PROJETO INTEGRADO PARA RASTREAMENTO DE PADRÕES UTILIZANDO OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS E COEFICIENTE DE CORRELAÇÃO DE PEARSON

Yuri Marchetti Tavares*, Nadia Nedjah†, Luiza de Macedo Mourelle†

*Rua Primeiro de Março, 118 - Ed. Barão de Ladário - 17º ao 21º andar
Diretoria de Sistemas de Armas da Marinha
Rio de Janeiro, RJ, Brasil

†Rua São Francisco Xavier, 524 - 1006 - Maracanã
Universidade do Estado do Rio de Janeiro
Rio de Janeiro, RJ, Brasil

Emails: yurimtavares@yahoo.com.br, nadia@eng.uerj.br, ldmm@eng.uerj.br

Abstract— The template matching is an important technique used in pattern recognition. The goal consists of finding a given pattern, given as a prescribed model, in a frame sequence. In order to evaluate the similarity of two images, the Pearson's Correlation Coefficient (PCC) is widely used. This coefficient is computed for each of the image pixels, which entails a computationally very expensive operation. This paper proposes the implementation of template matching using PCC together with Particle Swarm Optimization as an embedded system. This approach provides great versatility and implementation in portable equipments. In order to accelerate the execution of image comparison using PCC, the Particle Swarm Optimization (PSO) is exploited. An embedded co-processor were developed and is dedicated to the correlation computation. The obtained results indicate that the proposed design is up to 13,28 faster than the software based solution, considering the studied cases. So, the co-design implementation with PCC computation implemented in hardware and PSO in software is a viable way to achieve real-time template matching, which is a pre-requisite in real-word applications.

Keywords— Embedded systems, co-design, particle swarm optimization, template matching, correlation, tracking.

Resumo— O *template matching* é uma técnica importante para rastreamento de padrões em imagens. O objetivo consiste em encontrar um padrão, a partir de um modelo pré-estabelecido, em uma sequência de *frames*. Para avaliar o grau de similaridade entre duas imagens, o Coeficiente de Correlação de Pearson (*Pearson's Correlation Coefficient* - PCC) é amplamente utilizado. Esse coeficiente é calculado para cada pixel, o que é computacionalmente muito custoso. Este artigo propõe a implementação do *template matching* em um sistema embarcado, conferindo-lhe grande versatilidade e possibilitando seu uso em equipamentos portáteis. Para acelerar o processamento, foi utilizada a Otimização por Enxame de Partículas (*Particle Swarm Optimization* – PSO) e foi desenvolvido um coprocessador dedicado para o cálculo da correlação. Os resultados obtidos indicam que o projeto proposto é 13,28 vezes mais rápido que a solução baseada em software, considerando os casos estudados. Assim, o projeto integrado, obtido com implementação do PCC em hardware e PSO em software, é um ótimo caminho para atingir processamento em tempo real para *template matching*, que é um pré-requisito para aplicações no mundo real.

Palavras-chave— Sistemas embarcados, co-design, otimização por enxame de partículas, template matching, correlação, rastreamento.

1 Introdução

Com o desenvolvimento e aprimoramento de sensores e equipamentos inteligentes capazes de capturar, armazenar, editar e transmitir imagens e vídeos, a aquisição de informação por meio destes tornou-se uma importante área de pesquisa. Na área de segurança e defesa, esse tipo de pesquisa é de grande relevância para reconhecimento e rastreamento de padrões referentes a alvos em sequências de imagens, podendo proporcionar soluções para o desenvolvimento de sistemas de vigilância (Narayana, 2007), monitoramento, controle de fogo (Ali et al., 2009), guiamento (Choi and Kim, 2014), navegação (Forlenza et al., 2012), biometria remota (Benfold and Reid, 2011), armas guiadas (Olson and Sanford, 1999), entre outros.

Um padrão é um arranjo ou coleção de objetos que são similares entre si, caracterizado pela arrumação de seus elementos. Uma das técnicas amplamente utilizadas para encontrar e rastrear padrões em imagens é identificada por *template matching* (Ahuja and Tuli, 2013)(Mahalakshmi et al., 2012). Dentre as técnicas de *template matching* utilizadas para rastreamento de padrões (alvo), a correlação é um método bastante conhecido e muito utilizado. Esta tarefa se resume basicamente em encontrar ocorrências de uma imagem de tamanho reduzido, considerada como modelo ou *template*, dentro de uma imagem de tamanho bem maior do que o do *template*. Esta operação é muito custosa computacionalmente, sobretudo quando são considerados grandes modelos a serem encontrados em um extenso conjunto de imagens (Sharma and Kaur, 2013).

Este artigo propõe a implementação de *template matching* via um projeto integrado de software/hardware, utilizando a técnica de otimização

baseada em enxame do partículas (PSO), implementada em software, e cálculo do coeficiente de correlação, implementado em hardware via um coprocessador dedicado. Para avaliar a solução proposta, é realizada uma comparação do tempo de processamento com e sem o hardware dedicado, que pode ser usado de maneira sequencial ou em pipeline.

Para tal, a Seção 2 apresenta alguns trabalhos relacionados; a Seção 3 define o conceito de correlação; a Seção 4 descreve a arquitetura do hardware proposto; a Seção 5 descreve a metodologia empregada e a implementação do sistema; a Seção 6 apresenta e analisa os resultados; finalmente, a Seção 7 conclui o trabalho e aponta alguns trabalhos futuros.

2 Trabalhos relacionados

Em (Ngo et al., 2013) a metodologia co-design é utilizada para implementar um sistema automático de vigilância de vídeo em um dispositivo FPGA (*Field Programmable Gate Array*). Como a maioria das aplicações de processamento de imagens, o sistema deve ter alto desempenho para suportar aplicações em tempo real. As operações mais computacionalmente custosas foram implementadas no FPGA e o controle e a interface com o usuário foram desenvolvidas em software e executadas por processador de propósito geral. Para o rastreamento de objetos, a técnica de *background subtraction* foi utilizada. Embora eficiente, essa técnica necessita que o plano de fundo permaneça estático e não é aplicável quando os alvos ou o plano de fundo mudam ao longo dos frames.

Em (Mahalakshmi et al., 2012), várias técnicas de *template matching* são discutidas. Uma arquitetura baseada em correlação espectral e sua implementação em FPGA são propostas. A correlação é calculada no domínio da frequência por meio da Transformada Rápida de Fourier em duas dimensões e multiplicação complexa.

Em (Tavares et al., 2015), os autores realizaram uma comparação do desempenho dos algoritmos genéticos *vs.* PSO para execução de *template matching*. O PSO obteve melhores resultados, em termos de tempo de processamento e robustez, e demonstrou alta convergência e grande simplicidade de implementação. Esta abordagem demonstrou ser boa solução para o problema de rastreamento de padrões. Contudo, em aplicações de segurança e defesa, o sistema precisa estar embarcado no próprio equipamento. Diferentemente dos trabalhos citados, o principal propósito deste artigo foi desenvolver um sistema portátil, capaz de processar imagens em tempo real, com mudanças de comportamento do alvo e do plano de fundo. Além disso, ao contrário dos trabalhos encontrados, o cálculo da correlação foi implementado no domínio do tempo.

3 Correlação de Imagens

O *template matching* é utilizado para rastrear padrões em processamento de imagens por meio da similaridade entre duas entidades, tais como pixels, curvas ou formas de mesmo tipo. O padrão ser reconhecido dentro de uma imagem é comparado com o modelo previamente armazenado, para todas as possibilidades.

Dentre as técnicas de *template matching* para rastreamento de padrões, a correlação é um método bastante conhecido e utilizado. O coeficiente de correlação de Pearson (PCC) é utilizado como medida de similaridade entre duas variáveis e pode ser entendido como um índice adimensional com valores situados ente -1 e 1 inclusive, que reflete a intensidade do grau de relacionamento entre duas variáveis. Coeficiente igual a 1 significa uma correlação perfeita positiva entre as duas variáveis. Coeficiente igual a -1 significa uma correlação negativa perfeita entre as duas variáveis. Coeficiente igual a zero significa que as duas variáveis não dependem linearmente uma da outra. Para aplicações em imagens, o PCC pode ser calculado conforme a Equação 1:

$$corr = \frac{\sum_{i=1}^{N}(p_i - \overline{p})(a_i - \overline{a})}{\sqrt{\sum_{i=1}^{N}(p_i - \overline{p})^2}\sqrt{\sum_{i=1}^{N}(a_i - \overline{a})^2}}, \quad (1)$$

onde p_i é a intensidade do pixel i na imagem padrão (*template*); $\overline{p}$ é a média das intensidades dos pixels da imagem padrão; a_i é a intensidade do pixel i no pedaço de imagem original A; $\overline{a}$ é a média da intensidade dos pixels no pedaço de imagem A. O *template* e o pedaço de imagem A devem ter o mesmo tamanho.

4 Arquitetura do Hardware

A parte mais custosa do *template matching*, no caso abordado, é o cálculo do PCC. Para obter melhor tempo de processamento e possibilitar aplicações em tempo real, esse cálculo foi implementado em coprocessador dedicado, aproveitando o paralelismo intrínseco do hardware. Além disso, o processo de rastreamento do maior ponto de correlação foi auxiliado pelo PSO, que é executado em software, pelo processador de propósito geral. Esta abordagem, chamada *co-design*, é uma metodologia para desenvolver um sistema integrado, utilizando componentes de hardware e software, para atender requisitos de desempenho e limitações de custo (Nedjah and Mourelle, 2007).

A macroarquitetura do sistema integrado proposto é apresentada na Figura 1. O sistema contempla um processador (PS) para execução do passo do PSO, um coprocessador para o cálculo do valor do coeficiente correlação, os blocos de

memória dedicados (BRAM IMG e BRAM TMP) para armazenar a imagem original e o template, respectivamente, e os blocos do controle de acesso a essas memória (GET IMG e GET TMP).

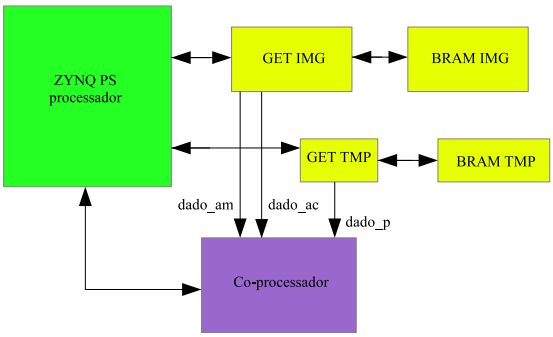


Figura 1: Macroarquitetura do sistema proposto

4.1 Coprocessador dedicado

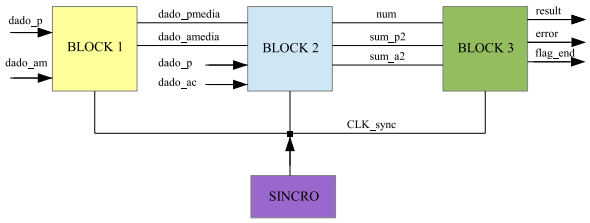


Figura 2: Macroarquiteura do coprocessador

A Figura 2 apresenta a arquitetura proposta para o coprocessador. Este é responsável por realizar o cálculo da correlação entre duas imagens, conforme definido na Equação 1. Sua estrutura é projetada na forma de *pipeline*, onde cada um de seus três blocos corresponde a um dos seus três estágios. A cada transição de subida do sinal do *clock*, 3 informações são recebidas pelo coprocessador: `dado_p` (um pixel da imagem de referência representado por 8 bits); `dado_ac` (um pixel da imagem a ser comparada também de 8 bits); e `dado_am` (um pixel da próxima imagem a ser comparada de 8 bits).

Todas as imagens a serem comparadas foram definidas com 64x64 pixels, totalizando 4096 pixels que são representados por 4 KBytes. Os cálculos são realizados em cada bloco e passados para o próximo, a cada pulso de sincronismo. Esse pulso é gerado pelo componente Sincro a cada 4103 transições de subida do sinal do *clock*. Como saída, o coprocessador fornece o valor da correlação calculado (`result`), em complemento a 2, codificado em 32 bits, o flag indicando término (`flag_end`) e um sinal de erro (`error`) que indica que o resultado não é válido, sendo oriundo de uma divisão por zero.

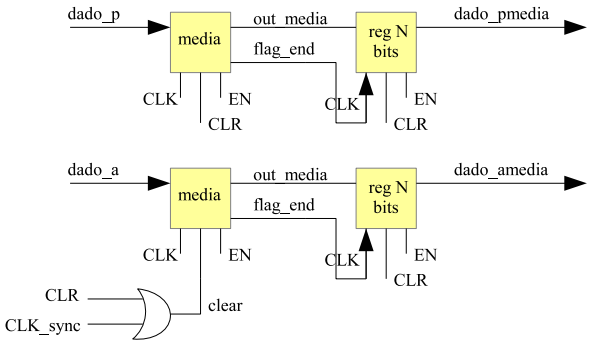


Figura 3: Microarquitetura do Bloco 1

4.1.1 Primeiro Estágio

A Figura 3 exibe o bloco 1 que forma o primeiro estágio do *pipeline* e é responsável por calcular a média dos pixels das duas imagens a serem comparadas. Possui registradores de saída que são carregados somente quando a tarefa do estágio é completada. Com pulso de sincronismo, somente o componente `media`, responsável por calcular a média da imagem a ser compara com o *template* é reiniciado.

4.1.2 Segundo Estágio

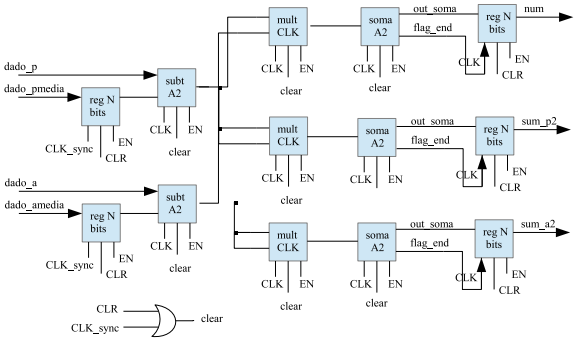


Figura 4: Microarquitetura do bloco 2

A Figura 4 exibe a arquitetura do bloco 2 que forma o segundo estágio do *pipeline*. Este é responsável por calcular os 3 somatórios da Equação 1. É composto por 2 componentes `subt_A2` que realizam a subtração em complemento a 2 dos pixels das imagens com as médias calculadas pelo bloco 1; por 3 componentes `mult_CLK` que realizam, em um pulso de *clock*, a multiplicação dos resultados oriundos dos componentes `subt_A2`; e por 3 componentes `soma_A2` que realizam os somatório, em complemento a 2, dos resultados das multiplicações provenientes dos componentes `mult_CLK`. Da mesma forma que no caso do bloco 1, o bloco 2 possui registradores de saída que são carregados somente quando a computação do estágio é completada. Com o pulso de sincronismo, os compo-

nentes `subt_A2`, `mult_CLK` e `subt_A2` são reiniciados.

4.1.3 Terceiro Estágio

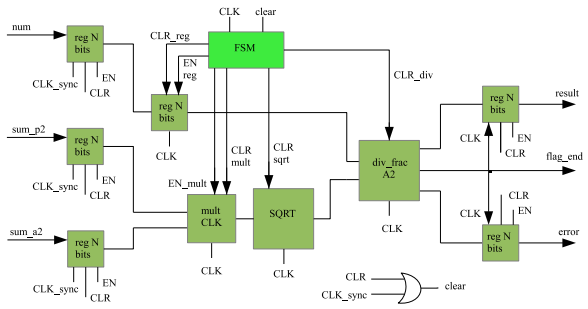


Figura 5: Microarquitetura do Bloco 3

A Figura 5 exibe a arquitetura do bloco 3, compondo o terceiro estágio do *pipeline*. Este é responsável por calcular a multiplicação principal, a raiz quadrada e a divisão da Equação 1. É composto pelo componente `mult_CLK` que, em um pulso de CLK, executa a multiplicação dos somatórios do denominador da Equação 1; pelo componente `SQRT` que calcula a raiz quadrada; e pelo componente `div_frac_A2` que realiza a divisão, proporcionando um resultado com a precisão de 2^{-24}. Este último componente é aquele que fornece os sinais de saída do coprocessador. A operação deste bloco é controlada por uma máquina de estado (`FSM`), coordenando a atuação dos seus componentes. Com o pulso de sincronismo, a máquina retorna ao seu estado inicial. Da mesma forma que os blocos 1 e 2, possui registradores de saída que são carregados somente quando a tarefa do estágio é completada.

O método numérico *babylonian* foi utilizado para cálculo da raiz quadrada. Baseado nele, o componente `SQRT` do bloco 3 foi implementado, em hardware, utilizando um processo iterativo de M passos com base na Equação 2:

$$x_{i+1} = 0,5 \times (x_i + \frac{S}{x_i}), \qquad (2)$$

onde x é o radical, S é o radicando ($x = \sqrt{S}$), x_i é o valor de x na iteração atual e x_{i+1} é o valor da próxima iteração. O número de iterações M é definido de acordo com a precisão do resultado requerida.

Ademais, o Algoritmo 1 foi utilizado para o cálculo da divisão entre dois números de N bits, com resultados de 24 bits nas casas fracionárias. Esse algoritmo foi implementado, em hardware, para o desenvolvimento do componente `div_frac_A2`.

Algoritmo 1 Divisão $Q = A/B$

$N_B := 0$
$RA_{\langle N-1 \ldots \frac{N}{2} \rangle} :=$ A
$RA_{\langle \frac{N}{2}-1 \ldots 0 \rangle} := 0$
while $RA_{\langle N-1 \ldots \frac{N}{2} \rangle} > B$ **do**
  $RA :=$ deslocar RA para direita
  $N_B := N_B + 1$
end while
$Q := 0$
for $j = 1, N_B$ **do**
  $RA :=$ deslocar RA para esquerda
  $Q :=$ deslocar Q para esquerda
  if $RA_{\langle N-1 \ldots \frac{N}{2} \rangle} > B$ **then**
    $RA_{\langle N-1 \ldots \frac{N}{2} \rangle} := RA_{\langle N-1 \ldots \frac{N}{2} \rangle} - B$
    $Q_0 := 1$
  else
    $Q_0 := 0$
  end if
end for
for $j = 1, 24$ **do**
  $RA :=$ deslocar RA para esquerda
  $Q :=$ deslocar Q para esquerda
  if $RA_{\langle N-1 \ldots \frac{N}{2} \rangle} > B$ **then**
    $RA_{\langle N-1 \ldots \frac{N}{2} \rangle} := RA_{\langle N-1 \ldots \frac{N}{2} \rangle} - B$
    $Q_0 := 1$
  else
    $Q_0 := 0$
  end if
end for

4.2 Controladores das memórias

Os blocos de memória dedicada `BRAM_TMP` e `BRAM_IMG` armazenam o *template* e a imagem principal, respectivamente. Estes foram implementados na lógica programável do chip Zynq (PL). O componente `BRAM_TMP` pode armazenar até 4096 pixels de 8 bits, totalizando 4K bytes (o que corresponde ao tamanho do *template*). Já o componente `BRAM_IMG` pode armazenar até 573x463 pixels de 8 bits cada, totalizando 260K bytes. Como as bordas da imagem principal são completadas com zeros, o tamanho máximo desta imagem é 510x400 pixels.

Os controladores `GET_TMP` e `GET_IMG` são responsáveis por proporcionar o acesso às memórias dedicadas `BRAM_TMP` e `BRAM_IMG`, respectivamente e disponibilizar ao coprocessador os dados, no momento correto. Os processo de acesso em leitura e escrita às memórias é sincronizado pelo sinal do clock (`CLK`) e pelo sinal de sincronismo (`CLK_sync`). Além dessa função principal, esses controladores de memória também permitem que o processador acesse as memórias dedicadas.

5 Metodologia e Implementação

O projeto assim proposto foi implementado utilizando a placa *Smart Vision Development Kit* rev 1.2 (SVDK) da *Sensor to Image*. Esta placa dispõe de um chip XC7Z015, da Xilinx, que possui sistema de processamento (PS), baseado em processador dual-core ARM Cortex-A9, e lógica programável (PL) em um único chip. Esse componente fornece a escalabilidade e flexibilidade de um

FPGA com o desempenho, capacidade e facilidade de uso associados com ASIC e é perfeito para a implementações do tipo co-design. Para avaliar o desempenho do projeto integrado assim proposto, foram utilizados três cenários: *(1)* SO: PSO implementado em software somente e executado pelo processador; *(2)* HS: o PSO implementado como um projeto integrado de software/hardware com o coprocessador funcionando em modo sequencial; e *(3)* HP: o PSO implementado como um projeto integrado de software/hardware com o coprocessador funcionando em *pipeline*.

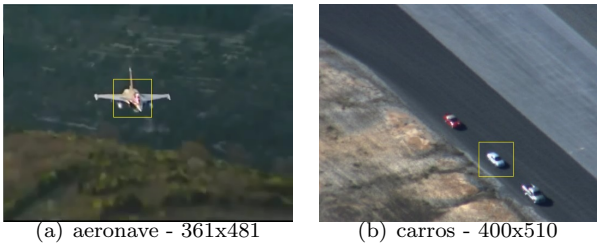
Para o primeiro cenário, o código do PSO foi desenvolvido em C para ser executado no processador, com os seguintes passos: *(1)* geração da população inicial das partículas, com posições e velocidades aleatórias; *(2)* para cada partícula do enxame, extração de um pedaço (recorte), de mesmo tamanho do *template* e cálculo do PCC na posição da partícula, que coincide com a do pixel central do recorte associado; *(3)* armazenar o melhor valor encontrado pela partícula e o melhor valor encontrado pelo enxame; *(4)* repetir os passos 2 e 3 até que o critério de parada seja atingido. Todas as imagens foram convertidas para tons de cinza e as bordas da imagem principal foram completadas com zeros.

Para o segundo e terceiro cenários, o passo 2 do PSO foi substituído por: *(1)* passar as informações de posição atual das partículas para o coprocessador dedicado e *(2)* aguardar o retorno dos valores da correlação para cada uma delas. As imagens utilizadas durante os testes foram transmitidas para a placa SVDK por meio do software MATLAB, utilizando porta serial e copiadas nas memórias pelo processador. O algoritmo PSO *global best* tradicional foi utilizado, com PCC como função objetivo, e os parâmetros foram configurados, empiricamente, da seguinte forma: 50 partículas; coeficiente inercial $w = 1$; coeficiente cognitivo $c_1 = 1,5$; coeficiente social $c_2 = 2$; e velocidade máxima igual a 10. Como critério de parada foi estabelecido valor de 0,95 para PCC ou número máximo de 30 iterações.

6 Resultados

A implementação na placa SVDK ocupou 11% de seus flip-flops, 39% de suas LUT (*Look-up Table*), 25% dos buffers e 69% de suas BRAM (*Block RAM*). Por limitações de tempo impostas pela ferramenta de síntese, principalmente no uso das BRAM, o coprocessador executou as tarefas com um clock de 25 MHz. Contudo, o processador ZYNQ pode fornecer para as BRAMs um clock de até 250 MHz.

As Figuras 6(a), 6(b) e 6(c) exibem as imagens utilizadas e os *templates* de cada uma marcados pelo quadrado interno. Estas foram extraídas do vídeo disponível em (YouTube, 2016) e dos primeiros *frames* dos *benchmarks* EgTest02 e EgTest05 disponíveis em (Collins et al., 2005).



(a) aeronave - 361x481 (b) carros - 400x510

(c) pickup - 400x510

Figura 6: Imagens

Tabela 1: Comparação do tempo processamento (ms) de 1 e 50 cálculos de PCC

#PCC	Caso	Aeronave	Carros	Pickup
1	1	2,010	2,010	2,010
	2	0,404	0,403	0,401
	Ganho_{2para1}	4,97	4,99	5,01
50	3	112,828	114,760	114,977
	4	20,095	20,076	20,051
	5	8,462	8,462	8,461
	Ganho_{4para3}	5.61	5,71	5,72
	Ganho_{5para4}	2,37	2,37	2,37
	Ganho_{5para3}	13,33	13,56	13,59

Primeiramente, foram feitos 3 recortes de 64x64 pixels dos *templates* identificados em cada imagem. O valor do PCC foi calculado para cada *template* com relação a ele mesmo. O cálculo da correlação foi executado pelo processador sozinho (caso 1) e com o auxílio do coprocessador (caso 2). Os resultados encontram-se na parte superior da Tabela 1, que diz respeito a 1 cálculo de PCC.

Depois disso, foram extraídos 50 pedaços aleatórios das imagens, de 64x64 pixels, e comparados com seus respectivos *templates*. Foi comparado o tempo gasto pelo processador para realizar os

Tabela 2: Comparação do tempo médio de processamento (ms) do PSO

Cenário	aeronave	carros	pickup
SO	2190,86	2702,17	2278,73
HS	406,14	451,87	394,82
HP	176,03	203,48	174,26
$\text{Ganho}_{HSparaSO}$	5,39	5,98	5,77
$\text{Ganho}_{HPparaHS}$	2,31	2,22	2,26
$\text{Ganho}_{HPparaSO}$	12,44	13,28	13,08

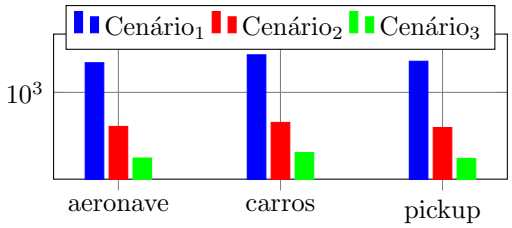


Figura 7: Diagrama em barras do tempo médio de processamento (escala logarítmica)

50 cálculos de correlação sozinho (caso 3), com o auxílio do coprocessador sequencialmente (caso 4) e sua estrutura *pipeline* (caso 5). Os resultados encontram-se na parte inferior da Tabela 1 que diz respeito a 50 cálculos de PCC.

Por fim, o projeto integrado proposto foi avaliado na execução da tarefa de rastreamento de padrões. Para tal, a Tabela 2 exibe os resultados médios para execução de 100 rastreamentos dos *templates* nas imagens utilizando os 3 cenários definidos na Seção 5. A Figura 7 ilustra essa comparação. Observa-se que a abordagem co-design, considerando o cenário com *pipeline* (HP), conseguiu melhorar o tempo de processamento em 12,44, 13,28 e 13,08 vezes para as imagens *aeronave*, *carros* e *pickup*, respectivamente, comparando-se com a solução somente de software (SO).

7 Conclusão

A abordagem co-design proposta, com implementação do PCC em hardware e PSO em software, é um ótimo caminho para atingir processamento de rastreamento de padrões em tempo real, o que é um pré-requisito para aplicações no mundo real. Conseguiu-se acelerar o tempo de processamento das imagens em até 13,28 vezes. Para melhorar ainda mais os resultados, o coprocessador pode trabalhar em frequências maiores, quando forem superadas as limitação da síntese, e pode-se limitar a janela de rastreamento do alvo.

Referências

Ahuja, K. and Tuli, P. (2013). Object recognition by template matching using correlations and phase angle method, *International Journal of Advanced Research in Computer and Communication Engineering* **2**(3): 1368–1373.

Ali, A., Kausar, H. and Khan, M. I. (2009). Automatic visual tracking and firing system for anti aircraft machine gun, *Applied Sciences and Technology (IBCAST), 2009 6th International Bhurban Conference on*, IEEE, pp. 253–257.

Benfold, B. and Reid, I. (2011). Stable multi-target tracking in real-time surveillance video, *Computer Vision and Pattern Recognition, IEEE Conference on*, pp. 3457–3464.

Choi, H. and Kim, Y. (2014). Uav guidance using a monocular-vision sensor for aerial target tracking, *Control Engineering Practice* **22**: 10–19.

Collins, R., Zhou, X. and Teh, S. K. (2005). An open source tracking testbed and evaluation, url: http://vision.cse.psu.edu/data/vivideval/datasets/datasets.html, acessado em Agosto 2016, *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* **2**(6): 35.

Forlenza, L., Fasano, G., Accardo, D. and Moccia, A. (2012). Flight performance analysis of an image processing algorithm for integrated sense-and-avoid systems, *International Journal of Aerospace Engineering* .

Mahalakshmi, T., Muthaiah, R., Swaminathan, P. and Nadu, T. (2012). Review article: an overview of template matching technique in image processing, *Research Journal of Applied Sciences, Engineering and Technology* **4**(24): 5469–5473.

Narayana, M. (2007). *Automatic Tracking of Moving Objects in Video for Surveillance Applications*, PhD thesis, University of Kansas.

Nedjah, N. and Mourelle, L. d. M. (2007). *Co-design for system acceleration: a quantitative approach*, Springer Science & Business Media.

Ngo, H. T., Ives, R. W., Rakvic, R. N. and Broussard, R. P. (2013). Real-time video surveillance on an embedded, programmable platform, *Microprocessors and Microsystems* **37**(6): 562–571.

Olson, T. L. and Sanford, C. W. (1999). Real-time multistage IR image-based tracker, *AeroSense'99*, International Society for Optics and Photonics, pp. 226–233.

Sharma, P. and Kaur, M. (2013). Classification in pattern recognition: A review, *International Journal of Advanced Research in Computer Science and Software Engineering* **3**(4): 3.

Tavares, Y., Nedjah, N. and Mourelle, L. d. M. (2015). Utilização de otimização por enxame de partículas e algoritmos genéticos em rastreamento de padrões, *12º Congresso Brasileiro de Inteligência Computacional* .

YouTube (2016). Rafale - avião caça de alta tecnologia (brasil), url: http://www.youtube.com/watch?v=e3wi-i_hdvq, acessado em Agosto 2016.