Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

CAMILA DUARTE RODRIGUES LOPES

DESENVOLVIMENTO DE UM SISTEMA DE DETECÇÃO E CLASSIFICAÇÃO POR CORES PRIMÁRIAS UTILIZANDO A BIBLIOTECA DE VISÃO COMPUTACIONAL *OPENCV*

Campina Grande, Paraíba
Dezembro de 2011

CAMILA DUARTE RODRIGUES LOPES

DESENVOLVIMENTO DE UM SISTEMA DE DETECÇÃO E CLASSIFICAÇÃO POR CORES PRIMÁRIAS UTILIZANDO A BIBLIOTECA DE VISÃO COMPUTACIONAL *OPENCV*

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento de Sinais

Orientador:
Professor Edmar Candeia Gurjão, Dr. Sc.

Campina Grande, Paraíba
Dezembro de 2011

CAMILA DUARTE RODRIGUES LOPES

DESENVOLVIMENTO DE UM SISTEMA DE DETECÇÃO E CLASSIFICAÇÃO POR CORES PRIMÁRIAS UTILIZANDO A BIBLIOTECA DE VISÃO COMPUTACIONAL *OPENCV*

Relatório de Estágio Supervisionado submetido à
Unidade Acadêmica de Engenharia Elétrica da
Universidade Federal de Campina Grande como parte dos
requisitos necessários para a obtenção do grau de
Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento de Sinais

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professor Edmar Candeia Gurjão, Dr. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Dedico este trabalho aos meus pais e aos melhores amigos que poderia ter: meus irmãos de sangue e os irmãos que a vida me deu.

AGRADECIMENTOS

Agradeço em primeiro lugar a todos na minha família, que sempre me incentivaram em todos os momentos, não medindo esforços para que sempre pudesse me dedicar exclusivamente aos estudos.

Deixo meu registro de gratidão a todos que de maneira direta ou indireta contribuíram para o êxito na minha trajetória durante a graduação e nessa reta final de curso. Expresso minha admiração e agradecimento aos professores e funcionários do Departamento de Engenharia Elétrica, com quem tanto aprendi e que me transmitiram conhecimentos essenciais para minha formação não só no âmbito acadêmico, mas também para outras áreas da minha vida. Agradeço em especial ao Prof. Dr. Edmar Candeia Gurjão, pela orientação durante o Estágio e pelo esforço no sentido de me auxiliar em todas as atividades, e aos colegas e funcionários do Laboratório de Automação e Processamento de Sinais – LAPS.

A todos os meus bons amigos, sempre presentes nas minhas melhores lembranças e com quem tanto aprendi sobre a vida.

Divido o mérito da minha graduação e deste trabalho com todas estas pessoas, desejando ter a sorte de contar sempre com todo esse apóio durante as etapas da minha vida que ainda estão por vir.

"É o peso, não a quantidade de experiências, que tem de ser observado."

Isaac Newton.

RESUMO

Este trabalho consiste no relatório final elaborado ao término das atividades desenvolvidas durante o Estágio Supervisionado realizado no Laboratório de Automação e Processamento de Sinais – LAPS localizado no Departamento de Engenharia Elétrica da Universidade Federal de Campina Grande, durante o segundo semestre de 2011. O foco principal do trabalho foi o desenvolvimento de um sistema de classificação de objetos segundo as cores primárias (vermelho, verde e azul). O trabalho realizado teve como plano de fundo o atual contexto de separação de garrafas PET em seu processo de reciclagem. A ideia do projeto é desenvolver um sistema de baixo custo, que use apenas um computador com uma *webcam* conectada. Deste modo, trabalhos posteriores poderão aperfeiçoar o sistema, construindo o aparato físico, por exemplo. Verificou-se a eficácia das atividades do Estágio no sentido de consolidar os conhecimentos da graduação, bem como no desenvolvimento da visão de um problema do mercado e de uma possível solução para o mesmo.

Palavras-chave: Sistemas de classificação e seleção; espaço de cores RGB, Visão Computacional, OpenCV.

ABSTRACT

This work is a final report written at the end of the activities developed during the supervised stage accomplished at the *Laboratório de Automação e Processamento de Sinais* – LAPS located at UFCG, during the second half of 2011. The work was developed inside the context of separation of PET bottles in their recycling process. The main focus of this study was the development of a basic system of detecting and sorting objects according to the primary colors (red, green and blue). The project main idea was to implement a low cost system, which uses only a computer and a webcam connected to it. Thus, further work may improve the system, building the physical apparatus, for example. The effectiveness of the activities during the stage was reached to consolidate the knowledge of undergraduate course as well as the development of the vision of a market problem and a possible solution to it.

Keywords: Sorting systems; RGB colors; Computer Vision; OpenCV.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Os processos industriais são sempre alvos de estudos para realização de aperfeiçoamento e aumento da eficiência das linhas de produção. Algumas áreas específicas da indústria costumam precisar de sistemas de seletores e classificadores de artefatos baseados em uma ou mais características do próprio objeto a ser classificado. Essas características costumam ser físicas, como cor, formato e tamanho, por exemplo.

A indústria de alimentos é pioneira no uso destes sistemas. Em (RUOYU, ZA e YINLAN, 2010), temos o exemplo de um sistema de seleção de tomates pela sua cor.

Além da seleção de alimentos, outras aplicações de sistemas de classificação se destacam pela sua utilidade. O crescente número de empresas que utilizam a reciclagem de matérias é notório, bem como é de conhecimento geral o volume de resíduos que são resultados do elevado consumo de certos produtos pela sociedade em geral. As garrafas plásticas do tipo PET são ótimos exemplos desses materiais que, se mal armazenados e deixados para decomposição na natureza, podem trazer graves consequências para o meio ambiente (Estima-se que o tempo de decomposição das garrafas PET esteja por volta de 400 anos) (ABIPET, 2011).

Por esse e outros motivos, a indústria da reciclagem de garrafas PET procura cada vez mais evoluir no beneficiamento destes produtos para atender a demanda de vários setores da economia que utilizam produtos fabricados a partir do PET reciclado, como o setor da construção civil, onde este material é utilizado para fabricação de telhas, tubos e conexões.

Porém, para que o PET reciclado possa ser utilizado, é importante que ele apresente certo grau de pureza, que varia de acordo com a aplicação desejada. Esse grau de pureza pode ser obtido fazendo-se uma triagem no início do processo de reciclagem. O que se observa comumente é essa triagem sendo feita manualmente, fazendo com que o processo perca em velocidade e esteja sujeito a erros humanos.

Tendo em vista os fatos citados, percebe-se a importância de um sistema de classificação integrado ao processo de reciclagem do PET.

O presente relatório é referente ao Estágio Supervisionado realizado no Laboratório de Automação e Processamento de Sinais da Universidade Federal de

Campina Grande, realizado no segundo semestre do ano de 2011 com duração de 360 horas.

O objetivo do trabalho foi a implementação de um sistema de detecção e classificação por cores primárias que pudesse ser posteriormente aperfeiçoado e embarcado para o uso na indústria de reciclagem de garrafas PET. O sistema foi implementado utilizando processamento digital de imagens em linguagem C/C++ e a biblioteca de Visão Computacional *OpenCV* (OpenCV Wiki, 2011).

1.1 LABORATÓRIO DE AUTOMAÇÃO E PROCESSAMENTOS DE SINAIS – LAPS

Como já foi citado, o Estágio descrito neste relatório foi realizado no Laboratório de Automação de Processamento de Sinais – LAPS, localizado no bloco CJ do Departamento de Engenharia Elétrica da Universidade Federal de Campina Grande. Estando sob a coordenação do Prof. Dr. Edmar Candeia Gurjão.

O LAPS tem como principais linhas de pesquisa o Rádio Definido por *Software* - RDS (com destaque para o desenvolvimento de aplicações para a plataforma USRP com a ferramenta GNU Radio), Processamento Digital de Sinais (principalmente Áudio) e a Amostragem Compressiva. (LAPS, 2011)

O laboratório conta com computadores funcionando com sistemas operacionais *Linux Ubuntu* e *Windows*, plataformas para RDS USRP, DSPs da *Texas Instruments* e instrumentos de medição, como osciloscópios e analisador de espectro.

Além das pesquisas de iniciação científica e tecnológica, trabalhos de conclusão de curso e estágios, o laboratório é ainda utilizado na graduação para aulas práticas de Processamento Digital de Sinais e Redes de Computadores.

O trabalho realizado também teve como objetivo introduzir o tema de processamento digital de imagens, aplicado a algoritmos de classificação ao âmbito de pesquisa do LAPS.

2 FUNDAMENTAÇÃO TEÓRICA

Para a fundamentação e compreensão das atividades desenvolvidas, é dada uma visão geral das ferramentas de programação utilizadas (Ambiente de programação *Eclipse* e biblioteca *OpenCV*) e sobre as técnicas de separação por cores.

2.1 TÉCNICAS DE SEPARAÇÃO E CLASSIFICAÇÃO

Inúmeras são as formas de classificar e separar objetos de acordo com suas características físicas. Sistemas baseados em processamento de imagens, medição de grandezas físicas (peso, por exemplo), análise óptica, técnicas de magnetização, dentre outros, são alguns dos exemplos de ferramentas utilizadas nessa tarefa.

Observando a literatura acerca do tema, observa-se que a principal forma de separação se dá pela análise da imagem do objeto a ser classificado. Os parâmetros a serem extraídos e as formas de tratamento desses dados é que variam bastante de acordo com a aplicação. Por exemplo, em (SCAVINO, WAHAB, *et al.*, 2009) o sistema utiliza pré-processamento para identificação do formato da garrafa através do reconhecimento de contornos imagem. Já em (RUOYU, ZA e YINLAN, 2010), é feita a identificação de cores para realizar a separação de tomates em verdes ou maduros.

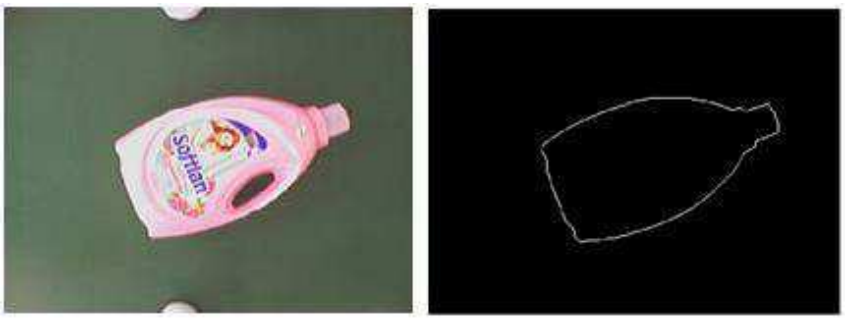

Figura 1. Processamento de Imagem para detecção do formato de garrafas plásicas. (SCAVINO, WAHAB, *et al.*, 2009)

2.2 SEPARAÇÃO POR CORES

Para a separação e classificação de garrafas PET, a identificação por processamento de imagens pode ser utilizada mais de uma vez para tornar a seleção

cada vez mais eficiente ao final do processo. A separação feita pela identificação de cores poderia ser tomada como um primeiro passo no processo.

Basicamente, um algoritmo de separação por cores irá processar a imagem em cores[1] adquirida e extrair os parâmetros de intensidade para cada camada do pixel. Por camada, entende-se os canais R (do inglês *Red* – vermelho), G (do inglês *Green* – verde) e Blue (do inglês *Blue* – azul). Essas cores são as chamadas cores primárias e no espaço de cores RGB, qualquer outra cor pode ser definida através de combinações entre essas três cores, atribuindo-se pesos a cada componente.

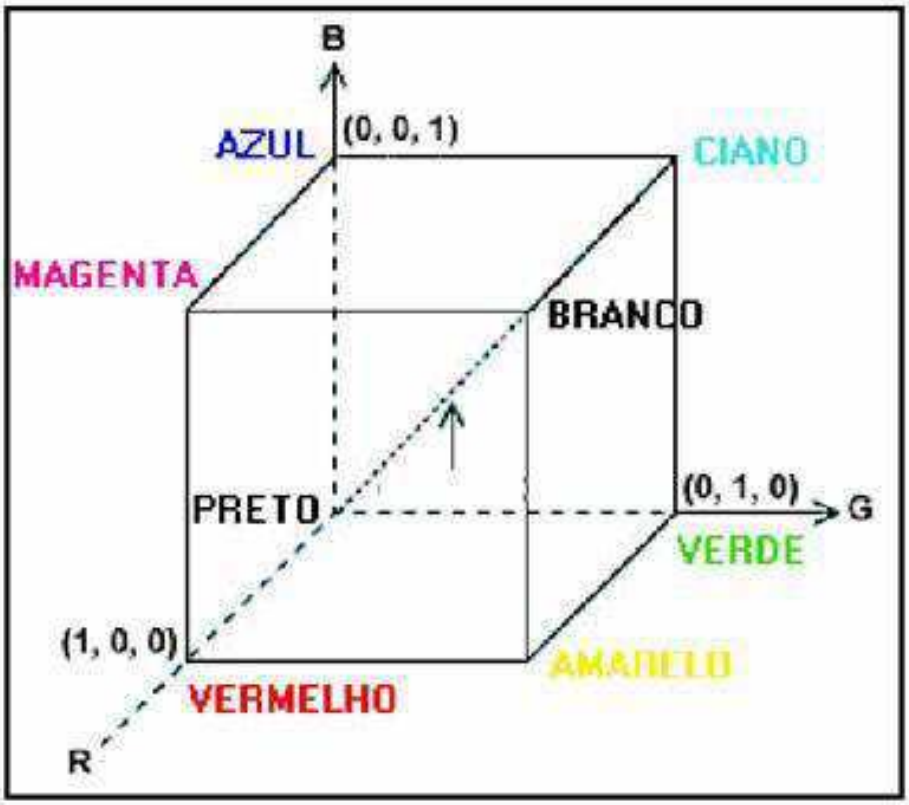

Figura 2. Representação do Espaço de cores RGB. (FONTE: http://www6.ufrgs.br/engcart/PDASR/formcor.html)

Podemos concluir que, reconhecendo a cor predominante do objeto que entra no sistema, ao utilizar a técnica de separação por cores o sistema poderá tomar decisões acerca do destino do material

[1] No presente trabalho, a referência a imagem em cores irá sempre denotar o espaço RGB (Red – Green – Blue), porém nada impede o uso de outros espaços como o HSV (Hue – Saturation – Value), já que a conversão entre estes espaços são bem definidas.

2.2.1 ALGORITMOS

Existem outros tipos de algoritmos de separação por cor que utilizam processamento mais robusto do que apenas identificação de intensidade nas três camadas de cores. Alguns requerem treinamento dos sistemas a partir de parâmetros conhecidos, utilizando algumas técnicas bem conhecidas, como redes neurais (SCAVINO, WAHAB, *et al.*, 2009).

Como o objetivo deste trabalho era apenas criar um protótipo para iniciar as pesquisas na área e introduzir o projeto às pesquisas do LAPS, optou-se por seguir uma linha mais prática onde o algoritmo fosse compreendido mais facilmente, inclusive deixando o sistema mais flexível e maleável a possíveis melhoramentos em futuros trabalhos.

2.3 BIBLIOTECA DE FUNÇÕES – *OPENCV*

A *OpenCV* (*Open Source Computer Vision Library*) é uma biblioteca destinada para aplicações na área de Visão Computacional cujo o início do desenvolvimento iniciou-se por volta do ano 2000 pela *Intel*®. Esta biblioteca é destinada ao desenvolvimento de aplicativos onde é necessária a aquisição de imagens (estáticas ou em movimento) para processamento digital.

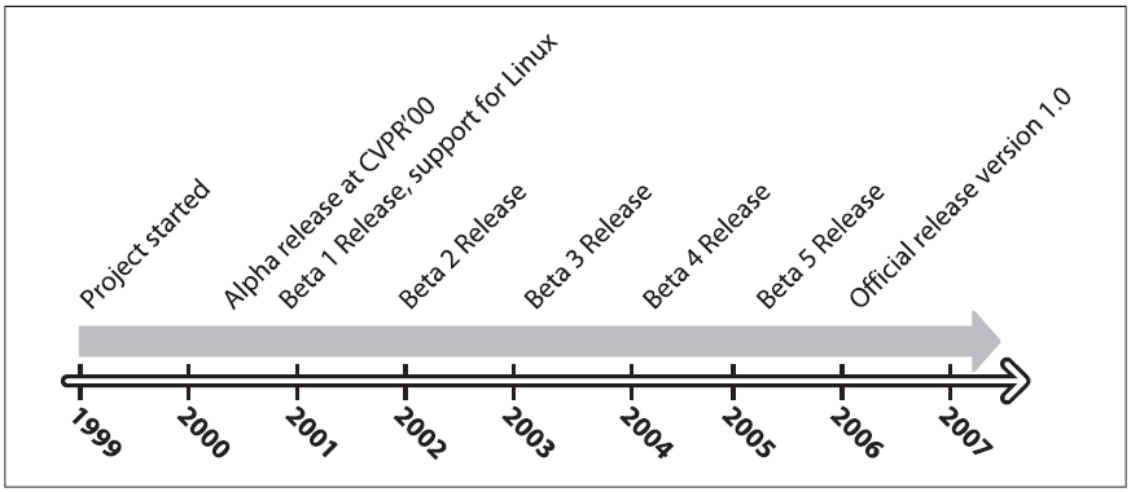

Figura 3. Linha do tempo da *OpenCV* até o ano de 2007. Atualmente, a versão mais recente é a 2.3.1, lançada em Agosto de 2011. (BRADSKI e KAEHLER, 2008)

A biblioteca é escrita em C e C++ e foi desenvolvida para rodar nos sistemas operacionais *Linux*, *Windows* e *Mac OS X*. As interfaces de programação podem ser desenvolvidas em C/C++, Python e outras linguagens. A *OpenCV* foi desenvolvida prezando a eficiência computacional tornando-se atrativa para aplicações em tempo real (BRADSKI e KAEHLER, 2008).

Um dos motivos do sucesso dessa biblioteca é a facilidade de implementação de aplicações de Visão Computacional. A biblioteca da *OpenCV* contém mais de 500 funções dos mais variados tipos e que podem ser úteis numa gama bastante variada de aplicações.

Tendo em vista as facilidades oferecidas pela *OpenCV* e pelo vasto material de documentação existente sobre a mesma, ela foi a biblioteca escolhida para realizar o tratamento necessário às imagens adquiridas pelo sistema desenvolvido.

2.4 PLATAFORMA DE DESENVOLVIMENTO – ECLIPSE

O Eclipse é uma plataforma destinada à integração de ferramentas de desenvolvimento na área de programação. É uma iniciativa *Open Source* e possui uma arquitetura extensível baseada no uso e desenvolvimento de *plugins.* O projeto Eclipse prevê a neutralidade quanto à linguagem utilizada, possuindo suporte a boa parte delas. Citando as mais conhecidas: Java, C/C++, HTML e Python.

O Projeto Eclipse foi originalmente criado pela *IBM*® em Novembro de 2001 e apoiado por um consórcio de fornecedores de *softwares*. A *Eclipse Foundation* foi criada em Janeiro de 2004 como uma corporação sem fins lucrativos independente para agir como uma espécie de administradora do projeto.

Durante as atividades do estágio, esta foi a plataforma escolhida para o desenvolvimento das atividades especificadas a seguir.

3 ATIVIDADE DESENVOLVIDA

Podemos relacionar as atividades desenvolvidas durante o Estágio Supervisionado na seguinte ordem:

- Estudo de sistemas de separação e classificação;
- Estabelecimento das diretrizes do projeto a ser desenvolvido;
- Leitura do manual e contato com uma máquina comercial separadora para reconhecimento de um sistema real[2];
- Escolha dos meios para efetivação do projeto;
- Desenvolvimento do programa e testes básicos do sistema.

A duas primeiras etapas já foram explicitadas até agora por meio da escrita da Fundamentação Teórica, onde foram expostos os principais pontos definidos como relevantes durante o Estágio.

A atividade de familiarização com uma máquina comercial foi realizada no intuito de se ter contato e entender o funcionamento de um sistema real de separação. A máquina estudada é destinada à separação de grãos pela sua cor na indústria alimentícia. Mais especificamente, a unidade analisada foi utilizada durante alguns anos para separação de grãos de milho pela São Braz S/A Indústria e Comércio de Alimentos, localizada na cidade de João Pessoa – PB. A máquina foi uma doação da São Braz ao Departamento e, primeiramente, cogitou-se a possibilidade da adaptação da mesma para trabalhar com a separação do PET moído. Porém, devido ao alto grau de deterioração de algumas partes da mesma, ela foi utilizada só para estudos.

Graças a esse contato direto com um sistema real, foi possível observar a complexidade desses processos, bem como o alto custo para adquirir uma máquina do tipo. Tal constatação só corrobora a iniciativa de pesquisas na área, que apesar de já apresentar resultados sólidos e produtos no mercado, ainda representa uma tecnologia muito cara e de difícil acesso. Deste modo, o desenvolvimento de um sistema que atue como protótipo para desenvolvimento de aplicações mais baratas e que utilizem materiais de fácil acesso para todos torna-se uma idéia altamente atrativa.

[2] A máquina estudada foi a 8400 Sorter, do fabricante Sortex.

3.1 MATERIAIS

Nesta seção, será apresentada a forma de utilização dos materiais necessários ao desenvolvimento das atividades.

A escolha dos materiais foi embasada na premissa da implementação de um sistema de baixo custo e contando com os materiais disponíveis no LAPS.

Basicamente, o projeto deveria contar com uma ferramenta de aquisição de imagens e um computador que recebesse essa informação e realizasse o processamento necessário.

Inicialmente, o projeto foi totalmente desenvolvido utilizando um notebook e sua própria câmera (*webcam*) integrada. Posteriormente, devido a uma melhor configuração física da aplicação, a aplicação foi rodada em um computador pessoal com *webcam* separada, conectada via USB.

As configurações do notebook utilizado são: Processador Intel® Petium® Dual Core @1.87GHz e Memória RAM 4 GB. O sistema operacional é o Windows Vista 32 bits.

3.1.1 INSTALAÇÃO DO ECLIPSE

O primeiro passo para o desenvolvimento é o *download* e instalação da plataforma Eclipse. O *download* do arquivo de instalação pode ser realizado diretamente no site do projeto (http://www.eclipse.org/downloads/). A versão baixada foi a 3.7.1 do Eclipse Classic. Após a instalação, é necessário realizar o *download* do *plugin* para as linguagens C/C++, já que a interface da aplicação foi escrita nestas linguagens. Essa tarefa pode ser feita diretamente no programa. Basta seguir o seguinte caminho na janela principal do Eclipse: *Help ->Install New Software*.

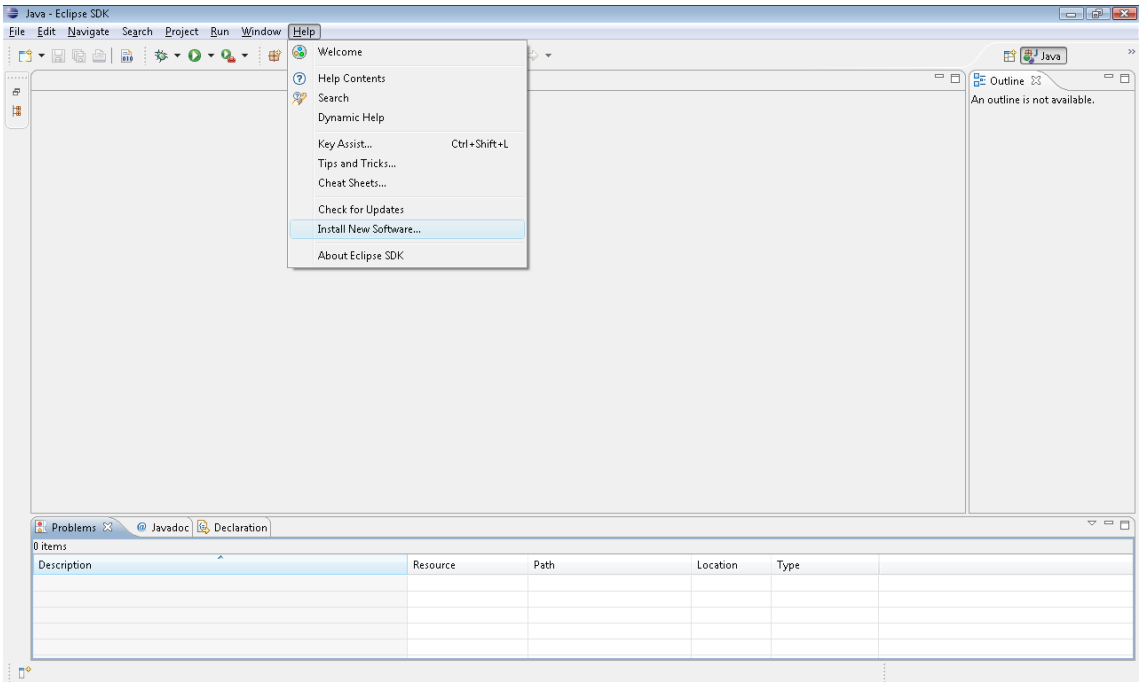

Figura 4. Tela do Eclipse indicando onde poderá ser feito o *download* de novos *plugins*.

Após a escolha, a seguinte janela será aberta:

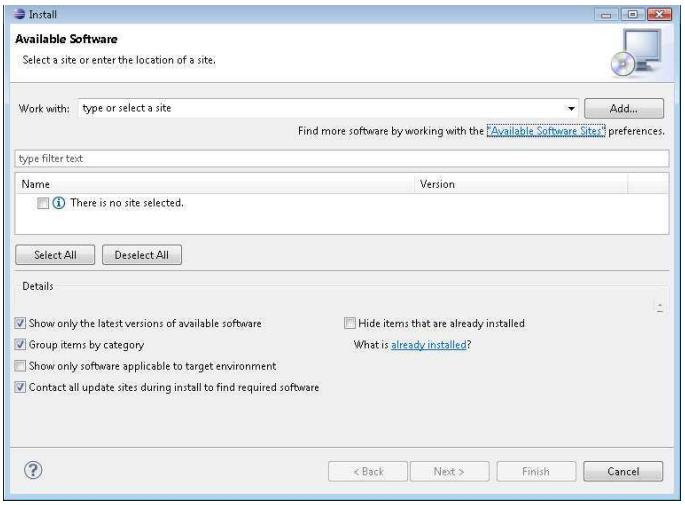

Figura 5. Janela indicando os *softwares* disponíveis.

O usuário deverá então clicar em *"Available Software Sites"* para ver a relação de *plugins* com seus respectivos sites de fonte. O *plugin* desejado para aplicações em C/C++ é o CDT. A versão baixada foi o CDT 8.0.1 for Eclipse Indigo.

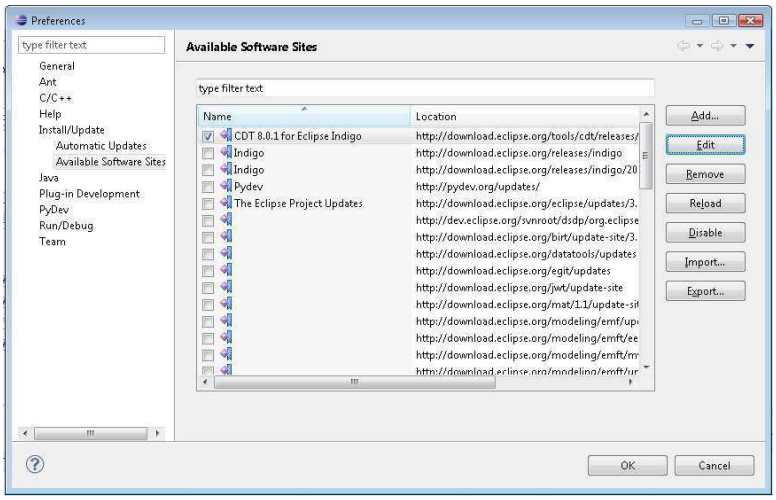

Figura 6. Janela com a lista dos sites fonte e seus respectivos *plugins*.

Caso o usuário não localize este *plugin* na lista apresentada, basta clicar em "*Add...*" (vide Figura 1Figura 6) e digitar as seguintes informações nos campos *Name* e *Location* (vide Figura 7):

- Name: CDT 8.0.1 for Eclipse Indigo
- Location: http://download.eclipse.org/tools/cdt/releases/indigo.

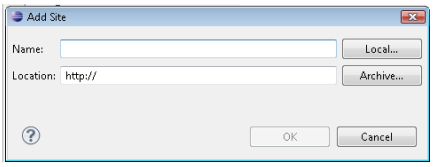

Figura 7. Opção de acrescentar um *plugin* e seu respectivo local caso a ferramente desejada não esteja relacionada.

Confirmada a escolha, a instalação prosseguirá e o programa irá solicitar o seu reinício para que as aplicações sejam validadas. Após isso, basta seguir o seguinte caminho: *Window -> Open Perspective ->Other...*. E selecionar a opção C/C++.

Após a instalação do Eclipse CDT, é necessário também realizar a instalação de compiladores C/C++ para o *Windows*. Estes compiladores serão chamados pelo Eclipse quando necessário.

Recomenda-se a instalação dos pacotes MinGW/Msys. Os tutoriais seguidos podem ser encontrados nas seguintes páginas da internet:

http://www.franciscosouza.com.br/2009/04/07/instalando-um-compilador-cc-no windows/

http://tommy.chheng.com/2009/05/12/opencv-with-eclipse-on-windows/

Páginas visitada em 18 de Novembro de 2011.

3.1.2 INSTALAÇÃO DA *OPENCV*

Para a instalação da biblioteca *OpenCV* também é possível encontrar diversos tutoriais disponíveis na internet. Porém, o usuário deve ter cuidado na hora da instalação, já que essa tarefa inclui mudanças nas variáveis do sistema do próprio *Windows*.

Por esses motivos, recomenda-se ao usuário baixar o arquivo executável de instalação no seguinte endereço:

http://sourceforge.net/projects/opencvlibrary/files/opencv-win/1.0/OpenCV_1.0.exe/download

Visitado no dia 18 de Novembro de 2011.

Após finalizada a instalação, o usuário deverá prosseguir da seguinte maneira: Abrir o Painel de Controle do *Windows* e selecionar o ícone de Sistema.

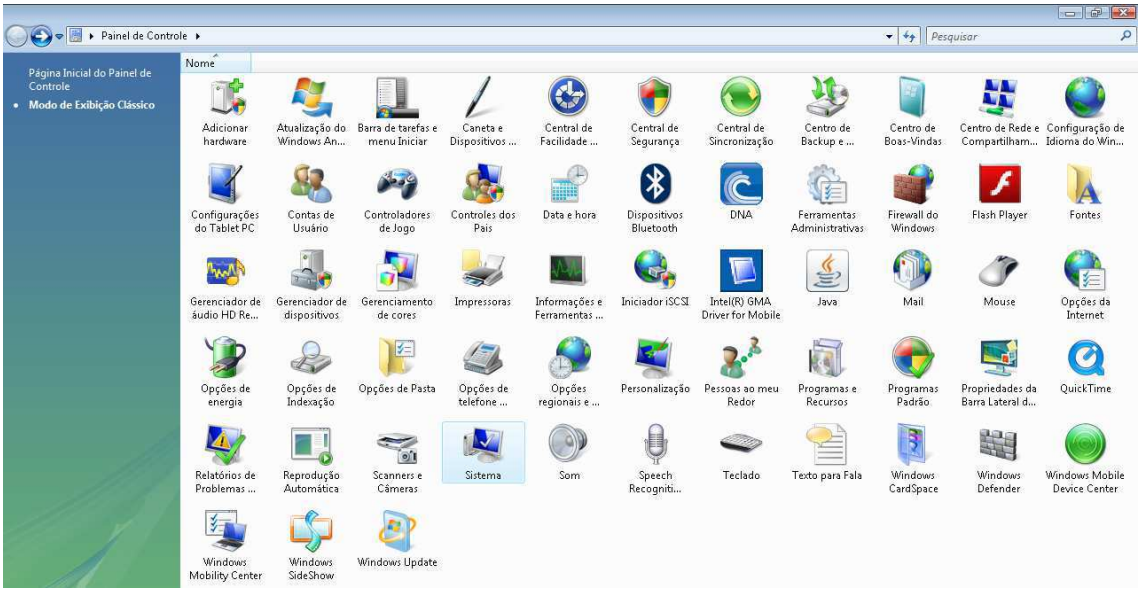

Figura 8. Painel de controle e seleção da opção “Sistema”.

O usuário terá então acesso a uma tela com informações gerais sobre o sistema. Deverá ser selecionada a opção “Configurações avançadas do sistema”, localizada no canto superior esquerdo da tela.

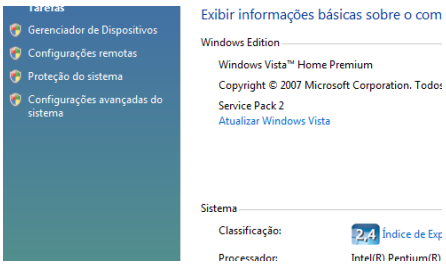

Figura 9. Caminho para realizar a mudança das variáveis do sistema.

O próximo passo é adicionar a seguinte identificação às variáveis do sistema: “C:\Program Files\OpenCV\”. Essa alteração deve ser feita selecionando “Variáveis do Ambiente...”, depois em “Variáveis do sistema” selecionar “Path” e clicar em “Editar...”. As janelas são mostradas abaixo:

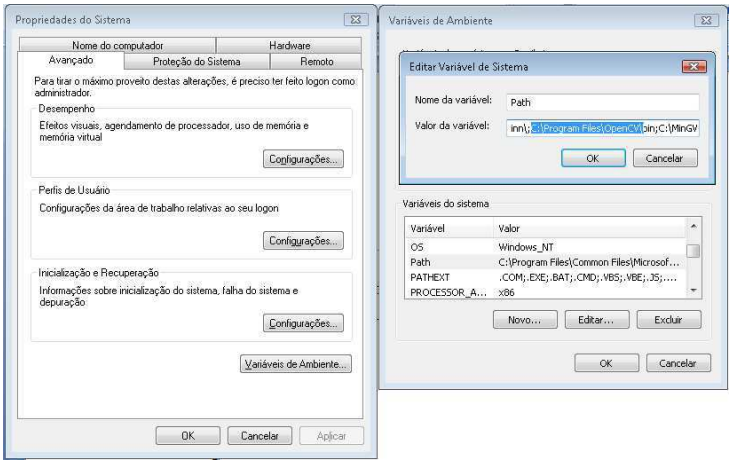

Figura 10. Telas onde é possível editar as variáveis do sistema.

Completadas todas as etapas relacionadas acima, o usuário deverá ser capaz de desenvolver um projeto em C/C++ no Eclipse utilizando as funções da *OpenCV* realizando os seguintes passos: criação de um projeto C/C++ no Eclipse (*File -> New-> C++ Project*); clicar com o botão direito na pasta do projeto localizada na aba *Project Explorer* e selecionar *Properties*. Ao chegar até esse ponto, as modificações devem ser feitas da seguinte o caminho *C/C++ Build->Settings->GCC C++ Compiler*. Os diretórios a serem adicionados são:

- C:\Program Files\OpenCv\cv\include
- C:\Program Files\OpenCv\cxcore\include
- C:\Program Files\OpenCv\otherlibs\highgui
- C:\Program Files\OpenCv\otherlibs\cvcam\include
- C:\Program Files\OpenCv\cvaux\include

E seus respectivos arquivos:

- C:\Program Files\OpenCv\cv\include\cv.h
- C:\Program Files\OpenCv\cxcore\include\cxcore.h
- C:\Program Files\OpenCv\otherlibs\highgui\highgui.h

- C:\Program Files\OpenCv\otherlibs\cvcam\include\cvcam.h
- C:\Program Files\OpenCv\cvaux\include\cvaux.h

Depois em *C++ Linker->Libraries*, as seguintes informações devem ser incluídas:

- cv
- highgui
- cxcore
- cvcam

Ainda na mesma opção, a indicação do diretório de procura dos arquivos cabeçalho deve ser indicada:

- C:\Program Files\OpenCV\lib

É importante salientar que essas informações devem ser incluídas todas as vezes que um projeto utilizando funções da *OpenCV* for criado.

Se todas as etapas foram concluídas corretamente, o usuário deverá ser capaz de compilar e executar um projeto em C/C++ utilizando a *OpenCV*. No Apêndice A é possível encontrar um código simples, uma espécie de programa *Hello world* para a *OpenCV*.

Cumprida toda a fase de preparação dos materiais e configuração das ferramentas utilizadas, foi iniciada a fase de desenvolvimento do código.

3.2 MÉTODOS

A idéia básica do sistema desenvolvido pode ser rapidamente explicada da forma:

i. É executada uma aplicação para que se possa capturar uma imagem da *webcam* e salvá-la em um arquivo .jpg. Essa imagem é a do local onde teoricamente os elementos a serem classificados estarão dispostos no

momento da seleção. O que foi feito no trabalho foi salvar uma imagem de fundo branco para utilizá-la como *background* de comparação. É necessário salientar a importância de uma boa iluminação na hora da captura;

ii. Tendo a imagem para comparação, deve-se agora realizar a captura de um quadro da *webcam*, contendo o objeto a ser classificado;

iii. Realizar o processamento da imagem seguindo os seguintes passos:

   a. Converter tanto a imagem do *background* como a imagem contendo o objeto a ser classificado do formato RGB para a escala em tons de cinza;

   b. Realizar a diferença absoluta entre as duas imagens;

   c. Caso o resultado não contenha nenhuma informação, concluí-se que não existe objeto na imagem ou o objeto é branco (transparente) e encerra-se o processamento;

   d. Caso o resultado da diferença seja diferente de zero, detecta-se algo e o algoritmo procede para identificar qual a cor do objeto (vermelho, verde ou azul)

iv. Por fim, o sistema informa se havia objetos ou não e sua cor.

Observando os passos acima, percebe-se que o sistema é capaz de realizar a identificação de cinco situações:

i. Não existe objeto ou o objeto é branco (transparente);

ii. Existe objeto vermelho;

iii. Existe objeto verde;

iv. Existe objeto azul;

v. Existe objeto, mas não é possível identificar sua cor.

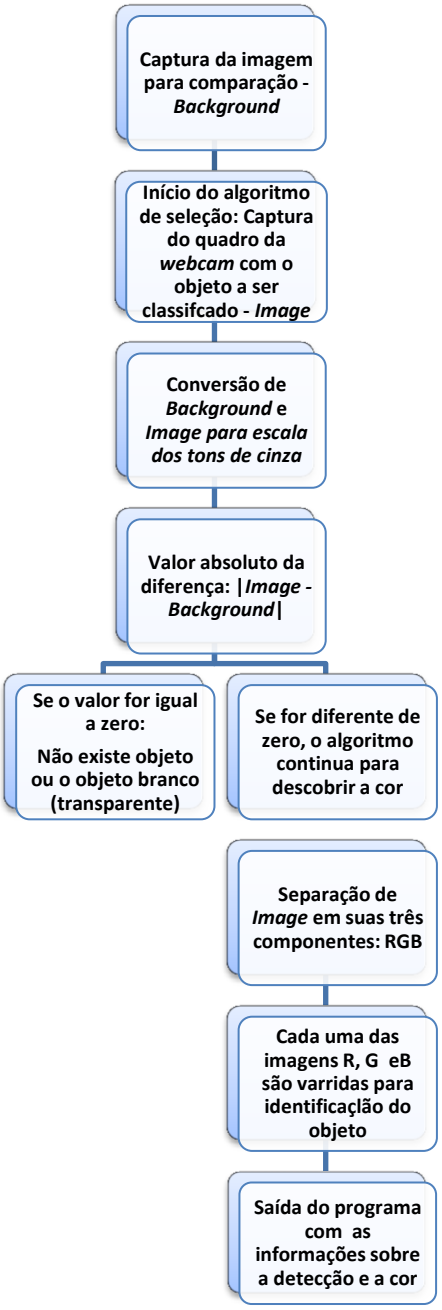

Figura 11. Grafo do algoritmo utilizado.

Para realização das etapas citadas acima, foram desenvolvidos dois projetos C/C++ no Eclipse: o *Background Capture* e o *Classificação_RGB*. O primeiro, já citado, captura a imagem de fundo e a salva em um arquivo que servirá como entrada no outro projeto. A principais funções e estruturas da *OpenCV* utilizadas por esse primeiro projeto são:

- *IplImage*: estrutura utilizada para alocar uma imagem. Contém várias informações como número de canais, altura, largura, profundidade, dentre outras;
- *CvCapture*: estrutura utilizada para captura de vídeo AVI, contém as informações sobre o arquivo;
- *cvCaptureFromCAM*: utilizada para indicar a fonte de dados de vídeo;
- *cvQueryFrame*: função utilizada para capturar um quadro do vídeo que está sendo capturado. Retorna uma imagem do tipo *IplImage*;
- *cvSaveImage*: Salva uma imagem em um arquivo;
- *cvNamedWindow*: Cria uma janela para mostrar uma imagem. O usuário pode definir o tamanho ou utilizar um parâmetro *autosize*;
- *cvMoveWindow*: Determina o local na tela onde a janela criada será exibida;
- *cvShowImage*: Indica a imagem a ser mostrada na janela criada;
- *cvWaitKey:* Função que lê as entradas do teclado. Pode ser utilizada para encerrar um laço ou terminar a aplicação;
- *cvReleaseImage*: Libera a memória que a imagem estava ocupando;
- *cvDestroyWindow*: Destrói as janelas criadas durante o programa.

O arquivo fonte do projeto pode ser visto no Apêndice B.

O outro projeto tem uma estrutura um pouco mais complexa. Seu arquivo fonte possui a função *main* e duas funções definidas como *decompose_rgb()* e *detecta()*. No Apêndice C é apresentado o código fonte do projeto.

A função *detecta()* é chamada nos seguintes casos: detectar a existência do objeto e depois para definir sua cor. Já a *decompose_rgb()* é chamada somente se for detectada a presença do objeto.

Além das funções da *OpenCV* já descritas, também são utilizadas os seguintes elementos da biblioteca no projeto:

- *cvCvtColor*: Função utilizada para conversão do espaço de cores. No programa é utilizada para conversão de RGB para GRAY (escala de cinzas);

- *cvAbsDiff*: Realiza a operação de diferença entre duas estruturas de entrada (no caso, duas imagens) e retorna o valor absoluto da operação;
- *cvCreateImage*: Cria a uma estrutura do tipo *IplImage* que contenha características pré-definidas pelo usuário, que são utilizadas como argumento da função;
- *cvGetSize*: Retorna os valores da altura e largura de uma dada imagem de entrada;
- *cvDestroyAllWindows*: Em uma aplicação onde é necessário o uso de várias janelas, essa função se torna útil para que não se tenha que usar a função *cvDestroyWindow* repetidas vezes.
- *cvGet2D*: Função que retorna o valor do pixel num dado ponto (x, y) da matriz que é a imagem de entrada

Com os algoritmos prontos, partiu-se para a fase dos testes

4 RESULTADOS

Os testes foram realizados da seguinte forma:

- Seleção de uma imagem para *background*. Optou-se pelo uso de uma folha de papel em branco;
- Com a imagem salva, colocamos a mesma no diretório do projeto *Classificação_RGB*;
- Para testar a seleção e o reconhecimento de cores foram utilizados papéis em branco com um desenho de uma só cor (vermelho, verde, azul e preto);
- Os papéis foram posicionados dentro do campo da lente da *webcam* de maneira a toda área capturada ser composta pelo papel;
- Foram testadas cinco possibilidades: Sem objetos ou objeto da mesma cor do plano de fundo; objeto vermelho; objeto verde; objeto azul e objeto de outra cor (objeto preto).

Os resultados são mostrados a seguir, onde podem ser visualizadas as janelas de saída da aplicação, bem como a sinalização da saída de texto diretamente no console do Eclipse.

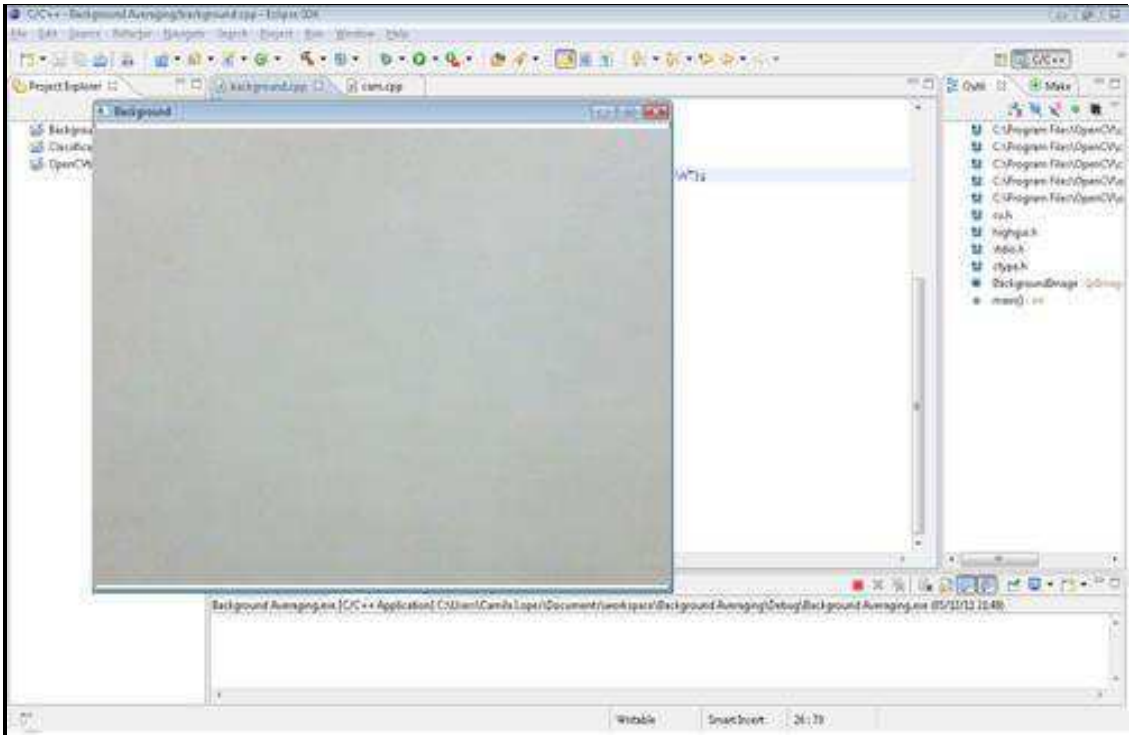

Figura 12. Tela do eclipse juntamente com a janela mostrando o *Background* capturado.

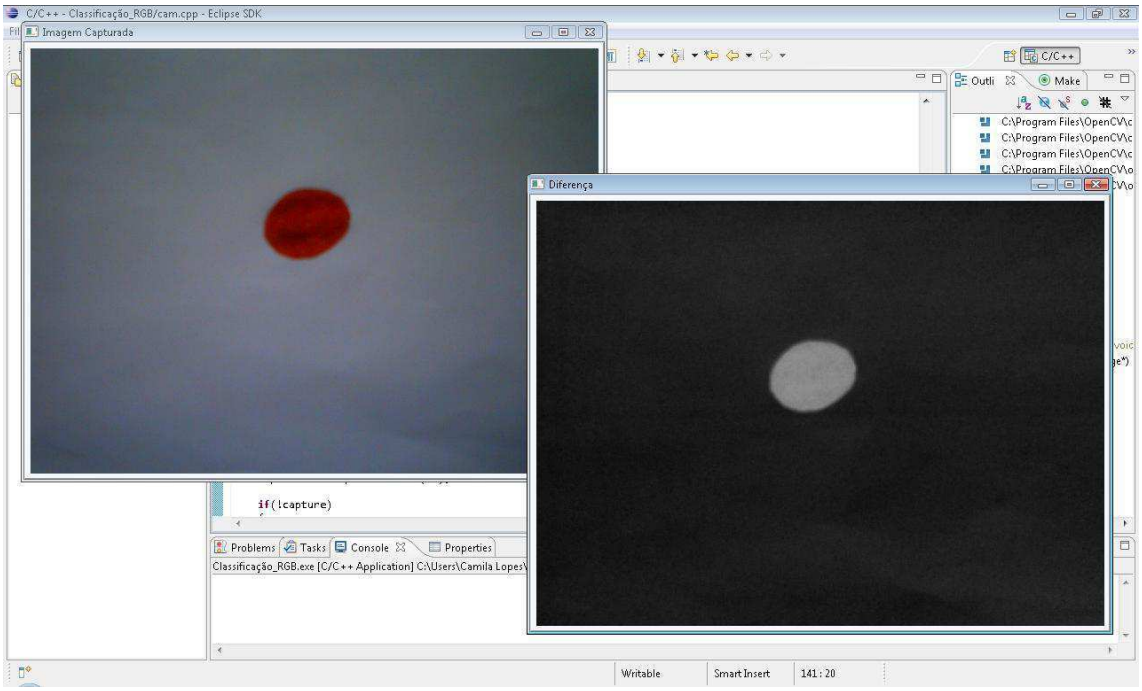

Figura 13. Telas do programa mostrando o quadro capturado e a diferença entre o *background* e a imagem capturada.

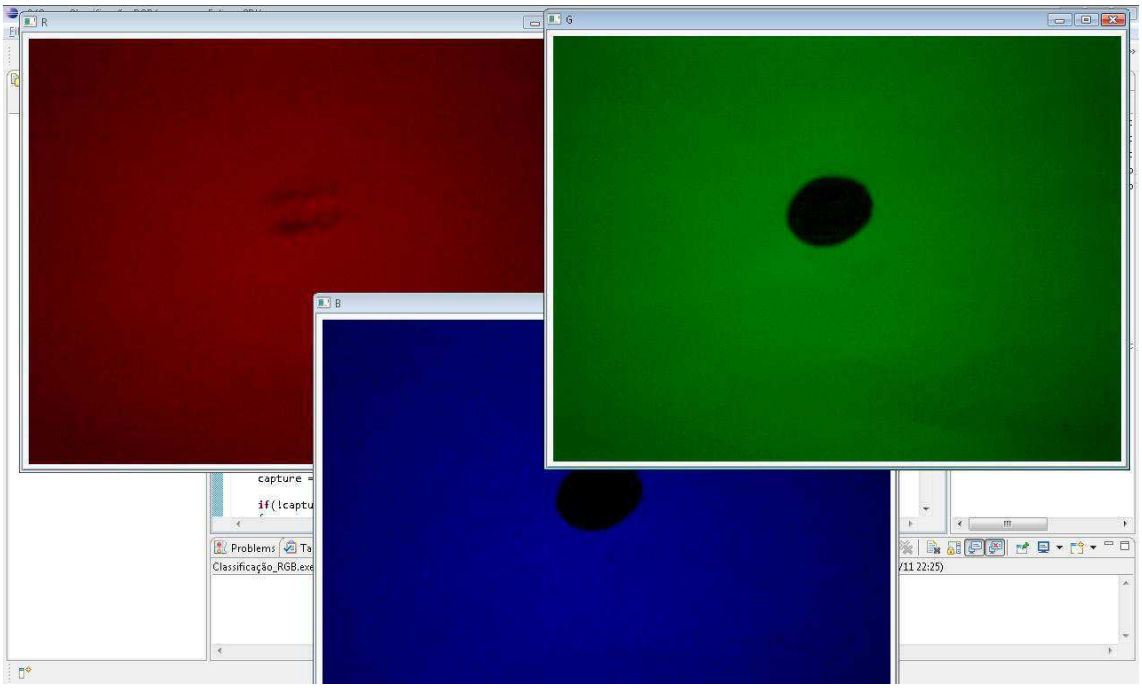

Figura 14. Telas do programa mostrando a decomposição da imagem capturada – Caso do objeto vermelho.

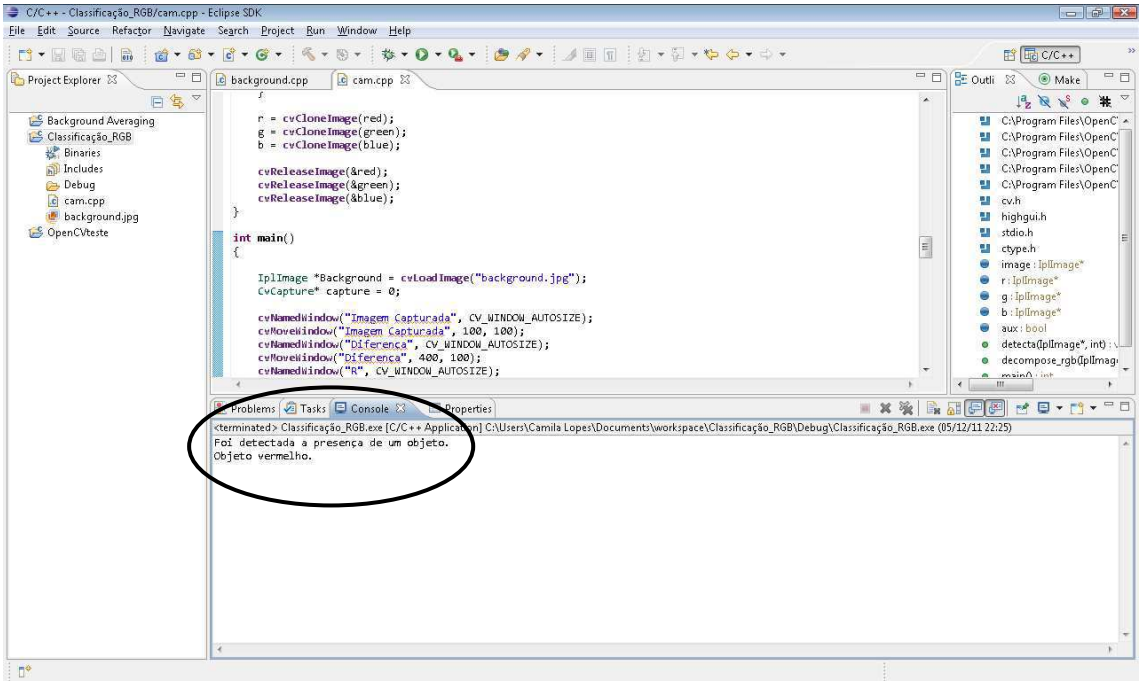

Figura 15. Console do Eclipse mostrando a saída final do programa sinalizando a detecção e a cor do objeto.

Para as outras cores o processo é semelhante, não sendo necessário mostrar todas as telas do passo a passo novamente. Apenas serão mostradas as telas de separação das cores.

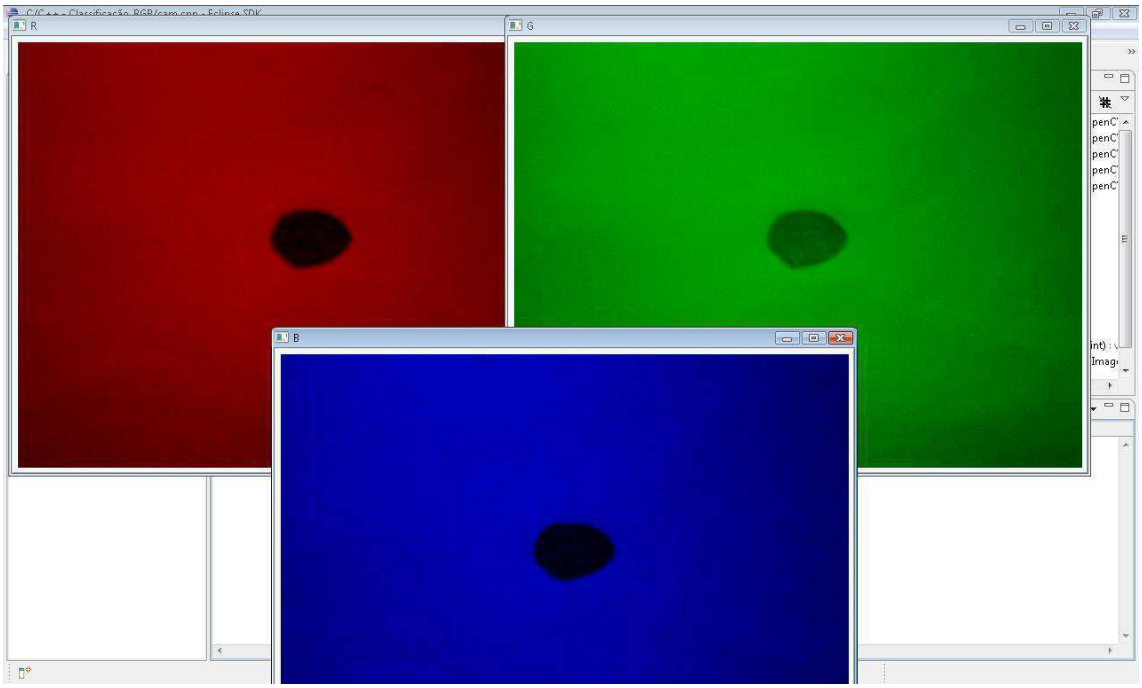

Figura 16. Telas do programa mostrando a decomposição da imagem capturada – Caso do objeto verde.

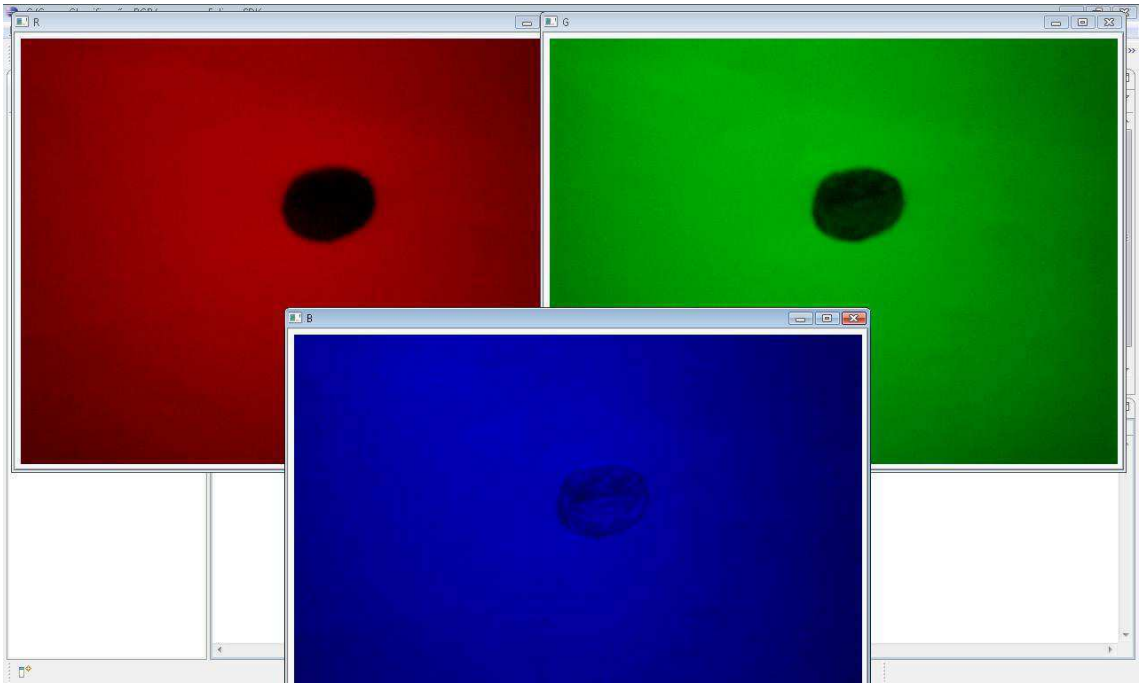

Figura 17 Telas do programa mostrando a decomposição da imagem capturada – Caso do objeto azul.

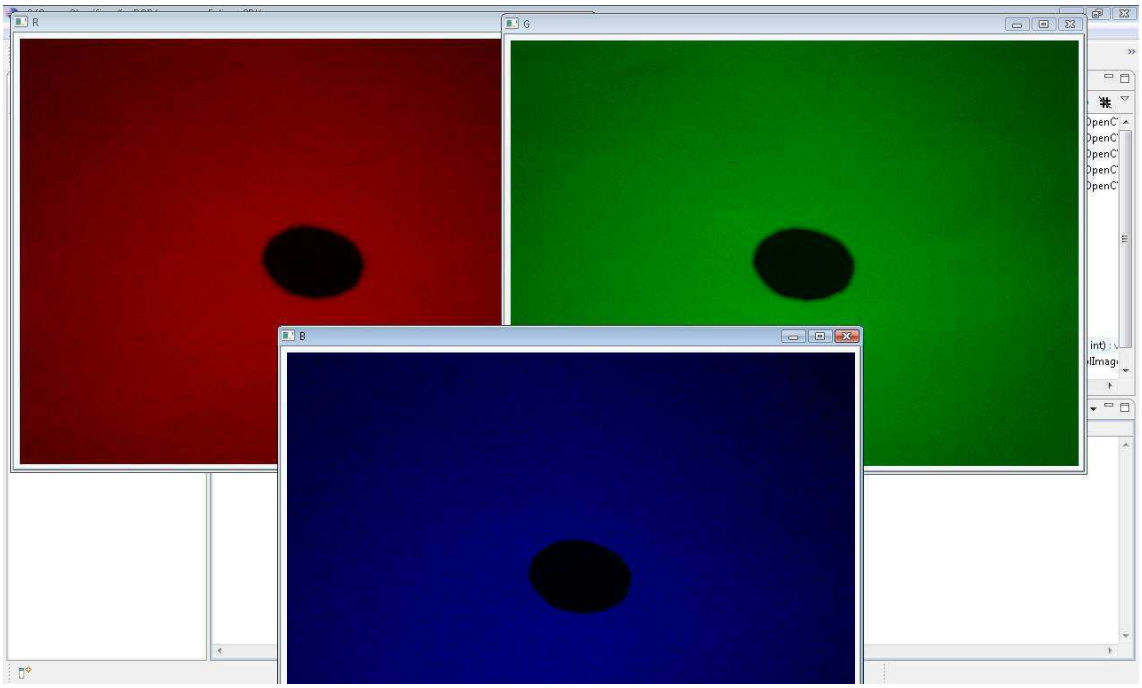

Figura 18. Telas do programa mostrando a decomposição da imagem capturada – Caso do objeto preto.

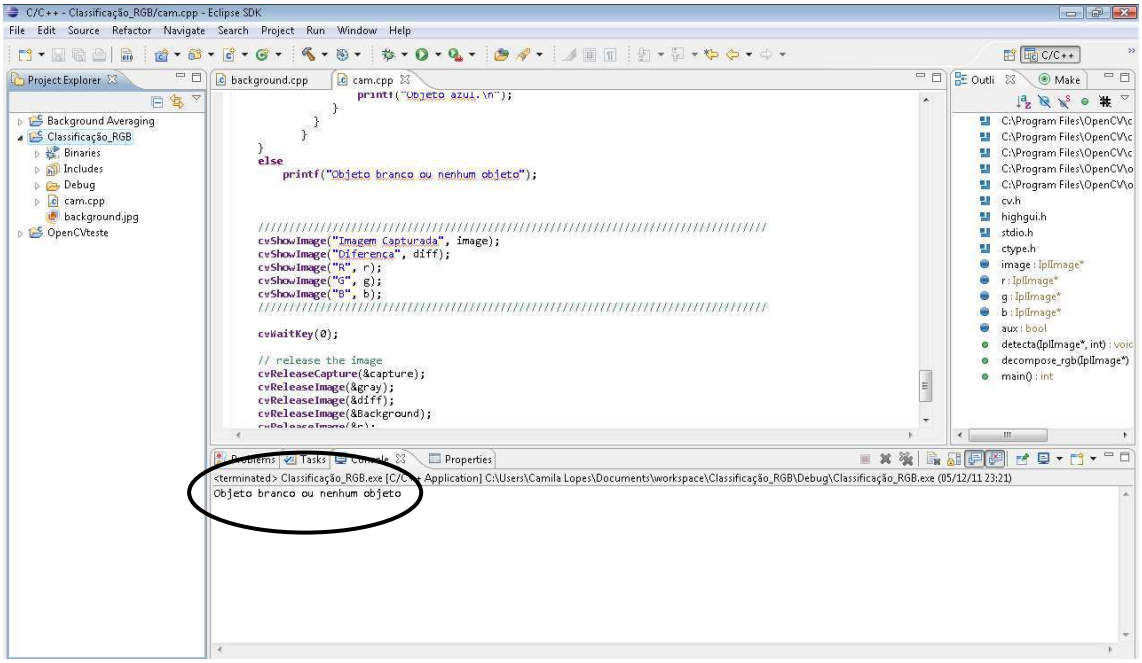

Figura 19. Tela do console para o caso do quadro capturado não conter oobjetos ou conter um objeto branco (transparente).

Verificou-se que o sistema realizou as tarefas propostas. Ele foi capaz de identificar a presença de algum objeto e, caso ele fosse de uma das cores primárias (vermelho, verde ou azul), também foi capaz de indicar corretamente a cor.

Nas figuras que mostram a decomposição de cor, pode-se sempre perceber que a componente da cor do objeto é sempre a que mostra o objeto de forma quase imperceptível. É através dessa característica que o aplicativo é capaz de indicar a cor na hora da seleção.

No caso da presença de objetos de outra cor (como foi exemplificado pelo objeto preto) o aplicativo apenas sinalizava a presença de um objeto, sem indicação de cores.

Notou-se também a importância de uma iluminação adequada na hora da captura do quadro da imagem proveniente da *webcam*. Durante os testes realizados, sempre que a iluminação não era suficiente, o sistema apresentou resultados duvidosos, pois a imagem não pode apresentar áreas de sombra nem qualquer alteração de cor que não seja o próprio objeto. Deste modo, sempre que a iluminação foi configurada de maneira adequada, o sistema é foi capaz da identificação. Posteriormente, a realização de testes repetidos e com variadas situações para que se possa afirmar o percentual de acertos do sistema torna-se interessante e necessária. Esta seria então, a primeira tarefa para a continuação das pesquisas.

5 CONSIDERAÇÕES FINAIS

O crescente aumento do consumo de produtos que geram resíduos, como as garrafas PET, é notável e considerado um problema em termos de poluição ambiental. Inúmeras são as iniciativas de reciclagem desses materiais, o que acabou por gerar o surgimento de um ramo de mercado para beneficiamento do plástico PET e seu posterior uso como matéria prima em diversas aplicações.

Um dos problemas nesse processo de reciclagem é a triagem dos materiais (separação de outros tipos de plástico e classificação por cor). O que se observa em grandes partes das indústrias do setor é essa triagem realizada manualmente, pela mão de obra humana, o que pode não ser muito eficiente em termos de velocidade e qualidade. Desta maneira, essa área é identificada como uma ótima aplicação das tecnologias de seleção e classificação automatizadas. O problema quanto a essa adoção é o preço elevado das máquinas comerciais que realizam esse tipo de processo.

Desta forma, o Estágio descrito neste relatório foi realizado no intuito de introduzir essa área de pesquisa no Laboratório de Automação e Processamento de Sinais da UFCG. Como atividade principal do Estágio, foi proposto o desenvolvimento de um sistema de detecção e classificação por cor. Ao final das atividades, foi possível verificar o funcionamento do sistema, que respondeu como esperado aos testes realizados.

Uma das idéias ao iniciar o desenvolvimento do sistema é dar o primeiro passo para implementação de algo mais robusto. Deste modo, o trabalho poderá ser aproveitado como ponto de partida para a continuação das atividades de aperfeiçoamento.

Ao decorrer do trabalho, foram utilizados conceitos adquiridos ao longo da graduação de Engenharia Elétrica, o que resultou em bons frutos e experiência pessoal significativa, de modo que as principais funções da disciplina de Estágio foram atingidas de maneira satisfatória.

BIBLIOGRAFIA

ABIPET. **Associação Brasileira da Indústria do PET**, 2011. Disponivel em: <http://abipet.org.br>. Acesso em: 4 Dezembro 2011.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV:** Computer Vision With the OpenCV Library. 1a Edição. ed. [S.l.]: O'Reilly, 2008.

FENG, G.; QIXIN, C. Study on Color Image Processing Based Intelligent Fruit Sorting System. **Proceedings of the 5th World Congress on Intelligent Control and Automation**, Hangzhou, 15 - 19 Junho 2004.

LAPS. **LAPS**, 2011. Disponivel em: <http://laps.dee.ufcg.edu.br>. Acesso em: 4 Dezembro 2011.

OPENCV Wiki. **Site da OpenCV**, 2011. Disponivel em: <http://opencv.willowgarage.com>. Acesso em: 2 Dezembro 2011.

RIISE, B. L. et al. Value Added Color Sorting of Recycled Plastic Flake from End-of-Life Electrical and Electronic Equipment. **Proceedings of the 2001 IEEE Internetional Symposium on Electronics and the Environment**, Denver, 7 - 9 Maio 2001. 223 - 228.

RUOYU, Z.; ZA, K.; YINLAN, J. Design of Tomato Color System Multi-Channel Real-time Data Acquisition and Processing System Based on FPGA. **Information Science and Engineering (ISISE), 2010 Internetional Symposium on**, Shanghai, 24 - 25 Dezembro 2010. 207 - 212.

SCAVINO, E. et al. Application of automated image analysis to the identification and extraction of recyclable plastic bottles. **Journal of Zhejiang University**, 2009.

APÊNDICE A – PROGRAMA TESTE DA *OPEN*CV

```
/ * hello.cpp
 *
 *  Created on: 29/11/2011
 *  Author: Camila Lopes
 *  Esse é um exemplo simples e introdutório. O programa lê uma imagem de um
 *  arquivo cujo caminho é especificado, inverte suas cores e mostra o
 *  resultado
 */

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <cv.h>
#include <highgui.h>

int main(int argc, char *argv[])
{
	IplImage* img = 0;
	int height,width,step,channels;
	uchar *data;
	int i,j,k;

	//A imagem a ser utilizada deve estar no mesmo diretório do projeto
	if(argc<2)
		printf("Utilizando a seguinte imagem: <insira o nome do arquivo
que contém a imagem>\n\7");

	//Carrega a imagem
	img = cvLoadImage(argv[1]);

	if(!img)
		printf("Não foi possível carregar o arquivo: %s\n",argv[1]);


	//Obtem os dados acerca da imagem carregada
	height = img->height;
	width = img->width;
	step = img->widthStep;
	channels = img->nChannels;
	data = (uchar *)img->imageData;

	printf("Processando uma imagem %d x %d com %d canais\n", height,
width, channels);

	//Cria uma janela para mostrar a imagem
	cvNamedWindow("mainWin", CV_WINDOW_AUTOSIZE);
	cvMoveWindow("mainWin", 100, 100);

	//Laço para inverter as cores
	for(i = 0; i < height; i++)
		for(j = 0;j < width; j++)
			for(k = 0; k < channels; k++)
```

```
                        data[i*step+j*channels+k] = 255-
data[i*step+j*channels+k];

        //Mostra a imagem invertida
        cvShowImage("mainWin", img );

        //Aguarda uma tecla do usuário para encerrar o programa
        cvWaitKey(0);

        //Libera o ponteiro da imagem e destrói a janela
        cvReleaseImage(&img);
        cvDestroyWindow("mainWin");

        return 0;
}
```

APÊNDICE B – PROGRAMA PARA CAPTURA DO PLANO DE FUNDO COMPARATIVO (PROJETO *BACKGROUND CAPTURE)*

```
/ * background.cpp
 *
 *  Created on: 29/11/2011
 *  Author: Camila Lopes
 *  Projeto criado para realizar a captura do plano de fundo que será
 *  utilizado como comparação da atividade de seleção.
 */
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <ctype.h>

//Cria variável do tipo IplImage para alocar a imagem da qebcam
IplImage *BackgroundImage = 0;

int main()
{
	CvCapture *BackgroundCapturedImage = 0;

	//Inicia a webcam
	BackgroundCapturedImage = cvCaptureFromCAM(-1);

	if(!BackgroundCapturedImage){
		fprintf(stderr,"Não foi possível realizar a captura da
imagem...\n");
		return -1;
		}
	//Captura do frame para ser utilizado como background
	for(int i = 0; i < 10; i++)
		BackgroundImage = cvQueryFrame(BackgroundCapturedImage);

	//Salva a imagem em um arquivo .jpg
	if(!cvSaveImage("background.jpg" , BackgroundImage))
		printf("Could not save: %s\n", "background.jpg");

	//Criação da janela para mostrar o resultado da captura
	cvNamedWindow("Background", CV_WINDOW_AUTOSIZE);
	cvMoveWindow("Background", 100, 100);
	cvShowImage("Background", BackgroundImage);
	//Aguarda o usuário apertar uma tecla para encerrar a aplicação
	cvWaitKey(0);
	//Libera o ponteiro para a imagem e destrói a janela
	cvReleaseImage(&BackgroundImage);
	cvDestroyWindow("Background");
	return 0;
}
```

APÊNDICE C – PROGRAMA PRINCIPAL PARA DETECÇÃO E CLASSIFICAÇÃO POR COR (PROJETO *CLASSIFICAÇÃO_RGB)*

```
/*
 * cam.cpp
 *
 *  Created on: 29/11/2011
 *  Author: Camila Lopes
 *  Projeto criado para realizar a captura do frame que será processado
 *  para detecção da presença de objeto e classificação dentro das cores
 *  primárias (vermelho, verde ou azul).
 */

#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <ctype.h>

//Declaração das variáveis globais do sistema
IplImage *image = 0;
IplImage *r = 0;
IplImage *g = 0;
IplImage *b = 0;
bool aux;


//Função utilizada para detectar a presença do objeto na imagem e também para
indicar sua cor
void detecta(IplImage *img, int c)
{
	int width  = img->width;
	int height = img->height;
	int valor = 0;
	int aux2 = 0;

	CvScalar s; //Tipo da OpenCV. É 4-tupla de variáveis tipo double

	switch(c){ //Switch é utilizado para sinalizar em que canal da imagem
o algoritmo
			//deve procurar o objeto

		case 0: //Caso de imagens com um só canal
			for(int i = 0; i < height; i++){
				for (int j = 0; j < width; j++){
					s = cvGet2D(img,i,j);
					if(s.val[0] > 50){
						valor++;
						aux2++;
```

```
                                        s.val[0] = 0;
                                }
                        }
                }
                if(valor > 1000) aux = true;
                else aux = false;
                break;

                case 1://Caso de imagens com componente no canal R
                        for(int i = 0; i < height; i++){
                                for (int j = 0; j < width; j++){
                                        s = cvGet2D(img,i,j);
                                        if(s.val[2] < 25){
                                                valor++;
                                                s.val[2] = 0;
                                        }
                                }
                        }
                        if(valor > aux2) aux = true;
                        else aux = false;
                        break;

                case 2://Caso de imagens com componente no canal G
                        for(int i = 0; i < height; i++){
                                for (int j = 0; j < width; j++){
                                        s = cvGet2D(img,i,j);
                                        if(s.val[1] < 25){
                                                valor++;
                                                s.val[1] = 0;
                                        }
                                }
                        }
                        if(valor > aux2) aux = true;
                        else aux = false;
                        break;

                case 3: //Caso de imagens com componente no canal B
                        for(int i = 0; i < height; i++){
                                for (int j = 0; j < width; j++){
                                        s = cvGet2D(img,i,j);
                                        if(s.val[0] < 25){
                                                valor++;
                                                s.val[0] = 0;
                                        }
                                }
                        }
                        if(valor > aux2) aux = true;
                        else aux = false;
                        break;

        }

}

//Função utilizada para decompor uma imagem em suas três componentes RGB
void decompose_rgb(IplImage *img)
{
        IplImage *red = cvCreateImage(cvSize(img->width, img->height ), img-
>depth, img->nChannels );
```

```
	red->origin = IPL_ORIGIN_BL;
	IplImage *green = cvCreateImage(cvSize(img->width, img->height ), img-
>depth, img->nChannels );
	green->origin = IPL_ORIGIN_BL;
	IplImage *blue = cvCreateImage(cvSize(img->width, img->height ), img-
>depth, img->nChannels );
	blue->origin = IPL_ORIGIN_BL;

	//Declara o ponteiro para acessar os dados da imagem
	uchar *pImg   = (uchar* )img->imageData;

	//Declara o ponteiro para escrever os dados das novas imagens
	uchar *pRed   = (uchar* )red->imageData;
	uchar *pGreen = (uchar* )green->imageData;
	uchar *pBlue  = (uchar* )blue->imageData;

	int i, j, rED, gREEN, bLUE;

	for( i = 0 ; i < img->height ; i++ ){
	  for( j = 0 ; j < img->width ; j++ ){
		rED = pImg[i*img->widthStep + j*img->nChannels + 2];
		gREEN = pImg[i*img->widthStep + j*img->nChannels + 1];
		bLUE = pImg[i*img->widthStep + j*img->nChannels + 0];

		//Vermelho
		pRed[i*img->widthStep + j*img->nChannels + 2] = rED;
		//Verde
		pGreen[i*img->widthStep + j*img->nChannels + 1] = gREEN;
		//Azul
		pBlue[i*img->widthStep + j*img->nChannels + 0] = bLUE;
	  }
	}

	//Copia as imagens para as variáveis globais
	r = cvCloneImage(red);
	g = cvCloneImage(green);
	b = cvCloneImage(blue);

	//Libera os ponteiros das imagens locais
	cvReleaseImage(&red);
	cvReleaseImage(&green);
	cvReleaseImage(&blue);
}

//Função principal
int main()
{
	//Carrega Imagem utilizada como background para comparação
	IplImage *Background = cvLoadImage("background.jpg");

	CvCapture* capture = 0;

	////////////////////////////////////////////////////////////////
	//Criação de todas as janelas utilizadas no programa
	cvNamedWindow("Imagem Capturada", CV_WINDOW_AUTOSIZE);
	cvMoveWindow("Imagem Capturada", 100, 100);
	cvNamedWindow("Diferença", CV_WINDOW_AUTOSIZE);
	cvMoveWindow("Diferença", 400, 100);
	cvNamedWindow("R", CV_WINDOW_AUTOSIZE);
```

```
        cvMoveWindow("R", 400, 100);
        cvNamedWindow("G", CV_WINDOW_AUTOSIZE);
        cvMoveWindow("G", 400, 300);
        cvNamedWindow("B", CV_WINDOW_AUTOSIZE);
        cvMoveWindow("B", 400, 500);
        ////////////////////////////////////////////////////////////////////

        //Inicia captura da webcam
        capture = cvCaptureFromCAM(-1);

        if(!capture)
        {
                fprintf(stderr,"Não foi possível realizar a captura da
imagem...\n");
                return -1;
        }

        //Seleciona um frame para o processamento
        for(int i = 0; i < 10; i++)
                image = cvQueryFrame(capture);

    ////////////////////////////////////////////////////////////////////
    //Cria nova imagem para versão em escala de cinza do frame capturado
    IplImage *gray = cvCreateImage(cvGetSize(image), IPL_DEPTH_8U, 1);
    gray->origin = IPL_ORIGIN_BL;

        //Cria nova imagem para versão em escala de cinza do background
        IplImage *gray1 = cvCreateImage(cvGetSize(Background), IPL_DEPTH_8U,
1);
        gray1->origin = IPL_ORIGIN_BL;

        //Conversão de cores (CV_RGB2GRAY: converte RGB para escala de cinzas
        cvCvtColor(image, gray, CV_RGB2GRAY);
        cvCvtColor(Background, gray1, CV_RGB2GRAY);

        ////////////////////////////////////////////////////////////////
        //Cria nova imagem para a diferença entre Image e Background
        IplImage *diff = cvCreateImage(cvGetSize(Background), IPL_DEPTH_8U,
1);
    diff->origin = IPL_ORIGIN_BL;

    //Função para realizar a operação do absoluto da diferença
    cvAbsDiff(gray, gray1, diff);

    //Primeira chamada da função detecta() para detectar presença
    detecta(diff, 0);

    //Variável auxiliar aux do tipo bool. True se detectar objeto
    if(aux){
        printf("Foi detectada a presença de um objeto.\n");
        //Como foi detectada a presença de um objeto a função de decomposição
é chamada
        //para que se possa detectar a cor
        decompose_rgb(image);

        //Segunda chamada da função detecta que avalia se o objeto é vermelho
        detecta(r, 1);

        //Se aux é False, a imagem é indicada como vermelha
```

```
        if(!aux){
                printf("Objeto vermelho.\n");
        }
        else{
                //Terceira chamada da função detecta que avalia se o objeto é
verde
                detecta(g, 2);

                //Se aux é False, a imagem é indicada como verde
                if(!aux){
                        printf("Objeto Verde.\n");
                }
                else{

                        //Quarta chamada da função detecta que avalia se o objeto
é azul
                        detecta(b, 3);

                        //Se aux é False, a imagem é indicada como azul
                        if(!aux){
                                printf("Objeto azul.\n");
                        }
                 }
            }
      }
      else //Se não houver detecção do objeto, aux é False
        printf("Objeto branco ou nenhum objeto");



      ////////////////////////////////////////////////////////////////////
      //Chamada as funções para exibição das imagens nas janelas criadas
      cvShowImage("Imagem Capturada", image);
      cvShowImage("Diferença", diff);
      cvShowImage("R", r);
      cvShowImage("G", g);
      cvShowImage("B", b);
      ////////////////////////////////////////////////////////////////////

      //Aguarda o usuário apertar uma tecla para encerrar a aplicação
      cvWaitKey(0);

      //Libera os ponteiro para as imagem e para a captura e destroi todas as
janelas
      cvReleaseCapture(&capture);
      cvReleaseImage(&gray);
      cvReleaseImage(&diff);
      cvReleaseImage(&Background);
      cvReleaseImage(&r);
      cvReleaseImage(&g);
      cvReleaseImage(&b);
      cvDestroyAllWindows();

      return 0;
}
```