PMC-581 Projeto Mecânico/Mecatrônico II

WebMosaicker - Geração Automática de Imagens Panorâmicas

Carlos Guestrin orientador: Fábio Cozman

Junho 1998

Resumo

O interesse por imagens panorâmicas tem crescido nos últimos quatro anos com a criação de algoritmos que automatizam o processo de geração deste tipo de imagens. Atualmente, este tipo de imagem está sendo utilizado em diversas áreas, como planejamento de missões robóticas e visualização de ambientes.

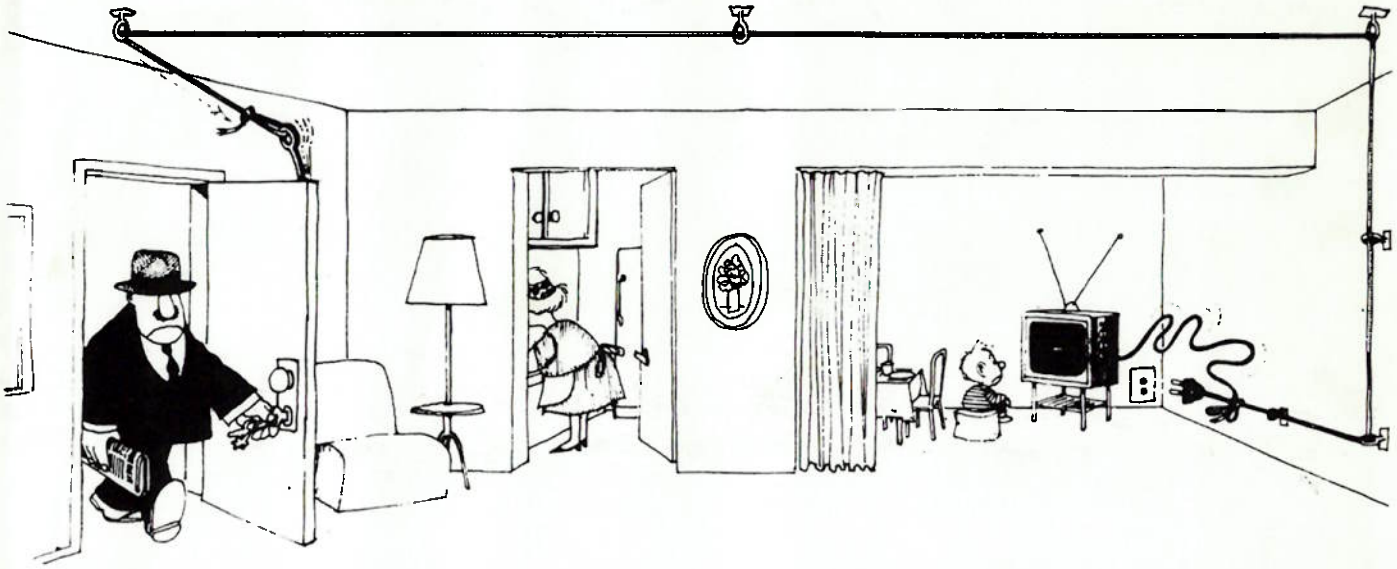
Este projeto visa o estudo e implementação de um sistema de geração automática de imagens panorâmicas a partir de uma sequência de imagens. Os principais pontos abordados neste projeto são: o registro de imagens, o modelo de movimento da câmera, a interface com o usuário e a aquisição de dados via rede.

Este texto apresenta a formulação implementação e teste do WebMosaicker, um sistema para geração automática de imagens panorâmicas. Neste relatório, objetivos do projeto são definidos, o problema que será abordado neste projeto é definido, possíveis soluções são apresentadas e os resultados da implementação de um protótipo do sistema também são apresentados. Além disso, são descritas a escolha e definição da solução final, a sua imaplementação e teste.

O WebMosaicker, implementado em Java, é capaz de ler uma sequência de imagens do disco ou da rede, e criar automáticamente uma imagem panorâmica. Foram utilizados os algoritmos de correlação de fase e de minimização da diferença das imagens para realizar o registro das imagens.

Os resultados foram analisados qualitativa e quantitativamente. Este relatório apresenta a primeira análise quantitativa da qualidade de imagens panorâmicas da literatura. Esta análise foi baseada numa formulação da razão sinal-ruído que criamos para analisar a qualidade de imagens panorâmicas.

A razão sinal-ruído foi de aproximadamente 70, enquanto, o tempo médio de processamento de cada imagem foi de 2 a 4 segundos.


QUINO

Aos meus pais.

Agradecimentos

Ao Professor Fabio Cozman pela sua orientação e ajuda neste projeto e na minha formação acadêmica.

Ao Professor Marcos Barretto pelo seu apoio acadêmico e pessoal.

Aos professores do Departamento de Engenharia Mecânica / Mecatrônica, especialmente os professores Marcelo Godoy Simões, Lucas Moscato e Jun Okamoto Jr., que propiciaram o desenvolvimento do projeto e meu desenvolvimento pessoal e intelectual durante o curso.

Ao Professor Eric Krotkov da Carnegie Mellon University pelas grandes oportunidades que estão fazendo diferença na minha vida.

Aos meus amigos que me apoiaram e tornaram a minha vida agradável nos meus anos de estudo.

A Tania por todo seu apoio e compreensão durante estes longos anos.

A todos aqueles que fizeram uma diferença neste projeto, nos meus estudos e na minha vida.

Conteúdo

1 Introdução

As cameras disponíveis atualmente possuem um campo de visão limitado, chegando até aproximadamente 100° em câmeras com lentes grande angular. Estas lentes, entretanto, causam uma grande distoção na imagem, o chamado efeito "olho de peixe", e diminuem a resolução angular de cada pixel da imagem (pixels por grau). Para obter imagens menos distorcidas e com maior resolução angular, é necessário utilizar lentes de até 60°.

Este projeto visa a criação de um sistema para a geração automática de panoramas a partir de uma sequência de imagens. A figura 1 ilustra o processo de geração de panoramas. Neste relatório, apresentamos a implementação deste sistema e a análise quantitativa dos resultados. Esta é a primeira formulação na literatura de uma análise quantitativa da qualidade da geração de imagens panorâmicas.

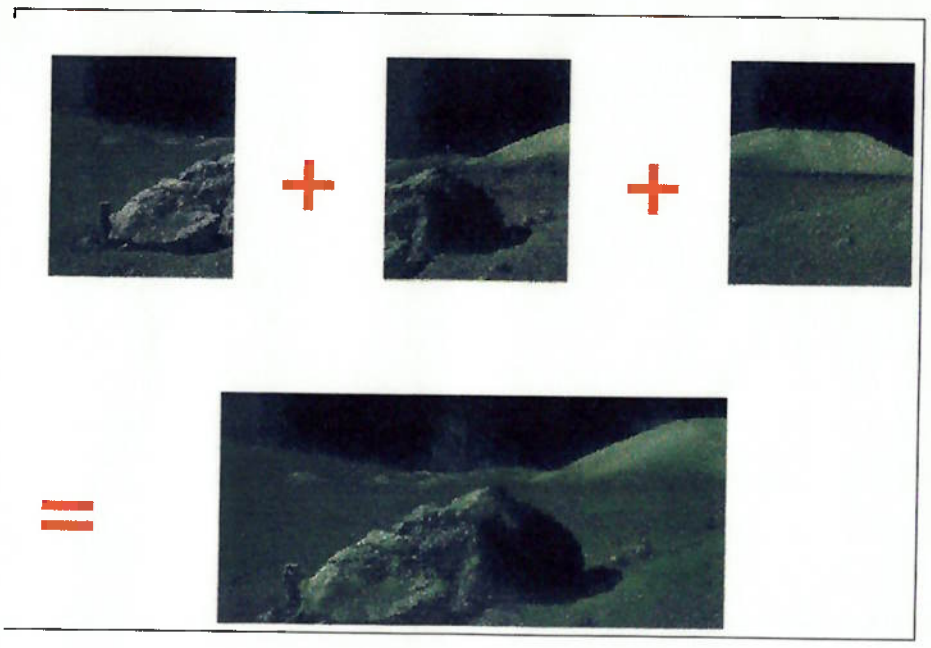


Figura 1: exemplo do processo de geração de panoramas.

Existem diversas aplicações nas áreas de Engenharia Mecatrônica que precisam de imagens de boa qualidade com um maior campo de visão. Frequentemente, é necessário obter imagens de todo o entorno, panoramas de 360°. Atualmente, imagens panorâmicas estão sendo utilizadas em diversas áreas, como por exemplo:

- **vigilância:** uma grande área pode ser vigilada com apenas uma câmera movimentando-se. Em vez do operador monitorar constantemente a câmera, ele pode visualizar um panorama que é atualizado enquanto a câmera se movimenta. Além disso, este enfoque poderia automaticamente disparar um alarme quando há movimentação no ambiente.
- **estimação de movimento em imagens:** enquanto a câmera é movimentada, o fundo é transformado em um panorama. O movimento de objetos nas imagens é obtido a partir resíduo após o fundo é subtraído.
- **estimação visual de posição:** a posição de robôs móveis é obtida a partir de panoramas de 360°. Panoramas são necessários porque uma imagem não contêm uma quantidade suficiente de informação para obter uma estimação precisa [3]. A figura 2 mostra um exemplo de panorama utilizado nesse sistema. A linha do horizonte, em vermelho na figura, é comparada com horizontes reanderizados em mapas para estimar a posição do robô.
- **planejamento de missões robóticas:** com um panorama de 360°, o problema de analisar o ambiente e planejar a atuação de um robô se torna mais simples. Isto foi aplicado na missão da NASA a Marte em julho de 1997. Panoramas foram utilizados para definir quais eram as pedras "interessantes" a ser analizadas, a figura 3 mostra o panorama utilizado nessa missão.
- **visualização virtual de ambientes:** uma única imagem fornece uma ideia limitada dos aspecto de um ambiente. Panoramas, por outro lado,

permitem uma maior imersão, aumentando o impacto de sistemas de visualização virtual. Exemplos incluem a visualização virtual de uma loja ou uma viagem virtual a uma floresta.

Figura 2: Panorama utilizado na estimação de posição de robôs; a linha do horizonte é indicada em vermelho.

Figura 3: Panorama utilizado pela NASA na missão *Mars Pathfinder*.

Este texto descreve um sistema desenvolvido para geração automática de panoramas a partir de multiplas imagens adquiridas rotacionando uma câmera. Inicialmente, os objetivos e especificações deste projeto são apresentadas. Posteriormente, é descrito um experimento realizado, através de uma implementação em AVS. Finalmente, o sistema final desenvolvido em Java é apresentado, incluindo uma análise quantitativa dos resultados.

2 Objetivos

O objetivo principal deste projeto é o estudo e implementação de um sistema para geração automática de imagens panorâmicas a partir de uma sequência de imagens.

Além disso, o sistema desenvolvido deve possuir uma interface gráfica que facilite a interação com o usuário. Afim de suprir as necessidades de

arquiteturas do tipo cliente-servidor, onde dados estao disponíveis em um servidor, esta inteface deve permitir a acquisição e processamento de imagens remotamente, utilizando uma rede de computadores ou a Internet. Além disso, esta interface deve permitir: a leitura de uma sequência numerada de imagens; visualizar o panorama; e configurar o processo de geração de panoramas.

Este projeto visa criar uma base para a geração de imagens panorâmicas. Estes panoramas serão gravadas em formatos padrão, desta forma facilitando a utilização dos resultados deste projeto em projetos futuros. Além disso, deseja-se projetar um sistema que permita a adição de novos componentes, permitindo a posterior utilização do sistema.

Os algoritmos utilizados devem ser capazes de registrar imagens com grandes deslocamentos. Entretanto, um limite máximo de deslocamento de 50% do tamanho da imagem é imposto, pois a maioria dos algoritmos não são capazes de lidar com deslocamentos superiores sem interação com o usuário.

Finalmente, deseja-se realizar uma análise quantitativa dos resultados. Geralmente, os resultados da geração de imagens panorâmicas são analisados qualitativamente. Entretanto, diversos algoritmos de registro de imagens só poderão ser comparados através de uma análise quantitativa. Portanto, é necessário desenvolver uma metodologia de comparação quantitativa dos resultados.

3 Especificação do Problema

O problema de geração de imagens panorâmicas é definido como a criação de uma imagem com grande campo de visão a partir de um conjunto de imagens de campo de visão menor. Os pontos abordados neste trabalho são:

- **registro de imagens:** algoritmos para determinar a correspondência entre (*registrar*) um par de imagens utilizando um dado modelo de movi-

mento de câmera;

- **modelo de movimento de câmera:** modelo matemático representa o efeito do movimento da câmera nas imagens;
- **tipo de panorama:** existem várias formas de projetar as imagens para gerar panoramas. Os tipos mais communs de projeção (tipos de panoramas) são: plano, cilíndrico e esférico;
- **visualização:** a forma que o panorama é visualizado pelo usuário;
- **interface:** definição de uma interface homem-máquina que facilite a utilização do sistema;
- **acesso à rede:** implementação de ferramentas que permitam o acesso a imagens disponíveis na rede.
- **análise quantitativa dos resultados:** formulação de uma metodologia de análise dos panoramas.

Para que este sistema seja útil, o tempo de processamento das imagens deve ser na ordem de segundos. Desta forma, o usuário pode utilizar o sistema em primeiro plano, obtendo os resultados "imediatamente".

Para criar o panorama, todas as imagens devem estar descritas no mesmo sistema de coordenadas. Portanto, deve-se determinar a relação (ou transformação) entre as imagens. Neste projeto, é definido que as imagens devem ser organizadas em sequência. Portanto, para determinar a transformação entre as imagens, basta determinar a transformação para cada par consecutivo de imagens de sequência. Esta transformação é um modelo do movimento da câmera.

O processo de geração de panoramas pode ser dividido nos seguintes passos, que são executados para cada nova imagem:

1. aquisição da imagem;

2. determinação da transformação entre a imagem atual e a anterior dado um modelo de movimento da câmera;

3. adição da imagem ao panorama.

Esta seção do trabalho descreve mais detalhadamente as questões que devem ser abordadas neste projeto.

3.1 Aquisição de Imagens

Este projeto assume que as imagens existem em formato digital. O objetivo é criar um sistema genérico. De acordo com a aplicação em vista, deve-se projetar um método de aquisição de imagens.

Entretanto, o modelo de movimento escolhido impõe condições na maneira que as imagens devem ser adquiridas. Portanto, este projeto deve definir diretrizes para aquisição de imagens.

Outro ponto importante, é o acesso à dados via rede. Este sistema deve possibilitar, de forma transparente, tanto a aquisição de imagens gravadas no disco local do computador, como imagens disponíveis via rede ou Internet.

3.2 Registro das Imagens

Conforme definido anteriormente, deve-se determinar a transformação entre cada par consecutivo de imagens. Esta transformação é um modelo do movimento da câmera. Por exemplo, se a câmera se movimenta sobre um plano perpendicular ao eixo focal, um modelo de translação das imagens seria adequado para representar este movimento.

Deseja-se descrever todas as imagens no mesmo sistema de coordenadas. Portanto, deve-se encontrar a transformação entre a imagem atual e um sistema de coordenadas arbitrário. Este sistema pode estar localizado em qualquer ponto. Para tornar a formulação mais intuitiva, neste trabalho é

assumido que o sistema de coordenadas do panorama coinside com o sistema de coordenadas de uma das imagens.

Desta forma, deseja-se determinar a transformação entre o sistema de coordenadas de cada imagem e o sistema de coordenadas de uma imagem de referência arbitrária. Não é possível comparar cada imagem com a imagem referência, pois elas podem não ter nenhuma intersseção (isto ocorre quando a distância entre as imagens é maior que o campo de visão). Portanto, deve se obter uma transformação relativa entre cada par de imagens e posteriormente compor as transformações.

Portanto, as transformações estimadas devem ser expressadas de uma forma que permita a composição de multiplas transformações. Desta forma, para determinar a transformação entre a imagem atual e uma imagem arbitrária, basta compor as transformações para cada para par. Se a transformação é expressada em coordenadas homogêneas, o problema se torna trivial, basta multiplicar as transformações.

Para cada par consecutivo de imagens, é determinada a transformação $\mathcal{T}_n^{n-1}$ que mapeia cada pixel da imagem anterior $n-1$ na imagem atual n. Para gerar o panorama, deseja-se determinar $\mathcal{T}_n^r$, a transformação que mapeia um pixel na imagem de referência r para o mesmo pixel na imagem atual n. Pode-se obter determinar esta transformação compondo as transformações relativas:

$$\mathcal{T}_n^r = \mathcal{T}_n^{n-1}\mathcal{T}_{n-1}^{n-2}\ldots\mathcal{T}_{r+2}^{r+1}\mathcal{T}_{r+1}^{r}, \text{ para } r \leq n; \quad (1)$$

$$\mathcal{T}_n^r = \mathcal{T}_n^{n+1}\mathcal{T}_{n+1}^{n+2}\ldots\mathcal{T}_{r-2}^{r-1}\mathcal{T}_{r-1}^{r}, \text{ para } r > n. \quad (2)$$

O processo de determinação dos parâmetros do modelo é denominado o *Registro* das imagens. Dependendo da transformação escolhida, este processo pode se tornar uma otimização não-linear nos parâmentros da transformação.

Outro ponto fundamental a ser abordado neste projeto é o máximo deslocamento entre as imagens com o qual o algoritmo pode lidar. No caso deste

projeto, como as imagens serão lidas do disco ou da rede, deseja-se minimizar o número de imagens necessárias para criar o panorama. Desta forma, deseja-se maximizar o possível deslocamento entre as imagens.

3.3 Adição de Imagens ao Panorama

Uma vez determinada a transformação entre o sistema de coordenadas do panorama e o sistema da imagem, o passo seguinte é descrever a imagem no mesmo sistema do panorama. Este processso é denominado *warping* [16].

No caso de translação entre as imagens, basta transladar o sistema de coordenadas. No caso geral, entretanto, o processo é mais complicado.

A transformação $\mathcal{T}_n^r$ mapeia um pixel no sistema de coordenadas da referência para um pixel da imagem n. Portanto, a resposta intuitiva para o problema de *warping* é aplicar a inversa dessa transformação para cada ponto da imagem. Este método, entretanto, não produz um resultado de boa qualidade. No caso geral, a imagem transformada não cai exatamente sobre os pixels da referência. Desta forma, devido a transformação de ponto flutuante para inteiros e a efeitos de mudança de escala, aparecem espaços no panorama que não são ocupados por nenhum ponto.

A solução é percorrer o panorama, aplicando $\mathcal{T}_n^r$ e testando se esta cai em dentro da imagem atual. No caso positivo, o ponto do panorama é atualizado com o valor do pixel da imagem. No caso geral, a transformação não gera um valor inteiro, mas um ponto entre pixels da imagem. Portanto, é necessário realizar uma interpolação (geralmente se utiliza interpolação bilinear).

Percorrer o panorama inteiro é muito caro em termos computacionais. Este problema pode ser solucionado determinando uma região que contém todos os pontos onde a imagem poderia mapear.

Devido a erros no registro das imagens, distorção da lente e, principalmente, a mudanças de iluminação e de abertura do diafragma da câmera,

quando a imagem adicionada ao panorama as intensidades próximas as bordas parecem discontínuas. Este problema deve ser abordado no projeto.

Existe também a questão de como o panorama é armazenado. Uma opção é armazenar o panorama em uma grande imagem. Outra opção é guardar todas as imagens e as transformações e gerar um "ponto de vista" quando desejado.

3.4 Tipos de Panorama

Uma imagem pode ser modelada como um plano no espaço. O tipo de panorama é o modelo utilizado para a projeção do panorama no espaço. O modelo mais simples é o plano, neste caso não é necessária nenhuma pre-transformação da imagem. Existem outros modelos como o cilíndrico, o esférico e a projeção manifold.

3.5 Visualização e Interface

Outros pontos a serem abordados são a questão da visualização e a interface do sistema.

Geralmente, Um panorama é maior que o tamanho do monitor. Para visualizá-lo inteiro, deve-se diminuir a sua resolução. Em alguns casos, não é interessante ver todas as partes ao mesmo tempo, pois não é possível visualizar os detalhes de cada parte. Deste modo, devem ser criadas formas visualizar partes ou todo o panorama.

A interface do sistema é um ponto importante. A interface deve possuir uma interface gráfica que torne mais intuitiva a operação do sistema. Além disso, deve ser criada de forma que o usuário não precise entender os detalhes do processo de geração de panoramas.

4 Possíveis Soluções

Uma pesquisa bibliográfica foi realizada para coletar referências relacionadas a geração de imagens panorâmicas. Nesta seção possíveis soluções para pontos importantes do projeto são apresentadas. Estas soluções são baseadas em referências e em experiência própria nas áreas de geração de panoramas, processamento de imagens e programação.

A escolha de linguagem de programação será realizada nesta parte do projeto. Esta escolha tem influência direta em que passos devem ser seguidos no projeto. Portanto, é fundamental definir este ponto cedo. Posteriormente, a escolha será reavaliada.

O ponto mais importante do trabalho é o registro das imagens. Portanto, mais ênfase será dada ao detalhamento de possíveis soluções para este problema. Este também é o ponto de maior custo computacional, portanto, deve-se buscar um algoritmo eficiente para este ponto.

4.1 Linguagem de Programação

Este ponto será analizado inicialmente, pois a escolha da linguagem tem influência em diversos pontos do projeto.

As questões principais que serão consideradas nesta seção são:

- acesso a rede;
- paradigma de programação;
- interface gráfica;
- dependência de plataforma.

Não serão consideradas linguagens de baixo nível, como linguagem de máquina por não possuirem ferramentas básicas para interfaces e pela complexidade de programação.

Das linguagens que possuem bibliotecas de interface gráfica disponíveis, as mais disponíveis foram selecionadas. Abaixo, essas linguagens são avaliadas de acordo com os critérios descritos acima:

- **C:** no geral, o código fonte é independente de plataforma, entretanto, os binários são específicos. Para acessar a rede, deve-se escreve um driver que será específico à plataforma. Esta linguagem possue muitas bibliotecas gráficas disponíveis como shareware. Também é uma linguagem familiar e utilizada extensivamente.
- **Pascal:** possue as mesmas características de C, mas não utilizada tão extensivamente.
- **Java:** Java utiliza o programação orientada a objetos, o que beneficia a reusabilidade e o desenvolvimento do software. Com esta linguagem é possível ler arquivos através da Internet de forma transparente. As ferramentas para interface gráfica são padronizadas, tornando o programa mais portável. Além disso, o byte-code é independente de plataforma, permitindo rodar o programa em qualquer plataforma que possua um interpretador. Também é possível rodar através de uma "browser" como o Netscape. Por outro lado, os programas nesta linguagem são mais lentos, pois o byte-code é interpretado.
- **C++:** possue características semelhantes a C, mas utiliza o paradigma de programação orientada a objetos. Existem ferramentas que permitem a programação visual, facilitando a criação de interfaces.
- **AVS ou Khoros:** estes ambientes de programação visual permitem a criação de progamas através da ligação de módulos em uma interface gráfica. Isto facilita muito a criação de protótipos. Alguns módulos básicos já estão disponíveis com estes ambientes, outros podem ser criado utilizando a linguagem C. Para o sistema final, entretanto, é necessário criar um programa otimizado e independente de um sistema

de programação visual.

A partir desta análise inicial, se torna evidente que Java possue as características desejadas de independência de plataforma, paradigma de programação mais adequado e disponibilidade de ferramentas para interface gráfica. O ponto mais forte é a acessibilidade a rede. Utilizado Java, a questão de aquisição das imagens através de uma rede se torna trivial.

Para realizar os testes iniciais, quando o acesso a rede não é fundamental, é mais adequada a utilização de um ambiente de programação visual.

Portanto, neste projeto, os testes iniciais serão realizados em AVS e a implementação final será feita em Java.

4.2 Aquisição de Imagens

A linguagem Java possue ferramentas para ler imagens nos formatos JPEG e GIF. Estes são os formatos padrão da Internet. Neste projeto, serão utilizadas imagens neste formato. Conforme colocado anteriormente, a linguagem Java pode ler de forma transparente imagens no disco local e imagens disponíveis via rede.

A princípio, a calibração da câmera é dada como conhecida. Esta calibração é necessária para alguns tipos de panorama, como o cilíndrico e o esférico. Na segunda etapa do projeto, será analizada a viabilidade de criar panoramas com câmeras não-calibradas [15].

4.3 Algoritmos de Registro de Imagens

Nesta seção serão apresentados alguns algoritmos disponíveis na literatura para o registro de imagens [1, 2, 6, 9, 14]. Dois pontos são analisados: o algoritmo propriamente dito e o modelo de movimento da câmera.

4.3.1 Registro Manual

Este é o método mais simples de registro de imagens. O usuário seleciona pontos correspondentes em cada imagem e esta correspondência é utilizada para determinar os parâmetros do modelo.

Para cada ponto selecionado, duas equações podem ser extraidas (uma para o eixo x e outra para o y). Portanto, o número mínimo de pontos é igual a metada do número de parâmetros do modelo. No caso de haver mais pontos que o necessário, é possível realizar uma otimização para determinar qual são os parâmetros que tornam o modelo mais adequado.

Este processo de selecionar manualmente as correspondências é longo e cansativo. Neste projeto, deseja-se utilizar um método que minimize a necessidade de intervensão do usuário.

4.3.2 Registro Baseado em "Features"

Intuitivamente, esta abordagem é uma automatização da abordagem manual. Pontos (ou features) das imagens são selecionados e rastreados através do video, em vez de selecionados manualmente.

Esta abordagem sofre pelas dificuldades de rastrear os pontos na sequência de video. Para poder seguir um ponto, é necessário que a região em torno do ponto tenha textura. Estes algoritmos estão sujeitos a erros em áreas de baixa textura ou de padrões repetitivos.

Além disso, selecionar pontos que potencialmente são bons para ser seguidos é uma terafa difícil. Se um número baixo de pontos é utilizado, o processo é sensível a erros em algum dos pontos. O deslocamento máximo entre as imagens está limitado a o máximo deslocamento que o algoritmo de rastreamento pode lidar.

Por outro lado, o custo computacional desta abordagem é baixo quando comparado com outros existentes. É possível criar um algoritmo robusto e

rápido utilizando esta metodologia. O algoritmo de *Morimoto et al.* é um bom exemplo [9]. Neste algoritmo, cada imagem é dividida em regiões verticais. Para cada região um ponto é selecionado para ser rastreado. A seleção é realizada utilizando um critério de alta magnitude da segunda derivada. O rastreamento utiliza um algoritmo de busca local de modo coarse-to-fine (utilizando uma pirâmide gaussiana).

Outro enfoque semelhante, implementado por *Hansen et al* [5], primeiro calcula o fluxo ótico da imagem. Posteriormente, os parâmetros do modelo de movimento são otimizados para melhor representar este fluxo.

4.3.3 Correlação de Fase

É possível obter a translação entre duas imagens utilizando a diferença de fase da transformada de Fourier das imagens. Este algoritmo foi proposto por *Kuglin e Hines* em [8].

Para criar uma intuição deste algoritmo, considere uma função contínua de uma dimensão. Dadas as funções $f(t)$ e $g(t)$ tais que $g(t) = f(t-a)$, g é uma versão transladada de f, as transformadas de Fourier diferem por um termo exponencial em a:

$$\mathcal{F}[g(t)] = e^{-a\omega j}\mathcal{F}[f(t)] \tag{3}$$

Portanto, a fase (Φ) das transformadas de Fourier diferem por um termo linear em a, a magnitude é invariante a translação:

$$\Phi[g(t)] - \Phi[f(t)] = -a\omega \tag{4}$$

A translação pode ser obtida calculado a transformada inversa de $\mathcal{F}[d(t)]$, onde:

$$\begin{aligned} \Phi[d(t)] &= \Phi[g(t)] - \Phi[f(t)] = -a\omega \\ \|[d(t)]\| &= 1 \end{aligned}$$

A inversa é uma função delta transladada de a:

$$\begin{aligned} \mathcal{F}[d(t)] &= e^{-a\omega j} \\ \Rightarrow d(t) &= \delta(t-a) \end{aligned} \tag{5}$$

Portanto, $d(t)$ é zero em todos os pontos, exceto a, que é a translação desejada.

Este mesmo raciocínio pode ser utilizado no caso de funções contínuas de duas dimensões. Dado que $g(x,y) = f(x-a, y-b)$ então:

$$\mathcal{F}[g(x,y)] = e^{-(a\omega_1 + b\omega_2)j}\mathcal{F}[f(x,y)] \tag{6}$$

Neste caso, a função $d(x,y)$ é definida:

$$\begin{aligned} \Phi[d(x,y)] &= \Phi[g(x,y)] - \Phi[f(x,y)] = -(a\omega_1 + b\omega_2) \\ \|\mathcal{F}[d(x,y)]\| &= 1 \end{aligned}$$

Portanto:

$$\begin{aligned} \mathcal{F}[d(x,y)] &= e^{-(a\omega_1 + b\omega_2)j} \\ \Rightarrow d(x,y) &= \delta(x-a, y-b) \end{aligned} \tag{7}$$

Um método análogo pode ser utilizado no caso de funções discretas bidimensionais, como imagens [4]. Portanto, para um par de imagens, é possível utilizar a diferença de fase para determinar a translação.

No caso de imagens ou outros tipos de dados de sensores, g não é uma copia perfeita ransladada de f. Neste caso, $d(x,y)$ não terá um único valor não-zero. A estimação da translação é feita determinando o máximo de $d(x,y)$. Iste efeito é especialmente prominente no caso de imagens, onde ruído, mudanças de iluminação, distorção da lente, efeito da discritização e translações de valor não-inteiro acentuam a diferença entre as funções.

A grande vantagem deste método sobre os de otimização iterativa é que a estimação obtida com *Correlação de fase* é global. Deste modo, não está sujeita a mínimos locais. Além disso, este método pode determinar translações de até 50% do tamanho da imagem.

O modelo rígido de movimento da câmera considera translação e rotação em torno do eixo ótico. É possível utilizar a transformada de Fourier para determinar os parâmetros deste modelo. Como a magnitude é invariante a translação é possível determinar a rotação no espectro de magnitude sem considerar a rotação. Posteriormente, pode-se determinar a translação no espectro da fase, compensando pela rotação estimada.

4.3.4 Minimização da Diferença entre as Imagens

O registro de imagens pode ser modelado como a minimização da diferença de intensidade entre as imagens [14]. Intuitivamente, no caso de imagens transladadas, este método é equivalente a mover uma imagem sobre a outra até que a diferença entre a área em comum seja mínima.

É comum formular este tipo de problema como a soma do quadrado da diferença de intensidade:

$$\begin{aligned} E^2 &= \sum_{x,y} [I_{t+1}(u,v) - I_t(x,y)]^2 \qquad (8) \\ \text{onde} \quad (u,v) &= \mathcal{T}[(x,y)] \end{aligned}$$

A transformação $\mathcal{T}$ leva o sistema de coordenadas de t para o de $t+1$. Esta tranformação é um modelo do movimento da câmera. Usando esta formuação o problema de registro de imagens é equivalente a determinar os parâmetros da transformação $\mathcal{T}$ que minimize E^2.

Geralmente, é utilizado um método iterativo de otimização para realizar esta minimização[14]. Estes métodos podem cair em um mínimo local, em vez do global desejado. Uma forma de lidar com este problema é utilizar um

método coarse-to-fine sobre uma pirâmide Gaussiana [13]. Outra técnica é fazer o coarse-to-fine no espaço de modelos, começando com um modelo mais simples (como translação) e posteriormente refinando a solução com modelos mais detalhados (como o projetivo) [7].

4.3.5 Modelos de Movimento da Câmera

Existem muitos modelos na literatura para o movimento da câmera [7, 9, 14]. A diferença entre estes modelos é o tipo de condição imposta no movimento da câmera. Em qualquer caso, para criar uma imagem panorâmica é necessário que o centro ótico da câmera não se mova, isto evita problemas de paralax. Estes são os modelos mais comuns (é importante notar que a origem do sistema de coordenadas da imagem deve estar no centro da imagem):

1. **Translação:** só existe translação sobre o plano da imagem:

$$\begin{aligned} u &= x + \delta_x \ , \\ v &= y + \delta_y \ . \end{aligned} \tag{9}$$

2. **Rígido:** uma transformação rígida bidimensional do plano da imagem, isto é, translação e rotação em torno do eixo ótico:

$$\begin{aligned} u &= x\cos\theta - y\sin\theta + \delta_x \ , \\ v &= x\sin\theta + y\cos\theta + \delta_y \ . \end{aligned} \tag{10}$$

3. **Similaridade:** o parâmetro extra representa uma mudança de escala da imagem:

$$\begin{aligned} u &= \mathcal{S}(x\cos\theta - y\sin\theta) + \delta_x \ , \\ v &= \mathcal{S}(x\sin\theta + y\cos\theta) + \delta_y \ . \end{aligned} \tag{11}$$

4. **Afim:** este modelo permite mudanças de escalas independentes para cada eixo e mudança do ângulo entre os eixos:

$$\begin{aligned} u &= m_0 x + m_1 y + m_2 \ , \\ v &= m_3 x + m_4 y + m_5 \ . \end{aligned} \tag{12}$$

5. **Projetivo:** a relação geral entre de mapeamento entre dois planos:

$$\begin{aligned} u &= \frac{m_0 x + m_1 y + m_2}{m_6 x + m_7 y + 1} \ , \\ v &= \frac{m_3 x + m_4 y + m_5}{m_6 x + m_7 y + 1} \ . \end{aligned} \tag{13}$$

6. **Pseudo-Projetiva:** este é uma simplificação do projetivo que é menos intenso computacionalmente:

$$\begin{aligned} u &= m_0 x + m_1 y + m_2 + m_6 x^2 + m_7 xy \ , \\ v &= m_3 x + m_4 y + m_5 + m_6 xy + m_7 y^2 \ . \end{aligned} \tag{14}$$

O modelo projetivo é o mais geral. Entretanto, a escolha do modelo depende da aplicação e das condições de aquisição de imagens.

4.4 Tipos de Panorama

O tipo de panorama é a superfície na qual o panorama esta projetado. De acordo com a escolha, o modelo de movimento tem significado diferente. O tipo de panorama deve ser escolhido de acordo com a aplicação. Estes são os tipos mais comuns de panorama:

- **Plano:** o panorama é um plano no espaço, conforme mostra a figura 4. A translação no modelo corresponde a uma translação sobre este plano e rotação tem o eixo de rotação perpendicular ao plano. Este tipo de panorama tem um campo de visão teórico máximo de 180°. Na prática o campo de visão é muito menor pois a escala das imagens é proporcional

a tangente do ângulo entre o vetor perpedicular ao plano e o eixo ótico da câmera, esta tangente tende a infinito com ângulos próximos a 90°. A melhor aplicação deste tipo de panorama é a criação de panoramas de objetos planos como quadros ou murais. A vantagem deste tipo de panorama é que não há distorção das imagens.

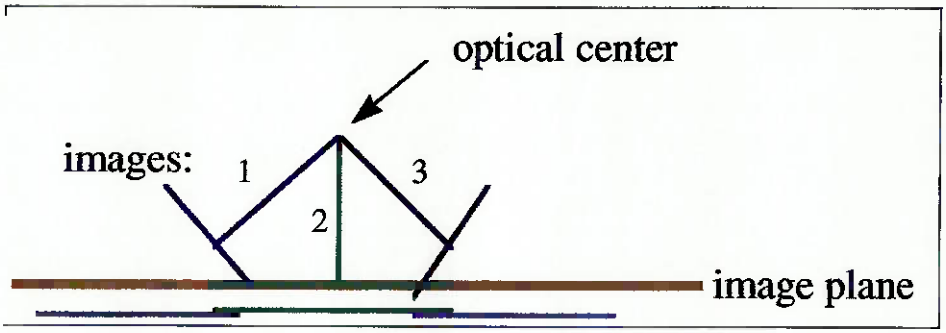


Figura 4: formação do panorama plano.

- **Cilíndrico:** as imagens são projetadas sobre um cilíndro logo após a aquisição. O eixo do cilíndro está sobre o centro ótico e tem direção vertical. Portanto, uma rotação da câmera em torno do eixo vertical é equivalente a uma translação das imagens. Com este modelo é possível obter panoramas de 360° na horizontal.
- **Esférico:** logo após a aquisição, as imagens são projetadas em uma esfera com centro no centro focal da câmera e raio igual a distância focal, conforme mostra a figura 5. Rotações da câmera em elevação ou azimuth são equivalentes a translação do modelo. Neste caso é possível obter um panorama de 360° na vertical e na horizontal.

No caso dos panoramas esférico e cilíndrico, as imagens são distorcidas durante a projeção inicial. Portanto, para visualizar um pedaço do panorama como se fosse uma imagem, deve-se compensar a distorção.

Existe também o panorama utilizando projeção manifold [10]. Este não foi considerado, pois assume pequenas translações entre as imagens, o que

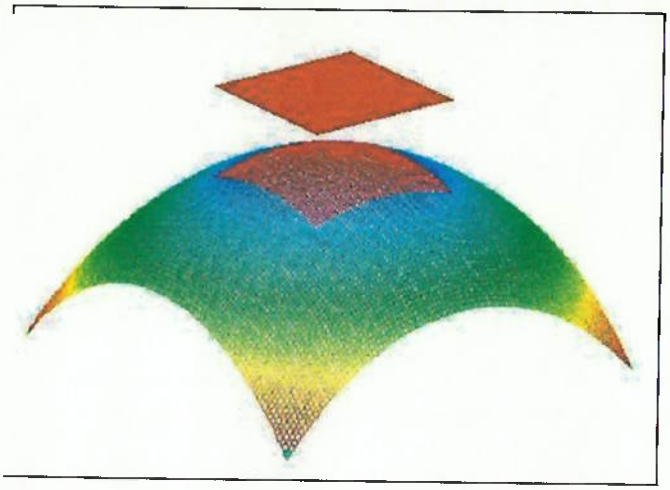


Figura 5: formação do panorama esférico.

não se aplica neste projeto.

5 Experimentos Iniciais

Um protótipo deste projeto foi criado em AVS. Esta seção descreve a implementação e os resultados obtidos com este protótipo.

5.1 Implementação em AVS

AVS (Advanced Visual Systems) produz um ambiente de programação visual de mesmo nome. Este ambiente permite criar programas ligando módulos em um fluxograma. O AVS vem com módulos básicos, principalmente módulos de visualização. O usuário pode criar os seus próprios módulos utilizando a linguagem C. Além disso, é possível criar uma interface gráfica utilizando bibliotecas disponíveis.

A figura 6 ilustra uma *Network*, um conjunto de módulos, criada para gerar imagens panorâmicas. Apenas o módulo *image viewer* é um módulo

padrão do AVS. Os outros foram implementados para testar as ideias deste projeto.

Figura 6: Network de módulos criada em AVS para geração de panoramas.

O sistema é interativo. O usuário seleciona uma nova imagem. Inicialmente, o sistema determina a transformação entre esta imagem e a anterior, colocando a imagem na posição correspondente no panorama. O usuário pode aceitar a posição escolhida, rejeitar a imagem, ou recalcular. No caso de recalcular, o usuário manualmente move a imagem até a posição desejada, podendo forçar a imagem a ficar nesta posição ou mandar o sistema estimar novamente a transformação utilizando esta nova posição como estimativa inicial.

5.2 Aquisição de Imagens

O sistema desenvolvido cria panoramas do tipo esférico. Portanto, as imagens devem ser projetadas em uma esfera logo após a aquisição.

O módulo *Calibration Parameters* contém os parametros de calibração da câmera. Estes parâmetros foram obtidos utilizando o programa de Luc

Robert para calibração de câmeras [12]. O módulo *Image Input* utiliza estes parâmetros para projetar cada nova imagem sobre a esfera. A projeção esférica utiliza um sistema de coordenadas no centro da imagem:

$$\begin{aligned} u &= x + x_{centro} \ , \\ v &= y + y_{centro} \ . \end{aligned} \tag{15}$$

A projeção transforma uma coordenada (u, v) em (θ, α):

$$\theta = \arctan \frac{u}{m_u} \ , \tag{16}$$

$$\alpha = \arctan \frac{v}{\frac{m_v}{m_u}\sqrt{m_u^2 + u^2}} \ , \tag{17}$$

onde m_u e m_v são as magnificações em cada eixo.

Os parâmetros x_{centro}, y_{centro}, m_u e m_v devem ser determinados através da calibração da câmera.

5.3 Registro de Imagens

Cada nova imagem é registrada com a anterior. O módulo *Mosaic Control* contrala este processo e contém a interface gráfica para as interações descritas anteriormente.

Inicialmente, o módulo *Mosaic Control* aloca uma imagem grande, maior que o tamanho máximo esperado do panorama. A primeira imagem é posicionada no centro desta imagem. Quando a posição da imagem a ser adicionada é maior que o tamanho do panorama, a imagem aparecerá do lado oposto. A figura 7 mostra um exemplo típico de panorama gerado desta forma.

O registro é realizado em dois passos. No primeiro passo, um estimativa de translação é obtida utilizando correlação de fase. Este processo é realizado

Figura 7: Resultado típico antes de aplicar a condição de corespondência.

numa imagem de um oitavo do tamanho inicial, pois o custo computacional deste método é alto. O módulo *Phase Mosaic* é responsável por este passo.

Posteriormente, o quadrado da diferença entre as imagens é minimizado utilizando o método de Levenberg-Marquardt para otimização não linear [11]. O módulo *LMM Mosaic* faz esta otimização utilizando um modelo de movimento genérico. Os modelos translação, rígido e afim foram testados.

A interação com o usuário foi essencial, pois assim é possível corrigir erros de registro enquanto o panorama é criado.

Experimentos foram realizados recuperar o modelo rígido através da transformada de Fourier. Este método mostrou-se muito sensível a variações de iluminação e ruído.

5.4 Adição de Imagens ao Panorama

Conforme colocado na seção 3.3: devido a erros no registro das imagens, distorção da lente e, principalmente, a mudanças de iluminação e de abertura do diafragma da câmera, quando a imagem adicionada ao panorama as intensidades próximas as bordas parecem discontínuas.

Neste protótipo, o problema das discontinuidades é abordado. Quando as imagens são adicionadas ao panorama, uma média ponderada entre o valor já existente no panorama (se não for nulo) e a imagem é calculada nas regiões próximas as bordas da imagem. Os pesos da média podenrada são proporcionais a distância da borda. Quanto mais próximo da borda, maior é o peso do panorama.

Utilizando este processo, foi possível minimizar essas discontinuidades.

5.5 Condição de Correspondência

Como uma condição adicional na geração de panoramas, quando o panorama atije 360°, o sistema pode usar a informação da posição de pontos correspondentes em cada ponta do panorama para determinar que parte da imagem constitue os 360°.

O usuário deve selecionar um ponto em cada ponta do panorama que represente o mesmo ponto do espaço. A figura 8 ilustra este processo.

Figura 8: Seleção de pontos correspondentes na imagem.

Com esta informação, o sistema cria um panorama contínuo, figura 9. Caso desejado, o usuário pode forçar que estes pontos tenham a mesma altura. No caso geral, isto não diminue os erros no registro das imagens, mas tem um efeito visual desejável.

Figura 9: Resultado final, o panorama é contínuo.

O módulo *Wrap Mosaic* realiza todo este processo.

6 Especificação da Solução Final

Nesta etapa, serão definidas as características da solução final. As soluções escolhidas estão baseadas na especificações possíveis soluções e experimentos iniciais apresentados anteriomente. A solução final tem as seguintes características:

- **registro de imagens:** uma estimativa inicial da translação entre as imagens é obtida utilizando a correlação de fase. Posteriormente, esta solução é refinada utilizando o algoritmo de Levenberg-Marquardt para minimizar a diferença de intensidade entre as imagens. Este algoritmo é aplicado numa metodologia coarse-to-fine no espaço da imagem, utilizando um pirâmide Gaussiana. Além disso, a metodologia coarse-to-fine também é aplicada no espaço do modelo de movimento, sendo que primeiro a otimização é realizada para um modelo de translação e, posteriormente, é utilizado um modelo mais detalhado, como o rígido.
- **verificação do registro pelo usuário:** o usuário deve ter a possibilidade de intervir, caso não esteja satizfeito com a qualidade do registro. Neste caso, ele deve selecionar um ponto correnspondente nas duas imagens. Este será utilizado como estimativa inicial no algoritmo de minimização da diferença das imagens.
- **tipo de panorama:** o sistema desenvolvido deve ser capaz de criar panoramas esféricos e planos. Os panoramas esféricos necessitam dos parâmetros de calibração da câmera. O programa deve permitir que o usuário digite os parâmetros. Existem algoritmos para recuperar os parâmetros a partir de um par de imagens. Futuramente, deseja-se adicionar esta funcionalidade. Portanto, deve-se prever isto no projeto do software.
- **visualização:** para visualizar a imagem panorâmica o programa deve ter uma janela que mostre o panorama enquanto este esta sendo construido.
- **linguagem de programação:** decidiu-se utilizar Java para implementar a versão final.
- **formato das imagens:** as imagens devem ser lidas nos formatos *gif* e *jpg*.

- **formato de gravação do panorama:** para permitir a utilização do panorama em outros programas. Portanto, o panorama deve ser gravado no formato *jpg*.
- **acesso à rede:** o programa deve ser capaz de acessar imagens através da rede. Estas imagens devem estar disponíveis na rede e serão acessadas utilizando o protocolo *http*.
- **remover a última imagem:** o sistema deve permitir a remoção de imagens do panorama na ordem inversa a que foram adicionadas.
- **configurar o sistema:** discretização do panorama, suavização na adição de imagens e interpolação são algumas das propriedades que devem permitir a configuração do usuário.
- **medida quantitativa da qualidade do panorama:** será realizada uma avaliação quantitativa da qualidade do registro das imagens no panorama. Para isto, será utilizada a razão sinal ruído.

7 Implementação Final do WebMosaicker

Nesta seção são apresentados alguns detalhes de implementação do WebMosaicker utilizando a linguagem Java. A figura 10 ilustra a interface do sistema implementado. Este é composto de uma janela principal, onde é realizada a interação com o usuário, e uma janela secundária, onde é visualizado o panorama.

7.1 Selecionando o Tipo de Panorama

Ao criar um novo panorama, o usuário pode selecionar o tipo de panorama, entre calibrado (esférico) ou não calibrado (plano). No caso do panorama esférico, é necessário digitar os parâmetros de calibração da câmera. Há uma

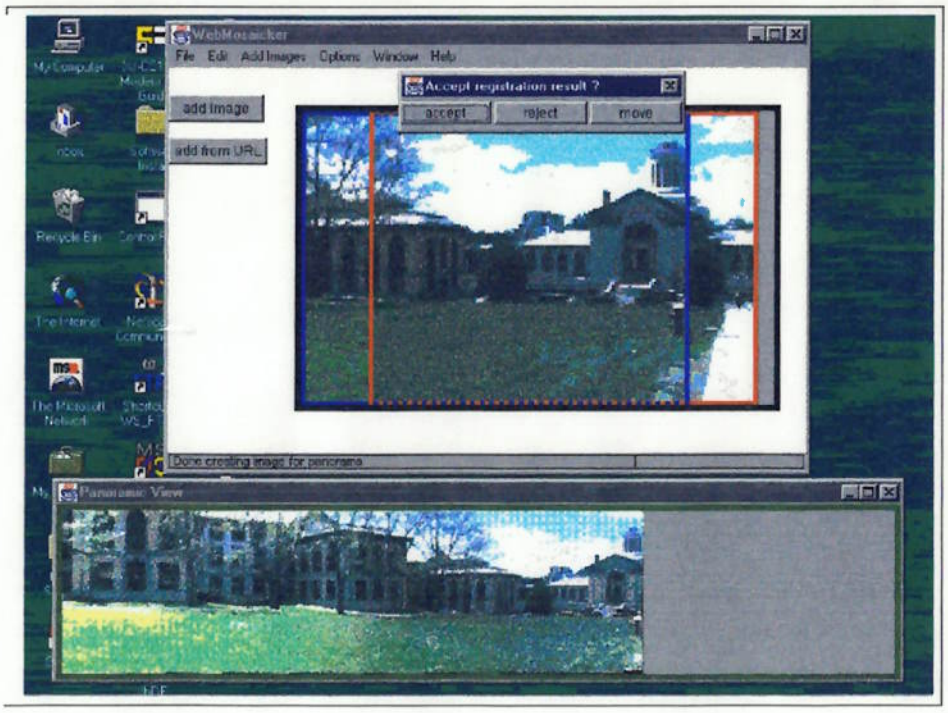


Figura 10: interface do WebMosaicker.

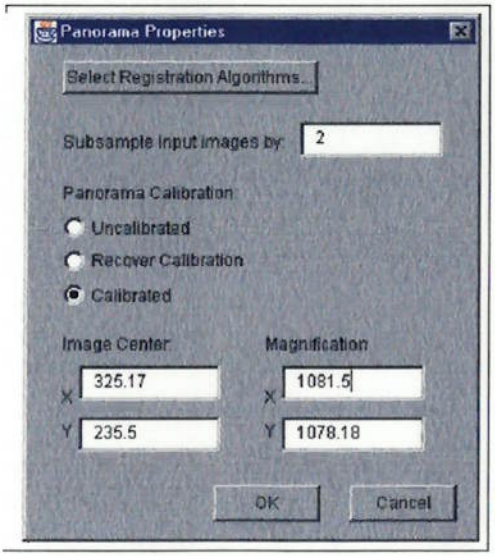


Figura 11: opções do WebMosaicker na criação de um novo panorama.

opção de recuperar os parâmetros. Esta opção está desabilitada atualmente, mas poderá ser imaplementada futuramente.

Outras opções para um novo para um novo panorama incluem: configuração do algoritmo de registro e a discretização das imagens. A figura 11 mostra a janela de opções da criação de um novo panorama.

7.2 Adicionando Imagens ao Panorama

O sistema implementado é capaz de ler imagens do disco ou da rede, a partir de uma *URL*, como mostra a figura 12. O sistema é capaz de ler imagens nos formatos *gif* e *jpg*.

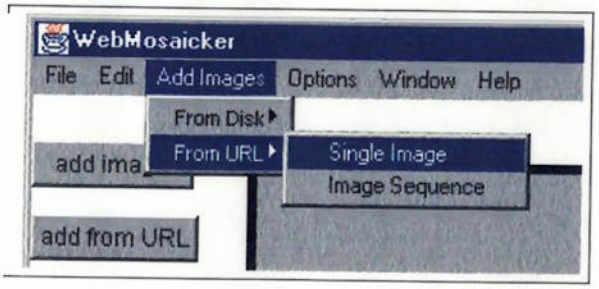


Figura 12: o WebMosaicker pode ler imagens únicas ou sequências numeradas, do disco ou a partir da rede utilizando uma URL.

Além disso, é possível ler uma sequencia numerada de imagens, tanto do disco como de uma URL. A figura 13 mostra a interface para realizar a leitura de uma sequencia de imagens. O nome padrão da imagem é digitado, substituindo # na posição onde será substituida a sequência numerada. A sequência vai do número inicial ao número final (inclusive), com o passo definido por *step*.

Após a imagem é carregada do disco ou da rede, ela armazenada na memória no formato padrão do Java para imagens: a classe *java.awt.Image*. Posteriormente, a imagem é registrada com a anterior. No caso da primeira

imagem, apenas o pre-processamento do registro é realizado e ela é colocada no centro do panorama.

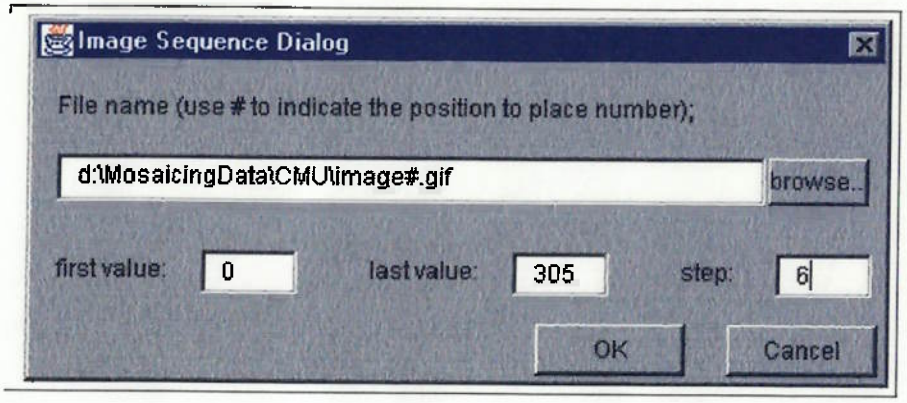


Figura 13: janela do WebMosaicker para ler uma sequência numerada.

7.3 Algoritmos de Registro das Imagens

O sistema implementado combina dois algoritmos de registro das imagens: a correlação de fase e a minimização da diferença das imagens.

A primeira estimativa de translação entre as imagens é obtida utilizando o algoritmo de correlação de fase. Este algoritmo, descrito anteriormente, foi implementado utilizando o algoritmo de Fast Fourier Transform (FFT) [11].

A minimização da diferença entre as imagens utiliza o algoritmo de Levenberg-Marquardt para otimização não-linear [11]. Este algoritmo é utiliza a estimativa de translação da correlação de fase como estimativa inicial da otimização. A otimização é realizada numa metodologia coarse-to-fine em uma pirâmide gaussiana [13]. Cada nível é composto de uma imagem com a metade da resolução do nível anterior, do gradiente na direção horizontal e na vertical. Um nível da pirâmide está ilustrado na figura 14. A otimização é iniciada pelo nível de menos resolução e o resultado é utilizado como estimativa inicial para o nível siguinte. Uma metodologia coarse-to-fine também é aplicada no

espaço do modelo de movimento. Inicialmente, é realizada uma otimização com o modelo de translação. Posteriormente, a solução é refinada utilizando o modelo rígido.

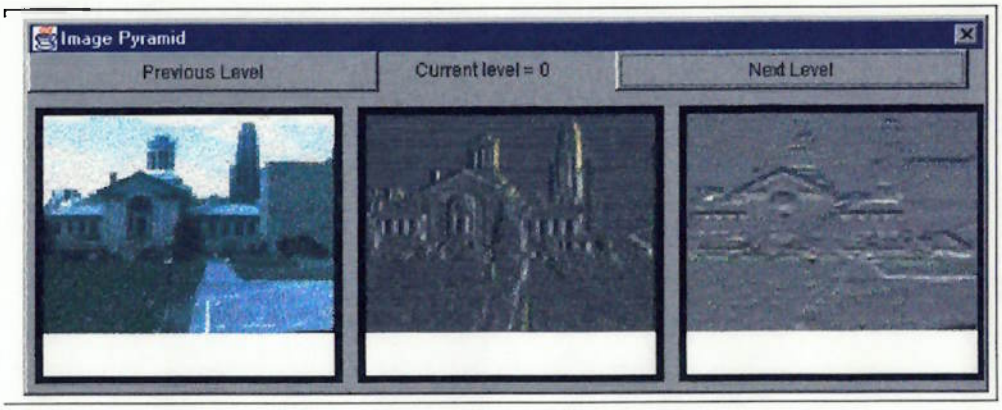


Figura 14: nível 0 da pirâmide da image utilizada no algoritmo de minimização da diferença de intensidade. Da esquerda para a direita: image, gradiente horizontal, gradiente vertical.

7.4 Interface de Java para Algoritmos de Resgistro de Imagens

O WebMosaicker foi imaplementado de modo que os algoritmos de registro de imagens tenham que implementar a interface *RegistrationAlgorithm*. Isto permite a simples adição de novos algoritmos, basta que estes implementem a interface *RegistrationAlgorithm*.

A interface é definida na figura 15. O método *resetRegistration* é chamado quando a primeira imagem é adicionada. O método *doRegistration* é chamado para cada imagem subsequente. Este método deve retornar a trasformação que leva um pixel da imagem anterior para um da imagem atual. A tranformação deve ser retornada como instancia da classe *Transformation*. O

```
import java.awt.*;
import java.awt.image.*;

public interface RegistrationAlgorithm
{
    public void resetRegistration( int x0, int y0,
                                   int w, int h, ColorModel model,
                                   int pixels[], int off, int scansize);

    public Transformation doRegistration( int x0, int y0,
                                          int w, int h,
                                          ColorModel model,
                                          int pixels[], int off,
                                          int scansize,
                                          Transformation initial_estimate);

    public void setStatusBar(StatusBarObject statusbar);

    public String toString();
}
```

Figura 15: interface *RegistrationAlgorithm*.

método *setStatusBar* passa uma referência da barra de status do WebMosaicker para o algoritmo informar o seu status. Finalmente, o método *toString* deve retornar uma *String* com o nome do algoritmo.

7.5 Verificação da Qualidade do Registro das Imagens

Após o registro da imagem atual, o sistema pergunta ao usuário se está satizfeito com a qualidade do registro, como mostra a figura 16. Neste caso, o usuário pode *aceitar*, a imagem é adicionada ao panorama, *rejeitar*, a imagem é removida, ou *mover*, neste caso, o usuário pode intervir no processo de registro das imagens.

Quando o usuário decide mover a imagem, a janela ilustrada na figura 17 é apresentada. O usuário, então, seleciona um ponto correspondente na imagem atual e na anterior. A partir desta correspondência, uma estimativa de translação entre as imagens é determinada:

$$\begin{aligned} \delta x &= P_{2,x} - P_{1,x} \quad ; \\ \delta y &= P_{2,y} - P_{1,y} \quad . \end{aligned} \tag{18}$$

Esta estimativa é então utilizada como estimativa inicial no minimização da diferença nas intensidades.

O usuário tem a opção de desabilitar esta verificação. Neste caso, a imagem é adicionada ao panorama diretamente após o registro automático.

7.6 Formato Interno do Panorama

Internamente, o programa guarda um vetor (*java.util.Vector*) com as imagens processadas pelo sistema imagens. Cada elemento, membros da classe *TransformedImage*, do vetor é composto de:

- uma imagem (*java.awt.Image*), projetada numa esfera no caso de panoramas esféricos;

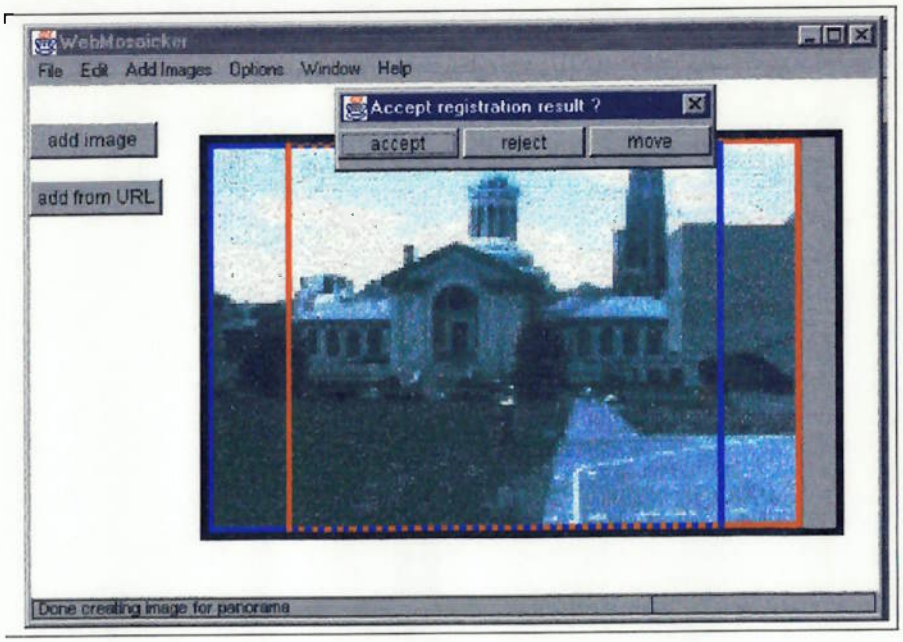


Figura 16: o usuário tem a opção de intervir no registro das imagens.

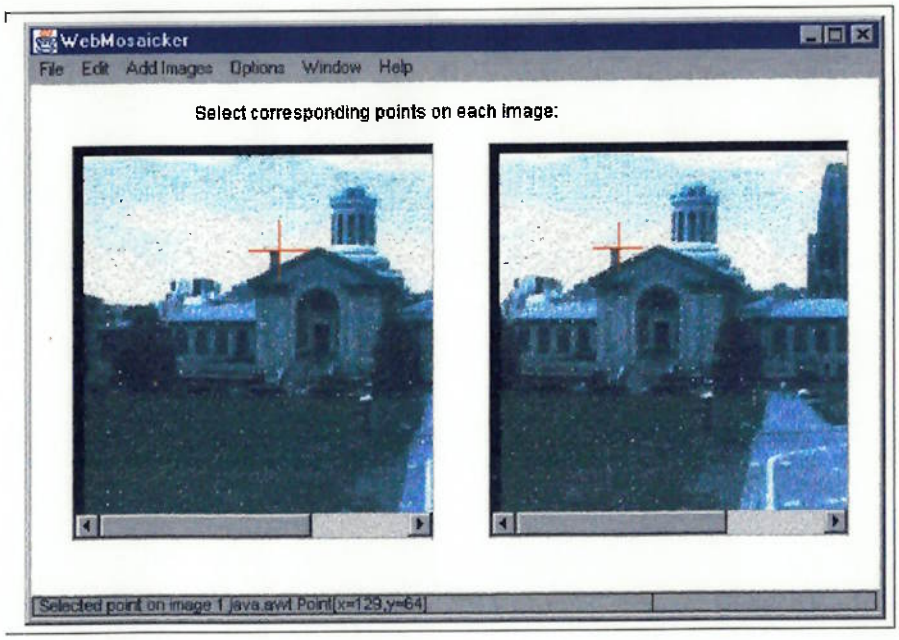


Figura 17: o usuário pode selecionar pontos correspondentes nas imagens. Estes são utilizados como estimativa inicial no registro das imagens.

- a transformação relativa entre esta imagem e a anterior no vetor;
- a transformação absoluta entre o centro de coordenadas da primeira imagem e a imagem atual. Esta transformação é formada pela multiplicação de todas as transformações relativas.

Este vetor é armazenado na classe *Panorama*. Além do vetor, a classe contém os parâmetros de calibração da câmera e outros parâmetros de configuração do sistema. Esta classe é a responsável pela manipulação das imagens que entram no panorama e do vetor de imagens. Um instância nova desta classe é criada no WebMosaicker sempre que um novo panorama é criado.

Por outro lado, o WebMosaicker contém um membro da classe *PanoramicImage*. Esta classe permite criar uma imagem panorâmica única a partir de um vetor de elementos de *TransformedImage*. A imagem é armazenda em uma matriz de inteiros presente nesta classe, podendo ser transformado em uma imagem do tipo *java.awt.Image* para mostrá-la ao usuário ou para gravá-la em formato *jpg*.

A seguir será descrito o processo realizado na criação do panorama visualizado na janela *PanoramicView* de visualização do panorama. Para cada nova imagem, após o registro ser aceito pelo usuário, o seguinte procedimento

é adotado:

O tamanho da matriz de inteiros é suficiente para colocar a nova imagem ?	
SIM	NÃO
adiciona a nova imagem na matriz;	aloca uma matriz cujo tamanho é maior de duas vezes o tamanho da imagem de entrada na direção do movimento , adiciona todas as imagens na nova matriz;

Portanto, a matriz de inteiros que representa a imagem panorâmica aumenta sempre na direção do movimento, quando necessário. Deste modo, não é necessário adotar a condição de correspondência descrita na seção 5.5 dos experimentos iniciais.

A desvantagem deste procedimento é que, quando o tamanho da matriz de inteiros não é suficiente, o usuário deve esperar até que todas as imagens são adicionadas a nova matriz. Este processo pode demorar 20 segundos quando o panorama é grande.

Por outro lado, este procedimento é vantajoso, pois evita a necessidade da condição de correspondência. Como todas as imagens originais são guardadas. não há perda de informações e o panorama pode ser gerado em qualquer ponto do processo.

7.7 Suavização na Geração do Panorama

Ao criar a imagem panorâmica, as imagens são adicionadas a uma matriz que representa o panorama. O método mais simples de realizar esta adição é de substituir os pixels da matriz pelos pixels da nova imagem, na posição

correspondente, como mostra a figura 18. Entretando, aparecem bordas de diferença de intensidade nas bordas das imagens. Isto é causado pelo ajuste automático de ganho da câmera, fazendo com que algumas imagens sejam mais escuras que outras.

Figura 18: Imagens adicionadas sem suavização.

Figura 19: Imagens adicionadas com suavização.

Uma solução para este problema é realizar uma suavização da região de borda. O resultado está ilustrado na figura 19. Este processo foi implementado realizando uma média ponderada entre a imagem anterior e a atual na região da borda. Os pesos variam linearmente com a distância da borda. Na borda, todo o peso é dado a imagem anterior. Quando a distância é maior ou igual a translação, o peso é zero para a imagem anterior e um para a imagem atual. Como há translação na vertical e na horizontal, o peso é o produto dos pesos devido a translação horizontal e a translação vertical, como descritos

anteriormente.

7.8 Configuração do Sistema

O sistema foi implementado visando a configuração de parâmetros específicos. A janela, ilustrada na figura 20, ilustra algumas das propriedades que podem ser configuradas pelo usuário. É possível configurar:

- a quantização da imagem panorâmica. Quanto maior a quantização, maior o custo de memória e há um aumento no tempo de processamento; por outro lado, a qualidade da imagem visualizada pelo usuário aumenta. Quando o panorama é gravado como *jpg*, ele é gravado com esta quantização.
- suavização na adição de imagens ao panorama.
- interpolação na adição de imagens ao panorama. Isto aumenta a qualidade visual, mas aumenta o tempo de processamento.
- tipo de quantização na leitura das imagens. Quando a imagem lida do disco é quantizada para um tamanho menor que o original, dois algoritmos podem ser utilizados: o rápido, de menor qualidade, e o suave. O algoritmo suave pode não funcionar corretamente na versão compilada para *Windows* devido a bugs do compilador.

A sugestão é de utilizar uma quantização baixa (quatro, por exemplo), sem interpolação e suavização, enquanto o panorama está sendo criado. Depois de terminado, pode-se modificar os parâmetros para obter um panorama de melhor qualidade.

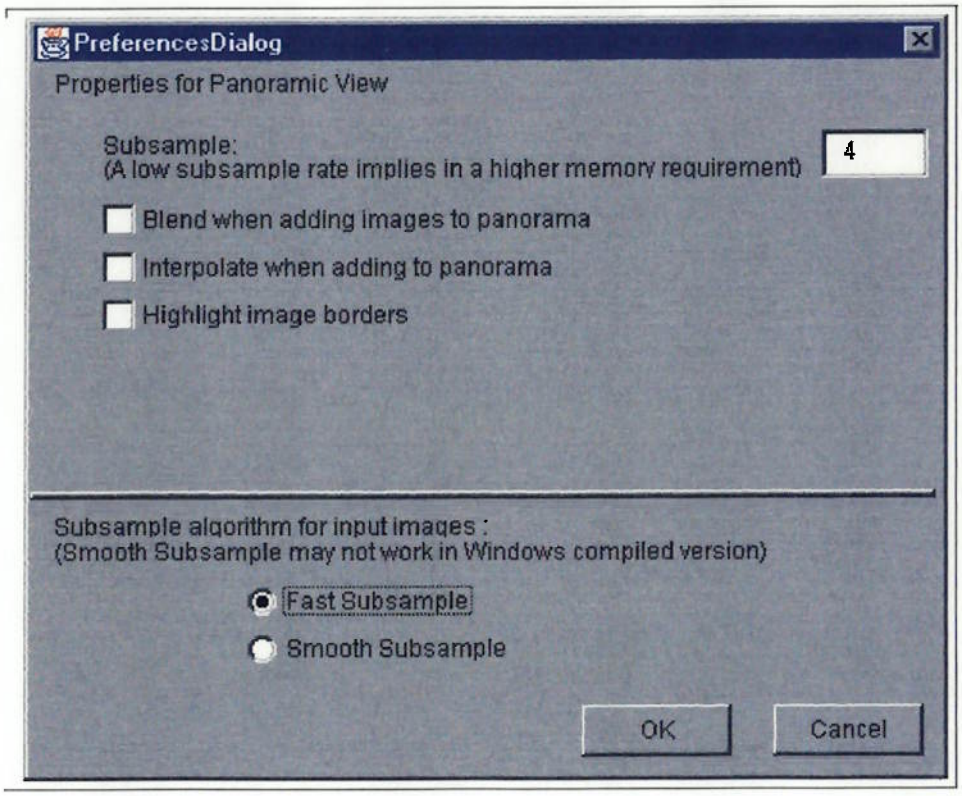


Figura 20: configuração do WebMosaicker.

8 Resultados e Análise

8.1 Exemplos

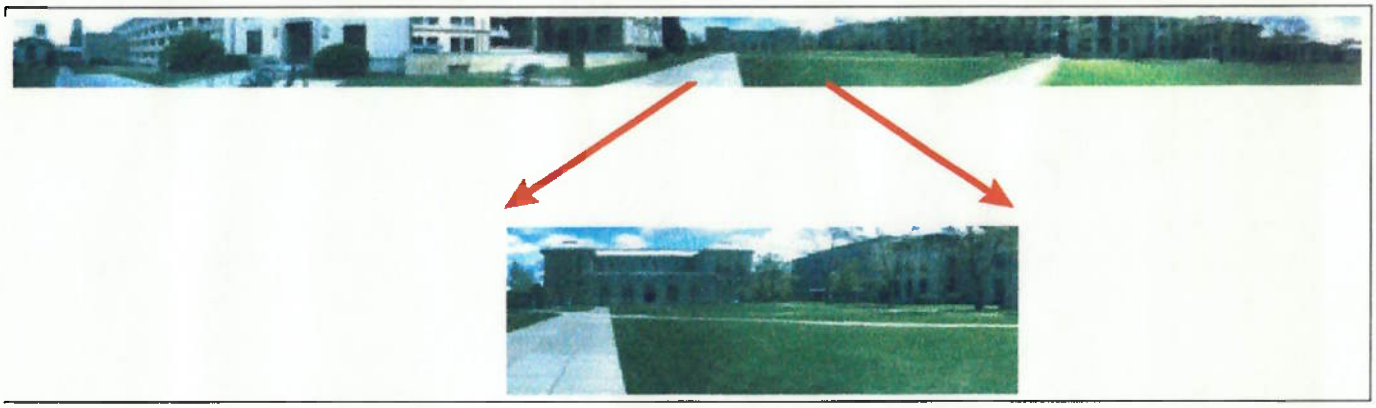


Figura 21: Universidade Carnegie Mellon.

Figura 22: Deserto do Atacama, Chile.

Figura 23: Missão Apollo 17, Lua.

Figura 24: Rio Allegheny, Pittsburgh.

Figura 25: Washington D.C., panorama gerado com imagens não calibradas.

8.2 Análise Qualitativa

Inicialmente, será realizada uma análise qualitativa dos resultados. Na Seção 8.1, alguns exemplos dos panoramas criados com o WebMosaicker são apresentados. Visualmente, principalmente quando analisado na tela do computador, as resultados são bons, apresentando registros coerentes das imagens e uma transição suave entre os pares de imagens.

Além da análise das imagens panorâmicas, deseja-se analisar a qualidade do registro das imagens. As figuras 26 e 27 ilustram a qualidade do registro das imagens. Cada uma das três imagens da figura 26 foi colocada em uma das bandas (vermelha, verde e azul) e registradas utilizando o algoritmo desenvolvido. O resultado é ilustrado na figura 27. As regiões em cinza indicam que o registro foi perfeito. Analizando a figura, é possível ver que as regiões onde as três imagens se sobrepõe é praticamente toda cinza. Os erros são principalmente devido a distorção da lente (região próxima a porta do lado esquerdo). O modelo rígido foi utilizado neste caso.

Figura 26: Imagens utilizadas para ilustrar a qualidade do registro.

Figura 27: Ilustração visual da qualidade do registro das imagens acima.

8.3 Análise Quantitativa

Além da análise qualitativa, onde podemos "ver se estamos satizfeitos" com os resultados, é necessário realizar uma análise quantitativa. Este tipo de análise nos permite determinar uma avaliação numérica da qualidade do panorama. Este valor númerico é importante para comparar algoritmos de registro de imagens, os efeitos da discretização do panorama e da suavização na sua geração.

Para realizar esta análise quantitativa da qualidade, foi criada e implementada uma formulação da razão sinal-ruído, que é comunmente utilizada em processamento de sinais, para imagens panorâmicas. Esta é a primeira formulação quantitativa da qualidade das imagens panorâmicas apresentada na literatura. Os detalhes e resultados serão apresentados nas próximas subseções.

Outro tipo importante de análise é a análise de desempenho. Neste caso, é desejado analisar o tempo de processamento em função das propriedades do panorama. Esta análise também será apresentada a seguir.

8.3.1 Razão Sinal-Ruído

O metodo de avaliação quantitativa da qualidade do panorama que foi desenvolvido utiliza a razão sinal ruído (*snr*). Esta razão é usualmente utilizada para avaliar a qualidade de sinais obtidos de sensores e em processamento de sinais. Nestes casos, o *snr* é dado pela razão entre a potência do sinal e a potência do ruído.

No caso das imagens panorâmicas, não existe uma definição formalizada para o "sinal" e o "ruído". Portanto, foi necessário formular uma forma coerente de calcular a razão sinal ruído. Portanto, foram utilizadas as seguintes definições:

sinal: a soma do espectro de potência das imagens que compõe o panorama;

ruído: a soma do espectro de potência da diferença entre as imagens originais e a projeção inversa do panorama no plano da imagem original, para cada imagem.

A equação 19 mostra a formulação matemática adotada para imagens panorâmicas.

$$snr = \sum_{i}^{n} \frac{\sum_{x,y} \|\mathcal{F}(Img_i(x,y))\|^2}{\sum_{x,y} (\|\mathcal{F}(Img_i(x,y) - Proj_i(x,y))\|)^2} \quad ; \qquad (19)$$

onde: snr é a razão sinal-ruído;

n é o número de imagens;

$\mathcal{F}$ é a transformada de Fourier;

Img_i é a imagem i;

$Proj_i$ é a projeção da imagem i.

8.3.2 Análise

A seguir, sãoapresentados alguns resultados obtidos utilizando o WebMosaicker e a metodologia apresentada anteriormente.

Nesta análise, uma comparação entre os panoramas calibrados (esféricos) e os não-calibrados (planos). Os dados utilizados foram obtidos utilizando as imagens do panorama da Universidade Carnegie Mellon apresentado na figura 21.

O "tempo de processamento", mensionado na análise, corresponde ao tempo médio de processamento de cada imagem. Isto é, o tempo de processamento de um panorama completo, no modo automático, dividido pelo número de imagens no panorama.

A "qualidade" corresponde a razão sinal ruído. Quanto maior a razão melhor a qualidade do panorama.

Efeito da Calibração na Qualidade

O gráfico da figura 28 mostra a qualidade do panorama em função da quantização da imagem panorâmica (matriz de inteiros).

A qualidade do panorama calibrado (esférico) é superior a do não-calibrado (plano). Isto se deve ao fato do modelo de movimento adotado, o de translação, não representar corretamente o efeito da rotação da câmera no panorama plano. O esférico, por outro lado, tem uma qualidade melhor, pois uma rotação da câmera corresponde a uma translação das imagens sobre a esfera.

Outro ponto interessante é a comparação do modo que a qualidade decai para um panorama calibrado com o modo para o não-calibrado. O não-calibrado decai quase lienarmente. Enquanto, o calibrado decai mais rapidamente para uma quantização maior que 2. Isto ocorre, porque o ganho de qualidade da calibração se perde mais rapidamente com a maior quantização.

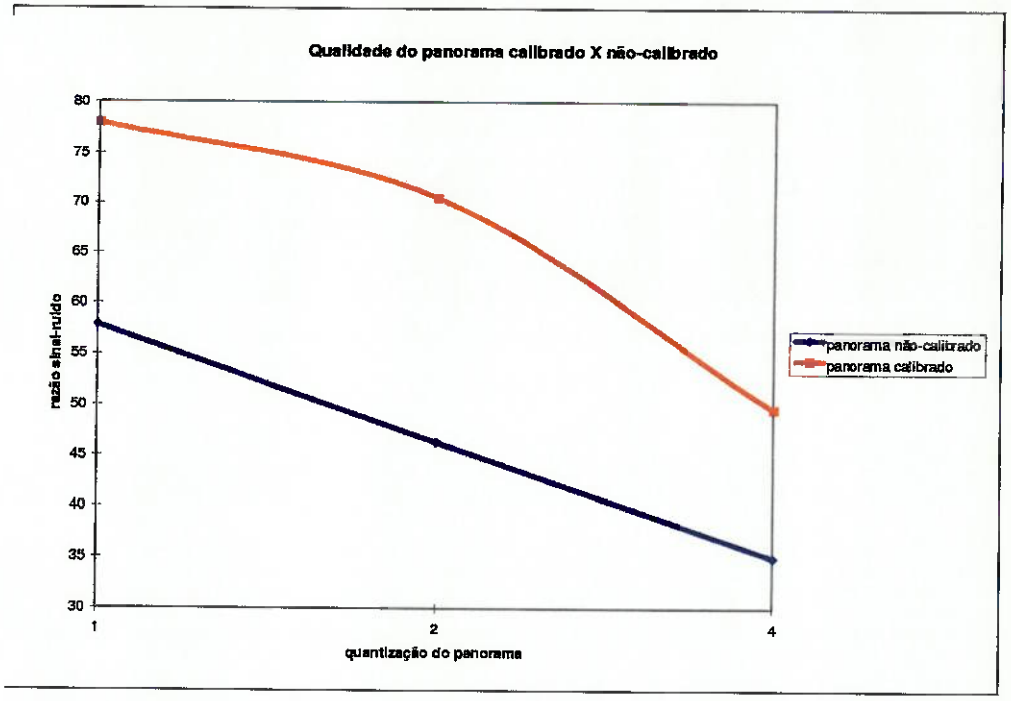


Figura 28: comparaçãoda qualidade do panorama plano com o esférico.

Tempo de Processamento

O gráfico da figura 29 ilustra o tempo de processamento em função da quantização do algoritmo de correlação de fase. Para uma quantização de 1, a imagem utilizada é de 256x128 pixels; quantização de 2, 128x64; etc.

Do gráfico, verifica-se que o tempo de processamento varia de 1.5 a 4.5 segundos. O tempo do panorama calibrado é obviamente superior ao do não-calibrado. O ponto interessante deste gráfico é a verificação que para o panorama não calibrado, o tempo decai com o nível de quantização. Por outro lado, para o panorama calibrado, o tempo de processamento decai até a quantização de 2, mas, para a quantização de 4, o tempo de processamento se estabiliza. Isto ocorre porque o tempo de projeção da imagem inteira se sobrepõe sobre o ganho no aumento da quantização da imagem.

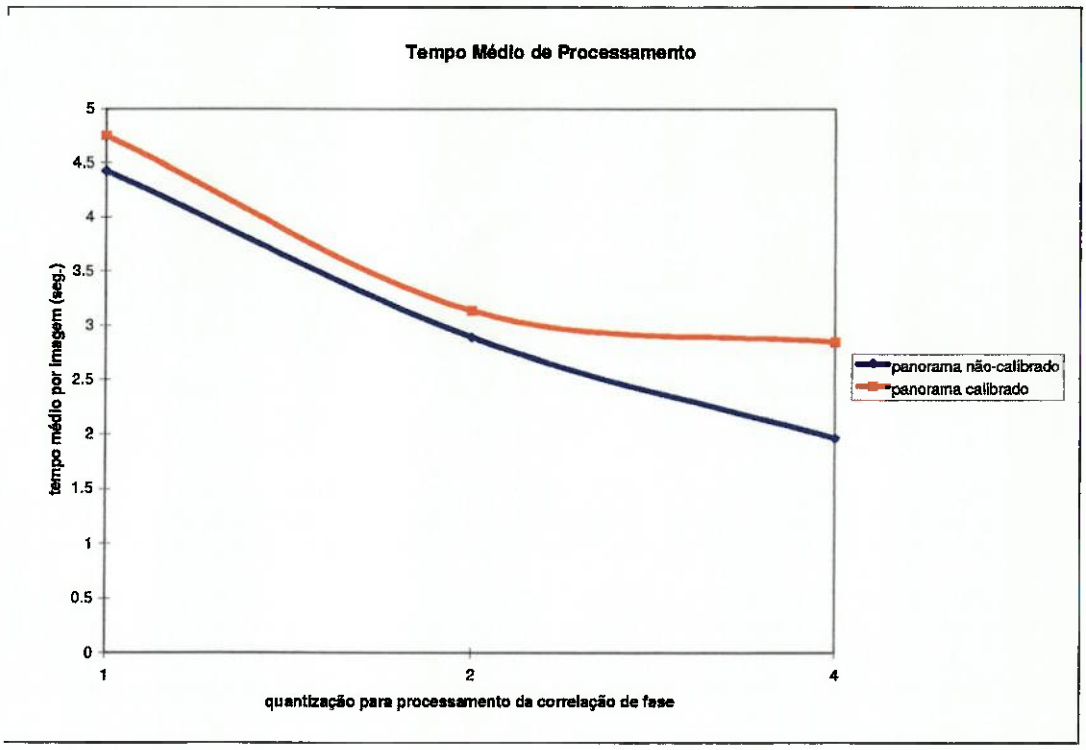


Figura 29: gráfico do tempo de processamento.

Ganho de Qualidade por Aumento de Tempo de Processamento

O gráfico da figura 30 extende a análise do último ítem para realizar uma comparação do ganho de qualidade para um aumento no tempo de processamento.

Verifica-se que, para ambos tipos de panorama, existe uma seção onde o ganho de qualidade é grande. Entretanto, este ganho diminue após este ponto.

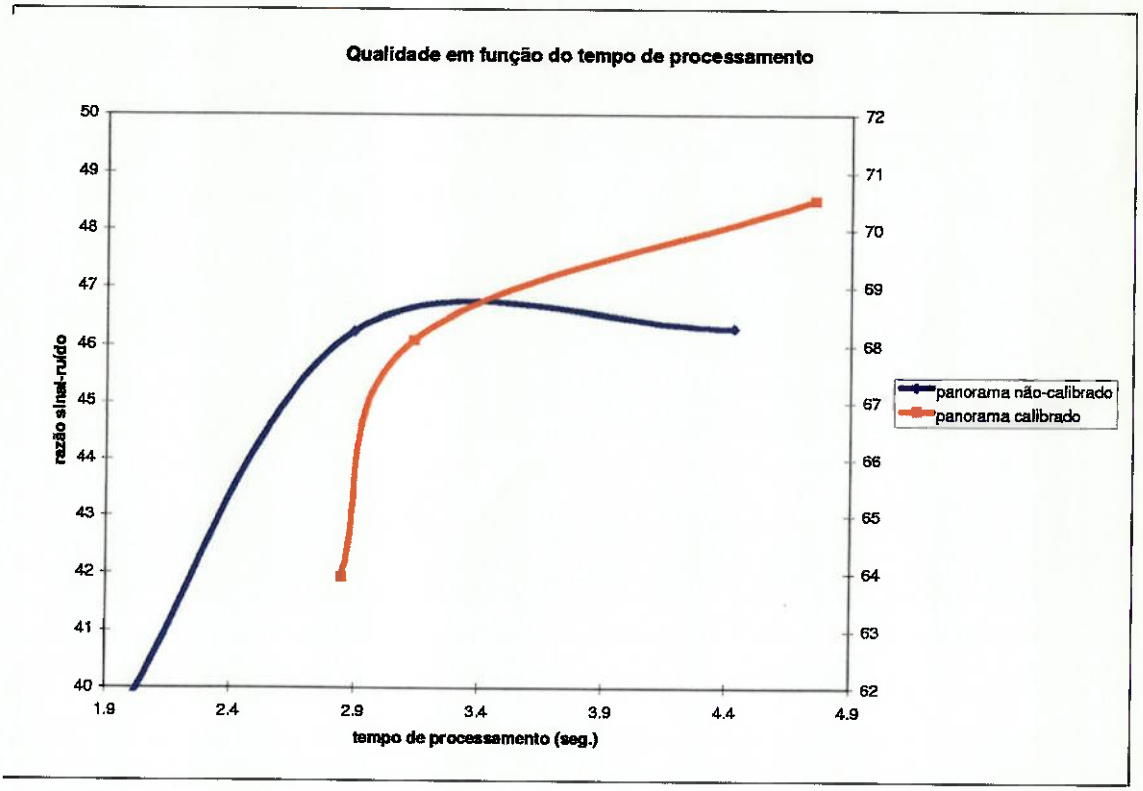


Figura 30: gráfico da qualidade do panorama em função do tempo de processamento.

Efeito da Quantização da correlaçãode Fase na Qualidade

A figura 31 mostra o efeito da quantização da correlação de fase na qualidade do panorama.

Para o panorama não-calibrado, a quantização não teve influência com uma variação de 1 para 2, pois o maior efeito na qualidade é devido ao tipo de panorama. Para quantização de 4, entretanto, a perda de resolução no algoritmo teve um efeito significativo, como esperado.

Por outro lado, o panorama calibrado teve o comportamento esperado, com uma perda de qualidade com a maior quantização do algoritmo.

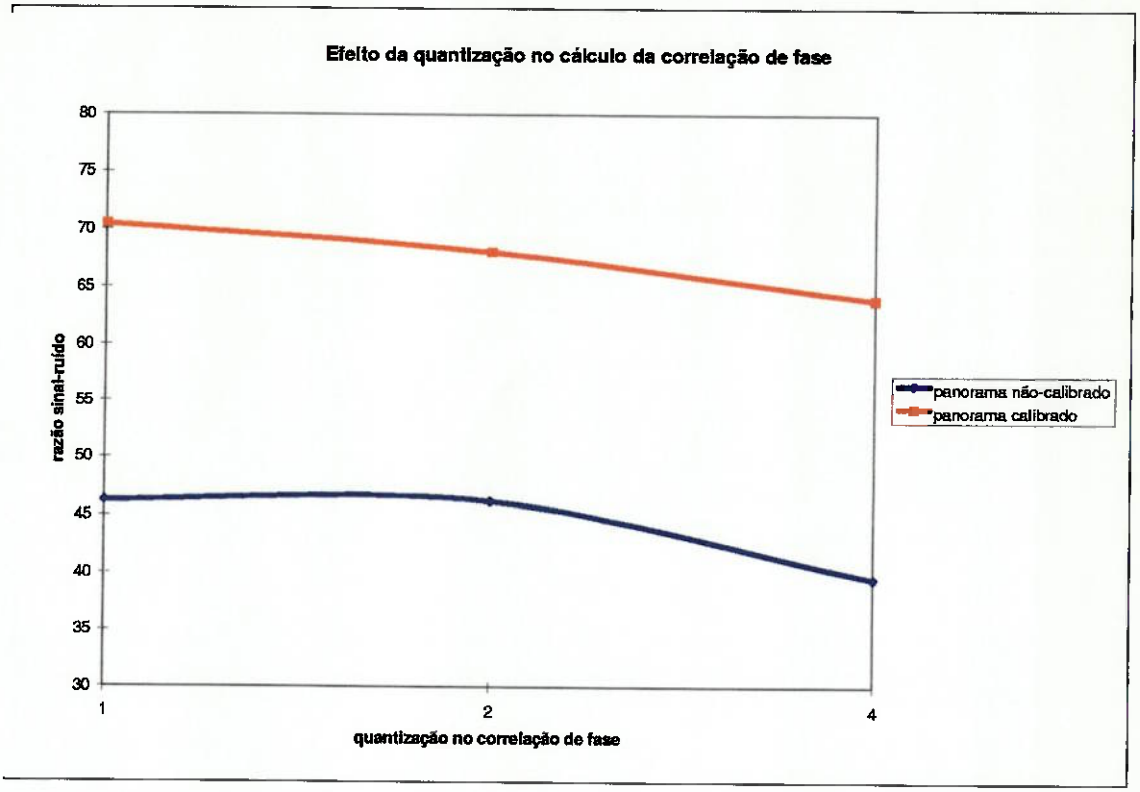


Figura 31: gráfico da qualidade do panorama em função da quantização do algoritmo de correlação de fase.

Efeito da Suavização na Qualidade

O gráfico da figura 32 mostra a qualidade do panorama em função da quantização da imagem panorâmica (matriz de inteiros), comparando o efeito da suavização na qualidade.

O principal resultado desta análise é que a suavização melhora a qualidade do panorama. Para o usuário, o panorama suavizado tem uma qualidade claramente superior. Entretanto, é interessante verificar essa superioridade numa análise quantitativa.

Além desse resultado, verifica-se que o decaimento na qualidade com a maior quantização do panorama foi mais acentuado para o panorama suavizado. Isto ocorre porque o ganho de qualidade devido a suavização se perde mais rapidamente com a maior quantização do panorama.

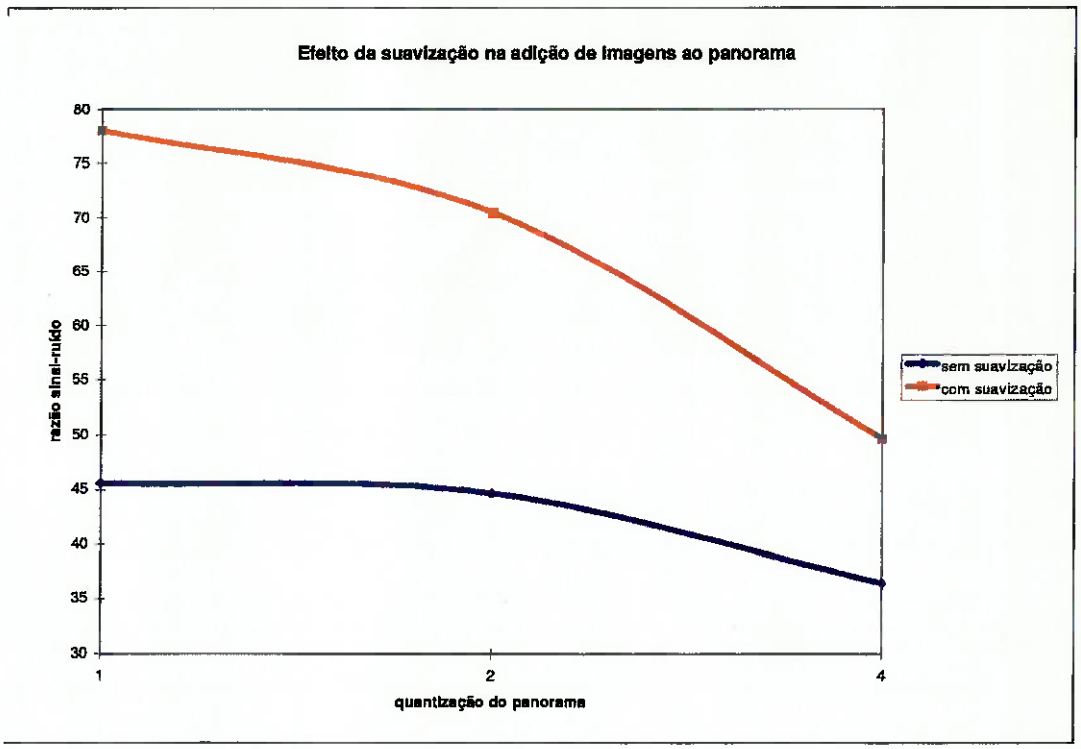


Figura 32: comparação do efeito da suavização na qualidade do panorama.

9 Conclusões

Este texto descreve o projeto *WebMosaicker - Geração Automática de Imagens Panorâmicas.* Inicialmente, o problema de geração de imagens panorâmicas foi definido, tendo em vista o objetivo principal de criar um sistema para geração de panoramas a partir de uma sequência de imagens. Posteriormente, foram apresentadas possíveis soluções com ênfase no registro das imagens. Experimentos iniciais foram realizados utilizando um protótipo implementado em AVS. Na etapa seguinte, foi definida e especificada a versão final. A implementação desta versão é descrita e testada. Os resultados da análise quantitativa e qualitativa são finalmente, apresentados.

A versão final, implementada em Java, tem uma interface que foi considerada amigável e de fácil utilização por usuários.

A capacidade de ler uma sequência numerada foi fundamental para aliviar o trabalho do usuário. Na prática, visualizamos a utilização do sistema como seguindo as seguintes etapa:

1. utilização no modo sem interação com o usuário e com uma sequência numerada para gerar um panorama inicial;
2. o usuário verifica se está satizfeito com o resultado;
3. caso contrário, o usuário verifica os pontos onde o registro das imagens não foi adequado. O modo sem interação com o usuário é utilizado nas partes onde o registro foi bom. Nos pontos de registro inadequado, o modo interativo é utilizado.

A análise qualitativa dos panoramas mostrou que os resultados são adequados para a utilização das imagens panorâmicas em aplicações que envolvem pessoas. Alguns exemplos desta aplicação são a realidade virtual e o mapeamento aéreo.

A análise quantitativa foi um dos pontos mais importantes deste trabalho, sendo a primeira citação na literatura de um método de análise quantitativa

de imagens panorâmicas.

Algumas conclusões interessantes da análise quantitativa:

- A razão sinal-ruído para os os panoramas esféricos (calibrados) esteve na órdem de 70. Enquanto, para os planos (não-calibrados) este valor foi de 50;
- o panorama esférico tem melhor qualidade que o plano;
- na correlação de fase, uma quantização de 2 resulta na melhor razão qualidade - tempo de processamento;
- a suavizão na criação da imagem panorâmica melhorou a sua qualidade;
- o tempo de processamento médio por imagem foi de 2 a 4 segundos.

O sistema implementado pode ser utilizado como uma base de testes para algoritmos de registro de imagens. Os algoritmos devem imaplementar a interface *RegistrationAlgoritm* para permitir a sua utilização no sistema. Vários algoritmos podem ser implementados e comparados com uma análise quantitativa.

A escolha de AVS para implementação do protótipo foi adequada, permitindo mudanças rápidas do sistema. Entretanto, para a implementação final, a linguagem Java é mais apropriada pela facilidade de acesso a rede, independência de plataforma, orientação a objetos, entre outras características.

Durante os experimentos, tornou-se claro que o sistema deve ser interativo. Deste modo, o usuário pode intervir no caso de possíveis erros no registro das imagens. Entretanto, é fundamental utilizar um sistema de registro automático, pois a seleção de pontos no caso manual é um processo longo e cansativo para grandes panoramas. Além disso, erros no registro das imagens ocorrem com baixa frequência na maioria dos casos.

O método de correlação de fase provou ser um bom estimador global da translação entre as imagens. O refinamento da solução utilizando a minimização da diferença entre as imagens, é necessário.

10 Trabalho Futuro

No WebMosaicker, há funcionalidades que podem ser adicionadas, como:

- permitir a adição de imagens a qualquer ponto do panorama; atualmente, a imagens só podem ser adicionadas se esta se sobrepõe a última imagem do panorama;
- permitir a remoção de qualquer imagem do panorama, não somente a última como é atualmente;
- permitir a interrupção da leitura de uma sequência numerada de imagens durante o seu processamento;
- salvar e abrir um panorama no formato interno no WebMosaicker;
- integrar o WebMosaicker com um digitalizador de imagens para que as imagens de entrada não tenham que ser gravadas no disco.

Existem vários pontos deste trabalho que podem ser extendidos novos trabalhos, como por exemplo:

- análise comparativa de algoritmos de registro de imagens, utilizando a metodologia de análise quantitativa desenvolvida;
- recuperação dos parâmetros de calibração da câmera para permitir a criação de panoramas esféricos com câmeras não-calibradas;
- desenvolvimento de um sistema de visualização da imagem panorâmica do tipo "realidade virtual".

11 Referências Bibliográficas

[1] L. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, December 1992.

[2] P. Burt and P. Anandan. Image stabilization by registration to a reference mosaic. *DARPA Image Understanding Workshop*, november 1994.

[3] F. Cozman and E. Krotkov. Automatic mountain detection and pose estimation for teleoperation of lunar rovers. *Int. Conference on Robotics and Automation*, 1997.

[4] E. Hall. *Computer Image Processing and Recognition.* Academic Press, 1979.

[5] M. Hansen, P. Anandan, K. Dana, G. Van der Wal, and P. Burt. Real-time scene atabilization and mosaic construction. *DARPA Image Understanding Workshop*, November 1994.

[6] M. Irani, B. Rousso, and S. Peleg. Computing occluding and transparent motions. *International Journal of Computer Vision*, 12(1):5–16, January 94.

[7] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using image stabilization. In *International Conference on Computer Vision and Pattern Recognition*, pages 454–460, March 94.

[8] C. Kuglin and D. Hines. The phase correlation image alignment method. *IEEE Conference on Cybernetics and Society*, September 1975.

[9] C. H. Morimoto and R. Chellappa. Automatic digital image stabilization. *IEEE International Conference on Pattern Recognition*, August 1996.

[10] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. *Computer Vision and Pattern Recognition*, June 1997.

[11] W. Press, W. Vetterling, and S. Teukolsky snd B. Flannery. *Numerical Recipes in C.* Cambridge Press, 1992.

[12] L. Robert. Camera calibration without feature extraction. *Proc. Intl. Conf. Pattern Recognition*, 1994.

[13] A. Rosenfeld. *Multiresolution Image Processing and Analysis.* Springer-Verlag, 1984.

[14] R. Szeliski. Image mosaicing for tele-reality applications. Technical Report CRL 94/2, Digital Equipment Corporation, Cambridge Research Lab, May 1994.

[15] R. Szeliski and H. Shum. Creating full view panoramic image mosaics and environment maps. *SIGGRAPH*, August 1997.

[16] G. Wolberg. *Digital Image Warping.* IEEE Press, 1990.


10 ¢
QUINO