14º Exercício Prático
Trabalho de Final de Semestre

Objetivo

O objetivo desta atividade é utilizar técnicas já aplicadas anteriormente nesta disciplina para otimizar um programa que separa cores específicas de uma imagem.

Materiais

1. Compilador GCC
2. Arquivos SeparaCor.c
3. Arquivo com textos: Lapis.ppm.

Introdução

O Departamento de Computação da Faculdade de Ciências da UNESP, campus de Bauru, participa de competições de futebol de robôs, na modalidade Very Small Size (atual-mente IEEE Very Small), desde 1998, com a realização do 1o Campeonato Brasileiro de Futebol de Robôs – CBFR 98. A pesquisa e o desenvolvimento em futebol de robôs mantém o objetivo de incentivar o uso de inovações tecnológicas, no campo da robótica, de baixo custo e com componentes encontrados no mercado nacional. O time de futebol da UNESP de Bauru é conhecido, desde a primeira edição desta competição no Brasil, como Carrossel Caipira devido sua estratégia de jogo.

No ambiente do futebol de robôs, nesta categoria, os robôs e a bola são identificados através de uma câmera utilizada como visão global, posicionada a 2m sobre o campo e alinhada ao seu centro, que captura imagens da arena. Estas imagens são processadas digitalmente obtendo as coordenadas dos robôs e da bola. A partir dessas coordenadas, uma estratégia escolhida e transformada em comandos que são enviados aos robôs por rádio. Os robôs recebem estes comandos e realizam as ações correspondentes, modificando a posição dos elementos presentes no ambiente real, que será capturado novamente pela câmera. A Fig. 1 apresenta uma ilustração do ambiente do futebol de robôs.

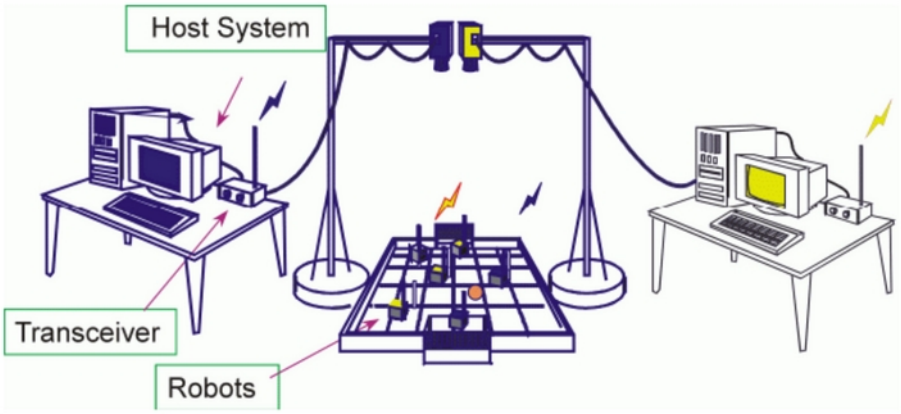

Fig. 1: Ambiente para futebol de robôs.
Fonte:www.mecatronicaatual.com.br/secoes/leitura/950 (2013)

O futebol robótico abrange diversas áreas do conhecimento. Na construção do robô são aplicados conceitos de mecânica, eletrônica e sistemas embarcados. Do ponto de vista do software, executado

no computador pessoal, estão envolvidos elementos de processamento de imagens, inteligência artificial e teoria de controle. Essa abrangência faz desta modalidade de futebol uma ferramenta pedagógica com possíveis aplicações na graduação.

O sistema deste time pode ser representado simplificadamente através do diagrama apresentado na Fig. 2 que indica as partes principais do processamento. A CÂMERA captura uma imagem do campo, esta imagem é então processada pelo módulo de VISÃO que determinará as posições atuais dos robôs a partir de suas etiquetas coloridas e a da bola que possui cor alaranjada. Com estas posições, o módulo de ESTRATÉGIA calcula os locais do campo que os robôs do time controlado deverão se posicionar.

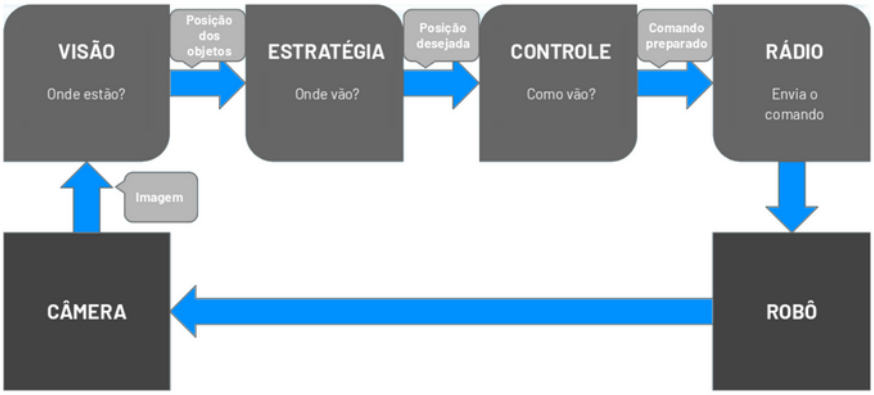

Fig. 2: Diagrama simplificado dos módulos do time Carrossel Caipira.

No ambiente de futebol de robôs toda a estratégia e o controle, tanto de baixo nível quanto de alto nível, são baseados na interpretação das imagens captadas pela câmera. Para que isso seja realizado, etiquetas de cores em destaque localizadas no topo dos robôs, conforme demonstra a Fig. 3, identificam cada um deles.

Fig. 3: Etiqueta de identificação do robô

O tempo de execução do ciclo de controle do sistema foi definido pela taxa de aquisição de imagens. Atualmente o time Carrossel Caipira tem a disposição uma câmera capaz de capturar 120 imagens por segundo com resolução 2560x1600 pixels. Portanto, a cada período de 8.3 ms, uma nova imagem refletindo o estado atual do campo torna-se disponível ao computador para processamento. Cada uma dessas imagens é capturada e digitalizada pela câmera, que disponibiliza, na forma de uma matriz com dimensões 2560x1600 pixels com três canais (componentes Red, Green, Blue – RGB). Cada um desses pixels deve ser analisado, quanto a sua cor, para identificar se

é uma cor de importância ao sistema, esta técnica é chamada de segmentação de cor.

Porem, hoje, o código de segmentação de imagens é antigo e foi feito para processar 30 imagens por segundo com resolução 640x480 pixels, um código inicial para separar as cores na nova resolução foi desenvolvido, porém com a nova resolução de 2560x1600 poucos quadros por segundo podem ser processados, inviabilizando a utilização deste código como está. Assim, estamos "contratando" você para ajudar no time de Futebol de Robôs Carrossel Caipira, otimizando o este código inicial que separa cores das etiquetas. Outro detalhes, como o sistema de software do time é composto por outros módulos, quanto menor o tempo de execução gasto com as imagens, mais tempo para os outros módulos.

A separação de cores é realizada pixel a pixel e deve acontecer a partir de uma cor padrão fornecida que identifica a cor de uma etiqueta sobre o robô na imagem capturada pela câmera. Considerando que cada pixel, com seus canais (R,G,B), determinam um ponto tridimensional, para saber quão próxima uma cor de um pixel está da cor padrão, usa-se a distância euclideana entre o ponto definido pelo (R,G,B) da etiqueta padrão e o ponto (R,G,B) do pixel analisado.

Nesta fase do desenvolvimento é preciso visualizar se o processo de separação está funcionando bem, assim para que se possa verificar o funcionamento, uma imagem de saída deve ser gerada, nessa imagem os pixels com cores próximas à cor padrão da etiqueta devem ter o seu brilho aumentado em 20% e os outros pixels mais distantes devem ter seu brilho reduzido para 30%.

Desenvolvimento

Use tudo o que você aprendeu nesta disciplina para otimizar o código fornecido. Você deve otimizar o programa todo, inclusive a parte de aumento e diminuição do brilho.

Executando o programa sem otimização

Um aluno do time forneceu o programa e te indicou a forma de compilar:

1. Compile o programa SeparaCor.c
    - gcc -masm=intel SeparaCor.c -o SeparaCor -lm
2. Execute o programa.
    - ./SeparaCor
3. Observe a imagem LapisSai.ppm para ver o funcionamento do programa e responda:

    Qual foi o tempo observado na execução[1]?

    Tempo 1: _____ s

Aplicando as técnicas de otimização

Tente usar as técnicas de otimização vista até agora, uma de cada vez, mas juntando com as otimizações anteriores a cada passo que você tiver sucesso. Use o Valgrind para identificar os *hotspots* e "ataque" primeiro os pontos de maior potencial de otimização. Sempre que uma das técnicas trouxer bons resultados, indique-as separadamente. Não esqueça de verificar sempre se a imagem gerada está correta.

1 **Todos os tempos deste trabalho devem ser medidos sem o Valgring**, mas o Valgrind pode ser útil para identificar os *hotspots* a otimizar.

Obs.: o código de processamento está repetido 120 vezes apenas para que os valores de tempos medidos fiquem mais estáveis e próximos. Baixar de 120 para 1 essa repetição não é uma otimização, pode ser feita nos testes, mas não deve ser feita nas medidas de tempo, pois no futebol de robôs a repetição acontecerá nesta taxa.

Algumas perguntas pela ordem de importância para orientar suas otimizações:

1. Existe alguma função das bibliotecas do compilador que está gastando muito tempo de execução do programa? Qual é a rotina e qual a porcentagem do tempo ela gasta? Tem como substituir a chamada dessa função para fazer de uma forma mais otimizada?
    - Analise se é possível substituir a comparação $y=\sqrt{x}$ por $y^2=x$
    - E a função pow()? Como pode ser substituída?
2. É possível desenrolar algum código?
3. Operações com inteiros podem ser uma boa ideia

Avaliando os resultados

Envie a avaliação dos resultados como descrito no arquivo “Avaliacao Dos Resultados.pdf”.

Conclusão

Este trabalho oferece a oportunidade do aluno experimentar a aplicação das técnicas de otimização de uma forma mais autônoma.