DESENVOLVIMENTO DE UM SISTEMA DE VISÃO PARA UMA FROTA DE ROBÔS

Kalyf Abdalla Buzar Lima (bolsista de ICV), Kelson Rômulo Teixeira Aires (Orientador, Departamento de Informática e Estatística – UFPI)

1. Introdução

Sistemas de visão, uma subárea da Inteligência computacional, têm como foco de estudo a utilização de processamento de imagem para geração de informações. Suas aplicações são aproveitas nas mais diversas áreas, fixando-se na robótica principalmente pelo seu baixo custo em hardware.

O custo dos sistemas de visão fica caro, contudo, no software. Os algoritmos utilizados por esses sistemas trabalham com cargas excessivas de dados na maioria das vezes superiores à 10^6 dados em uma única imagem.

Em resposta aos avanços tecnológicos, os sistemas heterogêneos surgem como uma expansão da arquitetura *multicore*, na qual se utiliza processadores de propósito específicos como de propósito geral para aumentar o desempenho de algoritmos que executem dessa forma.

Em especial, sistemas heterogêneos que utilizam GPGPU [1,2] (*General-Purpose computation on Graphics Processing Units*) possuem elevado potencial de avanço, visto que, é uma tecnologia que está na maioria dos computadores pessoais, contudo, é pouco explorada por seus usuários.

A computação paralela nos núcleos desse sistema possui tem como objetivo o ganho de tempo na execução de tarefas nas quais há independência nos dados trabalhados, o que corresponde à grande maioria dos algoritmos de processamento de imagem.

O problema de futebol de robôs já foi trabalhado diversas vezes entre os pesquisadores, seu grande desafio é quanto ao tempo do algoritmo que deve ser o mínimo possível pela dinamicidade das ações tanto de cada robô do time e do adversário.

Como diferencial, um sistema de futebol de robôs que utilizar a tecnologia de paralelismo na camada de detecção de objetos e localizar estes e inferir sua localização no espaço possui um tempo maior para a camada de planejamento de ações.

Para facilitar a programação desses tipos de sistemas a Apple propôs o OpenCL [3,4], um *framework* que permite ao programador a geração de código paralelo para sistemas heterogêneos [5,6] e ao mesmo tempo privando aquele da preocupação excessiva com controle de memória.

Palavras-chave: Futebol de robôs; Navegação de robôs; Processamento de imagens; Programação paralela.

2. Metodologia

Como a tecnologia da ferramenta é nova, não possui grande acervo de bibliografia disponível, os primeiros meses foram dedicados ao estudo das mesmas. Iniciando a fase de desenvolvimento, os algoritmos implementados foram:

- Segmentação por cores
    - Rede Neural K-Means; [7]
    - Estatística de componente; [8]
- Segmentação por linhas
    - Detector de contorno; [9]

- Descritor de imagem. [10]

Metodologia com vantagens e desvantagens, por ter uma gama enorme de implementação permitiu o conhecimento mais aprofundado da ferramenta e técnicas de implementação para cada algoritmo. Isso permitiu teste de desempenho entre tipos de algoritmos, criando um leque maior de estudo.

A desvantagem vem com a dificuldade de depuração de código. *Bug* em trecho de código tem maior gasto de tempo do que produzir o algoritmo desde o início e incompatibilidade entre modificações de SDKs (*Software Development Kit)* existentes.

3. Resultados

- Rede Neural K-Means

A rede neural K-Means apresentou ganhou de desempenho em 200% para número de classes entre os valores 5 e 10, essa decorrência se deve ao fato de o algoritmo ser parcialmente paralelizável.

- Estatística de componente

Foi o algoritmo que apresentou melhor custo-benefício, devido a sua simplicidade, seu desempenho foi rápido e interface que permite ao usuário seleção de área de interesse.

- Detector de contorno

Simples, desempenho alto, contudo, totalmente ineficaz para a proposta do projeto devido sua falha com ruídos, algo comum quando se está trabalhando com vídeos em tempo real.

- Descritor de imagem

Complexo, desempenho mediano, altamente eficiente para o objetivo, ineficaz devido ao tempo gasto que é muito superior aos outros algoritmos testados.

4. Discussão

- Rede Neural K-Means

A rede neural K-Means apresentou resultados satisfatórios, contudo, não é um algoritmo plenamente paralelizável e por se fixar em heurísticas não é o mais recomendado para o projeto. Um dos maiores problemas que foi observado é quanto o quesito variação de iluminação.

- Estatística de componente

Teve bons resultados com alta taxa de acerto, por ser plenamente paralelizável, pôde se utilizar o máximo potencial da GPU. Não apresentou grandes problemas de iluminação por não se limitar à um único *frame* e sim um agrupamento de *n* imagens do vídeo. Foi o escolhido para integrar o sistema também por sua confiabilidade, o sistema da UFRN utiliza uma variação deste para seu time de futebol de robôs.

- Detector de contorno

Simples, desempenho alto, contudo, totalmente ineficaz para a proposta do projeto, houve grande problema ao se trabalhar com degrades. Estes geravam contornos que não se apresentavam no mundo real e também devido sua falha com ruídos, algo comum quando se está trabalhando com vídeos em tempo real.

- Descritor de imagem

Algoritmo de complexidade razoável que apresentou desempenho mediano e eficaz, contudo, não apresentou tempo razoável para o sistema que necessita de tempo reduzido para ser

reaproveitado pela lógica de decisão. Por isso foi descartado, mas pode ser utilizado por outros sistemas que necessitam de detecções de padrões e futuros projetos.

5. Conclusão

Programação paralela se mostrou um diferencial para sistemas de visão de tempo-real, adaptações de algoritmos de processamento de imagem para o paralelismo elevou o desempenho em relação à CPU, afirmando o grande potencial da tecnologia CUDA em conjunto com o *framework* OpenCL.

A implementação de diversos algoritmos apesar de ter comprometido o cronograma de execução, permitiu o conhecimento aprofundado da ferramenta e a possibilidade da utilização da produção bibliográfica para estudos na área.

Referências

[1] NVIDIA, OpenCL Programming Guide for the CUDA Architecture Version 2.3. 2010, NVIDIA Corporation: Santa Clara, Califórnia.

[2] AMD, ATI Stream SDK OpenCL Programming Guide. June 2010. Advanced Micro Devices Inc.

[3] APPLE, Taking the graphics processor beyond graphics. June 2009, Apple Inc.

[4] http://gemma.apple.com/technologies/mac/snowleopard/opencl.html

[5] KHRONOS, Introduction and Overview. June 2010, Khronos Group.

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

[7] Kelson R.T.; ALSINA, Pablo J.; MEDEIROS, Adelardo A.D. A Global Vision System for Mobile Mini-Robots. Maio, 2001.

[8] CERQUEIRA, Auciomar C. T., LINS Filipe C. A., MEDEIROS, Adelardo A. D., ALSINA, Pablo J. ; A Versão 2006 da Equipe POTI de Futebol de Robôs. Abril 2006. Natal-RN.

[9] Gonzalez, Rafael C. E Richard E. Woods, Digital, 3rd ed, Prentice Hall, 2007.

[10] AIRES, Kelson R.T.; Segmentação de Planos Baseada em Homografia Afim, Fluxo Óptico e Reconstrução Métrica. Outubro, 2009. Natal-RN.