Desenvolvimento de ferramentas de teste, depuração e calibração para robô humanoide

Igor Albuquerque Silva
Instituto Tecnológico de Aeronáutica
Rua H8C, 322, CTA
12.228-462 - São José dos Campos / SP
Bolsista PIBIC - CNPq
asilvaigor@gmail.com

Celso Massaki Hirata
Instituto Tecnológico de Aeronáutica
Divisão de Ciência da Computação
Praça Marechal Eduardo Gomes, 50
12.229-900 – São José dos Campos / SP
hirata@ita.br

Marcos Ricardo Omena de Albuquerque Maximo
Instituto Tecnológico de Aeronáutica
Laboratório de Sistemas Computacinais Autônomos (LAB-SCA)
Divisão de Ciência da Computação
Praça Marechal Eduardo Gomes, 50
12.229-900 – São José dos Campos / SP
maximo.marcos@gmail.com

Resumo: *Devido à alta complexidade existente nos algoritmos de funcionamento de um robô humanoide, é fundamental possuir ferramentas de interface com o usuário para o desenvolvimento do sistema. Estas ferramentas possuem a finalidade de fornecer uma visualização dos algoritmos envolvidos de forma clara para o usuário. Após uma fase de pesquisa, foi decidido utilizar o framework rqt, uma plataforma baseada em Qt que permite o desenvolvimento de interfaces gráficas para ROS (Robot Operational System), assim possibilitando criar um sistema de telemetria para o robô.*
Com este intuito, foram desenvolvidas três ferramentas iniciais. A primeira, feita antes da pesquisa sobre o framework rqt, foi escrita utilizando apenas a plataforma Qt. Ela é uma interface que permite anotar imagens e gerar tabelas de cores utilizando as funções de treinamento de redes neurais presentes no software MATLAB. A segunda utiliza o framework rqt e é uma ferramenta de teste e calibração do algoritmo de visão computacional do robô humanoide. A última ferramenta é um controle remoto do robô que permite variar a sua velocidade para facilitar a depuração do código.
Palavras-chave: *robótica, interface gráfica, ros, rqt, qt.*

1. INTRODUÇÃO

A ITAndroids é uma equipe de alunos do ITA, supervisionada por um professor, que participa de diversas competições de robótica nacionais e internacionais. Uma das categorias em que a ITAndroids participa é a do robô humanoide, que consiste em desenvolver um time de robôs capazes de jogar futebol. Esta tarefa envolve uma série de desafios complexos que variam desde a construção do robô até a sua tomada de decisões.

Neste contexto, é fundamental a presença de algoritmos robustos para a realização das diversas ações do robô, tornando essenciais as ferramentas de testes, calibração e depuração. Equipes reconhecidas no cenário internacional, como a B-Human e a Nimbro-OP, possuem várias ferramentas com este intuito, como são apresentadas em Roffer *et al.* (2013) e em Allgeuer *et al.* (2013).

Essas ferramentas devem possibilitar o teste dos algoritmos em tempo real, criando um sistema de telemetria com o robô. Além disso, é interessante que elas também possibilitem testes sem a presença do robô, assim facilitando a depuração e calibração do código.

1.1 Ferramenta de Calibração da Tabela de Cores

Tabela de cores é uma tabela com todos os valores de um espaço de cores e um código simples para identificar aquela cor. Por exemplo, no espaço RGB o código (0, 0, 0) é da cor preta. Uma tabela de cores RGB possui todas as triplas e suas respectivas cores. Para a ITAndroids, as cores que são relevantes são aquelas contidas no jogo de futebol, portanto todas as triplas RGB são classificadas em verde, branco, preto, azul, laranja, rosa e amarelo. Uma tabela de cores é essencial para o início do algoritmo da visão computacional para o futebol de robôs. Ao receber uma imagem, o software deve classificar cada pixel em uma cor para, em seguida, agrupar os pixels de cores iguais e criar formas que podem ser classificadas em objetos.

Todavia, dependendo do campo e da iluminação em cada momento, as cores sofrem alterações de tons e, portanto, a tabela de cores deve ser modificada para maximizar o desempenho da visão do robô. Por isso, uma ferramenta simples e rápida que gere uma tabela de cores é extremamente útil para a ITAndroids.

Outro requisito da ferramenta de calibração da tabela de cores é que o usuário selecione as cores que farão parte da classificação para não limitar esta ferramenta a somente este problema.

1.2 Ferramenta para a Visão Computacional

A complexidade envolvida no problema da detecção dos objetos do campo exige um software capaz de exibir de maneira clara a saída do algoritmo, isto é, as detecções em cada imagem.

O código da visão computacional passa por diversas etapas para classificar os elementos do campo. Uma delas consiste em filtrar vários elementos detectados previamente. Estes possíveis elementos ficam armazenados dentro de vetores internos da parte da visão do código do humanóide, enquanto os objetos que passam no filtro são armazenados em estruturas chamadas "VisibleObjects", que são utilizadas em outras partes do código.

É de interesse da equipe que tanto os objetos que foram desclassificados quanto os classificados sejam visualizados na ferramenta da visão de forma clara e que suas variáveis internas também sejam disponibilizadas, para fins de depuração e calibração do algoritmo. A Figura 1 mostra uma ferramenta do time de robótica B-Human, apresentada em Roffer *et al.* (2013). Note que na imagem à direita as variáveis de cada trave estão escritas na foto, para fins de depuração. A ferramenta desejada deve conter uma opção similar à contida na Figura 1.

Outra necessidade da ferramenta é a possibilidade de alteração das variáveis da detecção dos objetos em tempo real para fins de calibração, por meio de barras que deslizam interativamente. Por meio destas barras, deve ser possível selecionar a melhor combinação de variáveis para a detecção dos objetos.

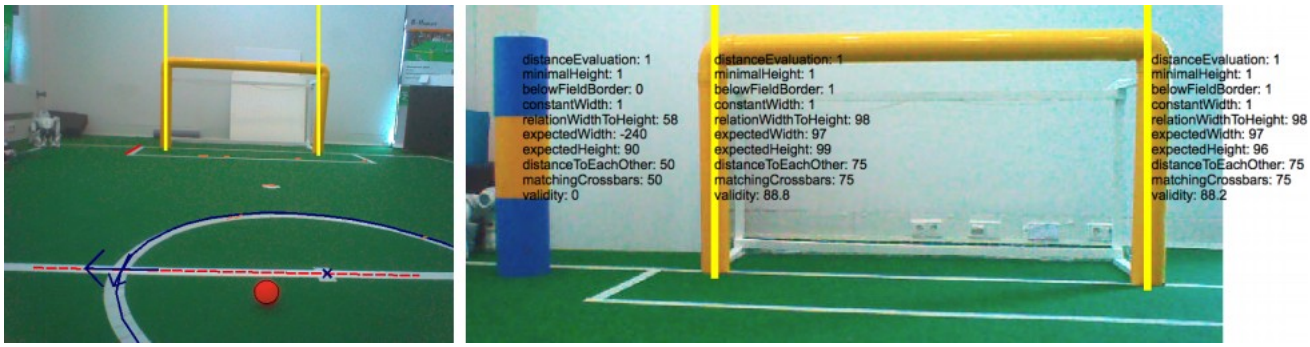

Figura 1: **Ferramenta de visualização da visão computacional do time B-Human.**

1.3 Controle Remoto

Devido às diversas situações possíveis em um jogo de futebol de robôs, é preciso uma forma de simular todas elas para o teste dos algoritmos desenvolvidos como o da visão, por exemplo. Assim, um controle remoto é fundamental para movimentar o robô pelo campo ao mesmo tempo que são testados estes algoritmos.

Além disto, um controle remoto permite visualizar facilmente os movimentos do robô em diferentes velocidades de caminhada para a análise deles.

2. RESULTADOS OBTIDOS

Todas as ferramentas foram desenvolvidas com sucesso. Com elas, um sistema de telemetria do robô foi desenvolvido que alavanca o avanço da equipe.

2.1 Ferramenta de Calibração da Tabela de Cores

Com o uso da plataforma Qt, foi desenvolvida esta ferramenta. Ela utiliza o Matlab Pattern Recognition App, presente em MATLAB Neural Network Toolbox (2016), uma ferramenta do Matlab que possibilita o treinamento de redes neurais que se mostrou extremamente rápida, demorando cerca de 5 minutos para gerar uma tabela de cores satisfatória. A ferramenta possui cinco abas:

- Captura de câmera: possibilita tirar fotos utilizando a câmera do dispositivo para servir de amostra, caso a ferramenta seja utilizada diretamente do robô.

- Anotação: possibilita pintar as imagens que são amostras para fornecer a entrada para a rede neural. Esta aba possibilita ao usuário selecionar quais cores serão utilizadas, como representado na Figura 2. Ela está ilustrada na Figura 3.
- Treinamento: possibilita o acesso ao Matlab Pattern Recognition App, que treina a rede neural.
- Teste: possibilita o teste da rede neural criada, importando uma imagem ou vídeo do dispositivo e classificando seus pixels.
- Ajuda: aba descrevendo cada função da ferramenta e como utilizá-la.

Figura 2: **Janela que possibilita a escolha das cores presentes na tabela e seus respectivos códigos numéricos.**

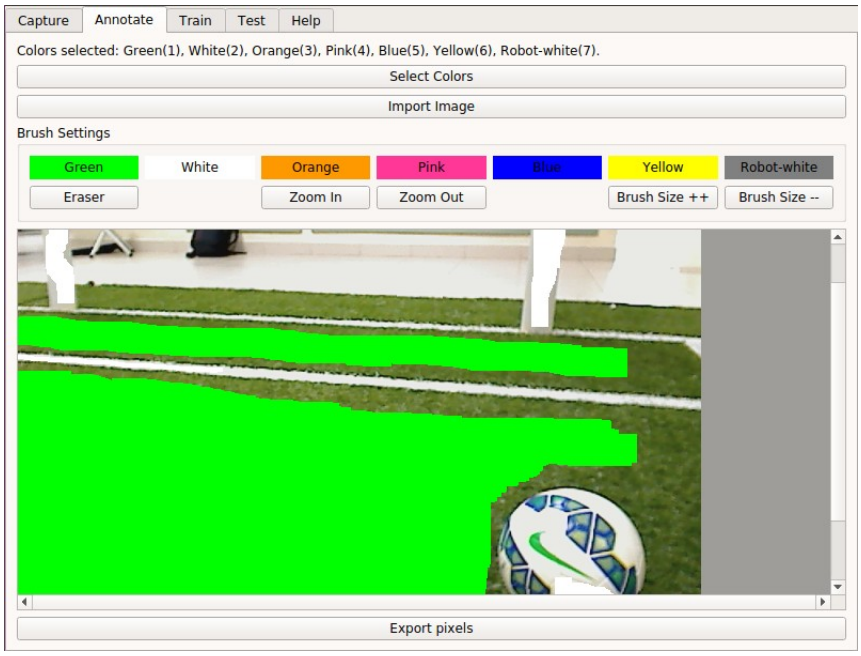

Figura 3: **Aba de anotação do programa.**

Assim, a ferramenta possibilita o treinamento de uma rede neural que classifica os valores de cores nas classes selecionadas. Para gerar a tabela de cores, basta executar o forward propagation para todos os valores de cores e armazenar os valores das classes obtidas. Notou-se que a ferramenta funciona corretamente, mas a arquitetura da rede neural ainda deve ser aprimorada para resolver o problema com perfeição. Na implementação atual a rede encontra-se desbalanceada, portanto as cores de maior ocorrência nas imagens (geralmente o verde) dominam na minimização da função de custo.

2.2 Ferramenta para a Visão Computacional

O framework utilizado para escrever a ferramenta da visão e as ferramentas seguintes foi o rqt, uma plataforma baseada em Qt desenvolvida para ROS (*Robot Operational System*), conjunto de frameworks para desenvolvimento de softwares de robótica, como descrito em Quigley *et al.* (2009). O rqt foi escolhido baseado no framework desenvolvido em Allgeuer *et al.* (2013), devido à sua facilidade de desenvolvimento e sua integração com a robótica, podendo ser utilizado para testes em tempo real utilizando o robô ou com dados coletados reproduzidos posteriormente.

Utilizando o rqt e a biblioteca OpenCV, introduzida em Bradski (2000), foi possível desenvolver uma interface gráfica que atende todos os requisitos da ferramenta para a visão. Esta ferramenta se comunica diretamente com o robô por meio do sistema ROS, sendo possível visualizar a saída do algoritmo da visão em tempo real, além de calibrá-lo. Assim, o usuário pode ver cada objeto detectado e também os não detectados. A Figura 3 mostra a interface da ferramenta.

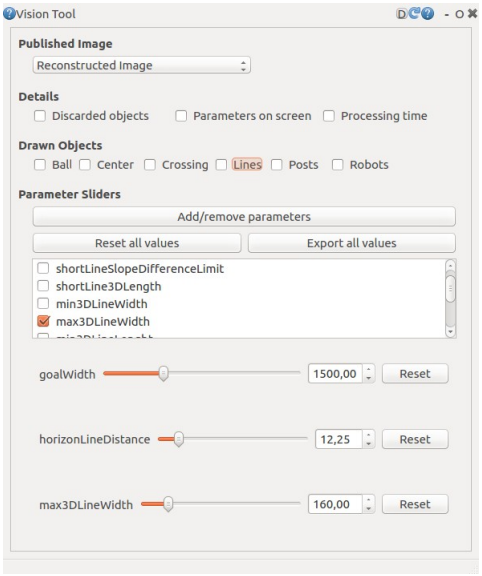

Figura 4: **Interface gráfica da ferramenta desenvolvida.**

A interface foi separada em quatro blocos. O primeiro, "published image" (imagem publicada), permite selecionar qual imagem será visualizada, isto é, a imagem original da câmera, a imagem segmentada em cores ou a imagem com os objetos detectados indicados. O segundo bloco, "details" (detalhes), permite algumas opções extras, isto é, visualizar os objetos descartados, visualizar os parâmetros internos do código para cada objeto e visualizar o tempo de processamento total do algoritmo da visão computacional. O terceiro bloco, "drawn objects" (objetos desenhados), permite selecionar quais objetos serão visualizados. O último bloco, "parameter sliders" (barras de parâmetros), permite calibrar os parâmetros do código por meio de barras iterativas.

Para gerar a imagem o sistema de publicação e subscrição do ROS foi utilizado. Este sistema permite que o robô publique uma imagem pela rede para o computador do usuário visualizar esta imagem. Além disso, o usuário publica quais informações ele deseja na imagem a partir da ferramenta desenvolvida, e o robô altera a imagem de acordo com estas informações. O maior benefício deste sistema é a possibilidade de salvar estas imagens em um vídeo e poder publicá-las posteriormente, assim testando o algoritmo da visão sem a necessidade do robô.

A Figura 5 mostra a ferramenta em funcionamento no framework rqt. Nele, é possível unir a ferramenta desenvolvida com um visualizador de imagens na mesma tela. Na figura, a ferramenta pode ser encontrada na parte esquerda da imagem, abaixo do logger padrão do ROS. Na parte direita da imagem se encontra um visualizador, que mostra a saída do código da visão computacional. Neste caso a bola, duas linhas e uma intersecção de linhas foram encontrados.

Um dos requisitos da ferramenta implementado com sucesso foi a visualização dos parâmetros internos do código para cada objeto. A Figura 6 ilustra este requisito em funcionamento. Nela os parâmetros internos das traves estão sendo visualizados em comparação com uma possível trave descartada em um filtro do código da visão computacional, que aparece em uma mancha branca da bola.

Os maiores desafios encontrado nesta etapa foram a compreensão do código da visão computacional da ITAndroids, o qual foi adaptado da liberação do código de 2012 da equipe Austin Villa, presente em Barrett *et al.* (2013), e a implementação do sistema de publicação e subscrição de imagens pelo ROS. No primeiro foi necessário compreender as estruturas de dados nas quais os elementos da visão são armazenados para ser possível desenha-los na tela. No segundo foi necessário pensar na estrutura de códigos para ser possível ter uma comunicação entre o robô e o computador.

2.3 Controle Remoto

A ferramenta de controle remoto do robô também foi desenvolvida utilizando o framework rqt e o sistema de publicação e subscrição do ROS. Essa interface permite controlar um robô humanóide, como ilustrado na Figura 7. Note que este tipo de robô é omnidirecional, isto é, ele consegue se mover para qualquer direção no plano. Por isso, a ferramenta permite comandar velocidades de translação e de rotação para controlar o robô. Além disso, é possível mover a cabeça do robô para facilitar o teste de algoritmos que envolvem a câmera.

A Figura 8 ilustra a interface gráfica desenvolvida.

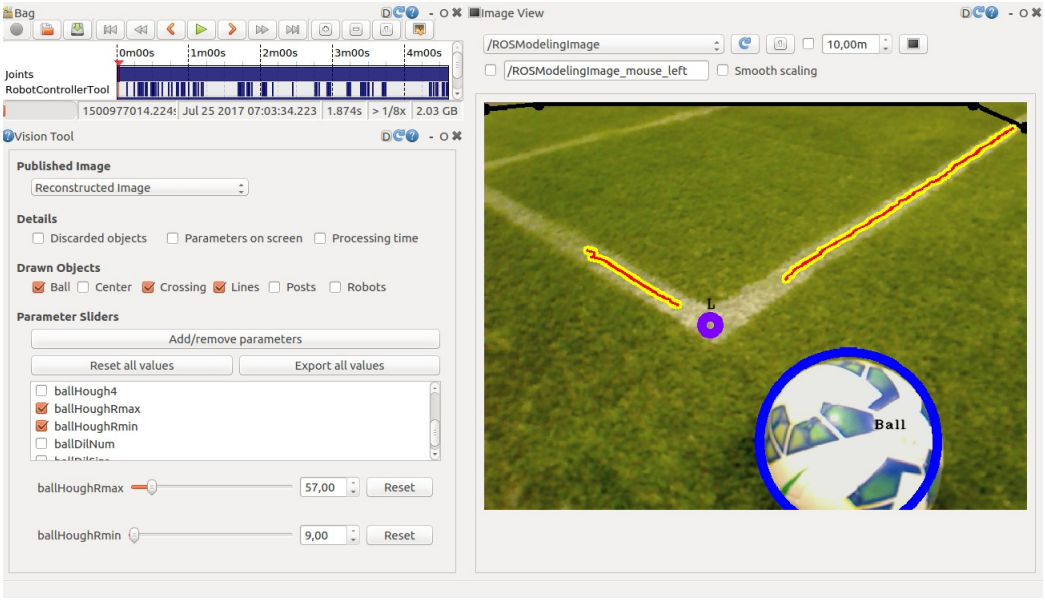

Figura 5: **Framework rqt em funcionamento.**

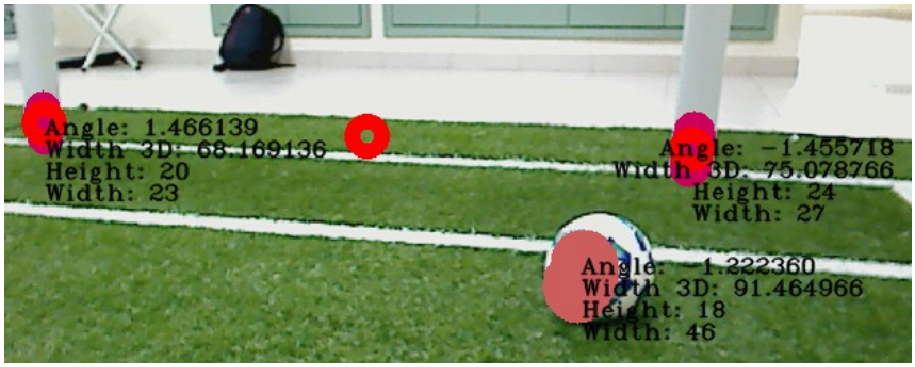

Figura 6: **A ferramenta permite visualizar os parâmetros internos do código para cada objeto.**

Nessa ferramenta, o computador do usuário publica a velocidade ou ação a ser adotada pelo robô e este executa a tarefa ao receber a mensagem. A direção da velocidade, assim como a movimentação da cabeça, são controladas pelo teclado do computador, como está ilustrado no diagrama presente na Figura 8. As ações a serem executadas, como chute e parada, também podem ser controladas no teclado ou por meio de botões. O usuário também controla a velocidade dos movimentos por meio de barras presentes na ferramenta.

3. CONCLUSÕES

O framework rqt é excelente para a criação rápida de ferramentas de interface com o usuário para a robótica. Estas ferramentas iniciais criadas agora serão modelo para a criação de mais ferramentas no futuro.

A ferramenta de calibração da tabela de cores se mostrou significativamente mais rápida do que aquela possuída anteriormente, além de ter uma interface gráfica mais completa. Com o auxílio desta ferramenta, é possível criar diversas tabelas de cores para cada ambiente sem muitas dificuldades. Melhorias ainda precisam ser implementadas no sentido de aprimorar o algoritmo de geração da tabela de cor, pois grandes mudanças na iluminação exigem diferentes tabelas de cores no algoritmo atual.

A ferramenta para a visão computacional já se mostrou extremamente útil para a equipe. Com suas funcionalidades, será possível maximizar o potencial do algoritmo de visão computacional da ITAndroids. Outras formas de visualização do algoritmo de visão computacional precisam ser implementados para deixar a ferramenta completa, como etapas intermediárias da detecção, por exemplo.

O controle remoto agora permite controlar o robô com facilidade, permitindo o teste de outros algoritmos da equipe, como o da visão computacional e o da localização no campo.

Outras ferramentas ainda serão desenvolvidas para completar o sistema de telemetria do robô. Um exemplo de ferramenta deve permitir a visualização da posição percebida pelo robô no campo. Assim, será possível testar o algoritmo

Figura 7: **Robô humanoide DARwIn-OP2 desenvolvido por Ha *et al.* (2013). A equipe ITAndroids possui uma unidade deste robô.**

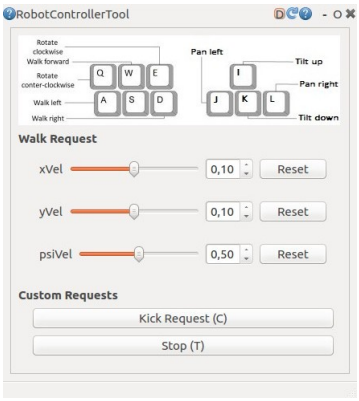

Figura 8: **Interface do controle remoto desenvolvido.**

de localização no campo de forma mais fácil. Com a arquitetura no rqt desenvolvida, é essencial o desenvolvimento de outras ferramentas para a progressão da equipe.

4. AGRADECIMENTOS

Agradeço ao CNPq pela oportunidade e pelo financiamento do projeto.

Agradeço ao professor coorientador Marcos Máximo pelo constante apoio durante toda a execução do projeto, mostrando alternativas e apresentando ideias novas que foram fundamentais para o produto final.

Agradeço a toda a equipe da ITAndroids pelo suporte e acompanhamento do projeto, em especial ao Samuel Cerqueira pela contribuição às ferramentas desenvolvidas.

Agradeço ao professor orientador Celso Hirata pela oportunidade da realização do projeto.

5. REFERÊNCIAS

Allgeuer, P. *et al.*, 2013. "A ros-based software framework for the nimbro-op humanoid open platform".

Barrett, S., Genter, K., He, Y., Hester, T., Khandelwal, P., Menashe, J. and Stone, P., 2013. "The 2012 UT Austin Villa code release". In *RoboCup-2013: Robot Soccer World Cup XVII*, Springer Verlag.

Bradski, G., 2000. "Open source computer vision library".

Ha, I., Tamura, Y. and Asama, H., 2013. "Development of open platform humanoid robot darwin-op".

MATLAB Neural Network Toolbox (2016), 2016. "Matlab neural network toolbox". The MathWorks, Natick, MA, USA.

Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A.Y., 2009. "Ros: an open-source robot operating system". In *ICRA Workshop on Open Source Software*.

Roffer, T. *et al.*, 2013. "B-human team description for robocup 2013".

6. RESPONSABILIDADE AUTORAIS

Os autores são os únicos responsáveis pelo conteúdo deste trabalho.