5
Protótipo e Resultados Obtidos

5.1.
Introdução

Este capítulo apresenta de forma minuciosa o protótipo desenvolvido durante este trabalho, que é uma ferramenta capaz de realizar o treinamento e a detecção de objetos em um único ambiente integrado, seguindo as técnicas de aperfeiçoamento propostas no capítulo 4 desta dissertação.

São apresentados a arquitetura do protótipo, o módulo para treinamento de objetos, o módulo para teste de performance do treinamento realizado, e por fim, o módulo para detecção de objetos em imagens ou vídeos digitais. Além disto, são apresentadas algumas telas do trabalho implementado a fim de ilustrar a utilização do mesmo por um usuário.

Também são apresentados os ganhos provenientes da utilização do protótipo desenvolvido neste trabalho em relação aos resultados previamente obtidos com a técnica proposta por Viola e Jones (2001).

5.2.
Arquitetura do Protótipo

O protótipo construído neste trabalho visa criar um ambiente integrado onde um usuário consiga tanto treinar uma nova cascata de classificadores, quanto detectar objetos previamente treinados em tempo real em imagens ou vídeos digitais. A definição dos módulos do protótipo foi feita de modo que cada um constitua uma unidade lógica bem definida e coesa, altamente encapsulada e com baixo acoplamento entre si. Para construção de cada um dos módulos foi utilizada a linguagem C.

O protótipo utiliza ainda duas bibliotecas muito conhecidas na literatura: a biblioteca **IUP** (Portable User Interface), produzida pelo Tecgraf – PUC-Rio (1994) que é uma API para construção de interfaces gráficas para o usuário, e a biblioteca **OpenCV** (Open Computer Vision), desenvolvida pela Intel (2006) que contém diversas estruturas de dados e algoritmos para manipulação de imagens e vídeos digitais.

Foram definidos os seguintes módulos para o protótipo:

• *Interface* – Responsável por construir toda a interface gráfica do programa, capturar e tratar todos os eventos das telas. Este módulo faz uso da biblioteca IUP.

• *Samples* – Contém rotinas para a criação de amostras (*samples*) de imagens para treinamento, rotinas de distorção em imagens como aumento, diminuição e rotação, entre outras.

• *Marker* – Possui funções que permitem a marcação de regiões nas imagens através do mouse e armazenar as informações destas regiões em um arquivo texto. É utilizado durante a criação de um conjunto de imagens positivas para treinamento.

• *Object Detect* – Capaz de analisar imagens ou quadros de um vídeo para detecção de objetos em tempo real e informar na imagem onde se encontram tais regiões. Recebe como parâmetro de entrada um arquivo de imagem ou vídeo onde os objetos devem ser procurados, e pelo menos um arquivo no formato *XML* que representa a cascata de classificadores previamente treinados.

• *Princ* – Responsável por realizar as inicializações em memória necessárias ao programa, como as estruturas de interface com o usuário ou o módulo de teste de desempenho da cascata de classificadores.

• *Test* – Módulo que realiza o teste de desempenho da cascata de classificadores construído através do algoritmo de treinamento. Exibe informações como percentual de acertos, falhas e falsos positivos em um conjunto de imagens para teste pré-selecionado.

Todos os módulos do protótipo foram desenvolvidos seguindo os padrões de programação propostos por Staa (2000) com o intuito de obter-se um artefato de software de alta qualidade, confiável e robusto.

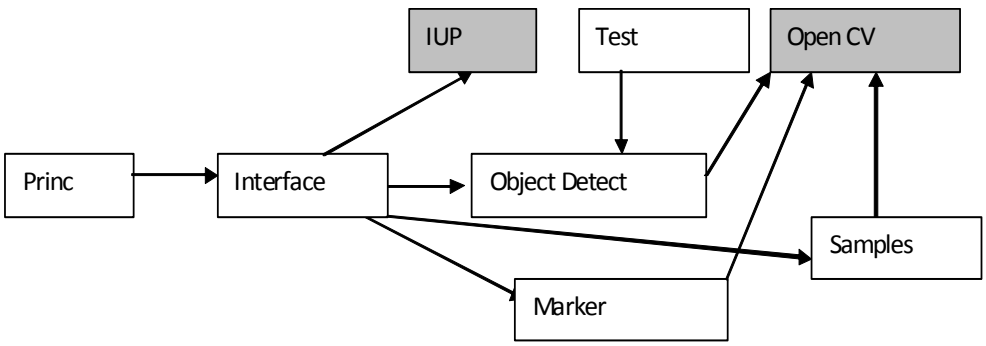

Figura 22. Representação da arquitetura dos módulos que compõem o protótipo

5.3. Treinamento

O módulo de treinamento desenvolvido no protótipo visa proporcionar um ambiente que o usuário seja capaz de treinar o algoritmo de detecção de objetos (conforme descrito no capítulo 3) de maneira simplificada.

Para tal tarefa, é necessário que o usuário forneça como entrada dois parâmetros fundamentais para que o treinamento possa ocorrer: um conjunto de **imagens positivas** e um conjunto de **imagens negativas**. De posse destas informações, o algoritmo é capaz de encontrar as características mais relevantes do objeto desejado e montar uma cascata de classificadores capaz de descartar regiões da imagem que não possuam o objeto de forma eficiente.

Viola e Jones (2001) especificam que as imagens que compõem o conjunto de imagens positivas possuam a mesma dimensão, e que o objeto para treinamento esteja posicionado da mesma forma em todas as imagens, apenas sendo aceitável ter pequenas rotações e transformações nos objetos das imagens. Para o conjunto de imagens negativas é especificado que nenhuma das imagens contenha o objeto desejado e que o número de imagens deste conjunto seja maior que o número de imagens positivas. No artigo escrito por Viola e Jones (2001) para o treinamento do objeto “face humana” foi utilizado um conjunto de imagens positivas contendo 5.000 faces, e um conjunto de imagens negativas contendo 10.000 imagens.

O módulo de treinamento foi construído de modo a facilitar a forma como o usuário possui armazenados estes conjuntos de imagens. No caso do conjunto de imagens negativas, os arquivos que o compõe podem estar contidos inteiramente em um único diretório ou ser representado por um arquivo texto que contenha o caminho físico completo de cada uma das imagens. Para o caso do conjunto de imagens positivas ele também pode ser informado de duas maneiras: 1 - um arquivo texto contendo o caminho físico completo de cada imagem juntamente com as localizações na imagem do objeto que se deseja treinar (cada imagem pode conter um ou mais regiões do objeto desejado, e cada região deve estar com os seus limites especificados no arquivo logo após o caminho completo da imagem); 2 – um arquivo binário com extensão “*vec*” que contém uma seqüência de imagens contendo apenas o objeto de interesse.

O protótipo permite que o usuário crie um conjunto finito de imagens positivas a partir de uma única imagem do objeto por meio de **distorções** da imagem original. O arquivo binário gerado possui a extensão “vec” e as imagens

contidas dentro dele possuem as dimensões definidas pelo usuário na interface do protótipo. A figura 23 mostra a tela de criação do conjunto de imagens positivas, onde o usuário informa parâmetros como o número de imagens positivas que deve ser gerado, a rotação a ser aplicada nos eixos X, Y e Z, a resolução em pixels desejada, etc. Uma vez iniciado, o processo de geração de amostras positivas pode ser acompanhado através do console de mensagens do protótipo.

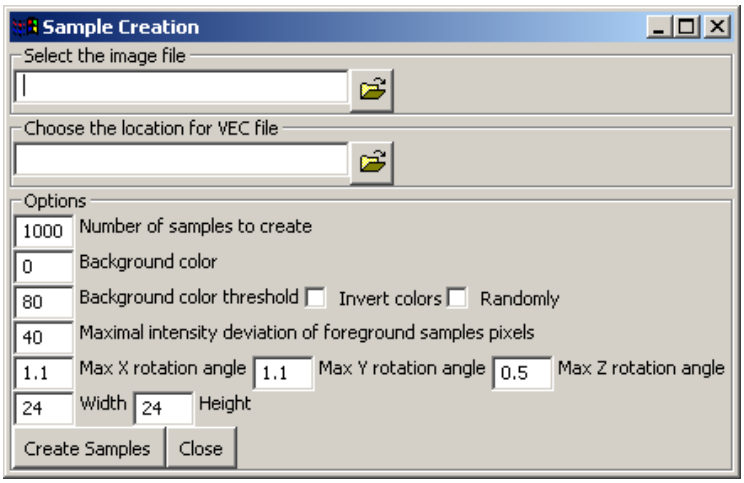

Figura 23. Criação de conjunto finito de imagens positivas a partir de uma imagem

O protótipo também permite que o usuário possa efetuar a marcação de áreas que contenham o objeto de interesse em uma ou mais imagens. Uma vez o usuário tendo fornecido o diretório em que se encontram as imagens com o objeto para treino, são exibidas de forma consecutiva cada uma das imagens existentes no diretório para que o usuário efetue a marcação da região do objeto. Ao término da marcação em cada uma das imagens, é gerado um arquivo texto com cada imagem correspondendo a uma linha no formato: *<caminho físico da imagem> <quantidade de objetos* N*> <ponto superior esquerdo 1> <ponto inferior direito 1>,..., <ponto superior esquerdo* N *> <ponto inferior direito* N *>*. É importante lembrar que como as regiões marcadas são retangulares apenas os pontos superior esquerdo e inferior direito são necessários. A figura 24 ilustra o processo de marcação de três regiões da imagem que contém o objeto que deve ser treinado e o seu respectivo arquivo texto gerado.

Tendo definido os conjuntos de imagens positivas e negativas a serem utilizados, o usuário pode optar por iniciar o processo de treinamento do objeto. Alguns parâmetros de como o treinamento deve ser realizado precisam ser informados pelo usuário. Entre eles podem-se citar o número de estágios que a

cascata gerada deve possuir, o número de amostras que devem ser utilizadas dos conjuntos de imagens positivas e negativas, a taxa de detecção mínima e a taxa de falso positivo máximo para cada estágio e se o objeto a ser treinado possui simetria vertical. Caso possua, o tempo de treinamento pode ser substancialmente otimizado, já que a metade de um lado da imagem é o "espelho" da outra metade. O protótipo já traz os valores padrões para o treinamento definidos por Viola e Jones (2001), porém o parâmetro que corresponde à resolução de pixels da imagem do objeto a ser treinado deve sempre ser informado conforme cada caso, evitando-se colocar resoluções maiores que 30x30 para não tornar o treinamento mais demorado ainda. Todos os parâmetros para o treinamento de um objeto são ilustrados na figura 25.

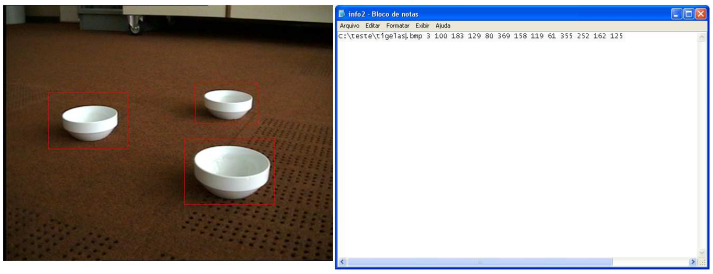

Figura 24. Marcação de áreas que contém o objeto para geração de conjunto de imagens positivas

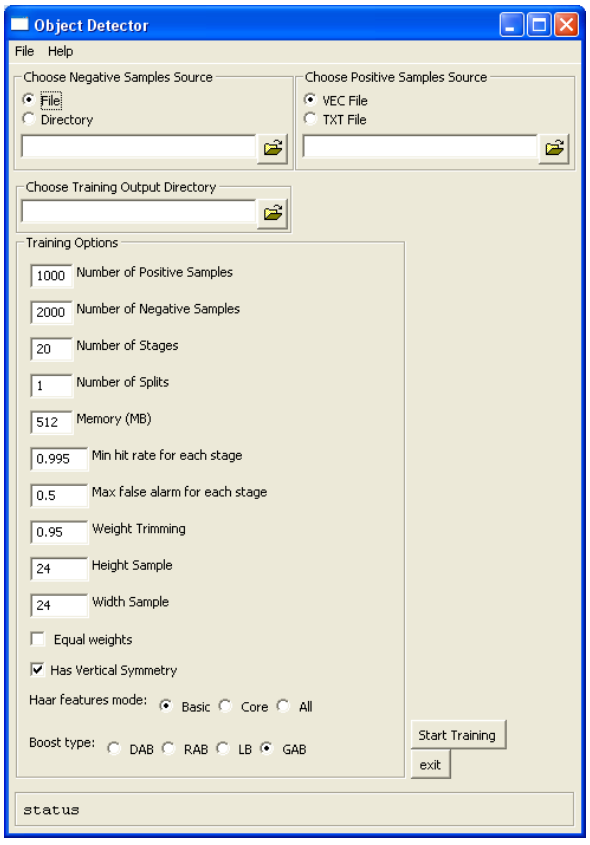

Figura 25. Parâmetros para se realizar o treinamento de um objeto

O processo de treinamento pode ser acompanhado por meio do console da aplicação, onde são informados continuamente o estágio atual do treino, os percentuais de acerto e falso positivos do estágio e a quantidade de características utilizadas em cada classificador. Ao término do treino o protótipo gera no caminho físico definido pelo usuário um arquivo *XML* contendo a cascata de classificadores pronta para ser utilizada pelo algoritmo de detecção.

5.4. Detecção de Objetos

O módulo de detecção de objetos recebe dois parâmetros de entrada: a imagem ou vídeo onde o objeto deve ser detectado, e uma lista de cascata de classificadores representando os objetos que devem ser procurados. Para cada um destes objetos é atribuído internamente uma cor específica, com a qual a imagem terá a região do objeto envolvida por um círculo com a cor do objeto correspondente. Desta forma quando mais de um objeto for buscado simultaneamente o usuário pode observar facilmente os objetos distintos que forem sendo encontrados.

O protótipo atualmente aceita apenas arquivos de vídeo com extensão "*AVI*" que não utilizem *codecs* de compactação (e.g. *DivX*) devido a restrições de compatibilidade e direitos autorais na biblioteca OpenCV (Intel, 2006). Apesar da restrição a um único tipo de arquivo de vídeo aceito pelo protótipo, o mesmo não ocorre em relação aos tipos de arquivos de imagens, onde as extensões mais utilizadas atualmente são aceitas pelo detector (e.g. "*bmp*", "*jpeg*", "*gif*" e "*tif*").

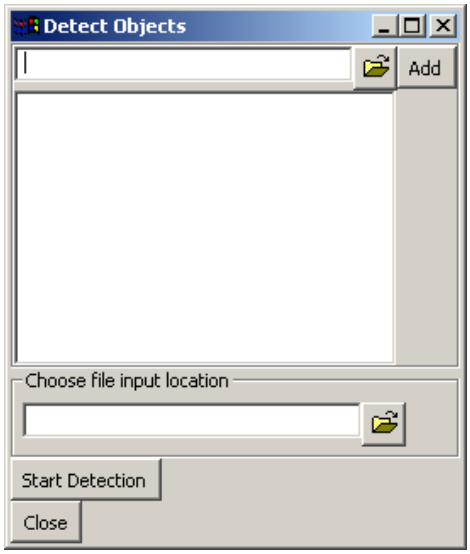

Figura 26. Parâmetros para a detecção de objetos

A figura 26 ilustra a interface exibida ao usuário antes do início da detecção de objetos, onde é necessário informar a lista de objetos a serem procurados e a imagem ou vídeo de entrada. O exemplo de detecção realizado pelo protótipo para objetos dos tipos “face” e “par de olhos” é mostrada na figura 27. As regiões que contém os objetos são exibidas envoltas por círculos.

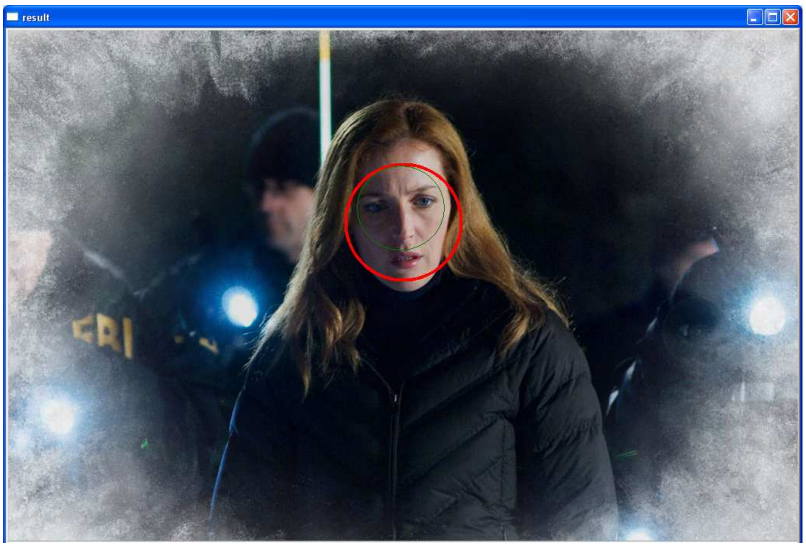

Figura 27. Detecção em imagem com resolução 1024x768 (extraída de X Files 2 (2008))

5.5. Resultados Obtidos

Viola e Jones (2001) constatam através de experimentos realizados que o algoritmo apresentado por eles possui uma das maiores velocidades de detecção e reconhecimento de objetos em relação a outros trabalhos amplamente conhecidos na literatura e considerados extremamente rápidos, como os trabalhos de Rowley et al. (1998) e de Schneiderman e Kanade (2000). Com os resultados obtidos através de experimentos com o protótipo construído neste trabalho, é comprovado um ganho substancial no desempenho da detecção de objetos em vídeos de alta resolução e também no desempenho para a detecção de objetos simultaneamente, questões até então não abordadas por Viola e Jones (2001).

Os experimentos realizados com o protótipo tiveram como foco a análise e a comparação com o trabalho de Viola e Jones (2001) das taxas de velocidade obtidas para a detecção de objetos. Devido ao fato do algoritmo desenvolvido no protótipo não ter sofrido alterações relacionadas à maneira de como o reconhecimento de objetos é realizado, os resultados referentes à quantidade de

objetos detectados corretamente e a taxa de falsos positivos foram descartados. Assim sendo, os experimentos realizados com o protótipo atendiam às seguintes configurações de teste:

1. Detecção de 1 a 4 objetos simultaneamente em uma imagem estática de diferentes resoluções;
2. Detecção de apenas 1 objeto ao longo de em um vídeo;
3. Detecção de 1 a 4 objetos simultaneamente ao longo de um vídeo.

Os testes foram realizados em uma máquina com processador *Intel Core 2 Quad*, 2.40 GHz, com 4 núcleos de processamento e 2 GB de memória *RAM*. Não foram encontrados algoritmos de trabalhos relacionados que estivessem disponíveis para testes de desempenho e comparação de resultados, somente resultados divulgados em trabalhos previamente publicados em artigos. Portanto, efetivamente foram testados o algoritmo original de Viola e Jones (2001) com o algoritmo implementado com as melhorias propostas ao longo deste trabalho. Cada um dos experimentos foi realizado com um tamanho inicial de Janela de Busca igual a 10% do tamanho da imagem testada, fator de escalonamento de 1,25 e fator de deslocamento horizontal/vertical igual a 1,5.

É importante salientar que apenas foram testados um máximo de 4 objetos por ser o número máximo de paralelização real conseguida no ambiente onde foram realizados os testes. Em um ambiente onde fosse possível ter um maior poder de paralelização, o número de objetos a serem detectados simultaneamente poderia ser aumentado de forma proporcional.

Experimento 1: A partir dos resultados deste primeiro experimento, ilustrados pelas figuras 28, 29 e 30, é possível notar o ganho através da paralelização do algoritmo de detecção de objetos. Enquanto o algoritmo que utiliza a paralelização aumenta muito pouco o seu tempo para detectar os quatro objetos, o algoritmo original de Viola e Jones (2001) praticamente quadriplica o seu tempo inicial. Isto ocorre devido ao algoritmo ser executado de forma seqüencial, ou seja, para cada um dos objetos procurados, é primeiramente criada a sua cascata de classificadores em memória para em seguida reiniciar a varredura na imagem. A figura 27 em três resoluções distintas foi utilizada como entrada de teste para este primeiro experimento.

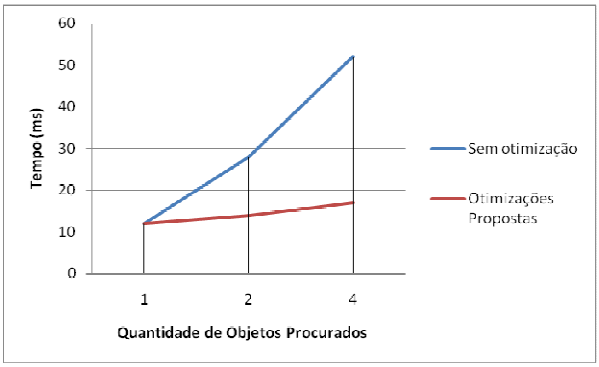

Figura 28. Tempo para detecção em imagem estática de resolução 320x240

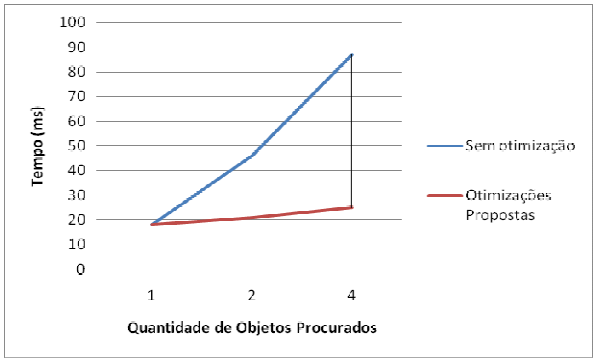

Figura 29. Tempo para detecção de imagem estática de resolução 640x480

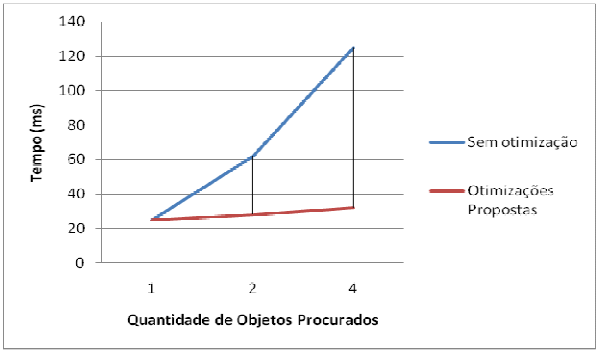

Figura 30. Tempo para detecção de imagem estática de resolução 1024x768

Experimento 2: Através do segundo experimento, ilustrado pelas figuras 33 e 34, nota-se a variação do desempenho médio do algoritmo ao longo do tempo de execução de dois vídeos testados, apenas considerando um único objeto para busca. Percebe-se que o desempenho inicial de ambos os algoritmos é similar. Isto ocorre porque no primeiro *frame* do vídeo, todas as regiões da imagem devem ser analisadas. A partir dos *frames* seguintes com a utilização da segmentação do plano de frente da cena é possível descartar boa parte das regiões de cada cena, o que acarreta a redução do tempo total de processamento. É possível notar também que nos momentos do vídeo em que

há mais “dinamismo” na cena, ou seja, mais regiões com uma grande quantidade de movimentos entre os quadros, ocorrem picos no tempo do processamento, pois o número de áreas que devem ser analisadas é muito maior. As figuras 31 e 32 ilustram respectivamente exemplos de momentos de um vídeo com grande e pequeno dinamismo entre suas cenas. Pode-se observar na figura 31 uma grande área branca, que é a área do plano de frente segmentada pelo protótipo, onde deve ter toda a sua área segmentada analisada pelo algoritmo de detecção de objetos. Já a figura 32 possui uma pequena área branca, implicando numa área menor a ser analisada pelo algoritmo. O tamanho da área a ser analisada influencia diretamente o tempo que o algoritmo de detecção leva para ser executado. Os altos picos da linha vermelha existente nas figuras 33 e 34 representam as cenas com grandes áreas a serem analisadas. Os pontos mais baixos da linha vermelha acontecem para trechos do vídeo com pequenas áreas de transições entre suas cenas, como é o caso ilustrado na figura 32.

Figura 31. Exemplo de alto dinamismo em uma cena

Figura 32. Exemplo de baixo dinamismo em uma cena

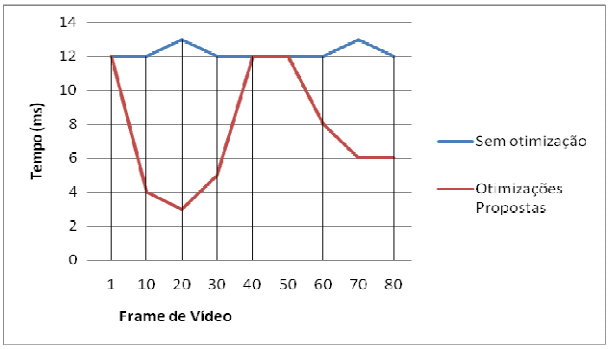

Figura 33. Tempo médio para detecção em cada *frame* de vídeo com resolução 320x240

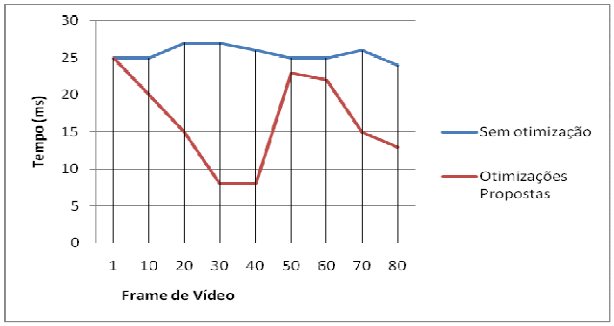

Figura 34. Tempo médio para detecção em cada *frame* de vídeo com resolução 1280x768

Experimento 3: Neste último experimento, ilustrado pela figura 35, pode-se verificar a grande diferença de desempenho do algoritmo proposto com o de Viola e Jones (2001). O algoritmo otimizado consegue manter o requisito de tempo real para a detecção paralela de 4 objetos distintos, enquanto que o algoritmo original mantém uma baixa taxa de quadros processados por segundo.

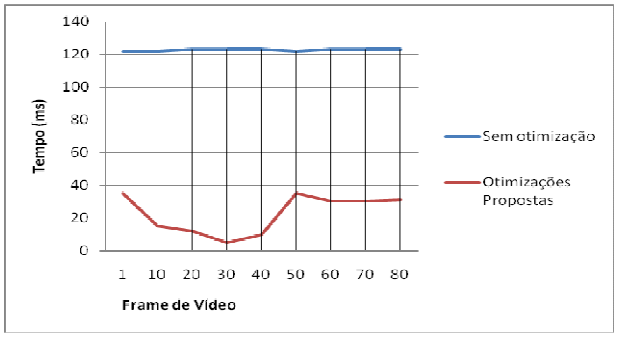

Figura 35. Tempo médio para detecção de 4 objetos simultaneamente em cada *frame* do vídeo com resolução 1280x768

Por meio dos aprimoramentos realizados no algoritmo de detecção de objetos, o protótipo é capaz de encontrar com alto desempenho objetos durante

a execução de vídeos digitais de alta resolução. É possível reduzir o tempo para detecção de objetos ao longo do vídeo em torno de 30% a 80%, dependendo da resolução do vídeo e a dinâmica de movimentação das cenas. Desta forma minimizamos consideravelmente a queda na taxa de exibição que ocorre ao utilizamos resoluções maiores de vídeos e buscarmos mais de um objeto simultaneamente.

5.6. Conclusão

Este capítulo apresenta o protótipo desenvolvido ao longo deste trabalho para treinamento e detecção de objetos. Com a utilização do protótipo apresentado o usuário dispõe de um ambiente integrado para o treinamento de novos objetos e suas respectivas detecções em imagens ou vídeos digitais.

O protótipo permite ao usuário que ele possa escolher um novo objeto qualquer para ser treinado pelo algoritmo de aprendizado. O protótipo disponibiliza uma maneira flexível para que o usuário possa criar o conjunto de imagens positivas e negativas de acordo com a sua necessidade. Por exemplo, o conjunto de imagens negativas pode ser formado por um diretório contendo várias imagens que não possuem o objeto desejado ou um arquivo texto que especifique a localização física de tais imagens. Para o conjunto de imagens positivas, o usuário pode aplicar distorções aleatórias sobre a imagem do objeto desejado ou ainda em um diretório com diversas imagens o usuário pode marcar manualmente a região que contém o objeto de cada imagem.

Após o treinamento o protótipo disponibiliza uma ferramenta de teste de performance para a cascata de classificadores gerada para que o usuário verifique se os resultados são satisfatórios. Caso não sejam, o usuário pode reiniciar o treinamento com novos parâmetros de configuração e verificar novamente o resultado.

A detecção dos objetos é realizada de maneira simples, onde é necessário apenas que o usuário informe as cascatas de classificadores dos objetos que devem ser procurados e uma imagem ou vídeo de entrada. O detector identifica se o parâmetro de entrada é uma imagem estática ou um vídeo e realiza o tratamento necessário conforme o tipo de arquivo identificado. Para cada tipo de objeto distinto encontrado na cena é desenhado um círculo em volta do mesmo a fim de facilitar a sua visualização pelo usuário.

Comparando o desempenho obtido pelo protótipo para a tarefa de detecção em relação ao trabalho de Viola e Jones (2001) foram obtidos ganhos em relação à velocidade de detecção superiores em até 80%, conseguindo-se obter uma taxa de quadros por segundo rápida o suficiente para manter o requisito de tempo real necessário a vídeos. Tal ganho apenas é possível através do uso do aprimoramento da técnica de Viola e Jones (2001) apresentada no capítulo 4 desta dissertação.