósito geral sintetizado. Com isso, grande adaptabilidade é alcançada.

A solução proposta foi validada frente a diferentes casos de uso. Nos experimentos realizados é possível observar que

os requisitos que orientaram o desenvolvimento da arquitetura foram atendidos. Os resultados obtidos permitem afirmar que tal arquitetura traz vantagens interessantes no que diz respeito ao desenvolvimento de sistemas de visão, visto que atende a requisitos operacionais encontrados em casos de inspeção reais da indústria, demanda menor esforço de desenvolvimento ao permitir que código legado seja utilizado e ao minimizar a exigência de implementações em *hardware*.

Na continuidade dos trabalhos serão avaliadas estratégias para reduzir o custo computacional exigido pela execução das rotinas implementadas em *software*. Duas abordagens estão sendo consideradas. A primeira delas, cujos trabalhos já foram iniciados, diz respeito ao uso de múltiplos processadores sintetizados [13]. Outra possibilidade a ser explorada é a utilização de um processador *hard-core* integrado ao FPGA, disponíveis em dispositivos SoC FPGA.

## AGRADECIMENTOS

Os autores agradecem ao Conselho Nacional de Desenvolvimento Cientifico e Tecnológico (CNPq) pelo apoio ao projeto.

## REFERÊNCIAS

[1] S. Vargas, M. E. Stivanello, M. L. Roloff, É. Stiegelmaier, and M. R. Stemmer, "Development of an online automated fabric inspection system," *Journal of Control, Automation and Electrical Systems*, vol. 31, no. 1, pp. 73–83, Feb 2020.

[2] A. C. Moreira, H. K. M. Paredes, W. A. Souza, P. H. J. Nardelli, F. P. Marafão, and L. C. P. Silva, "Evaluation of pattern recognition algorithms for applications on power factor compensation," *JCAE*, vol. 29, no. 1, pp. 75–90, Feb 2018.

[3] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2011.

[4] M. E. Stivanello and K. J. Marcellino, "A machine vision system for online metal can-end rivet inspection," *Journal of Physics: Conference Series*, vol. 1335, p. 012002, oct 2019.

[5] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision in C++ with the OpenCV library*, 2nd ed. O'Reilly Media, Inc., 2013.

[6] L. Chaparro and A. Akan, *Signals and Systems using MATLAB*. Academic Press, 2019.

[7] H. Jiang, R. Fan, Y. Zhang, G. Wang, and Z. Li, "Highly paralleled low-cost embedded hevc video encoder on ti keystone multicore dsp," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 4, pp. 1163–1178, 2019.

[8] A. Corporation. (2020, sep) Fpga vs. dsp design reliability and maintenance. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01023.pdf

[9] T. Kryjak, M. Komorkiewicz, and M. Gorgon, "Real-time hardware-software embedded vision system for ITS smart camera implemented in Zynq SoC," *Journal on Real-Time Image Processing*, vol. 15, no. 1, pp. 123–159, Jun. 2018.

[10] J. Zhou, G. Li, X. Wan, and J. Wang, "A real-time computer vision-based platform for fabric inspection part 2: Platform design and real-time implementation," *The Journal of The Textile Institute*, vol. 107, no. 2, pp. 264–272, 2016.

[11] D. G. Bailey, *Image Processing Using FPGAs*. MDPI, 2019.

[12] *Mutex core, Quartus II 9.1 handbook*, 5th ed., Intel Corporation.

[13] L. M. Mendes, M. E. Stivanello, and M. K. Dutra, "Avaliação de uma arquitetura fpga com múltiplos processadores para sistemas de processamento de imagens digitais," *Revista de Sistemas e Computação*, vol. 9, no. 2, pp. 326–332, 2019.

[14] L. Leiva, M. Vázquez, M. Tosini, O. Goñi, and J. Noguera, "Fpga based implementation of imagezero compression algorithm," *IEEE Latin America Transactions*, vol. 18, no. 02, pp. 344–350, 2020.

[15] C. Vakerlis, "Implementation of camera based computer vision algorithms on a reconfigurable platform," Master's thesis, Dept. of Electrical and Computer Engineering, University of Thessaly, 2015.

[16] R. Kapela, K. Gugala, P. Sniatala, A. Swietlicka, and K. Kolanowski, "Embedded platform for local image descriptor based object detection," *Applied Mathematics and Computation*, vol. 267, pp. 419–426, 2015.

[17] A. Lati, Z. Irki, S. Sakhi, A. Nemra, and M. Hamerlain, "Implementation of features detection and matching algorithms on fpga using nios ii," in *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*. IEEE, 2016, pp. 55–60.

[18] A. A. F. Hassan, A. H. Khalil, A. Omar, A. M. Sedek, M. I. Khalil, M. K. Ali, and M. M. A. Majed, "Implementation of hand gestures recognition using hardware/software co-design," Ain Shams, Tech. Rep., 2012.

[19] Y. Pan and R. Lu, "FPGA-accelerated one-dimensional Fourier reconstruction LCD defect detection algorithm," in *Tenth International Symposium on Precision Engineering Measurements and Instrumentation*, J. Tan and J. Lin, Eds., vol. 11053, International Society for Optics and Photonics. SPIE, 2019, pp. 776 – 781.

[20] L. Pyrgas, A. Kalantzopoulos, and E. Zigouris, "Design and implementation of an open image processing system based on nios ii and altera de2-70 board," *Journal of Engineering Science and Technology*, vol. 9, no. 5, pp. 50–54, 2016.

[21] J. N. Allen, H. S. Abdel-aty-zohdy, and R. L. Ewing, "Introducing rapidhdl: A new library to design fpga hardware in microsoft. net and automatically generate verilog netlists," in *2006 IEEE International Conference on Electro/Information Technology*, 2006, pp. 307–312.

[22] L. M. Garcés-Socarrás, A. J. Cabrera-Sarmiento, S. Sánchez-Solano, P. B. Jiménez, E. Ieno, and T. C. Pimenta, "Self-modifiable image processing library for model-based design on fpgas," *IEEE Latin America Transactions*, vol. 17, no. 05, pp. 742–750, 2019.

[23] *Introduction to the Intel Nios II Soft Processor*, Intel Corporation, 2017.

[24] *AutoVISION Software User Manual*, 84th ed., Omron Microscan Systems, Inc., 2017.

[25] M. E. Stivanello, S. Vargas, M. L. Roloff, and M. R. Stemmer, "Automatic detection and classification of defects in knitted fabrics," *IEEE Latin America Transactions*, vol. 14, no. 7, pp. 3065–3073, 2016.

**Ubiratan Ramos** É Engenheiro de Eletrônica pelo Instituto Tecnológico de Aeronáutica (ITA), Especialista em Sistemas de Telecomunicações pelo Instituto Nacional de Telecomunicações (INATEL), Mestre em Mecatrônica pelo Instituto Federal de Santa Catarina (IFSC). Foi Professor de Ensino Superior em diferentes instituições de ensino e atuou durante 25 anos na iniciativa privada nas áreas de Engenharia de Desenvolvimento de *Hardware* e *Software* e Telecomunicações. Atualmente é Professor no Instituto Federal Catarinense (IFC), nas áreas de Automação Industrial, Elétrica e Eletrônica.

**Maurício Edgar Stivanello** É Bacharel em Ciências da Computação pela Universidade Regional de Blumenau, Mestre em Engenharia Elétrica e Doutor em Engenharia de Automação e Sistemas pela Universidade Federal de Santa Catarina (UFSC) com período sanduíche na Universidade Tecnológica de Eindhoven (TU/e), Holanda. Realizou estágio de pós-doutorado no DAS - UFSC. É professor do Instituto Federal de Santa Catarina, campus Florianópolis, onde atua no ensino e pesquisa nas áreas de Programação, Mecatrônica e Visão Computacional.

**Marcelo Ricardo Stemmer** Graduou-se pela Universidade Federal de Santa Catarina em Engenharia Elétrica (1982). Realizou seu mestrado no Programa de Pós-Graduação em Engenharia Elétrica da UFSC (1985) e doutorou-se no instituto WZL da RWTH-Aachen, Alemanha (1991). Realizou estágio pós-doutoral no instituto LIP6 da Universidade Paris VI, França (2004). É professor do Departamento de Automação e Sistemas (DAS) da UFSC.