Controle de aeromodelo empregando visão computacional

Eduardo Barreto Alexandre[1,2], Marcelo Dornbusch Lopes[1], Thiago Rateke[1], Vito Francisco Chiarella[1,2], Antonio Carlos Sobieranski[1,2], Eros Comunello[1,2], Aldo Von Wangenheim[1]

[1]INCoD – Instituto Nacional para Convergência Digital
Universidade Federal de Santa Catarina - (UFSC)
Florianópolis - SC - Brasil

[2]4Vision Lab – Universidade do Vale do Itajaí (UNIVALI)
Rodovia SC 401, nº 5.025 - Florianópolis, SC - Brasil

{eduardobarreto, vitochiarella, asobieranski, eros.com}@univali.br

{thiago, marcelo}@incod.ufsc.br, awangenh@inf.ufsc.br

Abstract. *Control and navigation tasks are fairly widespread in the field of robotics, performing activities such as missions risk aid and navigating in unfamiliar terrain. This paper presents a methodology for the control and navigation predictive on model aircraft, employing techniques of computer vision and pattern recognition. The experiments performed demonstrate the ability of the methodology for real-time applications.*

Resumo. *Controle e navegação são tarefas bastante difundidas no campo da robótica, desempenhando atividades como o auxílio em missões de risco e navegação em terreno desconhecido. Este artigo apresenta uma metodologia para o controle e navegação assistida de aeromodelos, empregando técnicas de visão computacional e reconhecimento de padrões. Os experimentos realizados demonstram a capacidade da metodologia para aplicações de tempo real.*

1. Introdução

O Reconhecimento de Padrões (RP) é comprovadamente uma tarefa desafiadora no campo da Visão Computacional (VC) desde seus primórdios, em [Verhagen 1975] uma câmera é utilizada para reconhecer padrões estabelecidos respondendo com ações previamente definidas. Tal desafio tornou o RP um dos principais focos de estudo em diversas áreas e campos de aplicação, tais como: controle de braço mecânico, realidade aumentada, navegação autônoma de robôs e veículos, dentre outras [Nickel and Stiefelhagen 2007].

Diversos sistemas computacionais para uma aplicação mais específica como controle de robôs através do RP, podem ser encontrados de forma comercial. Neste contexto de controle robótico uma tarefa amplamente atendida é a navegação autônoma ou assistida, onde a VC realiza atividades como o auxílio à missões de risco, imersão e realidade aumentada, navegação em terrenos desconhecidos, entretenimento e jogos digitais de uma forma geral.

O presente trabalho demonstra uma metodologia para o controle de navegação em aeromodelos, empregando técnicas de VC e RP. Para o trabalho proposto foi utilizado um

modelo com quatro rotores de pequeno porte e hardware limitado, portanto a abordagem dos experimentos efetuados refletem a natureza do dispositivo.

Desta forma o restante do artigo foi organizado conforme a descrição a seguir: na Seção 2 foi apresentado a metodologia proposta e o conceitos de implementação, na Seção 3 foi descrito o experimento realizado, por fim na Seção 4 os resultados, as dificuldades e perspectivas futuras foram discutidas.

2. Metodologia Proposta

A metodologia para controle de navegação proposta foi ilustrada na Figura 1, o *stream* de vídeo que é capturado pela câmera do aeromodelo é transmitido para o sistema, as imagens do vídeo passam por uma etapa de atenuação onde um filtro anisotrópico é aplicado.

Para o controle do aeromodelo empregado neste trabalho foi construído um artefato de isopor com 3 bolas de borracha dispostas de forma trinagular, referenciado durante o restante do texto como Triângulo de Controle TC.

Após a filtragem as imagens são classificadas e segmentadas isolando o (TC). Com os vértices do triângulo identificados são então computadas operações trigonométricas sobre estes pontos, tais operações são convertidas em comandos de navegação para o dispositivo.

Figura 1. Visão geral da abordagem proposta

O dispositivo de aeromodelo escolhido para este trabalho foi o Parrot Ar.Drone[1], um multicóptero de quatro rotores. A comunicação entre o aeromodelo e o sistema que implementa a metodologia proposta emprega o protocolo IEEE 802.11(b/g)[2], para conexão sem fio.

A comunicação é implementada com o conceito de *sockets* UDP (User Datagram Protocol) ou TCP (Transmission Control Protocol), onde grupos de informações distintas são atendidas em portas específicas:

- Porta 5554 UDP: Usada para coletar dados gerais de telemetria do Ar.Drone, como seu estado atual, sua posição, velocidade, angulo, altitude e câmera.

[1]http://ardrone.parrot.com/
[2]http://standards.ieee.org/findstds/standard/802.11-2012.html

- Porta 5555 UDP: Um *stream* de vídeo é enviado pelo Ar.Drone por esta porta, o vídeo pode ser da câmera frontal ou da câmera vertical, com resoluções de 640x480 e 176x144 respectivamente.
- Porta 5556 UDP: Usada para o controle e configuração do Ar.Drone, qualquer tipo de comando trafega por esta porta. Movimentação, altitude máxima e mínima e qual câmera será transmitida.
- Porta 5559 TCP: Dados críticos devem ser transferidos ou recebidos por essa porta, a diferença para as outras é a utilização de *socket* TCP ao invés de UDP. Certificando que o dado enviado foi realmente recebido pelo aeromodelo, assim ela é usada majoritariamente para a obtenção e envio de dados de configuração.

2.1. Módulo de Pré-processamento

Este módulo implementa a captura do *stream* de vídeo, obtido da câmera frontal do Ar.Drone. Onde as imagens apresentam uma resolução de 640x480 pixels, estabelecendo um *socket* de comunicação que utiliza a porta 5555.

Com a finalidade de atenuar ruídos advindos da limitação do hardware de captura, bem como da transmissão sem fio, este módulo implementa um filtro de difusão anisotrópica baseado em [Weickert 1996].

O referido filtro é aplicado as imagens do vídeo capturado proporcionando um aprimoramento nas mesmas, resultando em uma atenuação que preserva as bordas dos objetos presentes nas imagens do *stream*. Esta etapa proporciona uma melhoria no resultado do Módulo de Classificação descrito na Seção 2.2.

Um gargalo foi identificado nesta etapa devido ao alto custo para a computação deste filtro, portanto, como solução esta parte do módulo foi construída empregando o conceito de GPGPU (*General-purpose computing on Graphics Processing Unit* - Unidade de Processamento Gráfico de Propósito Geral) utilizando a especificação OpenCL [3], que proporciona programação paralela para sistemas heterogêneos.

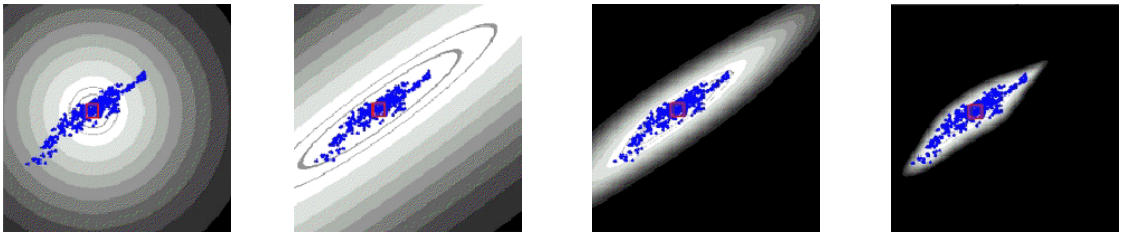
Tal subterfúgio proporcionou um aumento significativo no desempenho da abordagem como um todo, não tendo esta parte do código sendo executada na CPU (*Central Processor Unit*).

2.2. Módulo de Classificação

O Módulo de Classificação emprega a abordagem proposta por [Sobieranski et al. 2009] para o reconhecimento de padrões, esta abordagem estabelece um processo semi-supervisionado de classificação não-linear, empregando a Distância Polinomial de Mahalanobis (DPM) [Mahalanobis 1936] para tomada de decisão. A referida metodologia utiliza uma etapa de treinamento para a aprendizagem do padrão alvo.

Nota-se que a diferença entre as ordens do polinomio afeta consideravelmente a classificação final, sendo que quanto maior a ordem, mais restritivo e especializado a classificação se torna, como mostrado na Figura 2, onde a classificação de uma distribuição é demonstranda, onde a Figura 2(a) representa distância euclidiana ou DPM com ordem 0 (zero), a Figura 2(b) com ordem 1, a Figura 2(c) com ordem 2 e finalmente a Figura 2(d) com ordem 4.

[3] http://www.khronos.org/opencl/



(a) Distancia Euclidiana (b) DPM em ordem 1 (c) DPM em ordem 2 (d) DPM em ordem 4

Figura 2. Classificação de uma distribuição em várias ordens

A Figura 3 ilustra o processo de classificação e segmentação, uma imagem estática é capturada pela câmera do Ar.Drone Figura 3(a). A imagem adquirida deve conter o TC.

Na imagem estática capturada, Figura 3(b), são marcados com o mouse um conjunto de pontos que definem o padrão alvo. Este conjunto de pontos foi utilizado para a calibração do modelo de classificação. Na Figura 3(c) foi apresentado o resultado da segmentação.

A DM calcula a ponderação da distância de acordo com a variação estatística de cada componente, obtida pela seguinte equação:

$$d_M(x, y) = \|x - y\|_A = \sqrt{(x - y)^T A^{-1}(x - y)}, \quad (1)$$

onde $d_M(x, y)$ é a distância de Mahalanobis entre dois vetores de cores x e y e A^{-1} é a matriz inversa de covariância computada a partir de uma distribuição multivariada. Como demonstrado na Equação 1 a DM também reduz a norma do vetor se A é uma matriz identidade.

(a) Detalhe da câmera do Ar.Drone (b) Imagem capturada (c) Imagem segmentada

Figura 3. Metodologia de classificação

Maiores detalhes sobre a implementação e o *background* matemático envolvidos na metodologia de classificação e segmentação podem ser encontradas em [Sobieranski et al. 2010] e [Sobieranski et al. 2011].

2.3. Módulo de Controle

Este módulo recebe a imagem segmentada como vista na Figura 3(c), as coordenadas (*x,y*) de cada ponto do triângulo são então identificadas. Com isso é possível calcular a distância entre os vértices do triângulo identificado, computando a distância entre os pontos através do teorema de Pitágoras, Equação 2.

$$ladoAB = \sqrt{(x1 - x2)^2 + (y1 - y2)^2} \tag{2}$$

A próxima etapa efetua o cálculo dos ângulos dos vértices do triângulo, primeiro é calculado o cosseno de cada vértice (Equação 3). Em seguida é calculado o arco cosseno de cada vértice tranformando o valor em graus (Equação 4). O valor de π na Equação 4 é 3.1415, onde ocorre a tranformação do valor do vértice de radianos para graus.

$$cossenoA = \frac{(ladoAC)^2 + (ladoAB)^2 - (ladoBC)^2}{2 * ladoAC * ladoAB} \tag{3}$$

$$anguloA = arcosseno(cossenoA) * \frac{180}{\pi} \tag{4}$$

Também através de cálculos trigonométricos é possível calcular a área do triângulo, calculando primeiro o perímetro com a Equação 5 e o semi-perímetro com a Equação 6, representado pelo símbolo Ψ.

Finalmente a área do triângulo é obtida empregando a Equação 7. Estes valores de área e distância entre os vértices são então convertidos em comandos de navegação para o Ar.Drone.

$$perimetro = ladoAB + ladoAC + ladoBC \tag{5}$$

$$semi - perimetro = \frac{perimetro}{2} \tag{6}$$

$$area = \sqrt{\Psi * (\Psi - ladoAB) * (\Psi - ladoAC) * (\Psi - ladoBC)} \tag{7}$$

A metodologia proposta implementou três das quatro opções de movimentações possíveis pelo Ar.Drone, *roll, pitch e gaz*. O deslocamento na horizontal é controlado pela opção *roll*, o deslocamento para frente e para trás é controlado pela opção *pitch* e o deslocamento na vertical pela opção *gaz*. A Figura 4 mostra o cálculo dos vértices e a distância entre eles, definindo o triângulo de controle.

Para o deslocamento na horizontal e na vertical a metodologia emprega o centro de massa do triângulo como guia, este centro é obtido dividindo o perímetro do triângulo por 3, a posição deste centro é verificada em relação ao tamanho do frame.

Uma vez que este centro esteja se deslocando para algum lado, o algoritmo emite uma compensação mandando o aeromodelo para o lado oposto, estabilizando o centro do triângulo no centro do frame.

Já para o deslocamento do *pitch*, é necessário saber a distância entre o objeto de controle e o Ar.Drone, assim o algoritmo utiliza os três pontos do triângulo como referência para a criação de um *bounding box*. Na primeira imagem onde o triângulo é identificado, o tamanho do referido *bounding box* é armazenado como referência.

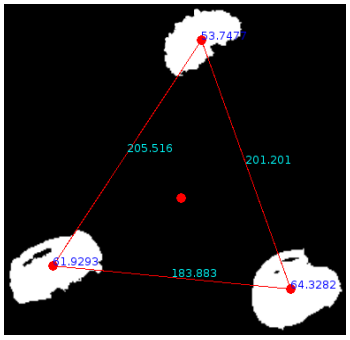


Figura 4. Triângulo de controle, com os valores dos ângulos dos vértices e suas distâncias, usados no controle do Ar.Drone.

Na análise das seguintes imagens do *stream* esse tamanho inicial de referência é comparado, verificando a proporção entre os tamanhos, desta forma ocorre a movimentação do aeromodelo para frente caso o tamanho atual seja menor que o tamanho de referência, e o inverso caso contrário.

Utilizando os métodos descritos ocorre a estabilização do Ar.Drone, permitindo ao usuário o envio de comandos de navegação para o aeromodelo, apenas movimentando o TC.

3. Experimento

A condução do experimento foi realizada em ambiente *indoor*, inicialmente através de observações empíricas foi percebido um segundo gargalo de custo computacional elevado, referente ao Módulo de Segmentação da metodologia proposta.

Para a eliminação deste gargalo foi adotada a mesma abordagem discutida na Subseção 2.1, portando o código do referido módulo para OpenCL. Conduzindo assim a uma avaliação quantitativa de desempenho da metodologia, empregando cenários de software e hardware distintos.

Consequentemente foram montados duas Configurações de Software (CS) e três Cenários de Hardware (CH) para a execução dos experimentos, tais configurações são descritas a seguir:

- CS 01: todo o algoritmo da metodologia sendo executado na CPU.
- CS 02: os Módulos de Pré-processamento e Segmentação sendo executados na GPU.

Cada CS foi avaliada sendo executada em cada Cenários de Hardware. As especificações dos CH empregados foram:

- CH 01: CPU AMD[4] Athlon X2 1.9 GHz, GPU NVIDIA[5] 9100m.
- CH 02: CPU Intel[6] i5 2.8 GHz, GPU NVIDIA GT 218.
- CH 03: CPU Intel i7 2.9 GHz, GPU NVIDIA GTX 560m.

[4]http://www.amd.com/br/pages/amdhomepage.aspx
[5]http://www.nvidia.com.br/page/home.html
[6]http://www.intel.com.br/content/www/br/pt/homepage.html

Além das diferenças de hardware e software, os experimentos consideraram também 3 ordens polinomiais diferentes, necessários para uma maior flexibilidade na hora da classificação correta nos mais variados cenários.

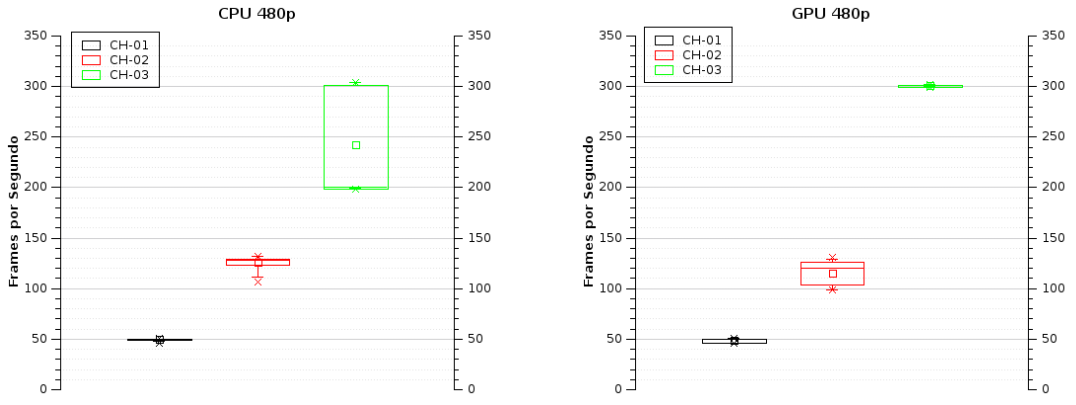
Apesar do quadrirotor de exemplo ser o Ar.Drone que contém uma resolução máxima de 640x480, outros quadrirotores podem conter variações nessas configurações, portanto, para nossos experimentos, duas resoluções foram usadas, sendo 640x480 uma e 1920x1080 outra, representando tanto o mínimo quanto o máximo teórico das câmeras desses aparelhos.

Os resultados dos experimentos foram plotados em gráfico estilo *box plot* Figuras 5 a 10, para os três cenários de hardware, as duas configurações de software e as duas resoluções distintas (640x480 e 1920x1080).

Utilizando a resolução de 640x480, pode ser observado que dependendo da ordem usada, tanto CS 01 quanto CS 02 são viáveis, porém, em nenhum desses casos CS 01 demonstra uma vantagem acima do CS 02, especialmente quando ordens maiores que 1 foram usadas: Figura 7(a), onde CH 01 demonstrou uma performance abaixo de 8 FPS e Figura 9(a), onde todos os CH falharam a obter FPS acima de cinco.

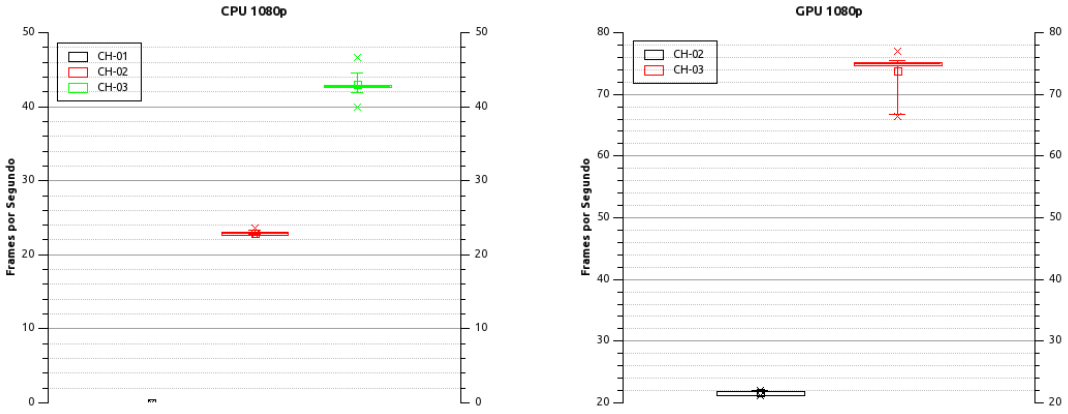
No caso da resolução 1080p, o CH 01 falhou ao executar os testes em CS 02 por uma limitação de memória interna de sua GPU (Figuras 6(b), 8(b) e 10(b)), portanto apenas CH 02 e CH 03 foram testados com CS 02.

Em CS 01, apenas em ordem 1 CH 02 e CH 03 conseguiram obter taxas de FPS em tempo real, CH 01 mostrou-se impraticável nessa resolução. Na CS 02 o CH 02 mostrou taxas aceitáveis na ordem 1 Figura 6(b) porém abaixo do necessário nas ordens 2 e 4 Figuras 8(b) e 10(b). Já o CH 03 manteve uma ótima performance em todas as ordens, Figuras 6(b), 8(b) e 10(b).



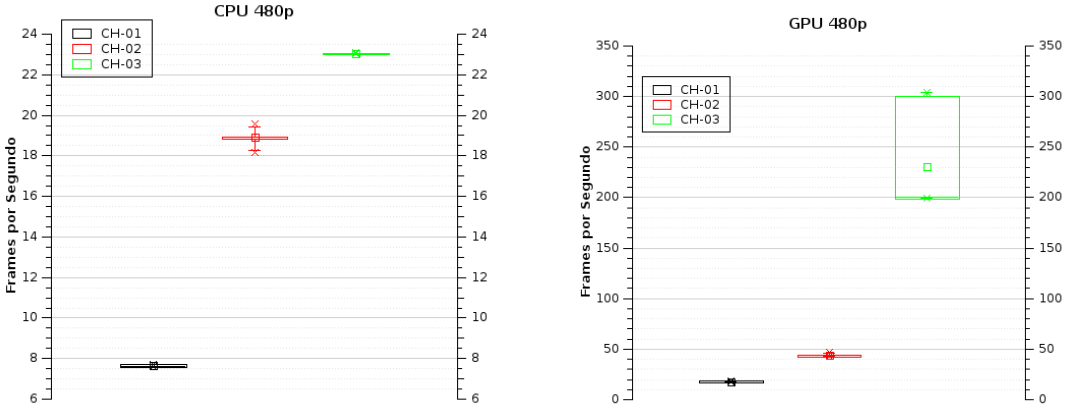
(a) 640x480 (b) 640x480

Figura 5. Gráficos de resultados dos experimentos com ordem polinomial 01



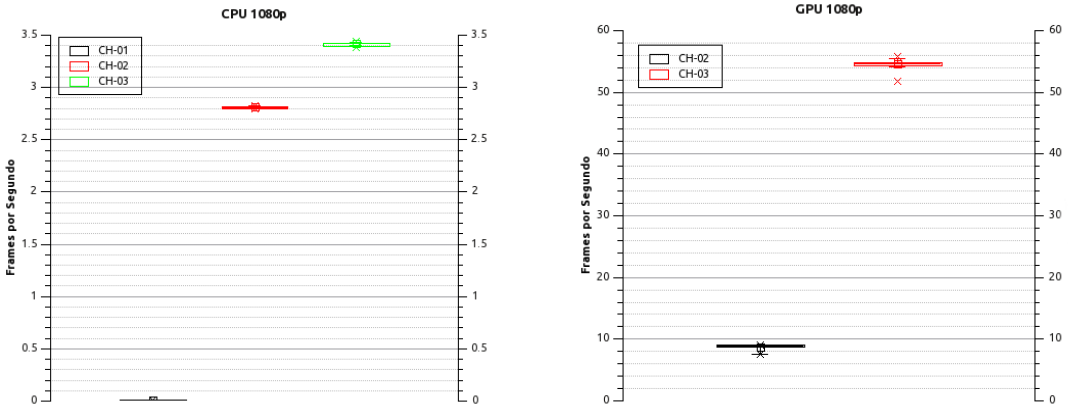
(a) 1920x1080 (b) 1920x1080

Figura 6. Gráficos de resultados dos experimentos com ordem polinomial 01



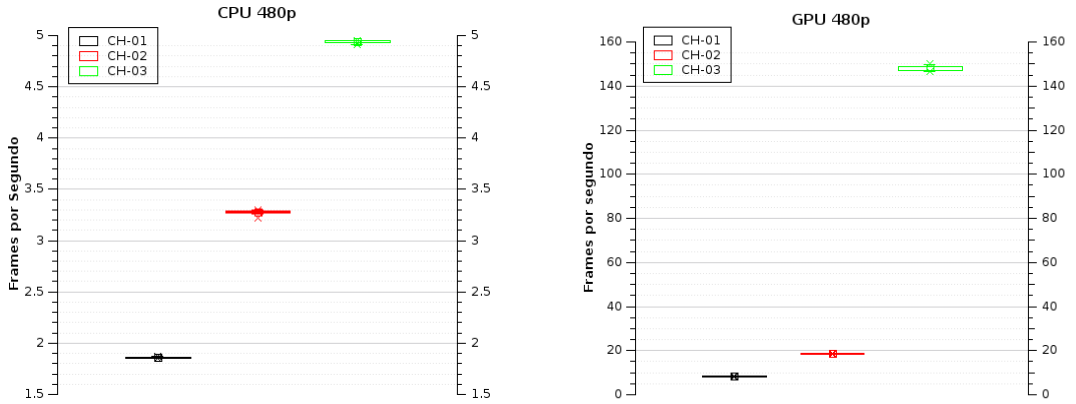
(a) 640x480 (b) 640x480

Figura 7. Gráficos de resultados dos experimentos com ordem polinomial 02



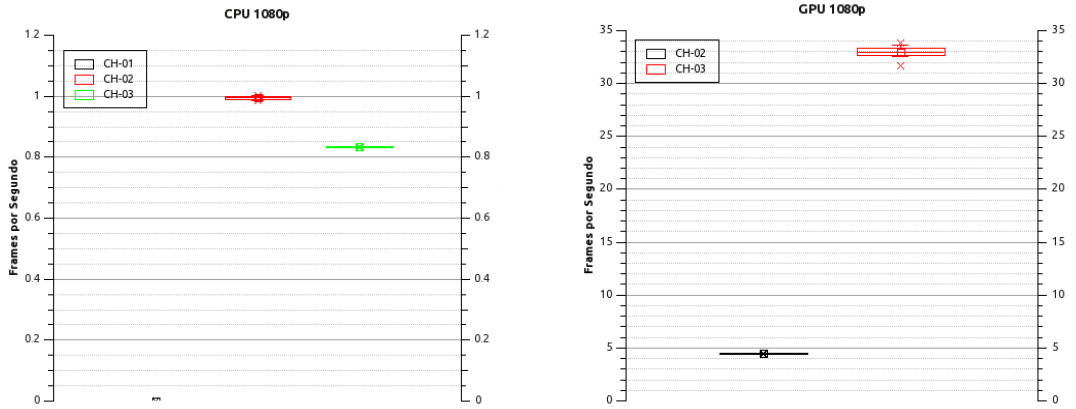
(a) 1920x1080 (b) 1920x1080

Figura 8. Gráficos de resultados dos experimentos com ordem polinomial 02



(a) 640x480 (b) 640x480

Figura 9. Gráficos de resultados dos experimentos com ordem polinomial 04



(a) 1920x1080 (b) 1920x1080

Figura 10. Gráficos de resultados dos experimentos com ordem polinomial 04

4. Discussão Final

Este artigo apresentou uma metodologia para o controle de navegação de aeromodelos baseado em VC, com três módulos bem definidos. O Módulo de Pré-Processamento que efetua correções nas imagens recebidas, o Módulo de Segmentação que usa uma métrica de classificação não-linear para a identificação do artefato de controle (ver Figura 3(b)), e o Módulo de Controle que efetua diversas operações trigonométricas as quais são convertidas em comandos para o Ar.Drone.

A abordagem proposta comprovou-se eficaz para o processamento em tempo real, uma vez que conseguiu atingir taxas de processamento com média de 33.01 FPS (0.46 de desvio padrão e 0.21 de variância). Tais valores foram obtidos na configuração de software que emprega GPGPU e no hardware mais atual (GPU NVIDIA GTX 560m), processando imagens com resolução de 1920x1080.

Pontos passíveis de expansão foram identificados, referentes a outros aspectos da trigonometria para o controle do aeromodelo. Como por exemplo, empregar a informação

dos ângulos dos vértices para realizar a movimentação de giro em torno do eixo do próprio dispositivo.

Diversas possibilidades de aplicação foram vislumbradas com a conclusão deste trabalho, para citar uma: aplicação em jogos digitais, enriquecendo assim o nível de interatividade homem-máquina.

Referências

Mahalanobis, P. C. (1936). On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55.

Nickel, K. and Stiefelhagen, R. (2007). Visual recognition of pointing gestures for human-robot interaction. *Image and Vision Computing*, 25(12):1875–1884.

Sobieranski, A. C., Comunello, E., and von Wangenheim, A. (2011). earning a nonlinear distance metric for supervised region-merging image segmentation. *Computer Vision and Image Understanding*, 115(2):127–139.

Sobieranski, A. C., Coser, L., Neto, S. L. M., Comunello, E., von Wangenheim, A., Giunta, G. D., and Cargnin-Ferreira, E. (2010). Reconhecimentoe quantificacao de expressoes de imunohistoqumica empregando aprendizado de metricas de distancia. *Revista Brasileira de Engenharia Biomedica*, 26:33–47.

Sobieranski, A. C., Neto, S. L. M., Coser, L., Comunello, E., von Wangenheim, A., Cargnin-Ferreira, E., and Giunta, G. D. (2009). Learning a nonlinear color distance metric for the identification of skin immunohistochemical staining. *Computer Based Medical Systems, IEEE Symposium on*, pages 1–7.

Verhagen, C. J. D. M. (1975). Some general remarks about pattern recognition; its definition; its relation with other disciplines; a literature survey. *Pattern Recognition*, 7:109–116.

Weickert, J. (1996). Anisotropic diffusion in image processing.