UM ROBÔ CARICATURISTA NA WEB

JOÃO H. Á. VALGAS FILHO MARIO F. M. CAMPOS RÉGIS NOGUEIRA

VERLab – Laboratório de Visão Computacional e Robótica
Departamento de Ciência da Computação, Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627, 31270-010, Belo Horizonte, MG, Brasil
E-mails: {javila, mario, regis}@dcc.ufmg.br

Resumo— O trabalho apresenta uma aplicação desenvolvida para web cujo principal objetivo é permitir o acesso da população a robôs. Um manipulador PUMA 560 esboça a caricatura da face de uma pessoa contida uma imagem enviada pela internet. O usuário envia o arquivo com a foto e monitora a robô caricaturista através de câmeras on-line. O processo como um todo inicia-se com o processamento da imagem captada onde são utilizadas técnicas específicas como detecção de bordas, afinamento e vetorização. Em seguida, ocorre a geração dos dados a serem enviados ao controlador do robô. Nesta etapa, os movimentos e trajetória do manipulador são otimizados levando em consideração pontos de singularidade e área de trabalho. Os algoritmos utilizados, a arquitetura do sistema bem como alguns resultados são descritos em detalhe.

Abstract— This paper presents a web based application whose goal is allowing people to interact with robots. A manipulator PUMA 560 paints a face on a board using a pain. The image which is going to be painted is uploaded and the user observes the task by webcams. The digital image processing (DIP) is the first job step where specific techniques are used as edge detection, thinning and vectoring. Next, comes data processing which is going to be sent to the robot controller. The robot trajectory is optimized observing the work area and singularities points. The algorithms and the architecture used as well as the results are described.

1 INTRODUÇÃO

A internet é hoje um dos principais meios de comunicação. Sua possibilidade de interação permite o desenvolvimento de serviços inviáveis há alguns anos atrás. Na área da robótica, uma aplicação utilizada é a teleoperação e o monitoramento de robôs permitindo o público em geral a ter acesso a tecnologias não convencionais. Taylor foi um dos primeiros a relatar um trabalho desta natureza (Taylor e Trevelyan, 1995). Desde então, alguns sites têm disponibilizado robôs (manipuladores e robôs móveis) para monitoramento e teleoperação (Simmons, 1998; Stein, 1999; Ho e Zhang, 1999; Siegwart e Saucy, 1999; Goldberg et al., 1996). Como o acesso à rede mundial é irrestrito, a interface deve ser a mais amigável possível, possibilitando pessoas sem nenhum conhecimento mais profundo a utilizarem e entenderem o serviço disponível. O monitoramento das tarefas utilizando câmeras on-line é uma técnica bastante utilizada.

O presente trabalho objetiva popularizar o acesso a robôs utilizando internet. A idéia é permitir que uma pessoa envie uma imagem de uma face e monitore um manipulador de 6 graus de liberdade desenhando a caricatura em um quadro. Caricaturas de pessoas públicas são comuns de serem encontradas em colunas de jornais e revistas. Os caricaturistas procuram enfatizar alguma característica física marcante encontrada na pessoa. Alguns procuram realçar olhos, nariz ou até mesmo uma mancha na pele. Uma vez disponível os traços que melhor representam a face de uma pessoa, pode-se executar uma caricatura com o mesmo objetivo. A imagem é processada com o objetivo de se obter estes traços a serem esboçados pelo manipulador. Técnicas como detecção de bordas, afinamento e vetorização foram utilizadas. Uma etapa de adequação das coordenadas ao espaço de trabalho do PUMA 560 foi necessária levando em consideração pontos de singularidade e área máxima do quadro. A Fig. 1 ilustra o manipulador esboçando uma caricatura.

Na seção seguinte serão abordados alguns trabalhos existentes na literatura. Na seção 3 será detalhada a arquitetura utilizada para execução da tarefa bem a metodologia utilizada. Os processos e técnicas utilizados serão bem detalhados. Alguns resultados são discutidos na seção 4, e na seção 5 são comentadas conclusões retiradas na realização do trabalho bem como propostas de futuros trabalhos.

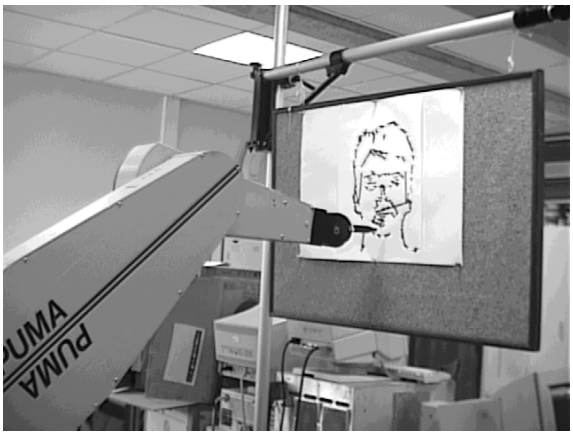

Figura 1. O manipulador PUMA 560 esboçando uma caricatura.

2 TRABALHOS RELACIONADOS

Um dos primeiros sites a disponibilizar a teleoperação de robôs pode ser visto em (Taylor e Trevelyan, 1995). Posteriormente, Simmons disponibilizou um robô móvel para ser teleoperado (Simmons, 1998). Este AGV (Autonomous Guided Vehicle), denominado XAVIER, navegava pelos corredores e salas da universidade através de comandos enviados pelo usuário através da web. No browser, uma interface foi construída exibindo a posição do robô em um mapa do local e uma imagem obtida por uma câmera disposta no XAVIER. Goldberg, com seu projeto denominado $Tele-Garden$ disponibilizou um manipulador para monitorar e preservar um jardim (Goldberg et al., 1996). Através da web, ao usuário era permitido movimentar o robô que possuía uma câmera em sua garra. Desta forma, era possível obter imagens de

pontos específicos do jardim. Além disso, outras funções eram disponíveis como jogar água nas plantas.

$PumaPaintingProject$ foi outro trabalho semelhante utilizando um PUMA 760 (Stein, 1999). O usuário desenhava formas e traços em um aplicativo que podia ser acessado por um navegador que suporta Java. Uma vez obtido o desenho feito neste aplicativo, o manipulador o repetia em um quadro utilizando pincéis. O monitoramento dos movimentos do robô era feito através de imagens obtidas do local. Para controle do robô, utilizou-se uma biblioteca em C++ para robótica (RCCL (Lloyd e Hayward, 1984)) rodando em uma máquina conectada via porta paralela ao controlador efetuando-se o controle de forma direta.

O presente trabalho ($PUMACaricaturista$) difere do projeto de Stein em dois aspectos básicos. O primeiro deles diz respeito a forma pela qual o desenho a ser reproduzido pelo manipulador é obtido. É necessário uma etapa de processamento de imagens e adequação ao espaço de trabalho. Já no $PumaPaintingProject$, as coordenadas a serem plotadas já são obtidas diretamente dos desenhos feitos no aplicativo. Um segundo aspecto que distingue os dois trabalhos é a forma de comunicação realizada com o robô. No caso do PUMA Caricaturista, houve a necessidade de se programar em VAL no controlador e a transmissão de dados foi feita via RS-232 de uma máquina.

3 DESCRIÇÃO DO SISTEMA

O processo inicia-se com o envio de uma foto pela internet e termina com o manipulador esboçando os traços desta imagem em um quadro utilizando um pincel. Uma visão geral do sistema é ilustrada na Fig. 2. Para exibição no browser, foi construída uma interface bem simples. É efetuado um upload do browser ao servidor de processos que inicia o processamento da imagem. Técnicas de processamento digital de imagens são realizadas para obtenção dos vetores que representam a face da pessoa. Estes vetores são armazenados em um arquivo que otimiza a quantidade de vetores gerados, eliminando os que possuem magnitude inferior a um dado valor. Além disso, efetua uma adequação das coordenadas ao espaço de trabalho do robô. Os vetores são então enviados (via RS-232) ao controlador em uma sequência pré-determinada. O programa em VAL, rodando no controlador, efetua o controle dos movimentos o robô de forma a esboçar a caricatura.

Paralelamente a estes processos, existem duas câmeras on-line que exibem ao usuário o que está acontecendo. Uma delas monitora todos movimentos do robô e a outra exibe o que está sendo desenhado no quadro. Estas webcams possuem um software comercial responsável pela atualização das imagens em intervalos pré-estabelecidos. O usuário pode escolher qual das câmeras deseja utilizar em um dado momento. Todas estas etapas são detalhadas a seguir iniciando-se pela interface do browser e finalizando com a programação do controlador.

3.1 A INTERFACE WEB

A interface deve combinar simplicidade e funcionalidade uma vez que pessoas sem nenhum conhecimento mais profundo podem acessar o site. As duas principais funcionalidades existentes são: upload de imagem e monitoramento das tarefas. Inicialmente, o usuário simplesmente submete o arquivo correspondente à imagem a ser desenhada. Depois, esta imagem é exibida em conjunto com a imagem do local para monitoramento dos movimentos do robô. Através de um botão, pode-se alternar entre as duas $webcams$, optando por uma visão lateral do manipulador ou frontal ao quadro. A primeira delas é a ilustrada na Fig. 3.

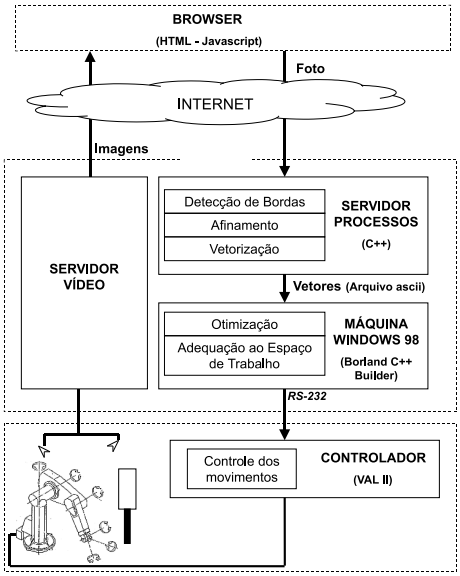

Figura 2. Visão Geral do Sistema.

Figura 3. A interface Web utilizada para teleoperar e monitorar o processo.

3.2 PRODUZINDO CARICATURAS

Uma vez disponível os traços que melhor representam a face de uma pessoa, pode-se executar uma caricatura. Fatores de escala distintos podem ser aplicados em determinados pontos específicos de forma a realçar nariz, boca ou olhos. No presente trabalho, estes fatores de escalas não foram aplicados sob a foto recebida, contudo, algumas características foram extraídas da face da pessoa. Extraiu-se o $esqueleto$ da imagem de forma a representá-la com certa fidelidade. No entanto, a iluminação, cor e outros são fatores decisivos no sucesso desta etapa.

O principal objetivo é obter o $menor$ número de

vetores que possam representar a face da pessoa na imagem recebida. Para isto utilizou-se técnicas de processamento de imagens específicas. Uma delas foi a vetorização que parte de algumas premissas do *esqueleto* da imagem. São elas:

- linhas com espessura menor possível (1 pixel);
- garantia de um e somente um caminho entre dois extremos;
- garantia de fácil determinação de pontos de intersecção de linhas e seus ponto finais.

Estas características podem ser obtidas aplicando-se detecção de bordas seguido de afinamento. Portanto, as técnicas utilizadas foram aplicadas na seguinte ordem: detecção de bordas, afinamento e vetorização. No entanto, O processamento implementado utiliza imagens em tons de cinza, assim, quando necessário a conversão foi efetuada. O *background* da foto deve ser o mais homogêneo possível para que não sejam detectadas bordas não desejadas.

DETECÇÃO DE BORDAS

Os métodos de detecção de bordas são numerosos e bem conhecidos. Em sua grande maioria, são baseados no princípio de que onde há uma variação abrupta na intensidade, é possível que haja presença de bordas. Dentre os métodos de detecção de bordas podem-se citar: Sobel, Roberts, Prewitt e Canny (Castleman, 1995). Foram realizados testes com todos estes métodos. O Canny detecta bordas muito sutis. Como o objetivo é obter as bordas mais relevantes, este método não obteve bom resultado. Dentre os outros, o que apresentou melhor imagem resultante foi o de Sobel. Posteriormente efetuou-se a binarização para eliminar bordas pouco expressivas. A técnica utilizada para determinação do limiar foi calcular a média dos níveis de cinza dos pixels.

Existem ainda métodos que utilizam transformada de Hough (Thecco e Verri, 1998). No entanto, não são indicados para a aplicação uma vez que detectam bordas em orientações preferenciais.

AFINAMENTO

O afinamento consiste em reduzir a imagem ao menor número de pixels capaz de representar seu *esqueleto*. No caso, existem várias técnicas disponíveis na literatura, dentre elas a baseada em operações morfológicas (Castleman, 1995), na transformação do eixo médio (Pavlids, 1982) e métodos iterativos (Zhang e Suen, 1984). É importante frisar que nem todas as imagens podem ser afinadas e os métodos existentes não garantem um bom resultado. A principal diferença dentre os métodos descritos é a complexidade computacional. O que apresenta menor complexidade é o que utiliza operações morfológicas. Contudo, como o tempo de processamento não é fator muito relevante no contexto do trabalho, optou-se pelo método iterativo de Zhang-Suen por ser de mais fácil compreensão e implementação.

Alguns conceitos apresentados como o de conexidade e de números de vizinhos são necessários para entendimento dos algoritmos demonstrados a seguir. Toda referência aos pixels da imagem é ilustrada na Fig. 4.

N_2	N_3	N_4
N_1	N_0	N_5
N_8	N_7	N_6

Figura 4. Referência dos pixels na imagem

- Conexidade $C(N_0)$: o número de transições preto-branco ou vice-versa, na sequência de vizinhos N_1, N_2,....., N_8, N_1. Na verdade é o número de objetos adjacentes ao pixel N_0.
- Vizinhos $B(N_0)$: o número de pixels vizinhos de N_0 diferentes de zero.

Antes da aplicação do método de afinamento, foi efetuado um pré-processamento que é descrito abaixo. Este algoritmo de exclusão de pontos é aplicado em 4 iterações na tentativa de eliminar projeções falsas ou minimizar suas ocorrências.

for todo pixel da imagem **do**
  if ($B(N_0) < 3$) ou ($C(N_0) < 2$)) **then**
    elimina pixel da imagem;
  end if
end for

Então, aplicou-se o afinamento. Basicamente o método consiste de algumas estratégias para escolha de pontos que podem ser excluídos. É importante notar que a exclusão dos pontos só termina quando não se verificar mais nenhum pixel a ser eliminado.

while houver pixel a ser removido **do**
  for todo pixel da imagem **do**
    if ($(2 \leq B(N_0) \leq 6)$ ou $(C(N_0) = 1)$ ou $((N_1 * N_3 * N_5) = 0)$ ou $((N_3 * N_5 * N_7) = 0)$) **then**
      elimina pixel da imagem;
    end if
    if ($(2 \leq B(N_0) \leq 6)$ ou $(C(N_0) = 1)$ ou $((N_1 * N_3 * N_7) = 0)$ ou $((N_3 * N_5 * N_7) = 0)$) **then**
      elimina pixel da imagem;
    end if
  end for
end while

VETORIZAÇÃO

Uma vez atingidas as características necessárias descritas anteriormente, pôde-se aplicar a vetorização sob a imagem afinada. Foi implementada a máquina de estados ilustrada na Fig. 5 para obtenção dos vetores. É necessário a rotulação dos pixels em ponto extremo e intermediário. Pontos extremos são aqueles que exprimem o fim ou início de algum segmento, podendo ser um ponto final, inicial ou de intersecção.

Um ponto é dito extremo se satisfizer a seguinte expressão: $B(N_0) = 1$ ou $C(N_0) > 2$ ou ($C(N_0) = 2$ e $4 < B(N_0) < 7$). Caso contrário, é dito como intermediário. De uma maneira geral, um ponto é extremo

quando for o início ou fim de um dado segmento. Caso os segmentos se cruzem, o ponto de intersecção também é dito como extremo.

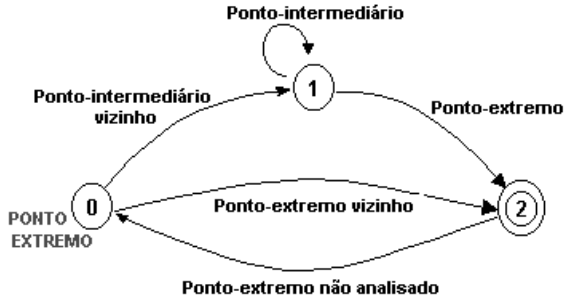

Figura 5. Máquina de estados para vetorização

Assim, uma vez rotulados os pixels, a máquina de estados é aplicada da seguinte forma: Inicialmente, é efetuada uma leitura linha a linha dos pixels da imagem a partir do vértice superior esquerdo. Quando um ponto extremo é identificado, ativa-se o estado 0. Verifica-se a vizinhança deste pixel (N_3, N_4, N_5... N_1e N_2) e caso seja encontrado um ponto intermediário, o estado 1 é ativado. Novamente verifica-se os vizinhos e este estado 1 é reativado ao se encontrar um ponto intermediário e só será desativado com a presença de um ponto extremo, quando atingi-se o estado final 2. Os pontos que foram visitados entre o estado 0 e 2 compreendem um segmento. Percebe-se a necessidade de haver um e somente um segmento entre dois pontos extremos, o que é garantido pelo afinamento e rotulação. Após o estado 2, caso seja encontrado um ponto extremo não analisado, ativa-se o estado 1 novamente. Assim, reinicia-se o processo de identificação de segmentos. É efetuada a identificação da linearidade dos segmentos detectados. Verifica-se a possibilidade de aproximá-los por uma reta ligando os pontos extremos. Evidentemente existe um erro nesta aproximação. Caso este erro não seja aceitável para um dado segmento, a saída é subdividí-lo em subsegmentos e tratá-los separadamente. Esta etapa é ilustrada na Fig. 6. Na coluna à esquerda (Situação 1), simplesmente uniu-se os pontos extremos (A e B) do segmento de curva e obteve-se um vetor com um erro que pôde ser tolerado. Já na outra situação, a união dos extremos não originou um vetor com um erro aceitável. Assim, subdividiu-se o segmento em dois, originando dois vetores (A-C e C-B). O ponto intermediário (ponto C) é o ponto do segmento mais distante da linha reta que une os pontos extremos. As subdivisões podem ser múltiplas, dependendo do erro considerado como aceitável. Em termos práticos, quanto menor o erro aceitável, melhor será a representação vetorial da imagem. Contudo, deve haver um compromisso com a quantidade de vetores gerados.

Finalmente, após a identificação e linearização dos segmentos, são armazenadas em um arquivo as coordenadas inicial e final destes vetores. Elas servirão de referência para os movimentos a serem efetuados pelo manipulador. Cada linha do arquivo representa um vetor composto pelas coordenadas de seu ponto inicial (Pi) e final (Pf).

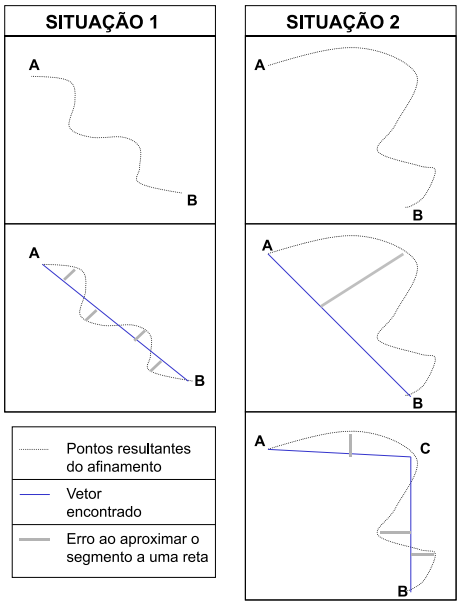

Figura 6. Etapas para identificação dos vetores a partir dos segmentos de curva.

3.3 ADEQUAÇÃO DA VETORIZAÇÃO AO ROBÔ

Todo o tratamento do arquivo contendo os vetores a serem esboçados assim como o sincronismo de envio de dados ao controlador foi efetuado em máquina local. Esta máquina estava conectada via RS-232 com o controlador.

Um dos objetivos desta etapa é minimizar a quantidade de vetores gerados sem comprometer a fidelidade com a imagem original. Uma alternativa encontrada foi eliminar todos os vetores com magnitude (distância entre os pontos inicial e final) inferior a 3 pixels. Outro objetivo é adequar as coordenadas encontradas. Isto requer uma adequação ao espaço de trabalho do manipulador (escala e transformação de referenciais). A área de alcance e pontos de singularidade foram levados em consideração.

Além disso, os vetores foram ordenados para serem enviados ao controlador. Esta sequência foi gerada de tal forma que o braço mecânico não executasse muitos movimentos desnecessários. É importante frisar que este ordenamento não é ótimo, ou seja, nem sempre a trajetória efetuada pelo robô será a menor. Contudo, para o contexto do trabalho, este aspecto não influenciou de forma significativa no tempo gasto para se desenhar os vetores. O algoritmo funciona da seguinte forma: Inicialmente é encontrado uma extremidade de um vetor localizada mais próxima da parte superior esquerda da área de trabalho. Posteriormente, a partir da outra extremidade, verifica-se a presença de uma coordenada inicial ou final num raio de 3 pixels. Caso não seja encontrado, este raio é incrementado até obter sucesso. Ao encontrar um novo ponto extremo, o processo se reinicia.

O envio destes dados ao controlador foi efetuado vetor por vetor. Assim, esta comunicação entre a máquina local e o controlador deve ser sincronizada. Numa primeira etapa, a máquina espera por uma confirmação de que o manipulador já está pronto para a tarefa. Após esta confirmação, um vetor é enviado

para ser desenhado no quadro. O controlador verifica que o desenho deste vetor foi efetuado e avisa que está pronto para receber mais um par de coordenadas. Então, inicia-se novamente o loop, que só é finalizado quando todos os vetores forem enviados.

PROGRAMAÇÃO DO CONTROLADOR

Uma vez especificado o protocolo de comunicação entre o controlador e a máquina local, consegue-se enviar os vetores de forma sequencial. No entanto, um pequeno tratamento dos dados recebidos ainda deve ser efetuado pelo controlador. Ao receber um conjunto de coordenadas (inicial e final), é verificado se o novo ponto inicial é igual ao ponto em que o pincel se encontra. Caso seja verdadeiro, plota-se o novo vetor. Caso contrário, afasta-se o pincel de aproximadamente 5 cm do papel, perdendo o contato com o mesmo. Então, desloca-se o pincel ao novo ponto inicial lido, aproxima-se novamente e esboça-se o novo traço. Este procedimento é repetido até que todos os vetores tenham sido enviados. Uma coordenada negativa indica o fim da transmissão. Esta etapa é ilustrada na Fig. 7.

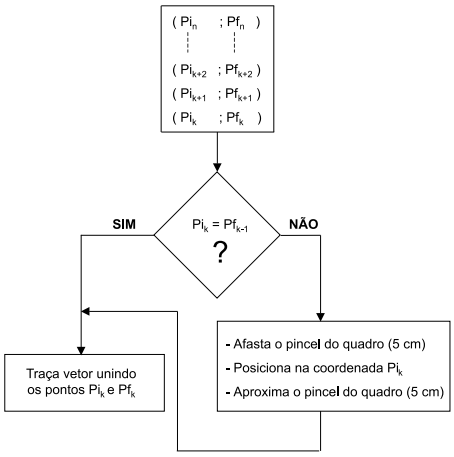

Figura 7. Programação existente no controlador para desenho dos vetores recebidos.

4 RESULTADOS

Como já descrito anteriormente, foi utilizado o robô comercial PUMA 560 que possui 6 graus de liberdade movimentado-se a uma velocidade de 6 in/s. Procurou-se uma região de trabalho onde o pontos de singularidade não acontecessem. A área de trabalho foi definido por um retângulo de 30 cm x 40 cm. O processamento de imagens ocorreu em um máquina Sun Sparc Station 4.0 com 64 Mb de memória RAM rodando Solaris 2.0. O programa de tratamento dos vetores e comunicação com o controlador rodou em uma máquina Pentium II 233 MHz com 64 Mb de memória RAM rodando Windows 98 conectada via RS-232 ao controlador do PUMA 560. Toda a programação foi efetuada em C++ (processamento de imagens), Borland C++ Builder (máquina local) e VAL II (controlador).

Alguns experimentos foram realizados e um resultado da etapa de processamento de imagens é demonstrado em detalhes na Fig. 8. A imagem rotulada é ampliada como pode ser visto na parte inferior da figura. Os pontos extremos estão representados pela cor branca e os intermediários por um nível de cinza intermediário. Como citado anteriormente, o afinamento é um pré-processamento à vetorização e deve atingir algumas características desejadas. Percebe-se que o processamento ocorreu com sucesso originando uma imagem com pixels rotulados da forma desejada. Após o rotulação, aplicou-se o algoritmo de vetorização e aproximação poligonal descrito anteriormente. O tempo gasto para processamento de uma imagem 640 x 480 pixels foi de aproximadamente 70 segundos.

Uma forma encontrada de validar estas etapas foi simular o desenho destes vetores. Utilizou-se o MATLAB para efetuar esta validação gerando as imagens ilustradas na Fig. 9. Na imagem à esquerda todos os vetores encontrados foram desenhados. Na tentativa de minimizar o número de vetores, aqueles que possuíam módulo inferior a um dado valor foram desprezados (imagem à direita). As simulações foram úteis para verificar a validade desta estratégia.

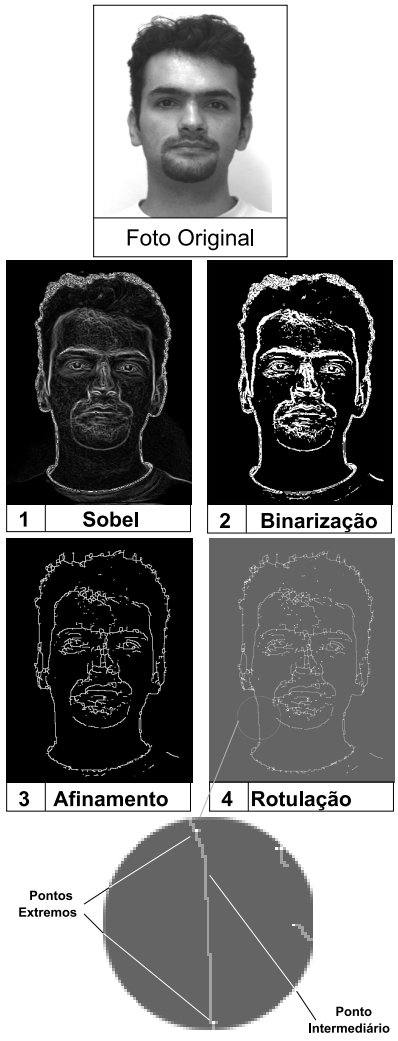

Figura 8. As etapas de necessárias para vetorização dispostas em ordem de processamento.

Algumas caricaturas realizadas pelo robô assim como suas respectivas imagens originais são ilustradas na Fig. 10. Como dito anteriormente, todo o tratamento dos vetores é feito sem intervenção do usuário.

Assim, a primeira caricatura não esboçou o rosto por completo da mulher na foto. Uma forma de resolver este problema é parametrizar o programa de tal forma que o usuário ajuste a área da foto a ser esboçada.

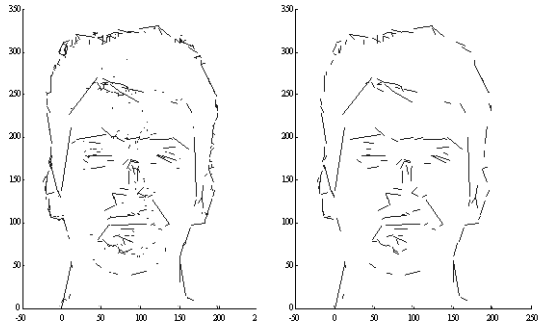

Figura 9. Simulação em MatLab dos desenhos a serem feitos pelo PUMA 560. Houve uma redução de 30% dos vetores.

Figura 10. Caricaturas efetuadas pelo PUMA 560 com suas respectivas fotos originais.

5 CONCLUSÕES

Este trabalho realizou a implementação de uma aplicação para web com a finalidade de possibilitar ao público um contato direto com robôs. De uma maneira geral, os resultados obtidos foram os esperados. Evidentemente, como já era esperado, dependendo da imagem enviada (background, iluminação, qualidade), o desenho não era reproduzido com fidelidade.

Alguns aspectos podem ser aprimorados em algumas etapas do projeto. Os algoritmos implementados não são ótimos, principalmente o algoritmo de vetorização que pode ser aprimorado utilizando operações morfológicas. A interface web pode possuir maior processamento com a utilização de applets em Java. Assim, os vetores poderiam ser gerados no próprio browser, reduzindo os dados trafegados pela rede mundial. Além disso, alguns parâmetros podem ser ajustados pelo próprio usuário (limiar, número de vetores) de forma bastante intuitiva. Um resultado preliminar poderia ser visualizado antes da submissão dos dados para o servidor permitindo um novo ajuste destes parâmetros. Outro futura implementação é uma fila de processos, permitindo que serviço fossem agendados. Um problema encontrado foi o ajuste do quadro de forma que ficasse exatamente no plano do espaço onde a ponta pincel pintava os traço. Sensores de torque e força podem ser acoplados ao pincel de tal forma que estes problemas de compliância sejam ajustados dinamicamente.

Por fim, novos projetos baseados em aplicação web possibilitando a população em geral a ter acesso a robôs ou tecnologias não convencionais podem ser adequados à arquitetura implementada. Além disso, as idéias e conceitos discutidos podem ser aplicados em áreas distintas como automação industrial e cartografia.

Referências Bibliográficas

Castleman, K. R. (1995). *Digital Image Processing*, Pretince Hall Signal Processing Series.

Goldberg, K., Santarromana, J., Bekey, G., Gentner, S., Morris, R., Sutter, C. e Wiegley, J. (1996). A telerobotic garden on the world wide web, *SPIE Robotics and Machine Perception Newsletter*, Vol. 1 of *5*, Canada. Reprinted in: SPIE OE Reports, 150, June 1996.

Ho, T. T. e Zhang, H. (1999). Internet-based telemanipulation, *Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering*, Canada, pp. 1425–1430.

Lloyd, J. e Hayward, V. (1984). Real-time trajectory generation in multi-rccl, *Journal of Robotics Systems*, number 3 in *10*, pp. 369–390.

Pavlids, T. (1982). *Algorithms for graphics and image processing*, Computer Science Press.

Siegwart, R. e Saucy, P. (1999). Interacting mobile robots on the web, *Workshop: Current Challenges in Internet Robotics - ICRA'99*, Detroit, MI, USA.

Simmons, R. (1998). Xavier: An autonomous mobile robot on the web, *Workshop: Robots on The Web - IROS'98*, Canada.

Stein, M. R. (1999). Painting on the world wide web: The pumapaint project, *Workshop: Current Challenges in Internet Robotics - ICRA'99*, Detroit, MI, USA.

Taylor, K. e Trevelyan, J. (1995). Australia's telerobot on the web, *26th International Symposium On Industrial Robots*, Singapore.

Thecco e Verri (1998). *Introductory Techniques for 3D Computer Vision*, Pretince Hall.

Zhang, T. Y. e Suen, C. Y. (1984). A fast parallel algorithm fot thinning digital patterns, *Comminucations of the ACM*, number 3 in *27*, pp. 236–239.