Aplicação de Algoritmos de Visão Computacional à Inspeção Industrial de Maçãs

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Daniel Cabrini Hauagge e aprovada pela Banca Examinadora.

Campinas, 10 de outubro de 2008.

Prof. Dr. Siome Klein Goldenstein
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Júlia Milani Rodrigues CRB8a / 2116

Hauagge, Daniel Cabrini

H29a Aplicação de algoritmos de visão computacional à inspeção industrial de maçãs / Daniel Cabrini Hauagge -- Campinas, [S.P. :s.n.], 2008.

Orientador : Siome Klein Goldenstein

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Visão computacional. 2. Reconstrução tridimensional. 3. Fluxo óptico. 4. Subtração de fundo. I. Goldenstein, Siome Klein. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Application of computer vision algorithms in the industrial inspection of apples.

Palavras-chave em inglês (Keywords): 1. Computer vision. 2. 3D reconstruction. 3. Optical flow. 4. Background subtraction.

Área de concentração: Visão Computacional

Titulação: Mestre em Ciência da Computação

Banca examinadora:

Prof. Dr. Siome Klein Goldenstein (IC-UNICAMP)
Prof. Dr. Jorge Stolfi (IC-UNICAMP)
Profa. Dra. Anna Helena Reali Costa (POLI-USP)

Data da defesa: 31/10/2008

Programa de pós-graduação: Mestrado em Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 31 de outubro de 2008, pela Banca examinadora composta pelos Professores Doutores:

Profª. Drª. Anna Helena Reali Costa
POLI / USP.

Prof. Dr. Jorge Stolfi
IC – UNICAMP.

Prof. Dr. Siome Klein Goldenstein
IC – UNICAMP.

Instituto de Computação
Universidade Estadual de Campinas

Aplicação de Algoritmos de Visão Computacional à Inspeção Industrial de Maçãs

Daniel Cabrini Hauagge[1]

Dezembro de 2008

Banca Examinadora:

- Prof. Dr. Siome Klein Goldenstein (Orientador)
- Prof. Dr. Jorge Stolfi
  Universidade Estadual de Campinas
- Prof.ª Dr.ª Anna Helena Reali Costa
  Universidade de São Paulo
- Prof. Dr. Ricardo da Silva Torres
  Universidade Estadual de Campinas
- Prof.ª Dr.ª Shin-Ting Wu
  Universidade Estadual de Campinas

[1]**Suporte financeiro de: Bolsa do CNPq (processo 132414/2006-6) 2006–2008.**

Resumo

Apresentamos nesta dissertação quatro algoritmos voltados para a classificação automatizada de frutas. A subtração de fundo baseada na distância de Mahalanobis. O rastreamento das frutas em uma esteira usando a subtração de fundo, casamento de padrões e fluxo óptico. A reconstrução tridimensional da fruta a partir de imagens dela na esteira, onde recuperamos a posição da câmera com relação a fruta usando fluxo óptico e uma estimativa grosseira do movimento da fruta. A forma da fruta é obtida a partir das silhuetas reprojetadas no espaço tridimensional usando duas abordagens diferentes. Finalmente, a localização do pedúnculo e cálice a partir do eixo de simetria da reconstrução tridimensional.

Realizamos testes com os quatro algoritmos. Obtivemos bons resultados com os dois primeiros. Para a reconstrução tridimensional verificamos bons resultados para algumas etapas do processo (fluxo óptico, estimativa inicial e otimização não-linear do movimento de câmera). Resultados fracos foram obtidos para a reprojeção das silhuetas usando os dois métodos. Analisamos as causas dos erros e propomos métodos que poderiam ser usados para melhorá-los. Os resultados da localização do pedúnculo e cálice foram insatisfatórios mas acreditamos que melhorariam se obtivéssemos uma reconstrução mais precisa.

Também criamos um sistema de captura que reproduz as condições dentro de um sistema comercial de classificação. Com este aparato construímos quatro grandes bases de dados com aproximadamente 3000 frutas, 35 imagens de cada uma, contendo quatro variedades de maçã. Outras 6 bases menores foram criadas.

Abstract

We present in this dissertation four algorithms targeted at the automated classification of fruits. Background subtraction based on Mahalanobis distance. Fruit tracking on a conveyor belt using background subtraction, pattern matching and optical flow. The 3D reconstruction of the fruit from its images on the conveyor belt, where we recover the camera position, with respect to the fruit, using optical flow and a rough estimate of fruit motion. The fruit's shape is recovered from the silhouette re-projected into 3D space using two different approaches. Finally, the location of the stem and calyx based on the symmetry axis of the 3D reconstruction.

We also present the results of tests conducted with the four algorithms. We obtained good results with the first two. For the three-dimensional reconstruction we obtained good results with some of the intermediary steps (optical flow, initial estimate and non-linear refinement of camera motion). Poor results were obtained for the re-projection of the silhouette's, using two approaches. We analyze the causes of these difficulties and suggest approaches that could improve them. The localization of stem and calyx was compromised by the poor 3D reconstruction so we believe that it will improve once we address the problems with the reconstruction algorithm.

We created an image capturing system that reproduces the conditions inside a commercial grading machine. With this device we acquired four big data sets with approximately 3000 apples, 35 images of each, comprising four varieties. Another 6 smaller data-sets were also created.

Agradecimentos

Agradeço a meus pais Roberto e Claudete e as minhas irmãs Nicole e Karina pelo apoio e carinho.

À minha namorada, Ariadne, que me deu suporte e foi sempre compreensiva.

Ao professor Siome Klein Goldenstein, meu orientador, pela confiança que depositou em mim e pela liberdade que me deu em pesquisar um tema novo para ele. Suas sugestões foram importantes no desenvolvimento desta dissertação.

Ao Valdir Bissiato, pela ajuda na construção do sistema de captura.

Aprendi muito sobre o problema tratado neste mestrado em viagens feitas ao sul do país, para a cidade de Fraiburgo, estado de Santa Catarina. Foi lá que colhi as imagens para as bases de dados que montei. Devo agradecer ao Arival Pioli, Silvio José Gmach e Stênio Ricardo Zanin da Fischer. Agradeço também o Ailton C. Bittencourt da Maquifrai Indústria e Comércio Limitado. Ao Ricardo Cecchini, Gilson, Marcelo e Dionisio da Renar. Ao José Perazzoli da Agropel Agroindústria Perazzoli Ltda.

Em um trabalho relacionado com o desenvolvido aqui, o de identificar frutas em uma imagem, trabalhei com o também colega de laboratório Anderson Rocha. O desenvolvimento deste trabalho foi uma experiência importante no desenvolvimento desta dissertação.

Ao professor Luíz Velho, do Instituto Nacional de Matemática Pura e Aplicada, por ter emprestado uma das câmeras usadas para capturar as bases de dados.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Algoritmos

Lista de Símbolos

t_i	instante de tempo
$\boldsymbol{P}$	transformação afim que leva o sistema de coordenadas da esteira para a câmera, $\boldsymbol{P} \in \mathbb{R}^{4\times4}$
$\boldsymbol{P}_{t_i}$	transformação afim que leva do sistema de coordenadas da fruta para o sistema de coordenadas da câmera no instante t_i
$\boldsymbol{P}_{t_it_{i+1}}$	transformação afim que leva do sistema de coordenadas da câmera no instante t_i para a câmera no instante t_{i+1}, $\boldsymbol{P}_{t_it_{i+1}} \in \mathbb{R}^{4\times4}$
$\tilde{\boldsymbol{P}}_{t_it_{i+1}}$	a matriz $\boldsymbol{P}_{t_it_{i+1}}$ corrompida por ruído
$\boldsymbol{P}'_{t_it_{i+1}}$	resultado da otimização não-linear aplicada a matriz $\tilde{\boldsymbol{P}}_{t_it_{i+1}}$
$\boldsymbol{M}$	matriz de parâmetros extrínsecos da câmera, $\boldsymbol{M} \in \mathbb{R}^{3\times3}$
$\boldsymbol{E}$	matriz essencial, $\boldsymbol{E} \in \mathbb{R}^{3\times3}$
$\boldsymbol{\rho}_{t_i}, \boldsymbol{\rho}_{t_{i+1}}$	região sendo rastreada, nos momentos t_i e t_{i+1}. Em boa parte do texto usamos $\boldsymbol{\rho}_{t_i} = \boldsymbol{\rho}$
$n_{\boldsymbol{\rho}}$	número de pontos dentro de uma região $\boldsymbol{\rho}$
λ	fator multiplicativo do modelo de iluminação, $\lambda \in \mathbb{R}^+$
$\delta\lambda$	incremento de λ, $\delta\lambda \in \mathbb{R}$
ϵ_λ	valor mínimo para λ, $\epsilon_\lambda \in \mathbb{R}$
σ	fator aditivo no modelo de iluminação, $\sigma \in \mathbb{R}$
$\delta\sigma$	incremento de σ, $\delta\sigma \in \mathbb{R}$
$\boldsymbol{\gamma}_i$	parametrização da matriz $\boldsymbol{P}_{t_it_{i+1}}$, $i \in [1, 2, 3]$, a dimensão de $\boldsymbol{\gamma}_i$ varia conforme i

$I(\boldsymbol{x})$	valor da imagem no ponto $\boldsymbol{x}$, $I(\boldsymbol{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}$
$\breve{I}(\boldsymbol{x})$	imagem alterada pela iluminação, $I(\boldsymbol{x}) = \lambda I(\boldsymbol{x}) + \sigma$.
$\boldsymbol{I}(\boldsymbol{\mu})$	todas as amostras da região $\boldsymbol{\rho}$ em forma vetorial, $\boldsymbol{I}(\boldsymbol{\mu}) : \mathbb{R}^{n_{\boldsymbol{\mu}}} \rightarrow \mathbb{R}^{n_{\boldsymbol{\rho}}}$. $\boldsymbol{I}(\boldsymbol{\mu}) = [I(\boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu}), \boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu}), \ldots, \boldsymbol{w}(\boldsymbol{x}_{n_{\boldsymbol{\rho}}}, \boldsymbol{\mu})]^T$
$\breve{\boldsymbol{I}}(\boldsymbol{\mu})$	o vetor $\boldsymbol{I}(\boldsymbol{\mu})$ afetado pela iluminação. $\breve{\boldsymbol{I}}(\boldsymbol{\mu}) = \lambda \boldsymbol{I}(\boldsymbol{\mu}) + \mathbf{1}\sigma$
nil	usado nos algoritmos para sinalizar que não foi encontrada uma resposta válida
$\boldsymbol{w}(\cdot)$	modelo de deformação do fluxo óptico, $\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}) : \mathbb{R}^{2+n_{\boldsymbol{\mu}}} \rightarrow \mathbb{R}^2$, onde $\boldsymbol{x} \in \mathbb{R}^2$ é a posição na imagem
$\boldsymbol{w}_{trans}(\cdot)$	modelo translacional de deformação
$\boldsymbol{w}_{esf}(\cdot)$	modelo esférico de deformação
$\boldsymbol{\mu}$	vetor de parâmetros de $\boldsymbol{w}(\cdot)$, $\boldsymbol{\mu} \in \mathbb{R}^{n_{\boldsymbol{\mu}}}$. As dimensões de $\boldsymbol{\mu}$ variam conforme o modelo de deformação
$n_{\boldsymbol{\mu}}$	dimensão de $\boldsymbol{\mu}$
$\delta\boldsymbol{\mu}$	incremento de $\boldsymbol{\mu}$, $\delta\boldsymbol{\mu} \in \mathbb{R}^{n_{\boldsymbol{\mu}}}$
$\epsilon_{\delta\boldsymbol{\mu}}$	valor mínimo para $\|\delta\boldsymbol{\mu}\|$, $\epsilon_{\delta\boldsymbol{\mu}} \in \mathbb{R}$
$\mathbf{1}$	vetor coluna de 1's, $\mathbf{1} = [1, 1, \ldots, 1]^T$. Dimensão implícita pelo contexto.
$\boldsymbol{J}$	jacobiano de $\boldsymbol{I}_{t_{i+1}}$ com relação a $\boldsymbol{\mu}$, $\boldsymbol{J} \in \mathbb{R}^{n_{\boldsymbol{\rho}} \times n_{\boldsymbol{\mu}}}$
$\boldsymbol{\Omega}$	matriz auxiliar no cálculo do fluxo óptico com robustez a iluminação. $\boldsymbol{\Omega} = [\boldsymbol{I}_{t+1}, \mathbf{1}, \lambda\boldsymbol{J}] \in \mathbb{R}^{n_{\boldsymbol{\rho}} \times (2+n_{\boldsymbol{\mu}})}$
$\delta\boldsymbol{\beta}$	incremento das variáveis do fluxo óptico com robustez a iluminação, $\delta\boldsymbol{\beta} = [\delta\lambda, \delta\sigma, \delta\boldsymbol{\mu}]^T \in \mathbb{R}^{2+n_{\boldsymbol{\mu}}}$
$\boldsymbol{n}_{rot}$	normal ao plano de rotação
$\boldsymbol{A}(\boldsymbol{\mu})$	matriz de transformação afim de $\boldsymbol{w}_{esf}(\cdot)$, $\boldsymbol{A}(\boldsymbol{\mu}) : \mathbb{R}^{n_{\boldsymbol{\mu}}} \rightarrow \mathbb{R}^2$
$\boldsymbol{b}(\boldsymbol{\mu})$	vetor de translação de $\boldsymbol{w}_{esf}(\cdot)$, $\boldsymbol{b}(\boldsymbol{\mu}) : \mathbb{R}^{n_{\boldsymbol{\mu}}} \rightarrow \mathbb{R}^2$
$\|\cdot\|$	norma L_2
$\|\cdot\|$	cardinalidade de um conjunto

Capítulo 1

Introdução

1.1 Motivação

Técnicas de visão computacional são amplamente empregadas na inspeção de bens manufaturados, no entanto, a inspeção de produtos agrícolas é ainda um desafio. A dificuldade em se desenvolver algoritmos para este propósito é que produtos agrícolas exibem uma variabilidade enorme quando comparados com produtos industrializados. Outro problema é que, embora alguns algoritmos mais sofisticados consigam lidar com esta variabilidade, o tempo de processamento é muito longo. Por estas razões o uso de humanos na classificação é ainda alto.

Escolhemos a maçã pois tem uma importância econômica alta, sua inspeção automatizada é desafiadora por vários motivos (forma irregular, ampla variedade de cores, muitos tipos de defeitos) e também porque os produtores nacionais têm necessidades específicas que não são estudadas por pesquisadores de outros países.

Outras aplicações desta pesquisa são em melhoramento genético onde as técnicas de detecção de defeitos e reconstrução tridimensional seriam úteis na hora de determinar as qualidades de novas variedades de fruta. Quando correlacionadas com a localização da colheita, as medições na fruta permitem fazer diagnósticos a respeito dos pomares de onde as frutas vieram (e.g. alguns tipos de defeitos são consequência de deficiências do solo e sua detecção forneceria uma forma precisa de avaliar a extensão do problema). Desenvolvimentos futuros permitiriam a classificação de frutas diretamente na árvore, cuja aplicação seria em colheitadeiras robóticas e sistemas de previsão de colheita.

1.2 Contribuições

A primeira contribuição desta dissertação são as bases de dados que criamos. Ao todo foram fotografadas mais de 3000 frutas, com uma gama enorme de defeitos e quatro variedades diferentes. Acreditamos que o uso futuro destas bases irá permitir o desenvolvimento de novos algoritmos para a detecção de defeitos.

A segunda contribuição é um algoritmo simples para o rastreamento de frutas na esteira. Desenvolvemos este algoritmo com as máquinas mecânicas em mente. Algoritmos mais sofisticados permitem máquinas mais simples, o que reduz substancialmente o custo da máquina, uma preocupação importante para o mercado nacional.

A nossa terceira contribuição é o algoritmo para a reconstrução tridimensional da fruta. Até o momento pouco trabalho tem sido feito neste sentido e o principal limitador é o poder de processamento disponível até então. Técnicas capazes de fazer esse tipo de aferição permitem fazer medições precisas de volume (o que eliminaria a necessidade de balanças eletrônicas) e permitem o uso de algoritmos mais sofisticados de classificação (que levam em consideração a localização do defeito, por exemplo).

Finalmente, também propomos uma nova maneira de localizar as extremidades da fruta em imagens, um problema amplamente estudado na literatura mas sem uma solução efetiva até o momento. A técnica é baseada na reconstrução tridimensional da fruta.

1.3 Organização

Esta dissertação foi dividida em oito capítulos. Seguindo o presente capítulo, a introdução, temos no Capítulo 2 uma descrição da produção da maçã, falando do fluxo de produção, normas de classificação e diferentes tipos de defeitos da fruta. O Capítulo 3 faz uma revisão bibliográfica dos algoritmos para a classificação de maçãs, mostrando de maneira separada os sistemas de aquisição, os algoritmos de localização do cálice e do pedúnculo e os algoritmos de localização de defeitos e classificação dos frutos. O Capítulo 4 descreve o nosso sistema de captura e as bases de dados que construímos com ele. O Capítulo 5 trata do algoritmo de subtração de fundo que usamos junto com alguns testes demonstrando sua performance. O Capítulo 6 mostra o algoritmo para rastreamento do fruto na esteira e apresenta os resultados obtidos. O Capítulo 7 traz o algoritmo para reconstrução tridimensional, com cada uma das etapas do processo descrita separadamente, algumas com resultados. Também descreve um método para encontrar as extremidades do fruto e os resultados do algoritmo. Finalmente, o Capítulo 8 traz as conclusões e trabalhos futuros.

Capítulo 2

Produção e Classificação da Maçã

A produção mundial da maçã gira em torno de 55 milhões de toneladas e os dez maiores países produtores podem ser vistos na tabela 2.1. A produção nacional é de cerca de 849.000 toneladas/ano (dado de 2005 [1]), concentrada principalmente no sul do país, com Santa Catarina produzindo 474.000 toneladas/ano, Rio Grande do Sul produzindo 328.000 toneladas/ano e Paraná com 45.000 toneladas/ano. Neste nível a produção torna o país auto-suficiente e aproximadamente 12% da safra é exportada, sendo que os principais mercados são: Holanda, Reino Unido, Suécia e Alemanha.

As principais variedades cultivadas no Brasil são: Fuji e Gala (ver Figura 2.2). A colheita se dá entre os meses de dezembro e abril, mas a classificação é realizada o ano todo, porque a fruta é armazenada em câmaras frias com atmosfera controlada que retardam a maturação.

A Figura 2.1 mostra um desenho esquemático com as principais partes da maçã.

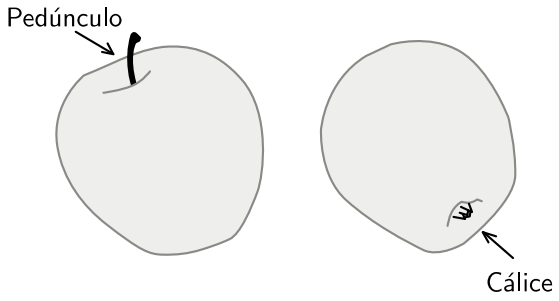


Figura 2.1: Anatomia de uma maçã.

Posição	País	Produção ($\times 10^6$ toneladas)
1º	China	25
2º	Estados Unidos	4.2
3º	Turquia	2.5
4º	Irã	2.4
5º	Itália	2.2
6º	França	2.1
7º	Polônia	2
8º	Rússia	2
9º	Alemanha	1.6
10º	Índia	1.5
14º	Brasil	0.8

Tabela 2.1: Os dez maiores produtores mundiais de maçã, mais o Brasil, no ano de 2005 [2].

2.1 Normas de classificação

A classificação da fruta tem vários papéis. O principal é facilitar o comércio da fruta a medida que estabelece padrões de qualidade que são amplamente conhecidos e aplicados de maneira uniforme, o que reduz a necessidade do comerciante inspecionar a fruta antes de comprá-la do produtor. Uma classificação precisa também permite que o produtor obtenha o máximo de lucro para cada classe. Frutas de uma classe superior não são vendidas como classes inferiores e a contaminação de lotes de frutas de uma classe superior com frutas de classes inferiores não reduz o valor de mercado do lote. Finalmente, a eliminação de frutas com alguns tipos de defeitos que progridem e atingem outras frutas evita que danos a poucos frutos se espalhem e aumentem os prejuízos.

Em linhas gerais, as normas de classificação avaliam duas características: o calibre (tamanho ou peso) e a qualidade da fruta.

A escolha entre o tamanho ou peso é aparentemente arbitrária, no Brasil a norma [10] faz referência apenas ao peso mas as legislações européia [28] e americana [32] permitem a classificação por diâmetro ou peso da fruta. No entanto, suas consequências para a classificação automatizada são importantes. Se a norma exige apenas a determinação do tamanho da fruta, apenas uma câmera é necessária. Para a medida do peso precisamos usar uma balança, visto que a densidade da fruta varia bastante. Quando desejamos avaliar características como cor e defeitos precisamos de câmeras. Portanto, quando a legislação normativa faz menção ao peso, sistemas mais avançados que avaliam atributos como cor e defeitos precisam de, no mínimo, dois tipos de sensores: câmeras e balanças.

Sobre a qualidade da fruta as principais características que a afetam são: coloração da casca, presença de defeitos, forma e teor de açúcar. Em países de clima temperado, onde a

maçã se adapta bem, a presença de defeitos é muito baixa (algo como 90% frutas livres de defeitos), já em países mais quentes como o Brasil a presença de defeitos é bastante alta (chegando a 90% de frutas *com* defeito). A consequência destas diferenças nas normas de classificação é que em países onde a ocorrência de defeitos é baixa a sua tolerância também é menor. A legislação americana praticamente não permite defeitos em suas três principais categorias ("U.S. Extra Fancy", "U.S. Fancy" e "U.S. No. 1"), com exceção de *russeting* que é permitido na categoria "U.S. No. 1" e dano por geada leve na sub categoria "U.S. No. 1 Hail", a diferenciação entre as classes é feita principalmente por cor. A legislação européia permite alguns poucos defeitos e especifica três classes ("Extra", "Categoria I", "Categoria II"). Já a legislação brasileira é a mais permissiva das três, permitindo mais defeitos e fazendo distinção entre os graus dos vários tipos de defeitos, especificando quatro categorias ("Extra", "Categoria I", "Categoria II" e "Categoria III").

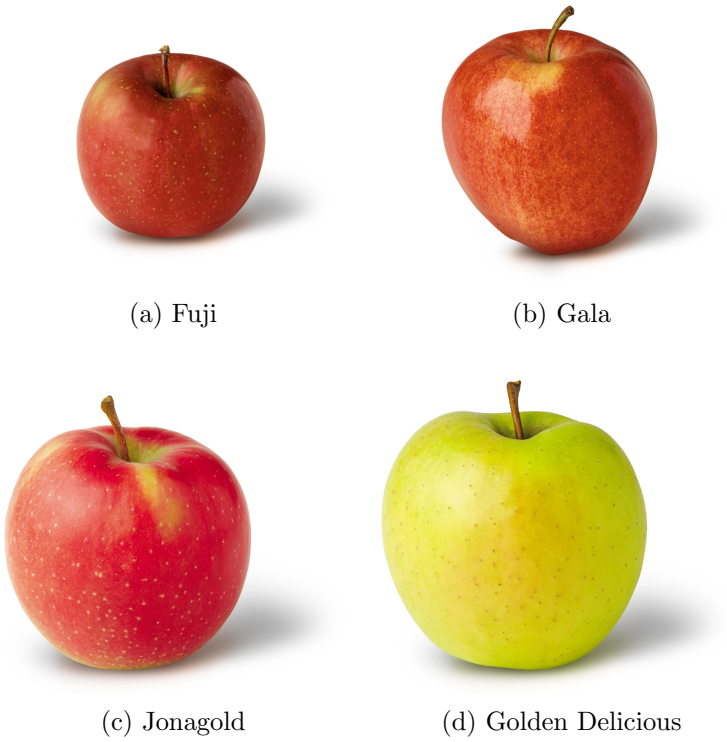
A consequência destas normas para a classificação automatizada da fruta é que em países onde a norma simplesmente não tolera defeitos os algoritmos precisam apenas detectá-lo; além disso, o impacto do erro na identificação destes defeitos é menor, visto que ocorrem com menor frequência. Para o caso do Brasil, um algoritmo completo teria que ser capaz de identificar cada tipo de defeito e também seu grau. Ademais, o impacto da classificação incorreta é maior.

2.2 Precisão da Classificação por Humanos

Uma das primeiras perguntas com a qual nos deparamos quando vamos desenvolver um algoritmo de classificação é quanto à precisão requerida. No caso da classificação de frutas as normas nos fornecem margens de erro que são toleradas mas uma informação ainda mais relevante seria o desempenho de um classificador humano (haja vista que será um humano quem determinará se a tolerância foi ou não violada).

Os autores de [29] fazem um estudo detalhando do desempenho de humanos na classificação segundo as normas Belgas. Duas variedades de maçã foram usadas: Jonagold, uma variedade bicolor, e a Golden Delicious, monocromática (Figura 2.2 na página seguinte). No artigo, os autores estudam a influência de três fatores na classificação: cor, forma e tamanho. O conjunto de teste incluía 150 maçãs da variedade Golden Delicious e 900 da variedade Jonagold. As maçãs foram escolhidas de modo que os parâmetros de cor fossem bem representados em toda a sua faixa. Todas as maçãs foram classificadas duas vezes por um especialista, sob iluminação uniforme e fundo preto. Em seguida uma matriz de confusão foi construída. O resultado mais interessante é que a inconsistência[1] combinada (levando-se em conta todos os fatores que determinam uma classificação) da

[1] A classificação é dita inconsistente se atribui a uma mesma fruta classes diferentes quando reapresentada ao classificador.



(a) Fuji (b) Gala

(c) Jonagold (d) Golden Delicious

Figura 2.2: Algumas das principais variedades de maçã no mercado. A Fuji e a Gala são as mais cultivadas no Brasil. A Jonagold e Golden Delicious são as mais populares no mundo.

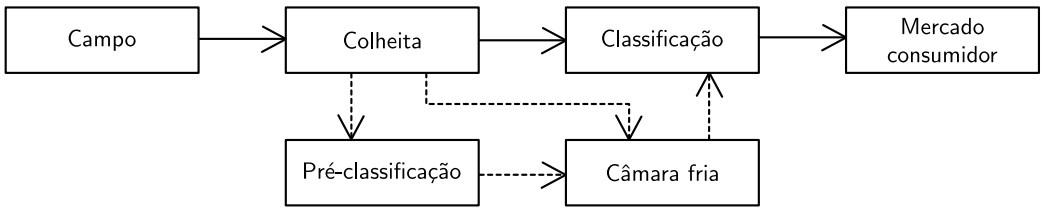


Figura 2.3: Fluxo de produção.

variedade Jonagold foi de 53.8% e para Golden Delicious foi de 35.5%. Estes testes foram conduzidos em ambientes controlados, bem iluminados e apresentando-se uma fruta de cada vez; como no ambiente de classificação a luz não é controlada, várias frutas têm de ser inspecionadas ao mesmo tempo e fatores como cansaço diminuem o desempenho do classificador humano, esperamos que a taxa real de inconsistência suba consideravelmente.

Para lidar com o erro humano o que vemos em barracões de classificação é que a mesma maçã é avaliada por mais de uma pessoa. Em um barracão que visitamos empregavam-se 60 pessoas para classificar de 17 a 22 toneladas de fruta por hora, sendo que a fruta poderia estar pré-classificada por tamanho e cor. Se considerarmos uma fruta média com 250g, um humano classificava de 3.7 a 4.7 frutas por segundo. Neste barracão uma mesma fruta poderia passar por até 10 pessoas para obter sua classificação final.

2.3 Fluxo de Produção

O fluxo da produção da fruta pode ser visto na Figura 2.3. Este esquema é geral, variantes existem principalmente em função da espécie cultivada e do tamanho do produtor. A dificuldade de automatização requer a intervenção humana em várias fases da processo. A primeira fase é a da colheita, que hoje é totalmente manual, embora exista pesquisa no sentido de criar colheitadeiras robóticas.

Após a colheita, a fruta pode seguir dois caminhos: a câmara fria ou classificação. Se for determinado que ela vai para a câmara fria, ainda existe a possibilidade de uma pré-classificação, que em geral é feita por peso. O que determina se uma fruta vai ou não para a câmara fria é principalmente o mercado consumidor e o tamanho do produtor. Para produtores pequenos, a opção de câmara fria é praticamente inexistente, a não ser que ele venda a produção para um produtor maior. Já para produtores maiores, é economicamente viável manter a fruta armazenada sob atmosfera controlada (para retardar a maturação) e vender ao longo do ano, aproveitando os preços mais altos da entre-safra. A fruta não é classificada antes de entrar na câmara fria porque sua qualidade se altera depois de meses de armazenamento devido a apodrecimento, mudança no grau de maturação e desidratação. Outro fator que faz com que o produtor deixe a classificação para depois da

câmara fria é que o processo de classificação acaba machucando a fruta, e estes machucados causariam seu apodrecimento durante o armazenamento. As frutas são armazenadas em caixas grandes (contendo aproximadamente 300kg de fruta) e reutilizáveis, chamadas *bins*.

A classificação recebe as frutas em *bins*, oriundas da colheita ou da câmara fria. O que é feito normalmente é retirar as frutas dos *bins* e colocá-las em esteiras rolantes para que pessoas, máquinas ou uma combinação dos dois avaliem cada fruto e atribuam a ele uma classe e calibre. Depois da classificação as frutas são embaladas e enviadas aos mercados consumidores.

2.4 Tipos de máquinas

O maquinário empregado na classificação vai desde mesas de classificação, onde humanos classificam a fruta visualmente e através do tato, passando por máquinas mecânicas e eletrônicas até chegar a máquinas computadorizadas. Para todas estas configurações é necessária a intervenção humana em algum grau.

As máquinas mecânicas conseguem fazer apenas a classificação por peso através de balanças instaladas embaixo de uma esteira que transporta a fruta individualmente. Este tipo de mecanismo é bastante impreciso e o mais provável é que desapareça.

Um pouco mais precisas mas ainda restritas à classificação por peso são as máquinas eletrônicas, que substituem o mecanismo de balanças por um sistema eletrônico. Este sistema de medição consegue precisões maiores ao fazer várias aferições e determinando o peso através de algum tipo de média. A medição eletrônica de peso é bastante comum ainda hoje em máquinas de médio porte, devido à sua precisão e ao baixo custo, visto que não necessitam de computadores, câmeras e software sofisticado.

Classificação por outros parâmetros que não peso são possíveis apenas com máquinas computadorizadas. Este tipo de equipamento consegue fazer medições de várias naturezas empregando sensores e algoritmos sofisticados. Um requisito comum é que os sensores não devem tocar a fruta para não danificá-la e também para maximizar o desempenho (frutas classificadas por segundo). Cada tipo de sensor consegue medir um conjunto de características:

- Câmera colorida: cor, forma, volume, presença de defeitos externos
- Microondas: firmeza, teor de açúcar
- Sonar: firmeza
- Câmera infra-vermelho: presença de defeitos externos e internos, teor de açúcar
- Raio-X: defeitos internos

Existe ainda uma divisão no tipo de máquinas: as usadas para pré-classificação e as usadas para classificação. As máquinas para pré-classificação não precisam trabalhar com tantas características, mas primam pela integridade da fruta, por esta razão dispensam ao máximo as esteiras rolantes em favor de tanques com água onde a fruta flutua através de pequenos canais. Outra diferença destas máquinas é que o destino final da fruta é o *bin* e não a caixa de papelão. Esta classe de máquinas trabalha geralmente apenas com o tamanho da fruta. As máquinas de classificação empregam todo tipo de sensor, trabalham com esteiras onde as frutas giram e são transportadas paras suas embalagens finais.

2.5 Defeitos da Maçã

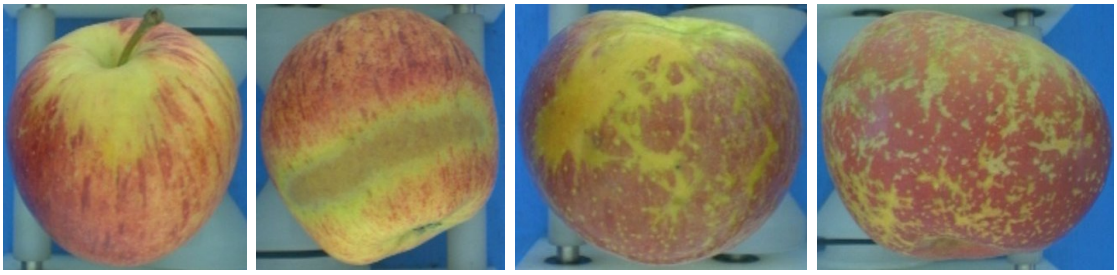
Existe um grande número de defeitos na maçã. A seguir apresentamos alguns dos principais encontrados por sistemas de classificação baseados apenas no aspecto exterior da fruta. Não incluímos defeitos internos que não fossem detectáveis pelo aspecto externo da fruta. Alguns tipos de podridão tornam o fruto muito frágil e, para evitar que se desfaçam e contaminem o ambiente de classificação, elas são manualmente retiradas antes de chegarem aos sensores para classificação. Estes tipos de podridão também não foram incluídos.

As descrições apresentadas a seguir se concentram principalmente no aspecto visual do defeito e o intuito é mostrar a grande variabilidade dos defeitos que devem ser identificados por sistemas automatizados.

Um dos principais defeitos encontrados na maçã é o *russeting*. As principais características são uma alteração na textura da casca, que se torna mais áspera, e sua cor amarelo escuro. Sua gravidade varia conforme a localização e extensão. Dependendo da causa, o contorno e localização do defeito podem variar consideravelmente. É natural de muitas variedades o aparecimento de *russeting* perto do pedúnculo, quando o contorno assume um aspecto de fractal. Esta manifestação do defeito é a menos problemática. Em alguns casos ela pode aparecer na região equatorial da fruta, com um contorno mais suave, um caso mais grave do defeito. Em outras, assume um desenho de uma rede que envolve a fruta. Uma outra manifestação do defeito aparece como pequenas manchas estreladas. Na Figura 2.4 podemos ver exemplos de frutas atingidas por *russeting*.

Outro defeito importante é o dano mecânico, causado por batidas, sua principal característica é um escurecimento da casca (Figura 2.5). Seu contorno assume formas variadas e depende do objeto que atingiu a fruta. É comum na forma globular (batida com um objeto plano) ou com algumas partes retas (batidas em quinas). Uma outra manifestação comum é nas extremidades da fruta quando estas são forçadas contra as paredes do *bin*, aparece em forma de anel.

Rachadura peduncular é um exemplo de defeito que, que como o próprio nome diz,



(a) na região do pedúnculo (b) na região equatorial (c) em forma de rede (d) em forma de estrela

Figura 2.4: Exemplos de *russeting*.

Figura 2.5: Exemplo de dano mecânico.

ocorre apenas no pedúnculo. Se manifesta como uma rachadura que envolve o cabo da fruta e tem a forma estrelada (Figura 2.6).

Queimadura de sol é um defeito que altera a matiz da cor da casca, fazendo-a puxar para o amarelo. Em casos menos severos a textura da casca ainda é visível. Seu contorno é difuso. A Figura 2.7 mostra imagens do defeito.

Sujeira de mosca é um defeito (que apesar do nome é causado por fungos e não por excreta de moscas) que se manifesta como agrupamentos de pontos pretos, com diâmetro de aproximadamente 1mm (Figura 2.8).

Mancha de cochonilha ou escama de São José se manifesta como manchas, de aproximadamente 5mm de diâmetro, sem um contorno definido, com o interior um vermelho intenso ou roxo e as bordas um vermelho mais suave (Figura 2.9).

Pequenas rugas na superfície aparecem em frutas desidratadas (Figura 2.10). Este é um tipo de defeito particularmente difícil de detectar pois não causa uma mudança de cor nem de forma geral da fruta.

O dano de lagarta enroladeira é uma mancha escura e sinuosa, com as bordas interiores pretas. Exemplo do problema pode ser visto na Figura 2.11.

Rachaduras na fruta podem ser causadas por vários motivos, um deles é o excesso de umidade. O problema não altera a cor da fruta mas expõe sua polpa. A Figura 2.12

Figura 2.6: Exemplo de rachadura peduncular.

Figura 2.7: Exemplo de queimadura de sol.

Figura 2.8: Exemplo de sujeira de mosca.

Figura 2.9: Exemplos de mancha de cochonilha.

Figura 2.10: Exemplo de fruta desidratada.

Figura 2.11: Exemplo de dano de lagarta enroladeira.

Figura 2.12: Exemplo de rachadura.

Figura 2.13: Exemplo de fuligem.

mostra um exemplo.

Um outro defeito é a fuligem, que aparece como manchas escuras na casca. A textura original da casca ainda é visível mas sua cor fica mais escura. Uma imagem do defeito pode ser vista na Figura 2.13.

A legislação brasileira [10] faz distinção entre os tipos de lesão cicatrizada (se apresenta como um escurecimento da casca), sendo um dos fatores que diferenciam este tipo de defeito o seu relevo. Defeitos que alteram a forma da fruta são considerados mais graves. Exemplos de lesão cicatrizada leve e grave podem ser vistos na Figura 2.14.

Um outro defeito interessante é a podridão carpelar, um defeito interno que altera a forma da fruta de uma maneira característica. A maçã cria vincos no sentido longitudinal, assumindo uma forma gomada (Figura 2.15).

Outras manchas que podem ser encontradas nas frutas, mas que não são defeitos, são: folhas, lenticelas e mudanças bruscas de cor. Sistemas de classificação têm de ser capazes de diferenciar, ou ignorar, estes tipos de manchas. A Figura 2.16 mostra alguns exemplos de manchas que não são defeitos.

(a) lesão cicatrizada leve (b) lesão cicatrizada grave vista de lado (c) lesão cicatrizada grave vista de cima

Figura 2.14: Exemplo de lesão cicatrizada.

Figura 2.15: Exemplo de podridão carpelar.

(a) Folha (b) Mudança brusca de cor (c) Mudança brusca de cor (d) Lenticelas

Figura 2.16: Exemplos de manchas que não são defeitos.

Capítulo 3

Revisão Bibliográfica

Apresentamos neste capítulo uma revisão dos principais trabalhos sobre a classificação de frutas. Dividimos o capítulo em seções que refletem cada uma das etapas do processo de classificação. De grosso modo, os algoritmos de classificação consistem das seguintes etapas:

1. Aquisição de imagens.

2. Localização da fruta na imagem usando alguma técnica de subtração de fundo.

3. Localização de cálice e pedúnculo.

   Não são todos os métodos que tratam este problema de maneira separada, muitos lidam com as extremidades como se fossem um tipo de defeito que não influência na classificação.

4. Localização de defeitos e classificação da fruta.

   Algumas técnicas não tentam segmentar o defeito. Das que segmentam o defeito, poucas tentam classificar os diferentes tipos de defeito.

3.1 Sistema de Aquisição de Imagens

Os autores de [21, 22] construíram um sistema de captura composto de um túnel de luz com uma câmera na parte superior da máquina. O túnel é composto por um cilindro com o interior pintado de branco, as lâmpadas são posicionadas lateralmente e verticalmente abaixo do plano do *carrier*. Duas camadas de difusores são usadas, a primeira entre a parte superior da máquina e as lâmpadas e a segunda entre a parede refletora e a fruta. Em artigos posteriores [23, 20] os autores alteraram o sistema para conter duas câmeras

apontando para dentro de um ângulo de 45° com a horizontal e 90° entre elas. A vantagem deste segundo sistema é que captura imagens de ambas as extremidades do fruto.

Os autores de [6] trabalham com um sistema elaborado de manipulação da fruta que consiste de um braço robótico com ventosa na extremidade. O bastão se prende à fruta e a faz girar dentro de uma câmara semi-esférica com iluminação controlada e uma câmera no topo. As imagens captadas são de alta qualidade e o ângulo de aquisição é controlado, no entanto a parte mecânica do sistema é muito elaborada, o que o tornaria mais caro e sujeito a falhas, além disso a necessidade de isolar cada fruta reduziria a velocidade de inspeção. Outro fator negativo é que cada câmera no sistema seria capaz de avaliar uma única fruta por vez.

Em [24] o sistema é composto por duas câmeras coloridas, uma em cima e outra embaixo, sendo que nesta máquina as frutas não giram, mas um sistema de espelhos e copo vazado garantem a captura de quatro imagens de cada fruto. Algumas desvantagens do sistema é que precisa de duas câmeras para avaliar cada fruto e a câmera que fica embaixo acumula sujeira na lente. Além disso, o uso de um jogo de espelhos nas laterais das esteiras desperdiça espaço de sensor (ao invés dos espelhos outras duas linhas poderiam ser observadas pela mesma câmera).

O sistema apresentado em [9] emprega câmara em formato de "V" invertido. Na parte superior encontram-se as câmeras e nas paredes, lâmpadas fluorescentes. As paredes são pintadas de branco para espalharem a luz. O sistema não emprega uma camada difusora.

Uma característica que é comum a todos os sistemas estudados é o controle da iluminação; todos os sistemas montam de alguma forma um ambiente que garanta iluminação difusa e uniforme.

3.2 Localização de Pedúnculo e Cálice

O autor de [38] usa duas imagens para localizar o cálice e pedúnculo. Na primeira imagem, monocromática, são localizadas regiões escuras, tanto defeitos quanto as duas extremidades da fruta são selecionadas nesta etapa. Na segunda imagem a fruta é iluminada com um padrão de listras, a idéia é determinar de maneira grosseira a geometria da região pela curvatura das listras. Para cada região escura da primeira imagem o autor extrai um conjunto de descritores das duas imagens (18 no total) e usa uma rede neural para determinar se a região é concava ou convexa. Regiões côncavas são classificadas como sendo uma das extremidades da fruta. Testes de laboratório apresentaram 95% de acerto na detecção. Um problema com este método é que não é necessariamente aplicável a outras frutas como kiwi, caqui e tomate, pois estas frutas não apresentam cálice e pedúnculo côncavos. Outro problema é que alguns defeitos também podem deformar a fruta e formar regiões côncavas.

De forma semelhante, os autores de [4] localizam as extremidades da fruta localizando depressões da fruta. Para determinar a geometria os autores empregam uma técnica de reconstrução tridimensional chamada *shape-from-shading*, em que a forma do objeto é determinada pela maneira como ele interage com a luz. Se supusermos um modelo lambertiano de interação com a luz e uma fonte distante, em que os raios cheguem paralelos até a superfície do objeto, podemos modelar a intensidade registrada no *pixel* $\boldsymbol{x}$ através da equação

$$I(\boldsymbol{x}) = l\rho\boldsymbol{n} \cdot \boldsymbol{s}, \tag{3.1}$$

onde l é a intensidade da fonte luminosa, ρ é o albedo da superfície, $\boldsymbol{n}$ é a normal à superfície, $\boldsymbol{s}$ é a direção da fonte luminosa, e $\cdot$ é o produto escalar. Através de algumas manipulações e simplificações desta equação os autores do artigo conseguem resolvê-la usando uma transformada de Fourier e obtêm uma reconstrução aproximada da superfície. Para localizar o pedúnculo e o cálice os autores usam o gradiente da superfície, regiões com gradientes mais altos são classificadas como extremidades da fruta. Um problema com o método aparece porque ele supõe que o albedo ρ é constante ao longo da superfície. O problema ocorre com defeitos, que tem um albedo diferente e aparecem como regiões em relevo, quando na realidade não são. Os testes apresentados aparentemente foram feitos com frutas secas (com a superfície livre de água), esperamos que problemas semelhantes aconteçam se frutas molhadas forem analisadas. O algoritmo também depende de uma fase de calibração para a determinação da direção da iluminação.

Já os autores de [20] usaram uma técnica mais simples: correlação normalizada com dois padrões, um obtido da média de vários cálices e outro da média de vários pedúnculos. Um problema com este método é que depende da posição da fruta, ou seja, só detecta as partes da maçã se esta estiver em posição semelhante à presente nos padrões usados para a correlação. O método também terá dificuldades com variedades bicolores.

Um outro método é usado em [24] e combina descritores baseados em dimensão fractal. Para determinar se uma região é ou não o cálice ou o pedúnculo os autores usam redes neurais.

Em [9] duas faixas do espectro infra-vermelho são capturadas. As frutas são resfriadas antes de passar pelo sistema de inspeção. A diferença de temperatura entre as extremidades e as outras partes da fruta são evidenciadas por uma das faixas do espectro usadas. Na outra faixa do espectro defeitos e extremidades são detectados, mas com a detecção na primeira imagem os autores conseguem diferenciar entre os dois tipos de manchas. Um problema com este método é que depende do resfriamento da fruta, o que nem sempre é possível.

O método apresentado em [30] detecta distorções nos padrões de reflexão especular para encontrar regiões côncavas. Os autores usaram três lâmpadas fluorescentes e o reflexo delas na casca da fruta foi analisado. Foi usada luz azul porque esta faixa é a menos usada

por outras técnicas para a localização de defeitos. Um problema com esta técnica é sua dependência na reflexão da luz: frutas molhadas, sujas, ou sem a cera da casca vão produzir outros padrões de reflexão.

Os autores de [3] diferenciam defeitos das extremidades da fruta ao rastrear as manchas escuras ao longo de várias imagens. As manchas causadas pelas extremidades são sombreadas em seu interior devido à depressão nessas regiões. A localização das sombras muda enquanto a fruta gira dentro do aparato de captura. As manchas causadas por defeitos se mantêm inalteradas nas várias imagens. Para detectar estes dois padrões os autores rastrearam as manchas ao longo de várias imagens. Em seguida as extraíram e redimensionaram de modo a corrigir a deformação imposta pela curvatura da fruta. As manchas corrigidas foram alinhadas e sobrepostas. Para manchas causadas por sombras existia bastante desacordo entre as várias imagens (regiões que estavam escuras em uma imagem estavam claras em outra), já para manchas verdadeiras a concordância era maior.

3.3 Localização e Classificação dos Defeitos

Um método para a segmentação dos defeitos é proposto em [21]. O algoritmo é composto por duas etapas: na primeira uma segmentação grosseira, baseada em estatísticas da população, e na segunda um refinamento, baseado em características locais da casca e/ou estatísticas de ordem da fruta. Para a primeira etapa, uma matriz de co-variância das componentes de cor é obtida de um conjunto de imagens de frutas sem defeitos. Esta matriz é utilizada para calcular a distância de Mahalanobis de cada *pixel* da imagem, sendo que ao invés de usar a média da população os autores usam a cor média da imagem sendo avaliada, isto confere robustez à iluminação.

Para a segunda fase, os autores desenvolveram dois algoritmos, que podem ser usados independentemente ou em sequência. Os algoritmos da segunda fase usam a segmentação obtida na primeira fase para produzir uma segmentação mais refinada. O primeiro método calcula estatísticas de ordem da fruta sendo avaliada para caracterizar a classe tecido sadio e tecido com defeito. Dadas as distribuições ele determina a distância de cada *pixel* até cada uma destas distribuições, atribuindo ao *pixel* a classe mais próxima. O segundo método trabalha com uma janela ao redor do *pixel*. Para cada *pixel* calcula a cor média do tecido sadio e tecido com defeito, dentro da janela e excluindo o *pixel* sendo avaliado. O *pixel* é alocado à classe mais próxima, tomando como distância a distância L_1 até o vetor de média.

Em [6], um modelo probabilístico é criado para detectar não só tecido sadio mas também defeitos, cálice e pedúnculo da fruta. O treinamento do sistema é feito por um especialista que classifica cada região da casca de um conjunto de frutas em: cor primária, cor secundária, dano do tipo I, dano do tipo II, cálice, pedúnculo e *russeting*. Dadas estas

rotulações os autores criam um modelo bayesiano para cada tipo de região, usando apenas as três componentes de cor de cada *pixel*. Dadas as imagens da fruta a ser classificada, o primeiro passo é determinar a qual classe pertence cada *pixel*. Classificados os *pixels*, extraem-se componentes conexos e características destes componentes são extraídos conforme a classe da região (e.g. para defeitos o eixo principal do defeito e a área, para cor primária apenas a cor média). Os testes de validação foram feitos com pêssegos, laranjas e maçãs (variedade monocromática Golden Delicious). Embora tenham obtidos resultados bons (e.g. a classificação dos *pixels* na classe de defeitos para maçã foi de 100%) a análise dos resultados é bastante limitada e o artigo não menciona que tipos de defeitos estavam presentes no conjunto de testes.

O trabalho feito em [34] é voltado para a indústria madeireira mas é aplicável ao problema de segmentar defeitos. Aqui a imagem original foi subdividida em pedaços de 32×32 *pixels*. Cada um destes pedaços foi classificado usando um mapa de Kohonen. O resultado deste mapa é uma imagem contendo os pedaços arranjados de maneira que aqueles correspondendo a tecido sadio estão em um dos cantos e os defeituosos no canto oposto. O operador determina uma fronteira que separa tecido doente de tecido são e o algoritmo usa esta informação para classificar outros pedaços de madeira. Uma desvantagem com esta abordagem é que o tecido sadio normalmente é muito mais comum do que o doente e se algum cuidado não for tomado para limitar o número de pedaços com tecido sadio o mapa fica extremamente enviesado (um mapa com pouquíssimas células correspondendo a tecido doente), o que por sua vez reduz a fronteira entre os dois tipos de tecido e dificulta a classificação.

Os autores de [23] desenvolveram um método capaz de distinguir vários tipos de defeitos sem a necessidade da segmentação e classificação prévia dos defeitos. A única rotulação do conjunto de treino é a classificação da maçã. Dada a imagem da fruta segmentada, o primeiro passo é separar o que é tecido sadio e o que é defeito. A tarefa é realizada usando-se um algoritmo de subtração de fundo, sendo que o modelo do fundo é baseado na cor do tecido sadio. Depois de removido o tecido sadio da imagem os componentes conexos dos defeitos são encontrados. De cada componente conexo são extraídas várias características que descrevem a geometria, cor, textura (desvio padrão da cor) e distância até as extremidades da fruta (totalizando 18 variáveis). Extraído o vetor de características cada defeito é associado a um tipo de defeito, obtido previamente com o algoritmo de k-vizinhos mais próximos, com uma probabilidade (distância de Mahalanobis). Para caracterizar a fruta como um todo dois parâmetros foram computados para cada tipo de defeito (centróide de cada *cluster)*: a somatória da probabilidade *a posteriori* da ocorrência daquele defeito e a variância das probabilidades dos defeitos associados ao *cluster*. Este novo vetor de característica era usado para associar uma classe à fruta como um todo (foram feitos testes usando redes neurais e *linear discriminant analysis*).

Em testes feitos com duas variedades, uma bicolor (Jonagold) e outra monocromática (Golden Delicious) os autores obtiveram uma taxa de acertos de 72% e 78%, respectivamente, para a classificação nas quatro categorias previstas pela legislação Belga ("Extra", "Categoria I", "Categoria II" e "Rejeitado"). Os erros se concentraram principalmente nas classes intermediárias e a classe "Extra", sem defeitos, obteve taxa de acerto de 89% e 96%. De todos os defeitos detectados, os que mais causaram erros na classificação foram batidas recentes (por apresentarem pouco contraste com a cor da fruta) nas duas variedades, extremidades confundidas com defeitos e um tipo de defeito que causava uma deformidade na casca da fruta (variedade bicolor) sem apresentar alteração da cor, o que a impedia de ser segmentada.

Capítulo 4

Aquisição de Imagens

Para realizar testes com os algoritmos desenvolvidos nesta dissertação, criamos um aparato que tenta aproximar os sistemas de aquisição mais comuns em máquinas classificadoras. Neste capítulo descrevemos a máquina que foi montada e algumas lições aprendidas com ela e também as bases de dados que foram montadas utilizando-a.

4.1 Sistema de Aquisição de Imagens

4.1.1 Hardware

Um desenho esquemático do sistema pode ser visto na Figura 4.1 na página seguinte. A câmera é posicionada imediatamente acima da fruta. Nas bases de dados que adquirimos usamos duas câmeras: uma iSight fabricada pela Apple e uma Dragonfly fabricada pela Point Grey Research Inc.[1].

Para evitar que reflexos especulares na fruta criem manchas brancas na imagem, o que poderia confundir os algoritmos de rastreamento e detecção de defeitos, é importante que a luz seja difusa. O espalhamento da luz é obtido pela camada de poliestireno leitoso posicionada entre a lâmpada e a fruta, o que chamamos de camada difusora. Este material é o mesmo usado em luminosos e uma das vantagens dele, além de sua boa capacidade de espalhar a luz, é que pode ser facilmente moldado quando quente. Uma camada refletora (de alumínio) exterior, garante que a luz das lâmpadas seja refletida para o interior da máquina; com isso diminui o número de lâmpadas necessárias além de reduzir a quantidade de luz ambiente que chega até a parte mais interior da máquina, o que torna o ambiente de captura de imagens menos suscetível a variações ambientais. Fizemos alguns testes com uma primeira versão da máquina, que usava materiais mais simples para a camada

[1]Point Grey Research Inc. `http://www.ptgrey.com`

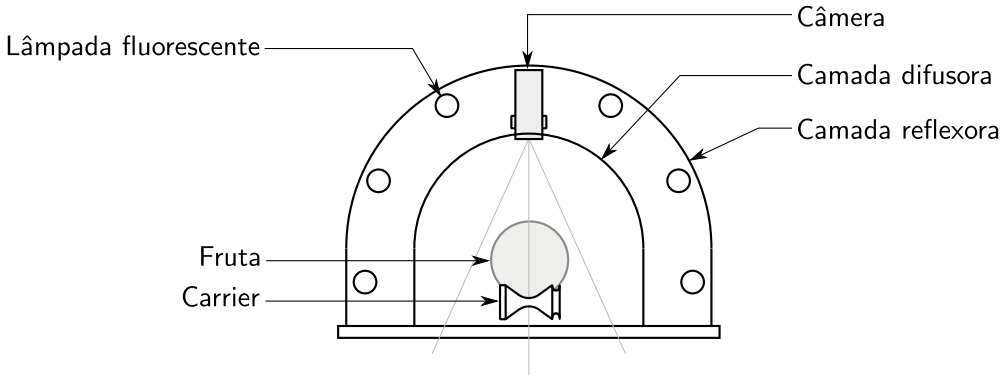


Figura 4.1: Esquema do sistema de aquisição de imagens.

(a) Luz ambiente (b) Luz fluorescente (c) Luz fluorescente + reflexor (d) Luz fluorescente + reflexor + difusor

Figura 4.2: Efeitos da luz fluorescente e das camadas reflexora e difusora.

difusora (papel manteiga). Para entender a importância de cada uma das camadas, os resultados podem ser vistos na Figura 4.2.

A fonte de luz deve ser intensa e uniforme, sua cor deve variar pouco ao longo do tempo. Por esta razão fizemos testes iniciais com lâmpadas halogênicas. Apesar de ter uma capacidade luminosa extraordinária vetamos o uso delas porque produzem muito calor, suficiente para derreter a camada difusora. Em máquinas maiores, onde a distância entre a camada difusora e a lâmpada forem maiores, talvez estas lâmpadas se mostrem mais viáveis. A alternativa que acabamos empregando são as lâmpadas fluorescentes, que consomem menos energia e operam a temperaturas mais baixas. Para evitar o problema de batimento usamos um reator eletrônico, cuja frequência de oscilação é de aproximadamente 28kHz.

Dentro da máquina a fruta pode ficar sobre um fundo plano, pintado de azul para aumentar o contraste entre a fruta e o fundo, ou então sobre um *carrier* que gira a fruta. O *carrier* é composto por duas roldanas (Figura 4.3), uma cilíndrica menor e a outra bicônica. O perfil da roldana bicônica foi projetado para que a velocidade angular

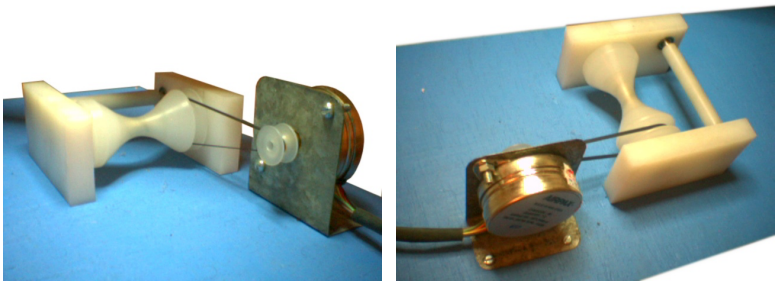


Figura 4.3: Carrier

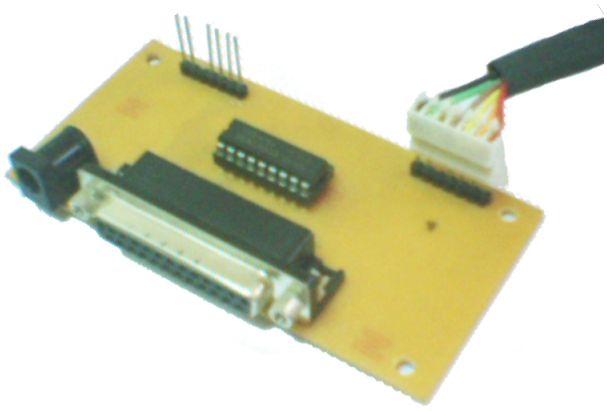


Figura 4.4: Circuito de controle do motor de passo.

da fruta (aproximada por uma esfera) fosse igual não importando o seu raio. As duas roldanas e seu suporte são feitos de nylon. Este material tem alguns problemas para nossa aplicação, uma delas é ser muito liso, o que faz com que a fruta deslize algumas vezes; a outra é sua cor — branco — que interfere na subtração de fundo, visto que é uma cor que ocorre na maçã. Também não é possível pintar a peça porque não existe tinta disponível no mercado para este material.

Ligado à roldana bicônica está um motor de passo unipolar[2]. Decidimos usar este tipo de motor porque permite um controle fino da rotação e a interface dele com o computador é simples de realizar. Sua conexão com o computador é feita através da porta paralela. Para controlar os sinais da porta paralela usamos a biblioteca parapin[3]. A placa junto com o motor de passo e o cabo paralelo pode ser vista na Figura 4.4.

[2]Jones on Stepping Motors `http://www.cs.uiowa.edu/~jones/step/`

[3]parapin — a Parallel Port Pin Programming Library for Linux `http://www.cs.uiowa.edu/~jones/step/`

4.1.2 Software

Para controlar a máquina durante a aquisição das imagens foi desenvolvido um programa usando a biblioteca OpenCV[4] para manipulação de imagens e interação com a câmera, GTKmm [5] para interface gráfica e *threads* e parapin para controle da porta paralela. Outra biblioteca que usamos foi a libraw1394[6] para controlar o foco e ganho da câmera iSight.

O programa é capaz de controlar o motor, sendo que a quantidade de passos que o motor dá é controlada pelo usuário. O usuário também pode determinar o número de fotos que devem ser tiradas. Um recurso que se mostrou bastante útil foi a detecção automática da fruta, que dispara a aquisição de imagens (pode ser desabilitado pelo usuário) e, quando a captura é finalizada, o armazenamento das imagens em disco (o que também pode ser desabilitado pelo usuário). O princípio de funcionamento da detecção de frutas é bastante simples e é baseado na subtração de fundo. Primeiro o usuário posiciona uma fruta dentro da máquina e configura a subtração de fundo. O usuário fornece uma máscara, que separa de grosso modo a imagem em fundo e objeto. A máquina gira a fruta e acumula imagens que serão usadas pelo subtrator de fundo para a fase de aprendizado. Uma vez configurado o subtrator de fundo, o usuário estabelece uma área mínima que acusa a detecção da fruta. Duas capturas de tela do software podem ser vistas na Figura 4.5 na página oposta.

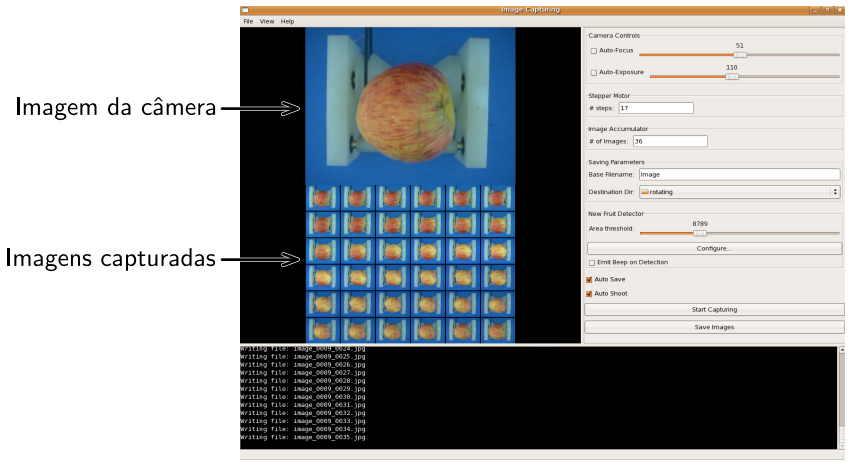
Para evitar que a interface ficasse congelada quando o motor está sendo acionado ou quando as imagens estão sendo armazenadas em disco usamos várias *threads*. Uma controlava o motor, outra salvava as imagens no disco e uma terceira cuidava da interface gráfica. Isso permitiu que tirássemos fotos ao mesmo tempo em que as imagens da fruta anterior estivessem sendo descarregadas para o disco rígido. O uso de *threads* e a detecção automática das frutas permitiu que a captura de imagens fosse muito rápida.

4.2 Bases de Dados

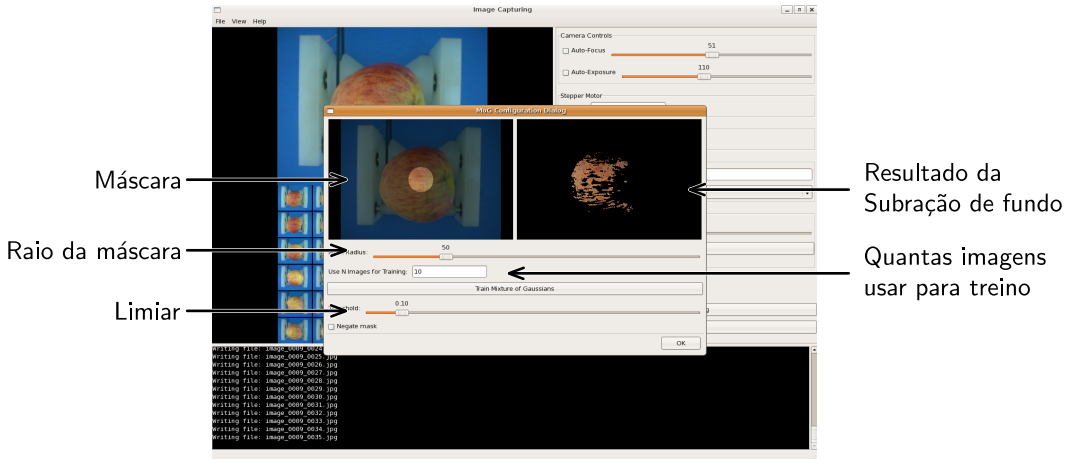
Com o sistema descrito na seção anterior construímos várias bases de dados, usando diversas variedades, frutas com defeitos e frutas com poucos defeitos, imagens adquiridas com as duas câmeras, frutas de diversos produtores, frutas girando de diversas maneiras e frutas com marcadores. O nome completo de cada base segue o padrão

[numero da base]_[nome],

[4]Open Computer Vision Library `http://sourceforge.net/projects/opencvlibrary`
[5]gtkmm — C++ Interfaces for GTK+ and GNOME `http://www.gtkmm.org/`
[6]IEEE 1394 for Linux `http://www.linux1394.org/`



(a) Tela de captura de imagens.



(b) Tela de configuração da detecção de fruta.

Figura 4.5: Software de captura de imagens.

onde "número da base" é um número de série de dois dígitos indicando a ordem de criação da base. É este o nome do diretório que contém a base completa. Em alguns casos também adicionamos "`_95jpg`" ao nome da base para indicar que as imagens foram armazenadas no formato jpeg usando qualidade de 95%. Como exemplo, o nome da primeira base de dados é `01_RoyalGala_95jpg`. Dentro do diretório principal de cada base de dados existem dois diretórios: `derived` e `original`. O primeiro contém dados que foram gerados a partir da base, são o resultado de experimentos; no segundo diretório está a base criada em sua forma original. Informações resumidas das principais bases de dados podem ser vistas na tabela 4.1 na próxima página.

Nome	Características	Variedades	Número de Frutas (aprox.)	Fotos por fruta	Posição da Fruta	Câmera	Tamanho da Imagem (pixels)	Defeitos	Extra
01_RoyalGala_95jpg	O número da fruta é consistente através dos diretórios (i.e. a fruta 1 no diretório rotating_sideways é a mesma do diretório rotating_random_axi). Em algumas sequências o sistema de autofoco da câmera faz com que a amplificação da imagem mude.	Royal Gala	290	35	r, s, pd, cx	iSight	640 x 480	Algumas fruta batidas	Imagens das frutas em uma esteira, usado para os testes com o rastreamento da fruta.
02_Agropel_95jpg	Algumas frutas são fotografadas mais do que uma vez. Isto ocorre porque algumas vezes a fruta não girava na primeira tentativa ou porque nem todos os defeitos eram visíveis com apenas um eixo de rotação. Algumas frutas estão muito molhadas. A segunda parte do nome da base corresponde ao nome da propriedade onde as imagens foram colhidas.	Royal Gala Fuji Suprema Imperatriz	1,800	36	r			Raspão Batida Sujeira de mosca	
03_Renar_95jpg		Fuji Suprema Granny Smith	700	36	r			Fuligem Sarna	
04_Fischer_95jpg		Fuji Red Fuji Gala Francesa	800	36	r			Vários tipos de podridão (branca, amarga, etc.) Excesso de umidade (a fruta racha) Queimadura de sol Dano de lagarta enroladeira Ausência de pedúnculo Rachadura peduncular Podridão carpelar Rusting (causada por geada, natural da variedade, etc.) Glomerela Bitter Pit Lesão aberta Lesão cicatrizada Mancha de sarna Desidratação Falta de cor Defeito de polinização	
06_appleWithDots	Fotos de maçã verde com marcadores. Rotação lenta.	?	1	200	s, p	PointGrey	1024 x 768	Nenhum	Imagens do padrão de calibração.
07_appleWithDots_02		?	1	200	s			Nenhum	
08_appleWithDots_03		?	1	250	r, s, p			Nenhum	Imagens do padrão de calibração.
10_appleWithDots_04		?	1	250	r, s, p			Nenhum	Imagens do padrão de calibração.
12_appleWithTexture_01	Fotos de maçã bicolor com textura. Rotação lenta.	?	1	250	r, s, p			Nenhum	Imagens do padrão de calibração.
16_appleWithTextureOnBlueBackground_01	Fotos de maçã bicolor com textura. Rotação lenta. Fundo azul.	?	3	150	r,s,p	iSight	640 x 480	Nenhum	Imagens do padrão de calibração.

Posição da Fruta: s - eixo de rotação paralelo ao plano da imagem, p - eixo de rotação perpendicular ao plano da imagem, r - eixo de rotação aleatório, cx - imagem do cálice, pd - imagem do pedúnculo.

Tabela 4.1: Dados das principais bases construídas.

Capítulo 5

Subtração de Fundo

Nos algoritmos que apresentaremos nos próximos capítulos fazemos uso da subtração para separar a fruta dos objetos à sua volta.

Subtração de fundo é um problema de classificação com duas classes: fundo e objeto. A entrada do algoritmo é uma imagem e a saída é uma máscara que diz quais *pixels* pertencem ao objeto e quais pertencem ao fundo.

Como o ambiente de aquisição de imagens é controlado, podemos escolher as cores dos objetos ao redor da fruta de modo a maximizar o contraste, facilitando assim a subtração de fundo. No entanto, a variabilidade de cores dos produtos inspecionados pede o uso de algoritmos flexíveis. Apesar do fundo se movimentar, a cor controlada dispensa a necessidade do uso de algoritmos mais sofisticados como os apresentados em [31].

Um dos algoritmos mais simples para a subtração de fundo e que obteve bons resultados em nosso caso, é o de *histogram-backproject*. Do ponto de vista formal, o algoritmo modela a função de densidade de probabilidade de cor (ou alguma outra informação disponível) do fundo e/ou objeto por uma função não-paramétrica. O modelo não paramétrico é simplesmente um histograma e sua construção é realizada com imagens previamente segmentadas. Uma segmentação grosseira é o suficiente. No nosso caso usamos um círculo centrado na fruta ou na esteira rolante. Para cada *pixel* da imagem selecionado pela máscara, incrementamos o *bin* correspondente no histograma. Depois desta fase, aplicamos um limiar aos valores do histograma, que se torna binário. Para calcular a máscara basta consultar o histograma binarizado, seu valor é a resposta final.

Um problema que verificamos foi que reflexos das cores do fundo na maçã faziam com que estas regiões fossem classificadas como fundo. A solução encontrada foi usar os valores do histograma para calcular a média e co-variância dos valores observados. Dadas essas duas variáveis substituímos os valores do histograma pela distância de Mahalanobis [5] do ponto original (ver Algoritmo 5.1). Este ajuste permite uma extrapolação dos valores da cor e agora pontos que antes eram erroneamente classificados como fundo passam a ser

classificados como objeto.

Algoritmo 5.1 Ajustar uma gaussiana sobre o histograma.

procedimento AjustarGaussianaSobreHistograma($\boldsymbol{H}$)

Entrada:
$\boldsymbol{H}$: histograma no espaço RGB
Saída:
$\boldsymbol{H}'$: o histograma $\boldsymbol{H}$ com os valores substituídos pela distância de Mahalanobis

$s \leftarrow 1$
$accum \leftarrow 0$
$\bar{\boldsymbol{x}} \leftarrow \mathbf{0}$
para todo *bin* (i, j, k) de $\boldsymbol{H}$ **faça**
  $v \leftarrow \boldsymbol{H}[i, j, k]$
  $h \leftarrow \sqrt{v}$
  $\boldsymbol{S}[s] \leftarrow [h * i, h * j, h * k]$ ▷ $\boldsymbol{S} \in \mathbb{R}^{bins_R bins_G bins_B \times 3}$
  $\bar{\boldsymbol{x}} \leftarrow \bar{\boldsymbol{x}} + [v * i, v * j, v * k]^T$
  $accum \leftarrow accum + v$
  $s \leftarrow s + 1$
fim para
$\bar{\boldsymbol{x}} \leftarrow \bar{\boldsymbol{x}}/accum$
$s \leftarrow 1$
para todo *bin* (i, j, k) de $\boldsymbol{H}$ **faça**
  $h \leftarrow \sqrt{\boldsymbol{H}[i, j, k]}$
  $\boldsymbol{S}[s] \leftarrow \boldsymbol{S}[s] - h * \bar{\boldsymbol{x}}$
  $s \leftarrow s + 1$
fim para
$\Sigma \leftarrow \frac{S^T S}{accum}$
para todo *bin* (i, j, k) de $\boldsymbol{H}$ **faça**
  $\boldsymbol{x} \leftarrow [i, j, k]^T - \bar{\boldsymbol{x}}$
  $\boldsymbol{H}'[i, j, k] \leftarrow \sqrt{\boldsymbol{x}^T \Sigma^{-1} \boldsymbol{x}}$
fim para
retorne $\boldsymbol{H}'$
fim procedimento

O algoritmo final é simples de implementar, sua execução é extremamente rápida e robusta a variações de iluminação. Os seus parâmetros são: o espaço de cor, o número de *bins*, o limiar e as imagens com as respectivas máscaras para o aprendizado.

5.1 Resultados

Na Figura 5.3 na página 36 podemos ver os resultados da aplicação do algoritmo à imagem da Figura 5.2 para três configurações. Para os três casos usamos todas as imagens (a Figura 5.1 mostra a região da imagem que foi usada para treino) de uma sequência com

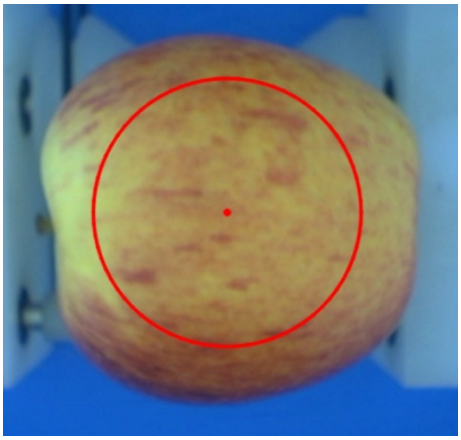


Figura 5.1: Círculo delimitando região usada para treino da subtração de fundo.

36 imagens de uma fruta girando. Cada componente do espaço de cor foi dividido em 64 intervalos para a construção do histograma.

Na primeira configuração usamos os canais H (matiz) e S (saturação) do espaço de cor HSV. Na segunda e terceira usamos os três canais de cor do espaço RGB sendo que na terceira ajustamos uma gaussiana ao valores do histograma e substituímos seus valores pela distância de Mahalanobis.

Podemos ver pelas máscaras geradas que o melhor resultado é aquele obtido com a terceira abordagem. O ajuste da gaussiana permite extrapolar as cores da região central da fruta para as bordas, mesmo com o reflexo azul do fundo. Dentre os dois primeiros o melhor resultado é o da primeira configuração, a exclusão do canal V confere uma robustez a variação de iluminação que a segunda configuração não consegue oferecer.

Descartamos o uso da distância de Mahalanobis com HS pois resulta em máscaras piores, visto que a distribuição da cor da fruta nestes canais não se aproxima de uma normal.

Os testes foram executados usando uma máquina com o processador AMD Athlon 64 X2 à 2.2GHz. As imagens eram de 640×480 *pixels*. Quando usamos o espaço RGB conseguimos uma performance de $\sim$ 180 quadros por segundo. Já para o caso em que usamos HS a performance foi de $\sim$ 90 quadros por segundo. A diferença no tempo de processamento é devido à conversão de espaço de cor que é necessária para o segundo caso (de RGB para HSV). O uso da distância de Mahalanobis não altera o tempo de processamento já que é calculado previamente e armazenado no histograma.

Figura 5.2: Imagem usada para testes da subtração de fundo.

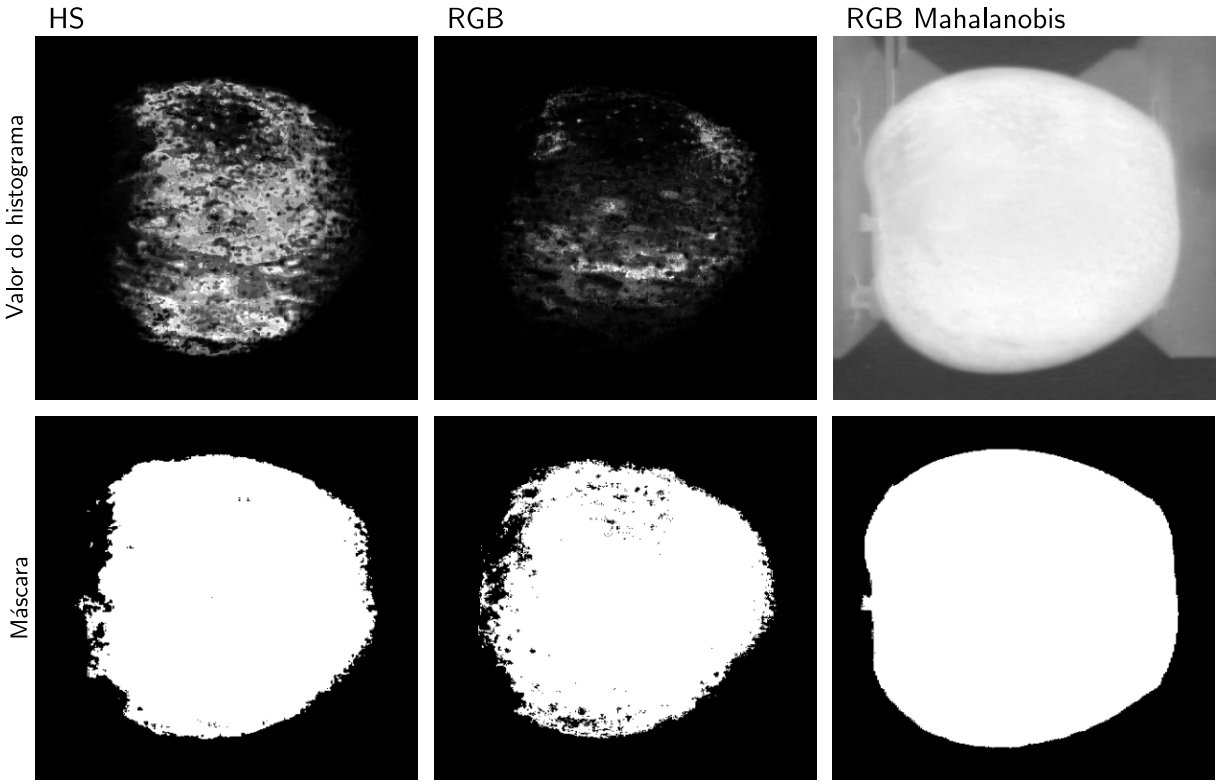


Figura 5.3: Resultados da subtração de fundo. HS significa que usamos os canais H e S do espaço HSV e RGB significa que usamos os três canais do espaço RGB. RGB Mahalanobis é o resultado obtido depois de ajustarmos uma gaussiana sobre os valores do histograma.

Capítulo 6

Rastreamento do Fruto

Descrevemos neste capítulo um algoritmo capaz de detectar e rastrear uma fruta em uma esteira mecânica. A esteira usada é mais simples do que aquelas usadas em máquinas computadorizadas e o intuito de seu desenvolvimento foi o de permitir que um sistema mais simples e barato pudesse ser usado para a classificação computadorizada.

Um dos principais problemas das esteiras mais simples é que não existe a garantia de velocidade e as frutas se movem para trás algumas vezes. Desta forma não assumimos que a velocidade da esteira era constante ou que as frutas só se moviam em um sentido.

6.1 Algoritmo

A primeira etapa do algoritmo é a localização das frutas. Usamos o algoritmo de subtração para remover a esteira da imagem. O treinamento é feito antes que as frutas entrem na esteira, este treinamento é rápido e pode ser feito toda vez que a máquina for iniciada.

Depois de removida a esteira, obtemos uma máscara em que as frutas aparecem unidas em muitos pontos. A fase seguinte é a separação das frutas na máscara. Para isso usamos a correlação com o padrão mostrado na Figura 6.1. O raio do círculo central é ajustado para coincidir com o raio médio das frutas sendo rastreadas. A região circular da máscara tem o valor 1 enquanto que a do fundo -1. É importante notar que o valor do fundo deve ser -1 e não 0. No segundo caso a máscara responderia com máxima intensidade a regiões uniformes e diferentes de zero, enquanto que no primeiro caso ela responde maximalmente a regiões circulares apenas (com o fundo mais escuro que o objeto central).

Após a correlação encontramos todos os componentes conexos e filtramos componentes com base em sua área. O que resta na máscara são pequenas manchas, uma para cada fruta. Usamos um algoritmo de fluxo óptico para rastrear estas manchas. Os pontos rastreados são os centros destas manchas.

Enquanto a fruta atravessa a esteira o sistema captura várias imagens dela, que serão

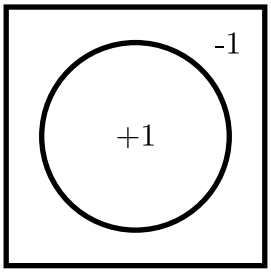


Figura 6.1: Máscara para correlação.

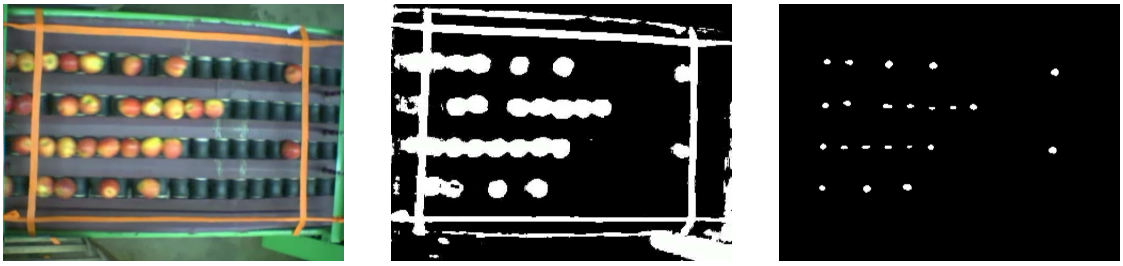
Figura 6.2: Região de descoberta e de entrega.

usadas mais tarde pelo sistema de classificação. A cada fruta dentro da esteira é associado um número identificador, desta forma conseguimos manter juntas todas as imagens desta fruta.

As extremidades da esteira são rotuladas região de descoberta (frutas entram por esta ponta) e região de entrega (frutas saem por esta ponta) (Figura 6.2).

Quando uma fruta chega à região de entrega todas as suas imagens são coletadas, junto com a sua localização e tempo presentes, e repassadas para o sistema de classificação.

Frutas que entram na esteira passam primeiro pela região de descoberta. Ao entrarem nesta região recebem um identificador. Para evitar que recebam mais do que um identificador, apagamos as manchas que correspondem a frutas previamente identificadas antes da etapa de rotulação de novos frutos.



(a) Imagem original. (b) Resultado da subtração de fundo. (c) Depois da correlação.

Figura 6.3: Resultados do rastreamento de frutos.

6.2 Resultados

Testamos o algoritmo em apenas uma sequência que foi capturada de uma esteira simples sob luz natural. Usamos uma câmera iSight que capturava imagens a 640×480 *pixels*. São registradas 93 frutas que se movem a uma velocidade aproximada de 10 *pixels*/quadro e o raio das frutas na imagem é de aproximadamente 20 *pixels*. Imagens das diversas fases do algoritmo aplicadas a um quadro da sequência podem ser vistas na Figura 6.3. O número máximo de frutas que aparecem simultaneamente na esteira é 23.

Todas as frutas foram rastreadas com sucesso, mesmo aquelas que deslizavam na esteira e se moviam em sentido oposto ao fluxo. Além das frutas, alguns artefatos da subtração eram selecionados na região de descoberta. No entanto, como sua aparição era aleatória o algoritmo de rastreamento logo os perdia e não chegavam à região de entrega.

Capítulo 7

Reconstrução Tridimensional

Apresentamos neste capítulo um algoritmo para a reconstrução completa da fruta, em contraste com técnicas apresentadas na literatura que faziam a reconstrução de partes isoladas [38, 4].

7.1 Motivação

A reconstrução completa da fruta tem várias aplicações. A mais direta é a medição do volume, o que diminui a necessidade de uma balança eletrônica. Também podemos calcular o diâmetro máximo no plano perpendicular ao eixo de simetria, informação usada em algumas normas de classificação. Outra medição que pode ser feita é o grau de simetria, frutas bem formadas tem maior valor de mercado. Podemos detectar alguns defeitos que se manifestam como uma deformação da fruta, como a podridão carpelar (a fruta assume um formato gomado) e problemas de polinização (a fruta cria um vinco profundo no sentido longitudinal).

Quando associada às imagens da fruta, podemos usar a informação da localização relativa de algumas características (longitude e latitude). A distribuição da cor é uma informação que afeta o valor de mercado, frutas que tem uma cor uniformemente distribuída tem valor maior. A gravidade de alguns defeitos varia conforme sua latitude, como o *russeting*, que é mais grave quando presente na região equatorial. Alguns defeitos ocorrem preferencialmente em algumas regiões (e.g. *russeting* e rachadura peduncular no pedúnculo, queimadura de sol na região equatorial), o que poderia ser usado por algoritmos de aprendizado para a detecção de defeitos.

A legislação brasileira diferencia os graus de vários defeitos com base no seu relevo. Defeitos salientes ou afundados são considerados mais graves que defeitos que não afetam a forma da fruta.

7.2 Restrições

Para maximizar o número de frutas inspecionadas por unidade de tempo devemos reduzir o número de imagens capturadas. Desta forma, devemos escolher algoritmos que não assumem pequenos movimentos de câmera entre as imagens.

Outra restrição é quanto ao número de correspondências, embora as principais variedades de maçã cultivadas no Brasil apresentem textura, em muitos casos encontramos frutas com pouca textura em algumas regiões (tipicamente partes que receberam pouca luz durante o desenvolvimento da fruta), por isso devemos escolher um algoritmo que use o mínimo de correspondências possível.

Como o ambiente de captura é controlado, podemos escolher a cor da esteira e a forma da iluminação de modo a maximizar o contraste entre a fruta e a esteira. Nestas condições obtemos silhuetas precisas, que nos dão mais informação sobre a forma da fruta do que as correspondências provenientes da textura. Desta forma usaremos algoritmos de reconstrução que se baseiam na silhueta da fruta.

A forma da fruta é bastante irregular, o que impede o uso de técnicas de reconstrução baseadas no casamento de padrões. Outra consequência é que não podemos assumir que a fruta se encontra alinhada no *carrier* ou que seu movimento é regular.

7.3 Revisão Bibliográfica

Uma técnica de reconstrução que foi usada com sucesso na literatura foi o *shape-from-shading* [4]. Uma limitação é que a reconstrução é por pedaços, para cada imagem obtemos uma aproximação da superfície da fruta mas não sabemos como as superfícies das diferentes imagens se encaixam. Como os autores fizeram algumas aproximações nas fórmulas, estas superfícies são versões deformadas da superfície real, o que dificulta a remontagem da fruta a partir destes fragmentos. Também não consegue lidar com a variação do albedo da fruta (reflexão especular, lama, variação da cera da fruta, fruta molhada, variações de albedo devido aos diversos tipos de defeito), que aparecem como descontinuidades na reconstrução. No entanto é rápida e permite a localização das estremidades do fruto com precisão.

Uma técnica que foi usada na reconstrução de asteroides e que poderia ser adaptada ao nosso problema foi apresentado em [7], que usa tensores trifocais para a reconstrução. Um problema com a técnica é que tensores trifocais são entidades matemáticas com 26 graus de liberdade (embora tenham 27 elementos a reconstrução recupera o objeto a menos de um fator de escala, o que reduz os graus de liberdade de um) e um número grande de correspondências deve ser obtido. Além disso, as correspondências devem ser entre três quadro. Fizemos testes iniciais com tensores trifocais e percebemos também que sua

determinação é extremamente sensível a ruído na localização dos pontos.

Existem várias técnicas de reconstrução baseadas na silhueta do objeto. Uma das mais conhecidas é a construção do *visual hull*, a intersecção dos cones formados pelo centro de projeção e as silhuetas do objeto [27]. Um problema com este método é que o objeto recuperado é facetado e um número grande de imagens é necessária para uma reconstrução precisa de superfícies suaves como as de uma fruta.

Um outro conjunto de técnicas baseadas em silhuetas são as diferenciais [19, 37], que obtêm resultados bons com superfícies suaves mas necessitam de muitas imagens e pouco movimento de câmera.

Pontos de fronteira (definidos mais adiante) são pontos especiais na silhueta que nos dão informação de profundidade. Um empecilho para o uso destas técnicas em nosso caso é que o formato da maçã resulta em um número insuficiente de pontos de fronteira para a reconstrução baseada somente neles [14].

7.4 Definições

Usamos três sistemas de coordenadas: esteira, fruta, câmera. O sistema de coordenadas globais coincide com o sistema da esteira. A esteira está contida no plano $z = 0$ com o eixo z apontando na direção da câmera (Figura 7.1 na próxima página). No sistema da câmera o eixo z aponta em direção à esteira. A origem do sistema de coordenadas da fruta é seu centro de massa. À medida que a fruta se move na esteira seu sistema de coordenadas assume um alinhamento diferente. Assumimos que este sistema é paralelo ao da esteira no instante inicial. Uma restrição ao sistema da fruta é que ela apenas tangencia a esteira, o que limita a distância entre a origem do sistema e o plano $z = 0$.

A mudança de coordenadas da esteira para a câmera é representada pela transformação afim $\boldsymbol{P} \in \mathbb{R}^{4\times4}$. A mudança do sistema da esteira para o da fruta é chamado $\boldsymbol{F}_{t_i}$. A mudança de sistema da fruta para a câmera é dado por

$$\boldsymbol{P}_{t_i} = \boldsymbol{P}\boldsymbol{F}_{t_i}^{-1}. \tag{7.1}$$

.

Em alguns momentos desejamos sinalizar a transformação entre dois sistemas de referência $\boldsymbol{P}_{t_i}$ e $\boldsymbol{P}_{t_{i+1}}$ (Figura 7.2), usamos a seguinte notação

$$\boldsymbol{P}_{t_i t_{i+1}} = \boldsymbol{P}_{t_{i+1}}\boldsymbol{P}_{t_i}^{-1}. \tag{7.2}$$

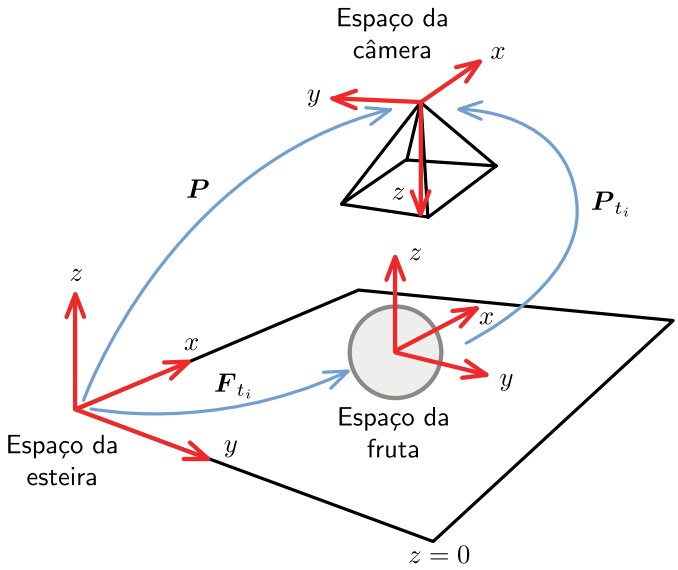


Figura 7.1: Sistemas de coordenadas da esteira, câmera e fruta.

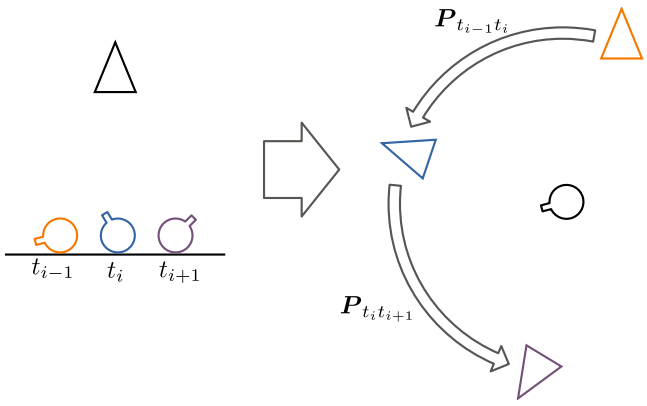


Figura 7.2: Movimento da fruta transformado em movimento de câmera.

7.5 Algoritmo

No algoritmo que apresentaremos a seguir quebramos o problema de reconstrução em duas partes: determinação do movimento de câmera e a reconstrução tridimensional. Fizemos esta divisão pois usamos entidades diferentes para resolver cada um dos sub problemas. Para determinar o movimento de câmera usamos correspondências, que embora sejam insuficientes para uma reconstrução total da fruta são suficientes para determinar os parâmetros da transformação afim que descreve o movimento de câmera. Assumimos também que os parâmetros extrínsecos e intrínsecos da câmera são conhecidos. Acreditamos que este requerimento seja razoável, visto que sua determinação poder ser feita previamente usando-se algoritmos de calibração [39] e, uma vez determinados, raramente mudam. Para a reconstrução usamos a silhueta da fruta, que oferece mais informação sobre a forma do que as correspondências.

Um problema que surge no uso das silhuetas é a determinação da profundidade de cada ponto dela. Qualquer ponto no raio que sai do centro de projeção e passa pelo ponto na silhueta poderia ter gerado a imagem do ponto. Para lidar com esta dificuldade assumimos primeiro que os pontos que geram a silhueta são coplanares e que este plano é paralelo ao plano da imagem. Embora isto seja verdade para a esfera é falso para o caso geral. No entanto, assumimos que a fruta se aproxima suficientemente de uma esfera para podermos usá-la.

Criamos duas abordagens para determinar a distância deste plano ao centro de projeção.

Na primeira, assumimos que a fruta pode ser aproximada por uma esfera. Dada a localização desta esfera no espaço e seu raio, calculamos a distância do plano que contém a curva geradora.

Na segunda encontramos pontos de fronteira e, por triangulação, determinamos a distância dos pontos até a câmara.

O algoritmo que apresentaremos nas próximas sub seções segue os seguintes passos

1. Captura das imagens da fruta na esteira.
2. Subtração de fundo.
3. Determinação de correspondências entre as imagens usando fluxo óptico.
4. Usando a informação obtida dos passos 2 e 3 fazemos uma estimativa grosseira do tamanho e posição da fruta em cada quadro e também da posição dos pontos no espaço da fruta.
5. Usando a informação do passo 4 estimamos a rotação da fruta entre dois quadros usando RANSAC [11, 13]. Junto com a informação sobre o centro da fruta e a

matriz $\boldsymbol{P}$ montamos uma estimativa para a matriz $\boldsymbol{P}_{t_i t_{i+1}}$.

6. Refinamento de $\boldsymbol{P}_{t_i t_{i+1}}$ usando o algoritmo de Levenberg-Marquardt [25]. Este passo é opcional.

7. Reposicionamento das silhuetas.

7.5.1 Fluxo óptico

Apresentamos agora o algoritmo usado para determinar a correspondência entre pontos em duas imagens na mesma sequência. Decidimos usar fluxo óptico pois é uma técnica madura e seu cálculo é rápido. O algoritmo que apresentaremos se enquadra na classe de algoritmos de fluxo óptico diferencial. O exemplo clássico pode ser visto em [26].

Como desejamos reduzir ao máximo o número de imagens necessárias para a reconstrução, exploramos algumas modificações no algoritmo original que permitiriam o rastreamento de pontos por uma distância maior.

7.5.1.1 Motivação

Um problema com o algoritmo proposto em [26] é a sua sensibilidade a iluminação. Se os parâmetros da iluminação variam conforme a sua posição na imagem (e.g. uma sombra que afeta apenas uma parte da imagem) o algoritmo clássico não consegue se adaptar a esta mudança e o ponto não pode ser rastreado.

No sistema de aquisição tomamos medidas no sentido de reduzir a variação da iluminação. Usamos difusores para espalhar a luz gerada pelas lâmpadas. A luz ambiente é bloqueada em parte por uma cobertura de metal. No entanto, quando realizamos testes notamos que existia uma mudança gradual da luz devido às aberturas nas duas extremidades da máquina. Embora seja possível reduzir um pouco o problema para o caso do nosso sistema de captura, acreditamos que o problema persistirá em máquinas comerciais.

Um modelo simples, mas que descreve de maneira satisfatória o efeito da iluminação sobre um ponto na imagem é dado pela equação

$$\breve{I}(\boldsymbol{x}) = \lambda I(\boldsymbol{x}) + \sigma, \tag{7.3}$$

onde $\lambda \in \mathbb{R}^+$ é um termo multiplicativo que afeta o contraste da imagem e $\sigma \in \mathbb{R}$ o seu brilho.

Para entender melhor a influência deste fator fizemos um experimento. Visto que a posição das lâmpadas é fixa podemos assumir que λ e σ são função apenas da posição na imagem e não do tempo. Para perceber esta influência acumulamos um histograma (usamos apenas o canal G do espaço RGB de cor) para cada *pixel*, e ao longo do tempo, de

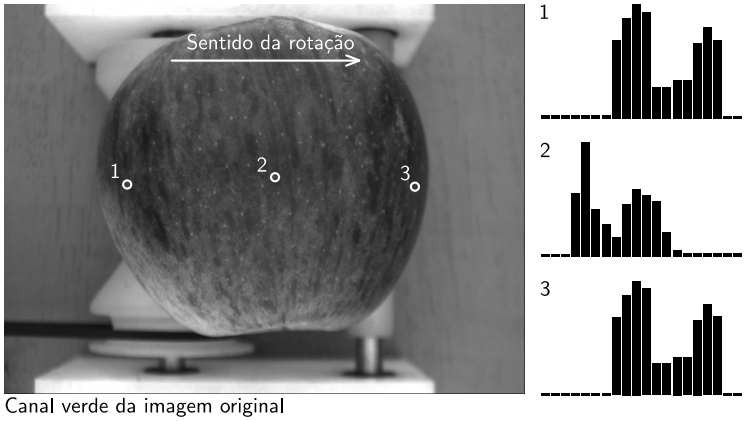


Figura 7.3: Variação de iluminação dentro do aparato de captura de imagens.

uma sequência em que a fruta gira. Desta maneira esperamos que o mesmo conjunto de pontos da fruta seja capturado por diferentes sensores ao longo da direção de movimento.

O efeito dos termos λ e σ sobre o histograma são os seguintes: o primeiro altera o intervalo de *bins* ocupados (e.g. se λ for igual a zero teremos apenas o primeiro *bin* com elementos) e o segundo corresponde a um deslocamento fixo nos valores (o formato do histograma não se altera mas sua posição no eixo x varia). A Figura 7.3 mostra uma das imagens capturadas e os histogramas para diferentes posições. O efeito de σ é muito mais pronunciado nas imagens visto que a posição do histograma ao longo do eixo x varia bastante. Já a variável λ não influencia muito, uma vez que a largura do histograma se mantém praticamente inalterada.

Outro problema com o algoritmo proposto em [26] é que ele assume um modelo translacional de movimento, o que dificulta o rastreamento de regiões que se deformam de alguma maneira (e.g. uma transformação afim). No caso da fruta, isto se manifesta como uma deformação da região em torno do ponto, à medida que ele se aproxima da borda da fruta (ver Figura 7.4).

No algoritmo que iremos propor a seguir usamos partes de dois artigos. Para incorporar o modelo de deformação esférica usamos partes de [15]. Incorporamos o modelo de iluminação da Equação (7.3) baseado em [18].

7.5.1.2 Algoritmo

Dadas duas imagens de uma cena em movimento I_{t_i} e $I_{t_{i+1}}$ e uma região $\boldsymbol{\rho}_{t_i}$ na primeira imagem, queremos determinar a localização de $\boldsymbol{\rho}_{t_i}$ na segunda imagem, o que chamaremos de $\boldsymbol{\rho}_{t_{i+1}}$ (Figura 7.5). A região $\boldsymbol{\rho}_{t_i}$ é formada pelos pontos na imagem $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_\rho}\}$, $\boldsymbol{x}_i \in \mathbb{R}^2$.

Devido ao processo de formação da imagem, ao movimento relativo entre objeto sendo

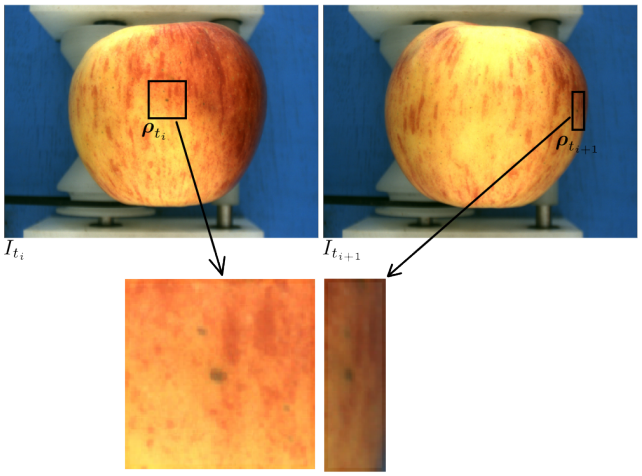


Figura 7.4: Deformação devido a curvatura da fruta.

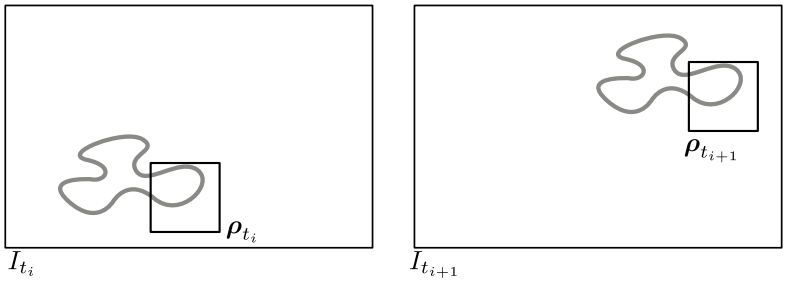


Figura 7.5: Fluxo óptico

rastreado e a câmera, e à geometria do objeto, a imagem da região rastreada se deforma ao longo do tempo. Usamos a função $\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}) : \mathbb{R}^{2+n_\mu} \rightarrow \mathbb{R}^2$ para corrigir esta deformação, onde $\boldsymbol{x}$ é a coordenada do ponto e $\boldsymbol{\mu}$ é vetor de parâmetros do modelo de deformação[1], constante para toda a região $\boldsymbol{\rho}_{t_i}$. A função $\boldsymbol{w}(\cdot)$ leva um ponto $\boldsymbol{x} \in \boldsymbol{\rho}_{t_i}$ até seu correspondente em $\boldsymbol{\rho}_{t_{i+1}}$. Daqui em diante chamaremos $\boldsymbol{\rho}_{t_i}$ de $\boldsymbol{\rho}$, já que sempre usamos $\boldsymbol{w}(\cdot)$ para recuperar a posição de um ponto no instante t_{i+1}.

Assumindo que o brilho do objeto rastreado não muda ao longo do tempo, podemos expressar a relação que existe entre a região nos dois instantes de tempo da seguinte maneira

$$I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}_{t_{i+1}})) = I_{t_i}(\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}_{t_i})). \qquad \forall \boldsymbol{x} \in \boldsymbol{\rho}$$

Impondo agora que $\boldsymbol{\mu}_{t_i} = \mathbf{0}$ e $\boldsymbol{w}(\boldsymbol{x}, \mathbf{0}) = \boldsymbol{x}$, e renomeando $\boldsymbol{\mu}_{t_{i+1}}$ para $\boldsymbol{\mu}$, chegamos a

$$I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu})) = I_{t_i}(\boldsymbol{x}). \qquad \forall \boldsymbol{x} \in \boldsymbol{\rho} \tag{7.4}$$

Somando-se todos os termos dos dois lados da igualdade chegamos a

$$\sum_{\boldsymbol{x} \in \boldsymbol{\rho}} \left[I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu})) - I_{t_i}(\boldsymbol{x}) \right]^2 = 0. \tag{7.5}$$

Com a Equação (7.5) podemos expressar o problema de fluxo óptico, determinar $\boldsymbol{\mu}$, como um problema de minimização

$$\arg\min_{\boldsymbol{\mu}} \sum_{\boldsymbol{x} \in \boldsymbol{\rho}} \left[I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu})) - I_{t_i}(\boldsymbol{x}) \right]^2. \tag{7.6}$$

Para o desenvolvimento das equações a seguir é conveniente expressar as amostras das imagens em notação vetorial

$$\boldsymbol{I}_{t_i}(\mathbf{0}) = \begin{bmatrix} I_{t_i}(\boldsymbol{x}_1) \\ I_{t_i}(\boldsymbol{x}_2) \\ \vdots \\ I_{t_i}(\boldsymbol{x}_{n_\rho}) \end{bmatrix} \qquad \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) = \begin{bmatrix} I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu})) \\ I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu})) \\ \vdots \\ I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_{n_\rho}, \boldsymbol{\mu})) \end{bmatrix}. \tag{7.7}$$

Com esta nova notação a Equação (7.6) fica

$$\arg\min_{\boldsymbol{\mu}} \|\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) - \boldsymbol{I}_{t_i}(\mathbf{0})\|^2$$

que pode ser expressa como o problema de minimizar a função objetivo

$$O(\boldsymbol{\mu}) = \|\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) - \boldsymbol{I}_{t_i}(\mathbf{0})\|^2 \tag{7.8}$$

[1]No modelo tradicional onde a região passa por uma translação $\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}) = \boldsymbol{x} + \boldsymbol{\mu}$, quando $\boldsymbol{\mu}$ é simplesmente o vetor de translação.

A função objetivo da Equação (7.8) é não linear. Uma maneira de encontrar a solução para este problema é o uso de técnicas iterativas, como o método de Newton. Para que estes métodos funcionem é necessário uma boa estimativa inicial, que pode ser simplesmente $\boldsymbol{\mu} = \mathbf{0}$ (assumindo que existe pouco movimento entre os quadros). A cada iteração perturbamos a nossa solução atual para encontrar um $\delta\boldsymbol{\mu}$ que minimiza a equação

$$O(\boldsymbol{\mu} + \delta\boldsymbol{\mu}) = \|\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu} + \delta\boldsymbol{\mu}) - \boldsymbol{I}_{t_i}(\mathbf{0})\|^2 \tag{7.9}$$

e então atualizamos nossa estimativa de $\boldsymbol{\mu}$ da seguinte forma

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \delta\boldsymbol{\mu}.$$

Para encontrar solução para a Equação (7.9) primeiro expandimos o termo $\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu} + \delta\boldsymbol{\mu})$ em sua série de Taylor ao redor de $\boldsymbol{\mu}$

$$\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu} + \delta\boldsymbol{\mu}) = \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) + \boldsymbol{J}\delta\boldsymbol{\mu} + \text{t.a.o} \tag{7.10}$$

onde t.a.o. significa termos de alta ordem e $\boldsymbol{J}$ é o jacobiano de $\boldsymbol{I}_{t_{i+1}}$ com relação a $\boldsymbol{\mu}$

$$\boldsymbol{J} = \left[\frac{\partial}{\partial\mu_1}\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu})\Big|\frac{\partial}{\partial\mu_2}\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu})\Big|\cdots\Big|\frac{\partial}{\partial\mu_{n_{\boldsymbol{\mu}}}}\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu})\right]. \tag{7.11}$$

Substituindo a Equação (7.10) na Equação (7.9) e desprezando os termos de alta ordem chegamos a

$$O(\boldsymbol{\mu} + \delta\boldsymbol{\mu}) = \|\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu} + \delta\boldsymbol{\mu}) - \boldsymbol{I}_{t_i}(\mathbf{0})\|^2 \approx \|\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) + \boldsymbol{J}\delta\boldsymbol{\mu} - \boldsymbol{I}_{t_i}(\mathbf{0})\|^2$$

que pode ser resolvido tomando o gradiente em relação a $\boldsymbol{\mu}$ e igualando a zero

$$\nabla_{\boldsymbol{\mu}} O(\boldsymbol{\mu} + \delta\boldsymbol{\mu}) = \mathbf{0}. \tag{7.12}$$

Lembrando agora que o gradiente de uma função da forma $\|f(\boldsymbol{x})\|^2$ é dado por

$$\nabla_{\boldsymbol{x}}\|f(\boldsymbol{x})\|^2 = 2f(\boldsymbol{x})\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

vemos que a Equação (7.12) tem duas soluções

$$\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) + \boldsymbol{J}\delta\boldsymbol{\mu} - \boldsymbol{I}_{t_i}(\mathbf{0}) = \mathbf{0} \tag{7.13}$$

ou

$$\nabla_{\boldsymbol{\mu}}\left(\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) + \boldsymbol{J}\delta\boldsymbol{\mu} - \boldsymbol{I}_{t_i}(\mathbf{0})\right) = \mathbf{0}. \tag{7.14}$$

Desprezamos a segunda solução já que é menos provável de ocorrer que a primeira (todos os elementos do vetor em (7.14) teriam que ser iguais a zero).

Isolamos $\delta\boldsymbol{\mu}$ na Equação (7.13) para encontrar

$$\delta\boldsymbol{\mu} = -(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T\left[\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) - \boldsymbol{I}_{t_i}(\mathbf{0})\right]. \tag{7.15}$$

Cálculo eficiente de $\boldsymbol{J}$

Retornamos agora ao cálculo do jacobiano de $\boldsymbol{I}_{t_{i+1}}$ com relação a $\boldsymbol{\mu}$

$$\boldsymbol{J} = [\eta_{ij}]$$

$$\begin{aligned} \eta_{ij} &= \frac{\partial}{\partial \mu_j} I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_i, \boldsymbol{\mu})) \\ &= \nabla_{\boldsymbol{w}} I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_i, \boldsymbol{\mu})) \frac{\partial}{\partial \mu_j} \boldsymbol{w}(\boldsymbol{x}_i, \boldsymbol{\mu}) \end{aligned} \tag{7.16}$$

Fazendo uso de

$$\nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}) = \left[\frac{\partial}{\partial \mu_1} \boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}) \middle| \frac{\partial}{\partial \mu_2} \boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}) \middle| \cdots \middle| \frac{\partial}{\partial \mu_{n_{\boldsymbol{\mu}}}} \boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}) \right]$$

podemos reescrever $\boldsymbol{J}$ em forma matricial

$$\boldsymbol{J} = \begin{bmatrix} \nabla_{\boldsymbol{w}} I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu})) \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu}) \\ \nabla_{\boldsymbol{w}} I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu})) \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu}) \\ \vdots \\ \nabla_{\boldsymbol{w}} I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_{n_{\boldsymbol{\rho}}}, \boldsymbol{\mu})) \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_{n_{\boldsymbol{\rho}}}, \boldsymbol{\mu}) \end{bmatrix}. \tag{7.17}$$

Observando o termo $\nabla_{\boldsymbol{w}} I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_i, \boldsymbol{\mu}))$ vemos que ele precisa ser recalculado toda vez que $\boldsymbol{\mu}$ muda e que isto seria um processo computacionalmente caro. Para reduzir o acoplamento das equações, voltamos à Equação (7.4)

$$I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu})) = I_{t_i}(\boldsymbol{x}),$$

que após a diferenciação de ambos os lados com relação a $\boldsymbol{x}$

$$\nabla_{\boldsymbol{w}} I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu})) \nabla_{\boldsymbol{x}} \boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}) = \nabla_{\boldsymbol{x}} I_{t_i}(\boldsymbol{x}).$$

Isolando $\nabla_{\boldsymbol{w}} I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}))$ na equação passada chegamos a

$$\nabla_{\boldsymbol{w}} I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu})) = \nabla_{\boldsymbol{x}} I_{t_i}(\boldsymbol{x}) \nabla_{\boldsymbol{x}} \boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu})^{-1},$$

que substituímos dentro da Equação (7.17) para obter

$$\boldsymbol{J} = \begin{bmatrix} \nabla_{\boldsymbol{x}} I_{t_i}(\boldsymbol{x}_1) \nabla_{\boldsymbol{x}} \boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu})^{-1} \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu}) \\ \nabla_{\boldsymbol{x}} I_{t_i}(\boldsymbol{x}_2) \nabla_{\boldsymbol{x}} \boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu})^{-1} \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu}) \\ \vdots \\ \nabla_{\boldsymbol{x}} I_{t_i}(\boldsymbol{x}_{n_{\boldsymbol{\rho}}}) \nabla_{\boldsymbol{x}} \boldsymbol{w}(\boldsymbol{x}_{n_r}, \boldsymbol{\mu})^{-1} \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_{n_{\boldsymbol{\rho}}}, \boldsymbol{\mu}) \end{bmatrix}.$$

Com esta nova formulação conseguimos mover o cálculo de $\nabla_{\boldsymbol{x}} I_{t_i}(\cdot)$ para fora do laço principal.

O pseudocódigo do fluxo óptico diferencial pode ser visto no Algoritmo 7.1 na página seguinte.

Algoritmo 7.1 Fluxo óptico diferencial.

1: **procedimento** FluxoÓpticoDiferencial($I_{t_i}, I_{t_{i+1}}, \boldsymbol{w}, [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_\rho}], k, \epsilon_{\delta\boldsymbol{\mu}}$)

Entrada:
$I_{t_i}, I_{t_{i+1}}$: imagens nos instantes t_i e t_{i+1}
$\boldsymbol{w}$: modelo de deformação
$[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_\rho}]$: pontos da região $\boldsymbol{\rho}$
k: número máximo de iterações
$\epsilon_{\delta\boldsymbol{\mu}}$: menor incremento permitido de $\boldsymbol{\mu}$
Saída:
$\boldsymbol{\mu}$: deslocamento da região $\boldsymbol{\rho}$ de t_i para t_{i+1}

2: $\boldsymbol{\mu} \leftarrow \mathbf{0}$
3: $\boldsymbol{I}_{t_i} \leftarrow [I_{t_i}(\boldsymbol{x}_1), I_{t_i}(\boldsymbol{x}_2), \ldots, I_{t_i}(\boldsymbol{x}_{n_\rho})]^T$
4: $\nabla_{\boldsymbol{x}} \boldsymbol{I}_{t_i} \leftarrow [\nabla_{\boldsymbol{x}} I_{t_i}(\boldsymbol{x}_1), \nabla_{\boldsymbol{x}} I_{t_i}(\boldsymbol{x}_2), \ldots, \nabla_{\boldsymbol{x}} I_{t_i}(\boldsymbol{x}_{n_\rho})]^T$ ▷ Cálculo da derivada espacial
5: **para** $iter \leftarrow 1 \ldots k$ **faça**
6: $\quad \boldsymbol{I}_{t_{i+1}} \leftarrow [I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu})), I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu})), \ldots, I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_{n_\rho}, \boldsymbol{\mu}))]^T$
7: $\quad \delta\boldsymbol{I} \leftarrow \boldsymbol{I}_{t_{i+1}} - \boldsymbol{I}_{t_i}$ ▷ Atualizar a derivada temporal
8: $\quad \boldsymbol{J} \leftarrow \begin{bmatrix} \nabla_{\boldsymbol{x}} \boldsymbol{I}_{t_i}[1] \nabla_{\boldsymbol{x}} \boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu})^{-1} \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu}) \\ \nabla_{\boldsymbol{x}} \boldsymbol{I}_{t_i}[2] \nabla_{\boldsymbol{x}} \boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu})^{-1} \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu}) \\ \vdots \\ \nabla_{\boldsymbol{x}} \boldsymbol{I}_{t_i}[n_{\boldsymbol{\rho}}] \nabla_{\boldsymbol{x}} \boldsymbol{w}(\boldsymbol{x}_{n_r}, \boldsymbol{\mu})^{-1} \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_{n_r}, \boldsymbol{\mu}) \end{bmatrix}$
9: $\quad$ **se** $\boldsymbol{J}^T\boldsymbol{J}$ é inversível **então**
10: $\quad\quad \delta\boldsymbol{\mu} \leftarrow -(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}\delta\boldsymbol{I}$
11: $\quad\quad \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \delta\boldsymbol{\mu}$
12: $\quad\quad$ **se** $\|\delta\boldsymbol{\mu}\| < \epsilon_{\delta\boldsymbol{\mu}}$ **então** ▷ Passo muito pequeno
13: $\quad\quad\quad$ **retorne** $\boldsymbol{\mu}$
14: $\quad\quad$ **fim se**
15: $\quad$ **senão**
16: $\quad\quad$ **retorne** *nil* ▷ Região contém pouca informação espacial, ponto foi perdido
17: $\quad$ **fim se**
18: **fim para**
19: **retorne** $\boldsymbol{\mu}$
20: **fim procedimento**

Robustez a variação de iluminação

Podemos modelar o efeito da iluminação com um termo multiplicativo λ (a mudança do contraste) e outro aditivo σ (mudança de brilho), é importante frisar que consideramos esses dois parâmetros constantes para toda a região. Se chamarmos de $\breve{\boldsymbol{I}}_{t_{i+1}}(\boldsymbol{\mu})$ a imagem capturada pelo sensor, podemos estabelecer a relação com $\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu})$, a região livre do efeito da iluminação, através da seguinte equação

$$\breve{\boldsymbol{I}}_{t_{i+1}}(\boldsymbol{\mu}) = \lambda \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) + \mathbf{1}\sigma, \tag{7.18}$$

onde $\mathbf{1}$ é um vetor coluna de 1's de dimensão $n_{\boldsymbol{\rho}}$. A função objetivo se torna

$$O(\boldsymbol{\mu}, \lambda, \sigma) = \left\| \lambda \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) + \mathbf{1}\sigma - \boldsymbol{I}_{t_i}(\mathbf{0}) \right\|. \tag{7.19}$$

De maneira análoga ao que foi feito com a Equação (7.10) podemos expandir a Equação (7.18) com relação a $\boldsymbol{\mu}$, λ, e σ

$$\begin{aligned} (\lambda + \delta\lambda)\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu} + \delta\boldsymbol{\mu}) + \mathbf{1}(\sigma + \delta\sigma) &= \lambda \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) + \mathbf{1}\sigma + \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu})\delta\lambda \\ &\quad + \lambda \boldsymbol{J}(\boldsymbol{\mu})\delta\boldsymbol{\mu} + \mathbf{1}\delta\sigma + \text{t.a.o.}. \end{aligned}$$

Agora substituímos na nova função objetivo (7.19)

$$O(\boldsymbol{\mu}+\delta\boldsymbol{\mu}, \lambda+\delta\lambda, \sigma+\delta\sigma) \approx \left\| \lambda \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) + \mathbf{1}\sigma + \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu})\delta\lambda + \lambda \boldsymbol{J}(\boldsymbol{\mu})\delta\boldsymbol{\mu} + \mathbf{1}\delta\sigma - \boldsymbol{I}_{t_i}(\mathbf{0}) \right\|^2. \tag{7.20}$$

Para encontrar os valores de $\delta\boldsymbol{\mu}$, $\delta\lambda$ e $\delta\sigma$ resolvemos a equação análoga a (7.13)

$$\begin{aligned} \lambda \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) + \mathbf{1}\sigma + \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu})\delta\lambda + \lambda \boldsymbol{J}(\boldsymbol{\mu})\delta\boldsymbol{\mu} + \mathbf{1}\delta\sigma - \boldsymbol{I}_{t_i}(\mathbf{0}) &= 0 \\ \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu})\delta\lambda + \mathbf{1}\delta\sigma + \lambda \boldsymbol{J}(\boldsymbol{\mu})\delta\boldsymbol{\mu} &= -\left\{ \lambda \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) + \mathbf{1}\sigma - \boldsymbol{I}_{t_i}(\mathbf{0}) \right\}, \end{aligned} \tag{7.21}$$

criando duas variáveis auxiliares

$$\boldsymbol{\Omega} = [\boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}), \mathbf{1}, \lambda \boldsymbol{J}(\boldsymbol{\mu})] \qquad \delta\boldsymbol{\beta} = \begin{bmatrix} \delta\lambda \\ \delta\sigma \\ \delta\boldsymbol{\mu} \end{bmatrix}$$

e substituindo na Equação (7.21)

$$\delta\boldsymbol{\beta} = -(\boldsymbol{\Omega}^T\boldsymbol{\Omega})^{-1}\boldsymbol{\Omega}^T \left\{ \lambda \boldsymbol{I}_{t_{i+1}}(\boldsymbol{\mu}) + \mathbf{1}\sigma - \boldsymbol{I}_{t_i}(\mathbf{0}) \right\}. \tag{7.22}$$

O pseudocódigo do fluxo óptico diferencial com robustez a iluminação pode ser visto no Algoritmo 7.2 na próxima página.

Algoritmo 7.2 Fluxo óptico diferencial com robustez a iluminação.

1: **procedimento** FLUXOÓPTICODIFERENCIAL-RI($I_{t_i}, I_{t_{i+1}}, \boldsymbol{w}, [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_{\boldsymbol{\rho}}}], k, \epsilon_{\delta\boldsymbol{\mu}}, \epsilon_\lambda$)

Entrada:
$I_{t_i}, I_{t_{i+1}}$: imagens nos instantes t_i e t_{i+1}
$\boldsymbol{w}$: modelo de deformação
$[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_{\boldsymbol{\rho}}}]$: pontos da região $\boldsymbol{\rho}$
k: número máximo de iterações
$\epsilon_{\delta\boldsymbol{\mu}}$: menor incremento permitido de $\boldsymbol{\mu}$
ϵ_λ: menor valor permitido para o parâmetro multiplicativo λ do modelo de iluminação
Saída:
$\boldsymbol{\mu}$: deslocamento da região $\boldsymbol{\rho}$ de t_i para t_{i+1}

2: $\boldsymbol{\mu} \leftarrow \mathbf{0}, \lambda \leftarrow 1, \sigma \leftarrow 0$
3: $\boldsymbol{I}_{t_i} \leftarrow [I_{t_i}(\boldsymbol{x}_1), I_{t_i}(\boldsymbol{x}_2), \ldots, I_{t_i}(\boldsymbol{x}_{\boldsymbol{\rho}})]^T$
4: $\nabla_{\boldsymbol{x}} \boldsymbol{I}_{t_i} \leftarrow [\nabla_{\boldsymbol{x}} I_{t_i}(\boldsymbol{x}_1), \nabla_{\boldsymbol{x}} I_{t_i}(\boldsymbol{x}_2), \ldots, \nabla_{\boldsymbol{x}} I_{t_i}(\boldsymbol{x}_{n_{\boldsymbol{\rho}}})]^T$ ▷ Cálculo da derivada espacial
5: **para** $iter \leftarrow 1 \ldots k$ **faça**
6: $\boldsymbol{I}_{t_{i+1}} \leftarrow [I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu})), I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu})), \ldots, I_{t_{i+1}}(\boldsymbol{w}(\boldsymbol{x}_{n_{\boldsymbol{\rho}}}, \boldsymbol{\mu}))]^T$
7: $\delta\boldsymbol{I} \leftarrow \lambda \boldsymbol{I}_{t_{i+1}} + \mathbf{1}\sigma - \boldsymbol{I}_{t_i}$ ▷ Atualizar a derivada temporal
8: $\boldsymbol{J} \leftarrow \begin{bmatrix} \nabla_{\boldsymbol{x}} \boldsymbol{I}_{t_i}[1] \nabla_{\boldsymbol{x}} \boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu})^{-1} \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_1, \boldsymbol{\mu}) \\ \nabla_{\boldsymbol{x}} \boldsymbol{I}_{t_i}[2] \nabla_{\boldsymbol{x}} \boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu})^{-1} \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_2, \boldsymbol{\mu}) \\ \vdots \\ \nabla_{\boldsymbol{x}} \boldsymbol{I}_{t_i}[n_{\boldsymbol{\rho}}] \nabla_{\boldsymbol{x}} \boldsymbol{w}(\boldsymbol{x}_{n_r}, \boldsymbol{\mu})^{-1} \nabla_{\boldsymbol{\mu}} \boldsymbol{w}(\boldsymbol{x}_{n_r}, \boldsymbol{\mu}) \end{bmatrix}$
9: $\boldsymbol{\Omega} \leftarrow [\boldsymbol{I}_{t_{i+1}}, \mathbf{1}, \lambda \boldsymbol{J}]$
10: **se** $\boldsymbol{\Omega}^T \boldsymbol{\Omega}$ é inversível **então**
11: $\delta\boldsymbol{\beta} \leftarrow -(\boldsymbol{\Omega}^T \boldsymbol{\Omega})^{-1} \boldsymbol{\Omega} \delta \boldsymbol{I}$
12: $\delta\lambda \leftarrow \delta\boldsymbol{\beta}[1], \delta\sigma \leftarrow \delta\boldsymbol{\beta}[2], \delta\boldsymbol{\mu} \leftarrow \delta\boldsymbol{\beta}[3]$
13: $\lambda \leftarrow \lambda + \delta\lambda, \sigma \leftarrow \sigma + \delta\sigma, \boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \delta\boldsymbol{\mu}$
14: **se** $\|\delta\boldsymbol{\mu}\| < \epsilon_{\delta\boldsymbol{\mu}}$ **então** ▷ Passo muito pequeno
15: **retorne** $\boldsymbol{\mu}$
16: **fim se**
17: **se** $\lambda < \epsilon_\lambda$ **então**
18: **retorne** *nil*
19: **fim se**
20: **senão**
21: **retorne** *nil*
22: **fim se**
23: **fim para**
24: **retorne** $\boldsymbol{\mu}$
25: **fim procedimento**

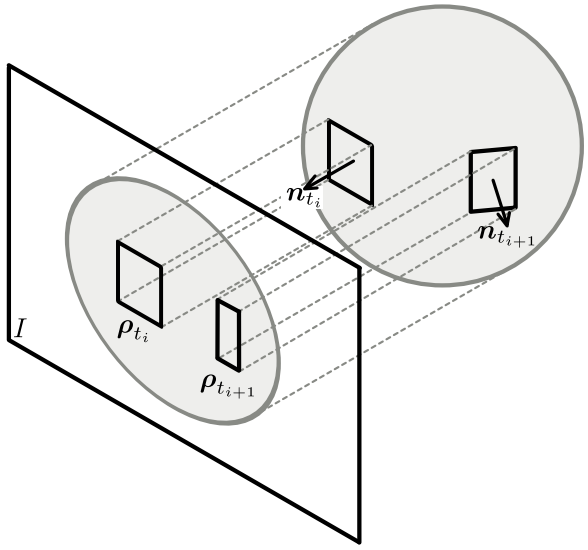


Figura 7.6: Modelo esférico de deformação da região de interesse $\boldsymbol{\rho}$.

Aplicação ao caso da fruta

Para o rastreamento de pontos na fruta usamos dois modelos para $\boldsymbol{w}(\cdot)$, o primeiro é o modelo translacional e o segundo é a aproximação da fruta por uma esfera. O primeiro é dado por

$$\boldsymbol{w}_{trans}(\boldsymbol{x}, \boldsymbol{\mu}) = \boldsymbol{x} + \boldsymbol{\mu}, \qquad \boldsymbol{\mu} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}.$$

Não entraremos em detalhes sobre este modelo pois é amplamente tratado na literatura [15, 17, 26].

O segundo modelo aproxima a fruta por uma esfera e assume um modelo ortográfico de projeção como pode ser visto na Figura 7.6. Nestas condições a projeção da fruta seria um círculo. Além disso, aproximamos a região $\boldsymbol{\rho}$ por um plano normal à esfera, sendo que a normal é computada no ponto central de $\boldsymbol{\rho}$, o qual chamaremos de $\boldsymbol{d} \in \mathbb{R}^2$. Para lidar com a deformação que ocorre à medida que $\boldsymbol{d}$ se aproxima das bordas do círculo lançamos mão de uma transformação linear $\boldsymbol{A}(\boldsymbol{\mu}) : \mathbb{R}^{n_{\boldsymbol{\mu}}} \to \mathbb{R}^{2\times 2}$ e um deslocamento $\boldsymbol{b}(\boldsymbol{\mu}) : \mathbb{R}^{n_{\boldsymbol{\mu}}} \to \mathbb{R}^2$ e a função de deformação toma a forma

$$\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{\mu}) = \boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{x} + \boldsymbol{b}(\boldsymbol{\mu}). \tag{7.23}$$

Se deixássemos os elementos de $\boldsymbol{A}(\boldsymbol{\mu})$ e $\boldsymbol{b}(\boldsymbol{\mu})$ livres, teríamos o que é comumente chamado de fluxo óptico afim. No entanto, temos informação extra que podemos usar para tornar o algoritmo mais rápido e preciso: o objeto sendo rastreado pode ser aproximado por uma esfera.

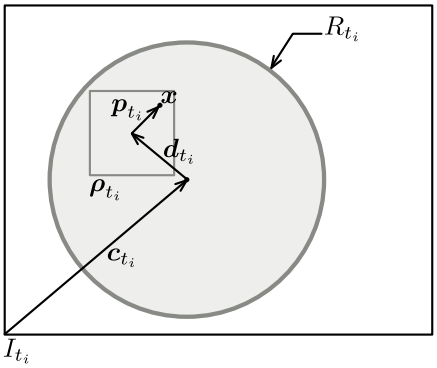


Figura 7.7: Principais variáveis no cálculo de $\boldsymbol{w}_{esf}$.

Agora iremos derivar as equações que definem $\boldsymbol{A}(\boldsymbol{\mu})$ e $\boldsymbol{b}(\boldsymbol{\mu})$. O vetor $\boldsymbol{\mu}$ representa o deslocamento na referência t_i. Usamos também as seguintes variáveis

$$\boldsymbol{c}_{t_i}, \boldsymbol{c}_{t_{i+1}}, \boldsymbol{d}_{t_i}, R_{t_i}, R_{t_{i+1}}$$

onde $\boldsymbol{c}_{t_i}$ e $\boldsymbol{c}_{t_{i+1}}$ são as coordenadas do centro do círculo, $\boldsymbol{d}_{t_i}$ é a coordenada do centro de $\boldsymbol{\rho}$ em t_i, R_{t_i} e $R_{t_{i+1}}$ os raios dos círculos, todos medidos em *pixels*.

Em primeiro lugar vamos derivar apenas as equações que definem $\boldsymbol{A}(\boldsymbol{\mu})$, a matriz que deforma a região $\boldsymbol{\rho}$ para compensar a curvatura da esfera. Começamos por substituir o $\boldsymbol{x}$ da Equação (7.23) por $\boldsymbol{p}$, que é o deslocamento dentro da região $\boldsymbol{\rho}$ a partir do ponto $\boldsymbol{x}$,

$$\boldsymbol{p}_{t_i} = \boldsymbol{x} - \boldsymbol{d}_{t_i}.$$

Para facilitar o raciocínio podemos trabalhar com os eixos x e y da imagem separadamente. Com isso precisamos olhar somente para a visão lateral da cena (Figura 7.8). Outra simplificação é que, para o instante t_i, a normal é perpendicular ao plano de projeção (vamos remover esta restrição mais para frente). Nestes termos chegamos à equação

$$\boldsymbol{p}_{t_{i+1}} = \begin{bmatrix} \cos\alpha_{t_{i+1}} & 0 \\ 0 & \cos\beta_{t_{i+1}} \end{bmatrix} \boldsymbol{p}_{t_i}. \tag{7.24}$$

onde $\alpha_{t_{i+1}}$ e $\beta_{t_{i+1}}$ são o ângulo que a normal $\boldsymbol{n}_{t_{i+1}}$, quando projetada nos planos zx e zy, faz com os eixos x e y (Figura 7.9), respectivamente.

Agora podemos generalizar a Equação (7.24) para o caso em que $\boldsymbol{n}_{t_i}$ não é perpendicular ao plano da imagem. Neste caso podemos quebrar o problema em duas partes. Na primeira, encaramos a situação como sendo a inversa à que tratamos anteriormente, criamos um ponto intermediário $\boldsymbol{p}_{aux}$ cuja normal é perpendicular ao plano da imagem e escrevemos a equação similar à Equação (7.24) que relaciona $\boldsymbol{p}_{t_i}$ com $\boldsymbol{p}_{aux}$

$$\boldsymbol{p}_{t_i} = \boldsymbol{C}_{t_i}\boldsymbol{p}_{aux}, \tag{7.25}$$

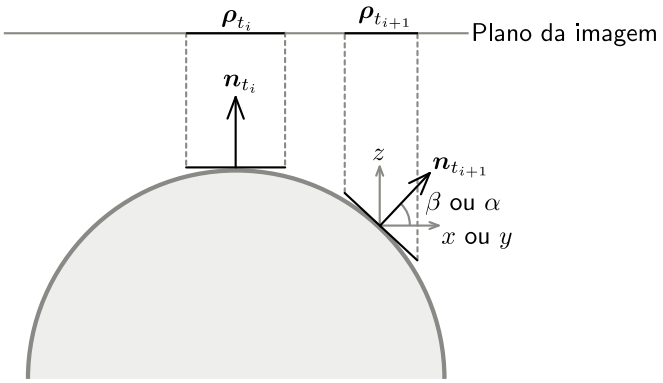


Figura 7.8: Visão lateral da projeção da esfera.

onde $\boldsymbol{C}_i = \begin{bmatrix} \cos\alpha_i & 0 \\ 0 & \cos\beta_i \end{bmatrix}$. Podemos agora relacionar $\boldsymbol{p}_{aux}$ com $\boldsymbol{p}_{t_{i+1}}$

$$\boldsymbol{p}_{t_{i+1}} = \boldsymbol{C}_{t_{i+1}} \boldsymbol{p}_{aux}. \tag{7.26}$$

Agora eliminamos $\boldsymbol{p}_{aux}$ das Equações (7.25) e (7.26) e chegamos a

$$\boldsymbol{p}_{t_{i+1}} = \boldsymbol{C}_{t_{i+1}} \boldsymbol{C}_{t_i}^{-1} \boldsymbol{p}_{t_i}. \tag{7.27}$$

Sob a projeção ortográfica o tamanho de um objeto não varia com sua distância ao plano de projeção. No entanto, este modelo de projeção é apenas uma aproximação e na realidade precisamos lidar com a variação do tamanho, que é resultado de dois fatores: um é a distância da fruta até a câmera e o outro é a forma irregular da fruta. Para corrigir a escala adicionamos o termo multiplicativo $R_{t_{i+1}}/R_{t_i}$ à Equação (7.27) ficando com

$$\boldsymbol{p}_{t_{i+1}} = \frac{R_{t_{i+1}}}{R_{t_i}} \boldsymbol{C}_{t_{i+1}} \boldsymbol{C}_{t_i}^{-1} \boldsymbol{p}_{t_i},$$

então a matriz $\boldsymbol{A}(\boldsymbol{\mu})$ é dada por

$$\boldsymbol{A}(\boldsymbol{\mu}) = \frac{R_{t_{i+1}}}{R_{t_i}} \boldsymbol{C}_{t_{i+1}} \boldsymbol{C}_{t_i}^{-1}. \tag{7.28}$$

Voltamos agora às normais $\boldsymbol{n}_{t_i}$ e $\boldsymbol{n}_{t_{i+1}}$, que foram introduzidas mas não definidas. Dada a função

$$f(x, y, z) = x^2 + y^2 + z^2,$$

cuja curva de nível $f(x, y, z) = R^2$ é a equação da esfera centrada na origem e de raio R, obtemos a equação de sua normal da seguinte forma

$$\frac{\nabla f(x, y, z)}{\|\nabla f(x, y, z)\|} = \frac{2 \begin{bmatrix} x \\ y \\ z \end{bmatrix}}{2\sqrt{(x^2 + y^2 + z^2)}} = \frac{1}{R} \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

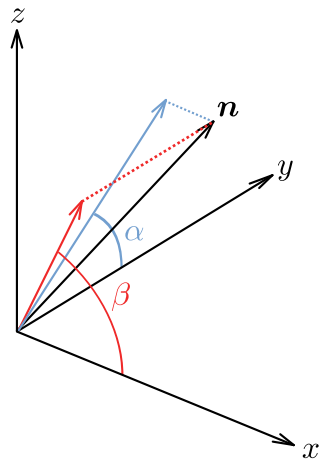


Figura 7.9: Ângulos α e β.

Os cossenos dos ângulos α e β são dados por

$$\cos\alpha = \frac{z}{\sqrt{z^2 - x^2}} \qquad \cos\beta = \frac{z}{\sqrt{z^2 + y^2}}.$$

Os valores de x e y são dados por

$$x = \boldsymbol{x}[x] - \boldsymbol{c}[x] \qquad y = \boldsymbol{x}[y] - \boldsymbol{c}[y].$$

Agora precisamos determinar z. Como assumimos que a projeção é ortográfica podemos deslocar a esfera ao longo do eixo z sem que sua projeção se altere. Desta forma assumimos que a coordenada z do centro da esfera é igual a zero. Isolando z na equação $f(x, y, z) = R^2$ e tomando apenas a resposta positiva chegamos a

$$z = \sqrt{R^2 - x^2 - y^2}$$

e a equação da normal, dados $\boldsymbol{c}$ e R,

$$\boldsymbol{n} = \frac{1}{R}\begin{bmatrix} x \\ y \\ \sqrt{R^2 - x^2 - y^2} \end{bmatrix}. \tag{7.29}$$

O vetor $\boldsymbol{b}(\boldsymbol{\mu})$ transporta o ponto da referência em t_i para t_{i+1}. Para entender sua derivação veja a Figura 7.10 na próxima página.

$$\boldsymbol{b}(\boldsymbol{\mu}) = \frac{R_{t_{i+1}}}{R_{t_i}}(\boldsymbol{d}_{t_i} - \boldsymbol{c}_{t_i} + \boldsymbol{\mu}) + \boldsymbol{c}_{t_{i+1}}. \tag{7.30}$$

A equação final para $\boldsymbol{w}_{esf}(\boldsymbol{\mu})$ é

$$\boldsymbol{w}_{esf}(\boldsymbol{\mu}) = \boldsymbol{A}(\boldsymbol{\mu})\left[\boldsymbol{x} - \boldsymbol{d}_{t_i}\right] + \boldsymbol{b}(\boldsymbol{\mu}).$$

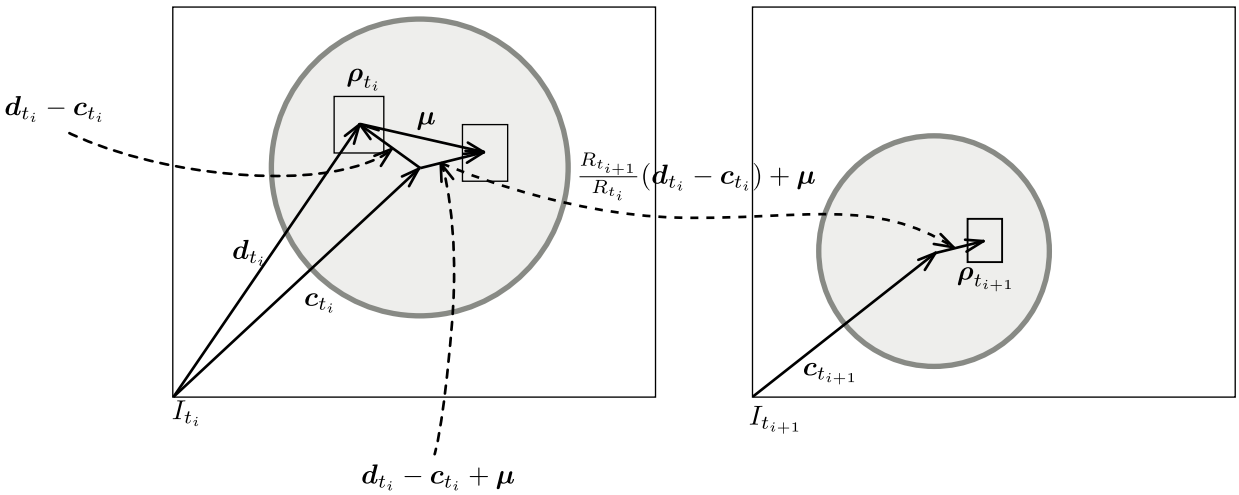


Figura 7.10: Derivação de $\boldsymbol{b}(\boldsymbol{\mu})$.

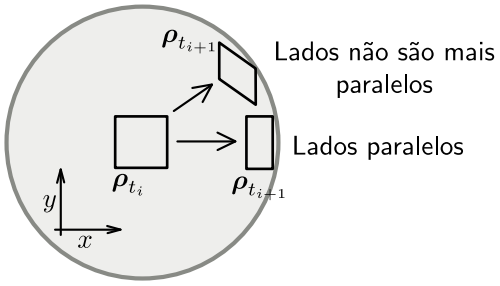


Figura 7.11: Deformação do quadrado tangente à esfera.

Para chegar à forma final de $\boldsymbol{w}_{esf}(\cdot)$ fizemos algumas simplificações, discutimos agora um dos problemas decorrentes desta simplificação. Se observarmos como a imagem de um quadrado contido no plano tangente a uma esfera se deforma à medida que este plano caminha sobre a superfície, veremos que ele se mantém retangular se a direção do deslocamento for paralelo a um de seus lados. Este tipo de deformação é corretamente modelada pelas nossas equações. Já quando a direção de deslocamento foge a essa regra o paralelismo das faces é perdido e é este efeito que $\boldsymbol{A}(\boldsymbol{\mu})$ não consegue modelar no presente desenvolvimento (veja a Figura 7.10). No entanto, como será visto na seção de testes, $\boldsymbol{w}_{esf}(\cdot)$ consegue reduzir o erro ao rastrear pontos sobre uma esfera. Para corrigir este erro teríamos que alterar $\boldsymbol{C}_i$ para que incorporasse a co-variância das coordenadas x e y , ou seja, os termos fora da diagonal principal teriam que ser diferentes de zero.

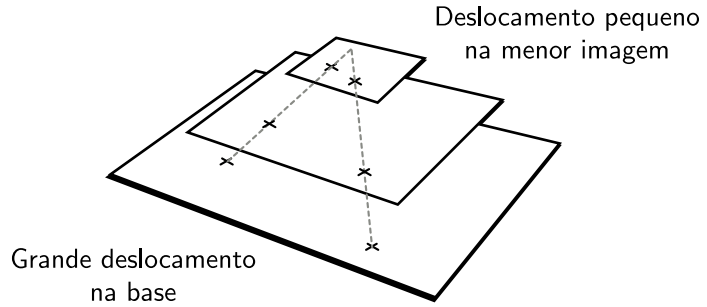


Figura 7.12: Fluxo óptico multi-escala.

Notas sobre a implementação

Discutiremos agora alguns detalhes da implementação do algoritmo que foi descrito até o momento. Os algoritmos serão descritos apenas brevemente pois usamos versões do que já é amplamente conhecido na literatura.

O primeiro aspecto importante é quanto ao uso de múltiplas escalas no rastreamento. A principal limitação dos algoritmos diferenciais de fluxo óptico é que, na sua forma mais simples, não são capazes de rastrear pontos que se deslocam mais do que alguns *pixels* por imagem, esta limitação é resultado da aproximação pela série de Taylor (ver Equações (7.10) e (7.20)) e também do tamanho da região $\boldsymbol{\rho}$. Para contornar esta limitação usamos imagens em várias escalas em forma piramidal. A base da pirâmide tem o tamanho da imagem original, a cada nível da pirâmide reduzimos o tamanho da imagem pela metade. O rastreamento neste esquema começa da escala mais grosseira (menor imagem). Nesta imagem um deslocamento no tamanho original se torna menor e em mais situações é possível rastrear o ponto. Uma vez rastreado o ponto em uma escala mais grosseira passamos a rastrear o ponto na próxima imagem da pirâmide (imagem maior) e usamos o $\boldsymbol{\mu}$ obtido da escala anterior (corrigido pelo tamanho da imagem) como uma estimativa inicial de movimento. O conceito está ilustrado na Figura 7.12.

Outro problema que não foi abordado até o momento é o de rastrear pontos que se aproximam da borda da imagem. Supondo uma região r quadrada de 7×7 *pixels*. Para uma imagem de 640×480 *pixels* a região r ao redor dos pontos começa a cair fora da imagem para pontos que estão a 3 pixels da borda ou menos. Em termos de área isso representa aproximadamente 1.5%. À primeira vista pode parecer que podemos desprezar estes pontos (declará-los perdidos uma vez que chegam perto da borda) mas quando começamos a trabalhar com múltiplas escalas a fração da área perdida cresce rapidamente (assumindo que o tamanho da janela r não se altera para as várias escalas). Para o nosso exemplo a borda no nível de maior resolução é de $(7-1)/2 = 3$. Se pensarmos na borda perdida de cada nível da pirâmide reprojetado na sua base veremos que a espessura da borda perdida dobra para cada nível da pirâmide e para 2, 3 e 4 níveis

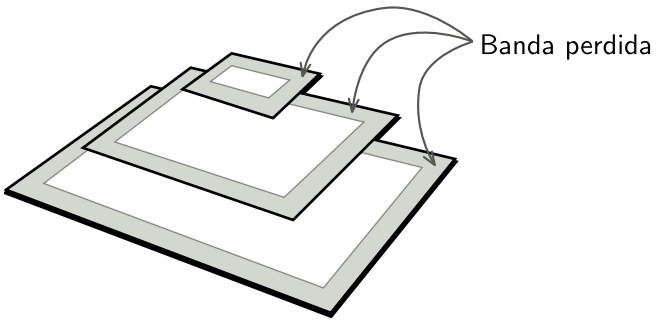


Figura 7.13: Banda perdida para pontos perto da borda.

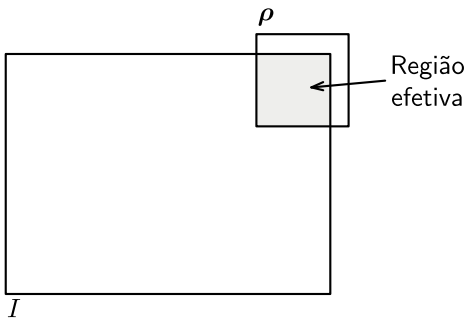


Figura 7.14: Interseção de r_{t_i}, $r_{t_{i+1}}$ e quadro da imagem.

as frações da área perdida são $\sim 3\%$, $\sim 6\%$ e $\sim 12\%$ (ver Figura 7.13). Fica evidente que é necessário tomar providências no sentido de continuar rastreando estes pontos. Para fazer isso passamos a fazer uso de sub janelas de r_{t_i} e $r_{t_{i+1}}$ que se encontram na intersecção das janelas originais e do retângulo da imagem (ver Figura 7.14).

A nossa implementação de multi-escala e rastreamento perto das bordas segue o que foi descrito em [8].

O algoritmo exposto rastreia pontos, no entanto não são todos os pontos de uma imagem que são rastreáveis. Para entender o porquê podemos imaginar uma sequência capturada de um muro uniforme. Se as bordas do muro não forem visíveis na imagem não conseguimos determinar se existe movimento ou não (ver Figura 7.15). O problema é que qualquer correspondência entre regiões da sequência minimiza o erro. Nas Equações (7.15) e (7.22) as matrizes $\boldsymbol{J}^T\boldsymbol{J}$ e $\boldsymbol{\Omega}^T\boldsymbol{\Omega}$ são singulares para estes casos. O algoritmo apresentado em [33] tenta encontrar na imagem pontos bons para rastrear calculando os autovalores da matriz $\boldsymbol{J}^T\boldsymbol{J}$, com $\boldsymbol{w}(\cdot)$ o modelo translacional de movimento, e filtrando pontos cuja matriz seria mal condicionada. Usamos a implementação do algoritmo disponível na biblioteca OpenCV. Embora esta implementação assuma que o modelo de deformação seja o translacional usamos ela também para os testes usando $\boldsymbol{w}_{esf}(\cdot)$.

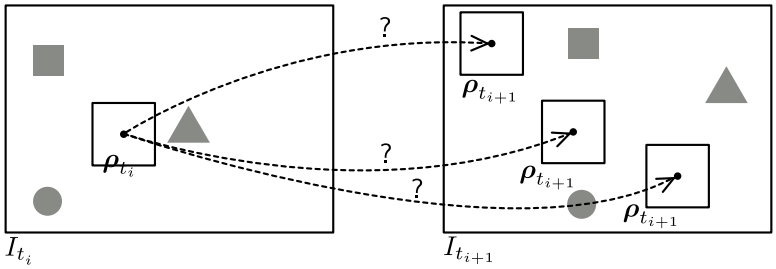


Figura 7.15: Pontos que não são rastreáveis.

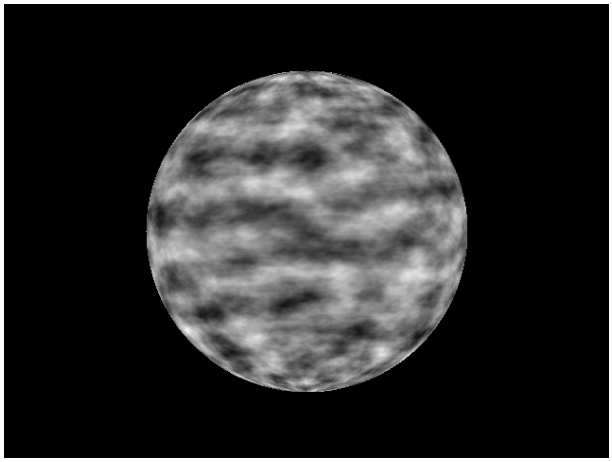


Figura 7.16: Imagem (sintética) da cena de teste.

7.5.1.3 Testes

Para entender o ganho que as duas alterações no fluxo óptico tradicional oferecem fizemos testes com uma cena artificial. Uma imagem da sequência pode ser vista na Figura 7.16. A cena consiste de uma esfera, com uma textura mapeada, girando sobre um fundo preto sob iluminação difusa e uniforme. Usamos o modelo ortográfico de projeção. Para cada teste usamos o algoritmo proposto em [33] para selecionar pontos para rastrear. Calculamos o fluxo real para estes pontos e comparamos com aquele obtido dos algoritmos testados.

O primeiro teste compara o modelo translacional $\boldsymbol{w}_{trans}$ com o esférico $\boldsymbol{w}_{esf}$ de deformação para diferentes velocidades de rotação da esfera. O resultado do teste pode ser visto no gráfico da Figura 7.17. Um erro menor é registrado pelo modelo esférico para todas as velocidades. O componente periódico das curvas é devido a um artefato da cena, a linha que se forma na esfera no encontro das duas extremidades da imagem da textura.

O segundo teste mede como a iluminação interfere no rastreamento. Como foi visto anteriormente (Figura na página 47) o componente σ da Equação (7.18) é o que mais varia, por esta razão variamos apenas ele no experimento. Usamos valores parecidos com aqueles

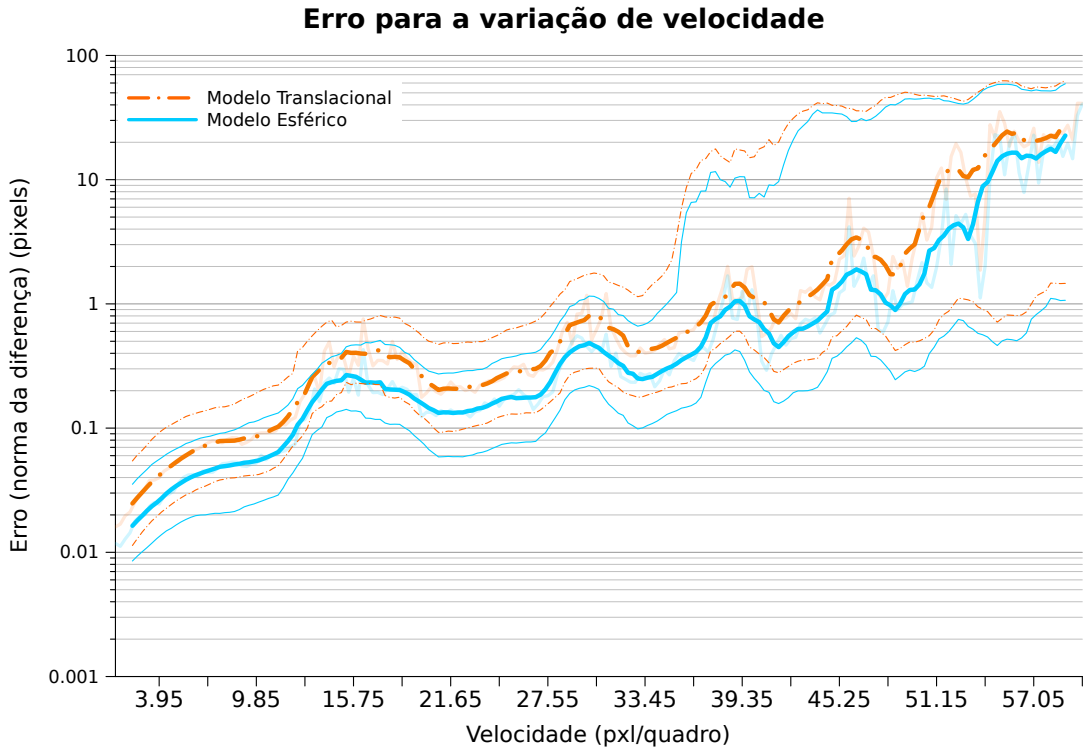


Figura 7.17: Erro no cálculo do fluxo óptico para diferentes velocidades e para ambos os modelos de deformação, $\boldsymbol{w}_{trans}$ e $\boldsymbol{w}_{esf}$. As linhas mais finas marcam os quartis inferior e superior. A velocidade é medida no ponto central da esfera (normal perpendicular ao plano da imagem).

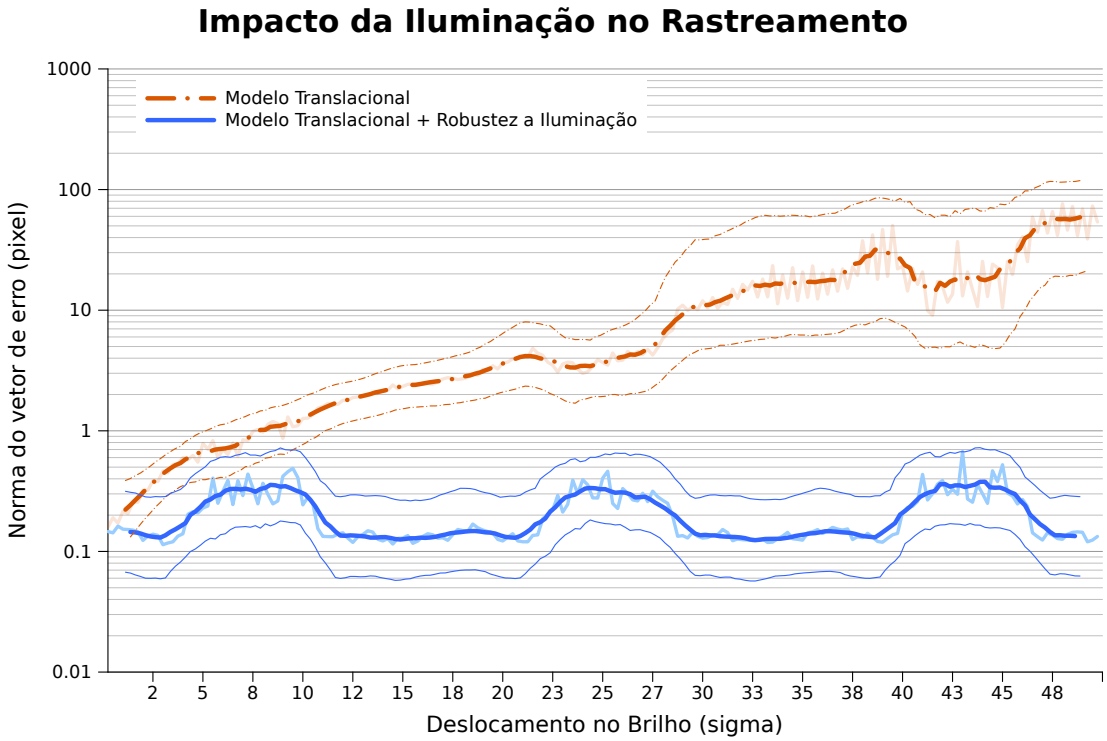


Figura 7.18: Impacto do componente σ para o modelo translacional de deformação $\boldsymbol{w}_{trans}$. As linhas mais finas marcam os quartis inferior e superior. A velocidade é medida no ponto central da esfera (normal perpendicular ao plano da imagem). A velocidade é medida no ponto central da esfera (normal perpendicular ao plano da imagem).

observados em imagens reais. Mantendo a velocidade de rotação fixa (15 *pixels*/quadro), variamos σ de 0 a 50 para os quadros ímpares da sequência e igualmente para toda a imagem. Os quadros pares da sequência foram mantidos inalterados. O valor de σ variou de 0 até 50 sendo que o valor mínimo de intensidade é 0 e o máximo é 255.

Os gráficos das Figuras 7.18 e 7.19 mostram os resultados do segundo teste para o modelo translacional e esférico de deformação, respectivamente. Vemos nos dois o mesmo comportamento, o algoritmo simples é severamente afetado pela variação de iluminação enquanto que aquele com com a robustez obtém os mesmos resultados independente do valor de σ. Um comportamento que observamos mas para o qual não temos explicação é que o componente periódico é invertido nas duas curvas nos dois gráficos. Na Figura 7.18 apresentamos as quatro curvas combinadas.

A Figura 7.21 ilustra o impacto da robustez a iluminação para imagens reais. Podemos ver uma melhora na estimativa, com o fluxo sendo calculado nas regiões mais escuras da

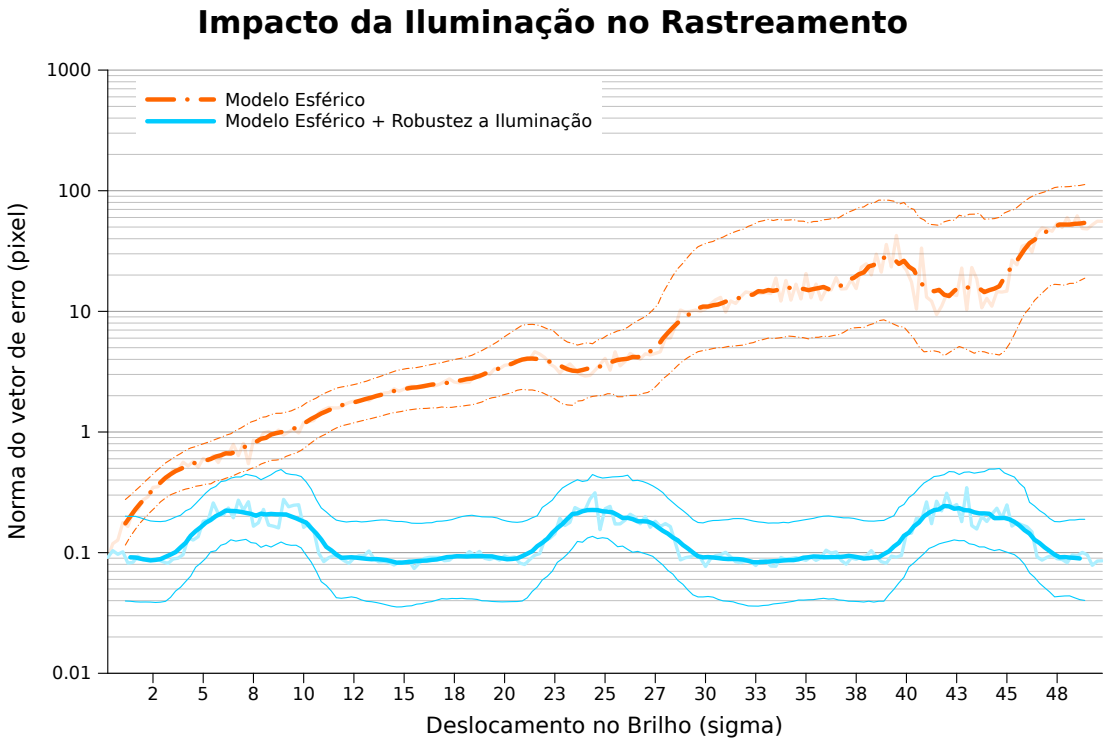


Figura 7.19: Impacto do componente σ para o modelo esférico de deformação $\boldsymbol{w}_{esf}$. As linhas mais finas marcam os quartis inferior e superior.

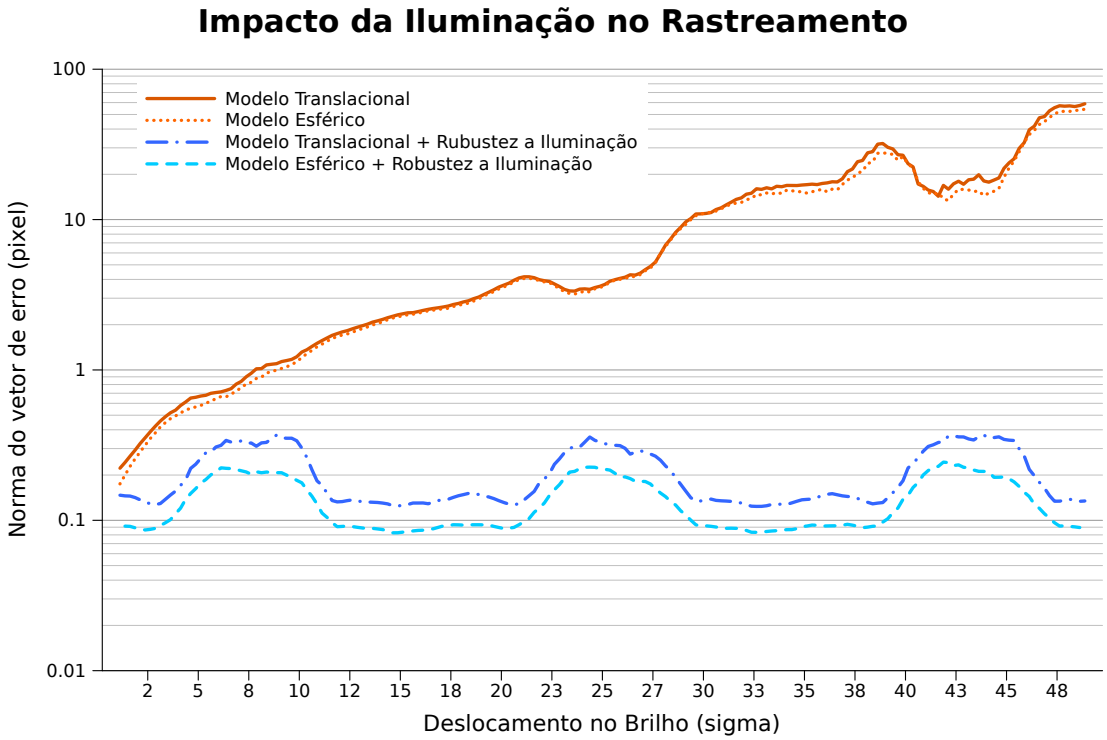


Figura 7.20: Impacto do componente σ para os dois modelos de deformação, $\boldsymbol{w}_{esf}$ e $\boldsymbol{w}_{trans}$

imagem e menos *outliers*.

7.5.2 Estimativa das Coordenadas dos Pontos

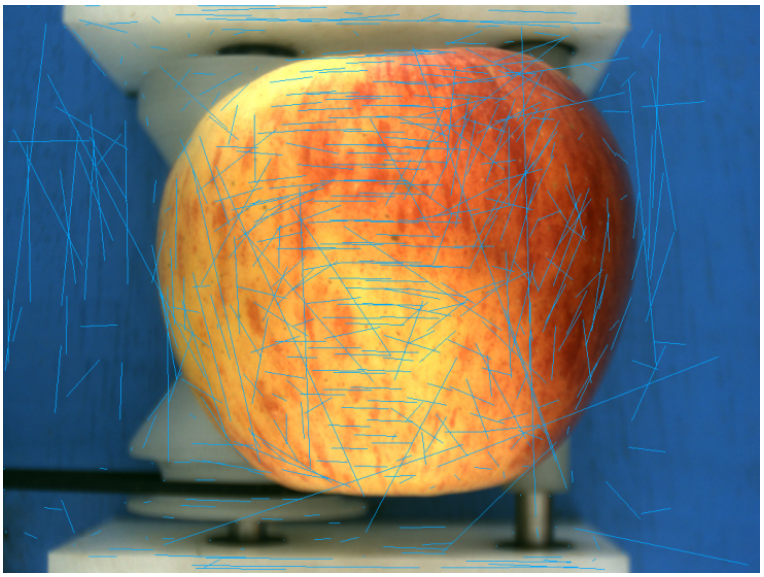
O algoritmo que apresentaremos agora tem por finalidade encontrar as coordenadas, na referência da fruta, dos pontos rastreados pelo algoritmo de fluxo óptico. Para tanto assumimos que o modelo de projeção é projetivo e que a fruta pode ser aproximada por uma esfera. Além disso assumimos que os parâmetros extrínsecos $\boldsymbol{P}$ e intrínseco $\boldsymbol{M}$ da câmera são conhecidos. Na primeira parte do algoritmo determinamos a coordenada do centro da fruta e seu raio. Em seguida, encontramos as coordenadas dos pontos rastreados na referência da esfera.

Antes de descrever o algoritmo faremos algumas observações a respeito da projeção de uma esfera usando um modelo projetivo de perspectiva. A primeira observação é que a projeção de uma esfera é sempre uma elipse. Para entender o porquê podemos pensar no cone cujo vértice coincide com o centro de projeção da câmera e que tangencia a esfera (ver Figura 7.22 na página 69). A imagem da esfera é a intersecção do plano de projeção com o cone, uma cônica. Se girarmos a câmera sem mover o centro de projeção mudamos o plano de projeção de lugar e a imagem da esfera anda na imagem. Se excluirmos o caso em que o plano de projeção passa pelo centro de projeção (um caso degenerado) e o outro em que a esfera intercepta o plano de projeção (fisicamente impossível), todas as outras configurações resultam em intersecções elípticas. A segunda observação é que o eixo maior da elipse sempre intercepta o eixo óptico.

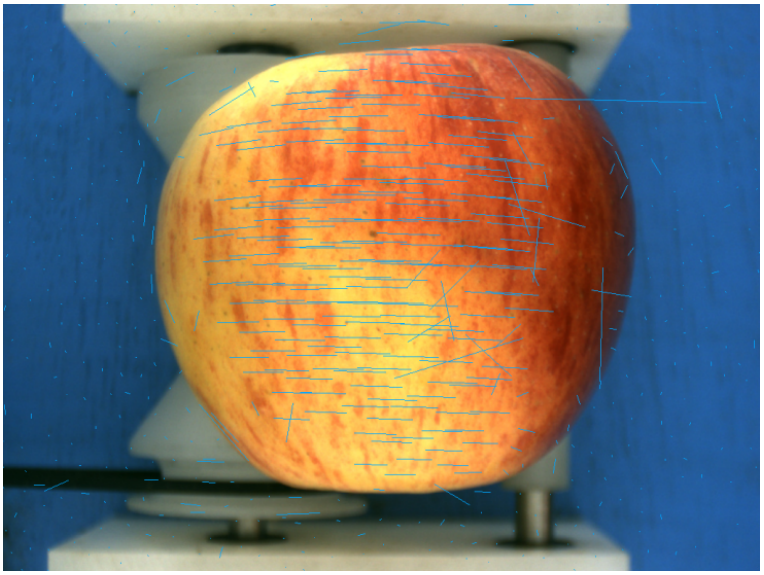
Começamos ajustando uma elipse à máscara da fruta usando mínimos quadrados. A abordagem é bastante direta. Calculamos a matriz de co-variância das coordenadas dos *pixels* correspondentes à fruta. Extraímos os autovalores e autovetores da matriz de co-variância. Os autovetores nos dão a direção dos eixos principais e os autovalores seus comprimentos (Figura 7.23). O algoritmo retorna o centro da elipse, o comprimento de seus dois eixos e o ângulo entre o eixo x e o eixo principal.

Encontrada a elipse, passamos para o problema de localizar duas retas que tangenciam a esfera em pontos diametralmente opostos e se interceptam no centro de projeção. Os pontos de intersecção entre a elipse e a reta que passa pela imagem do eixo óptico satisfazem este requerimento. Estas duas retas, junto com a reta que une a intersecção delas com o plano $z = 0$, formam um triângulo circunscrito à fatia da esfera (ver Figura 7.24). Como usamos retas que tocam a fruta em pontos diametralmente opostos, a circunferência dentro deste triângulo é maximal. Do triângulo extraímos o raio R da circunferência inscrita, que é o raio da esfera. Para encontrar a posição do centro da esfera determinamos o ponto de interseção da reta bissetriz do ângulo A da Figura 7.24 com o plano $z = R$.

Sabendo as coordenadas do centro da esfera e seu raio, calculamos a posição dos pontos



(a) modelo translacional



(b) modelo translacional + robustez a iluminação

Figura 7.21: Resultados do fluxo óptico, com e sem o modelo de iluminação.

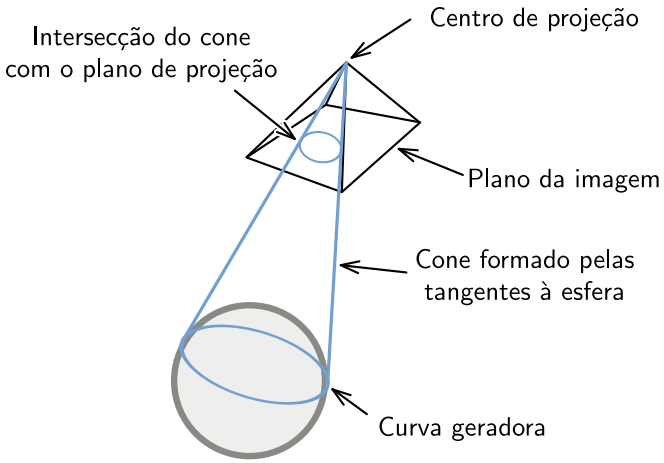


Figura 7.22: Projeção da esfera.

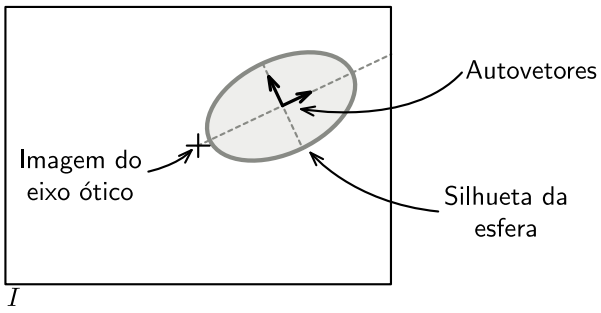


Figura 7.23: Projeção da esfera como uma elipse.

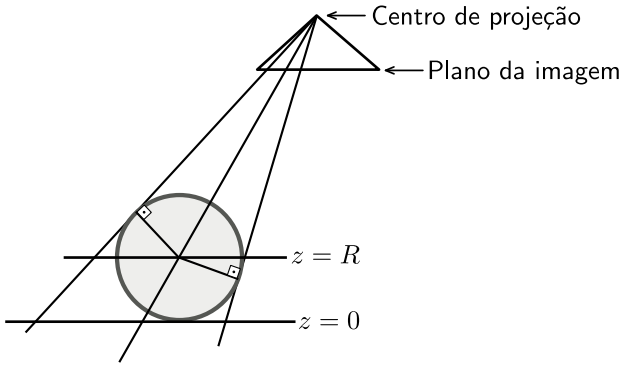


Figura 7.24: Fruta circunscrita por triângulo. Visão lateral da Figura 7.22.

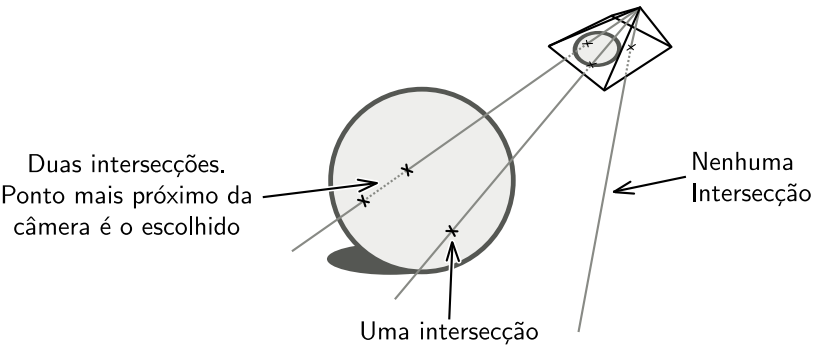


Figura 7.25: Obtenção das coordenadas dos pontos no espaço da esfera.

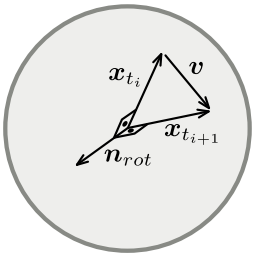


Figura 7.26: Vetores de diferença da posição.

rastreados em cada imagem como sendo a intersecção da esfera com a reta que começa no centro de projeção e passa pelo ponto na imagem (Figura 7.25). Se não houver intersecção descartamos o ponto, e se houverem dois pontos de intersecção ficamos com aquele que está mais próximo da câmera.

7.5.3 Estimativa de Movimento de Câmera

Dada a coordenada dos pontos rastreados na esfera precisamos determinar sua rotação de um quadro para o próximo. Se a captura fosse perfeita bastariam duas correspondências para determinar a rotação. Os vetores $\boldsymbol{v}_1$ e $\boldsymbol{v}_2$ (ver Figura 7.26) são coplanares ao plano de rotação. Para determinar a normal a este plano fazemos $\boldsymbol{n}_{rot} = \boldsymbol{v}_1 \times \boldsymbol{v}_2$. Para encontrar o ângulo de rotação subtraímos a componente paralela a $\boldsymbol{n}_{\text{rot}}$ de $\boldsymbol{v}_1$ e $\boldsymbol{v}_2$

$$\boldsymbol{v}_i^{\perp} = \boldsymbol{v}_i - \boldsymbol{v}_i^{\parallel} = \boldsymbol{v}_i - \boldsymbol{n}_{rot}(\boldsymbol{n}_{rot} \cdot \boldsymbol{v}_i)$$

e medimos o ângulo entre os dois

$$\theta = \cos^{-1} \frac{\boldsymbol{v}_1^{\perp} \cdot \boldsymbol{v}_2^{\perp}}{\|\boldsymbol{v}_1^{\perp}\| \|\boldsymbol{v}_2^{\perp}\|}.$$

Se a única fonte de imprecisão fosse a localização dos pontos poderíamos melhorar a estimativa fazendo a média de várias estimativas. No entanto, o maior problema ao

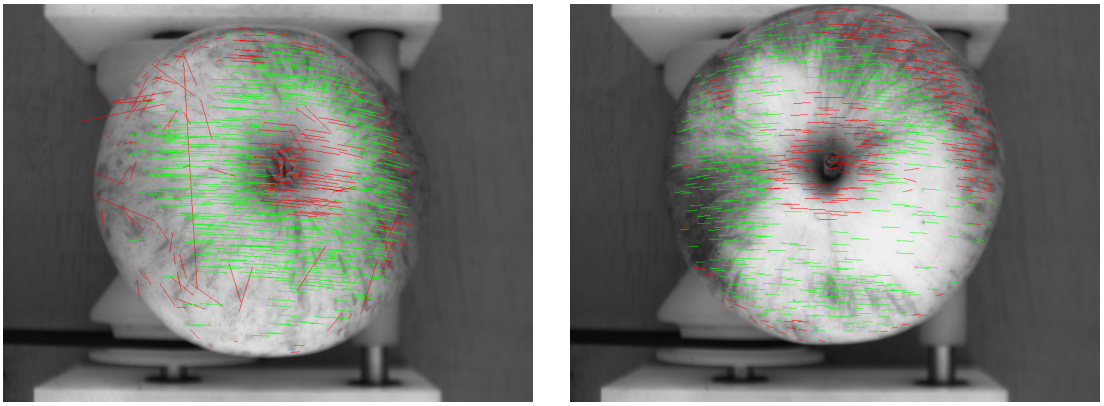


Figura 7.27: Correspondências selecionadas pelo RANSAC. Cor verde indica que foi selecionado, vermelho que foi rejeitado.

rastrear os pontos são os *outliers*, pontos aleatórios que não representam pontos verdadeiros corrompidos por ruído. Por serem pontos aleatórios sua presença em um grupo pode alterar de maneira arbitrária a média do grupo, técnicas robustas que conseguem ignorar sua contribuição precisam ser empregadas. Para determinar o ângulo de rotação de maneira robusta usamos o algoritmo RANSAC [12, 13]. A sua aplicação ao caso de determinação do ângulo pode ser vista no Algoritmo 7.3.

Determinado o plano e o ângulo de rotação, junto com a posição da câmera $\boldsymbol{P}$ e a posição do fruto $\boldsymbol{c}_{t_i}$ e $\boldsymbol{c}_{t_{i+1}}$ nos dois instantes de tempo, encontramos agora a matriz $\boldsymbol{P}_{t_i t_{i+1}}$ que nos leva da referência da câmera no instante t_i para o instante t_{i+1}

$$\boldsymbol{P}_{t_i t_{i+1}} = \boldsymbol{P} \begin{bmatrix} \boldsymbol{R} & -\boldsymbol{c}_{t_{i+1}} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_3 & -\boldsymbol{c}_{t_i} \\ \boldsymbol{0} & 1 \end{bmatrix}^{-1} \boldsymbol{P}^{-1},$$

onde $\boldsymbol{R}$ é a matriz de rotação do fruto.

7.5.3.1 Resultados

A Figura 7.27 mostra o resultado do RANSAC [11, 13] para duas imagens reais em que aparecem as extremidades da fruta. Podemos ver que o algoritmo consegue eliminar *outliers* corretamente e que é sensível o suficiente para eliminar as depressões na fruta (as duas extremidades). Também vemos aqui que alguns pontos foram incorretamente eliminados, principalmente na segunda imagem. Ajustamos os parâmetros do algoritmo de modo que fosse mais conservador porque o próximo passo da reconstrução, o refinamento, é muito sensível a *outliers*, então preferimos perder alguns pontos bons.

Algoritmo 7.3 RANSAC para determinação do ângulo e eixo de rotação.

procedimento RotaçãoEsfera-RANSAC($\boldsymbol{C}_{t_i}, \boldsymbol{C}_{t_{i+1}}, k, \epsilon_{con}, \epsilon_\phi, \epsilon_\theta, \epsilon_{\boldsymbol{n}_{rot}}$)

Entrada:
$\boldsymbol{C}_{t_i}, \boldsymbol{C}_{t_{i+1}}$: coordenada dos pontos, no espaço da fruta, nos instantes t_i e t_{i+1}
k: número máximo de iterações
ϵ_{con}: fração mínima de pontos dentro do consenso para parar o algoritmo
ϵ_ϕ: quanto o ângulo de um elemento pode diferir daquele do consenso
ϵ_θ: diferença máxima entre os ângulos de vetores sorteados
$\epsilon_{\boldsymbol{n}_{rot}}$: limiar do produto vetorial entre a normal do plano de rotação e o vetor candidato
Saída:
$\bar{\phi}$: ângulo de rotação
$\boldsymbol{n}^*_{rot}$: eixo de rotação
$status^*$: status dos pontos de $\boldsymbol{C}_{t_i}$ e $\boldsymbol{C}_{t_{i+1}}$

$n \leftarrow |\boldsymbol{C}_{t_i}|$ ▷ número de pontos, $|\boldsymbol{C}_{t_i}| = |\boldsymbol{C}_{t_{i+1}}|$
$minConsenso \leftarrow \epsilon_{con} * n$
$tamConsenso^* \leftarrow -1$
para $iter \leftarrow 1 \dots k$ **faça**
  $a, b \leftarrow$ Sortear2$(1 \dots n)$ ▷ escolhemos dois vetores distintos aleatoriamente
  $\boldsymbol{v}_1 \leftarrow \boldsymbol{C}_{t_{i+1}}[a] - \boldsymbol{C}_{t_i}[a]$
  $\boldsymbol{v}_2 \leftarrow \boldsymbol{C}_{t_{i+1}}[b] - \boldsymbol{C}_{t_i}[b]$
  $\boldsymbol{n}_{rot} \leftarrow \frac{\boldsymbol{v}_1 \times \boldsymbol{v}_2}{\|\boldsymbol{v}_1 \times \boldsymbol{v}_2\|}$ ▷ Eixo de rotação
  $\theta_1 \leftarrow$ ÂnguloEntreVetores$(\boldsymbol{C}_{t_i}[a] - \boldsymbol{n}_{rot} * (\boldsymbol{n}_{rot} \cdot \boldsymbol{C}_{t_i}[a]), \boldsymbol{C}_{t_{i+1}}[a] - \boldsymbol{n}_{rot} * (\boldsymbol{n}_{rot} \cdot \boldsymbol{C}_{t_{i+1}}[a]))$
  $\theta_2 \leftarrow$ ÂnguloEntreVetores$(\boldsymbol{C}_{t_i}[b] - \boldsymbol{n}_{rot} * (\boldsymbol{n}_{rot} \cdot \boldsymbol{C}_{t_i}[b]), \boldsymbol{C}_{t_{i+1}}[b] - \boldsymbol{n}_{rot} * (\boldsymbol{n}_{rot} \cdot \boldsymbol{C}_{t_{i+1}}[b]))$
  se $\|\theta_1 - \theta_2\| < \epsilon_\theta * \max(\|\theta_1\|, \|\theta_2\|)$ **então**
    $\theta \leftarrow \frac{\theta_1 + \theta_2}{2}$
    $\boldsymbol{\phi} \leftarrow$ ÂnguloEntreVetores$(\boldsymbol{C}_{t_i} - \boldsymbol{n}_{rot} * (\boldsymbol{n}_{rot} \cdot \boldsymbol{C}_{t_i}), \boldsymbol{C}_{t_{i+1}} - \boldsymbol{n}_{rot} * (\boldsymbol{n}_{rot} \cdot \boldsymbol{C}_{t_{i+1}}))$
    para $j \leftarrow 1 \dots n$ **faça**
      se $\|\theta - \boldsymbol{\phi}[j]\| < \epsilon_\phi * \|\theta\|$ **então**
        $\boldsymbol{v} \leftarrow \boldsymbol{C}_{t_{i+1}}[j] - \boldsymbol{C}_{t_i}[j]$
        se $\|\boldsymbol{v} \times \boldsymbol{n}_{rot}\| < \epsilon_{\boldsymbol{n}_{rot}} * \|\boldsymbol{v}\|$ **então**
          $status[j] \leftarrow \mathbf{V}$
        senão
          $status[j] \leftarrow \mathbf{F}$
        fim se
      senão
        $status[j] \leftarrow \mathbf{F}$
      fim se
      $tamConsenso \leftarrow |status = \mathbf{V}|$
      se $tamConsenso > tamConsenso^*$ **então**
        $\boldsymbol{\phi}^* \leftarrow \boldsymbol{\phi}$
        $status^* \leftarrow status$
        $\boldsymbol{n}^*_{rot} \leftarrow \boldsymbol{n}_{rot}$
        $tamConsenso^* \leftarrow tamConsenso$
      fim se
    fim para
  fim se
fim para
$\bar{\phi} \leftarrow$ Média$(\boldsymbol{\phi}^*)$
retorne $\bar{\phi}, \boldsymbol{n}^*_{rot}, status^*$
fim procedimento

7.5.4 Refinamento da Estimativa de Movimento de Câmera

Para reduzir o erro de nossa estimativa inicial precisamos primeiro de uma medida do erro. Optamos por usar a matriz essencial

$$\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R},$$

onde $\boldsymbol{t}$ e $\boldsymbol{R}$ vem da matriz de movimento da câmera

$$\boldsymbol{P}_{t_i t_{i+1}} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix}$$

e o operador $[\cdot]_\times$ representa a matriz equivalente ao produto vetorial

$$[\boldsymbol{x}]_\times \boldsymbol{y} = \boldsymbol{x} \times \boldsymbol{y}.$$

Se $\boldsymbol{x} = [x, y, z]^T$, então

$$[\boldsymbol{x}]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}.$$

Dada a correspondência $\boldsymbol{x}_{t_i}$ e $\boldsymbol{x}_{t_{i+1}}$, onde as coordenadas dos pontos são dadas em *pixels*, vale a seguinte equação [36]

$$\boldsymbol{x}_{t_i}^T \boldsymbol{M}^{-T} \boldsymbol{E} \boldsymbol{M}^{-1} \boldsymbol{x}_{t_{i+1}} = 0, \tag{7.31}$$

onde $\boldsymbol{M}$ é a matriz de parâmetros intrínsecos da câmera.

Se $\boldsymbol{C}$ é o conjunto de correspondências, podemos usar a Equação (7.31) para formular o problema como um problema de minimização, onde a função objetivo é dada por

$$O(\boldsymbol{\gamma}) = \sum_{\forall (\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_{i+1}}) \in \boldsymbol{C}} \left(\boldsymbol{x}_{t_i}^T \boldsymbol{M}^{-T} \boldsymbol{E}(\boldsymbol{\gamma}) \boldsymbol{M}^{-1} \boldsymbol{x}_{t_{i+1}} \right)^2,$$

e $\boldsymbol{E}(\boldsymbol{\gamma})$ representa a matriz essencial parametrizada pelo vetor $\boldsymbol{\gamma}$. A escolha da parametrização é importante e será discutida adiante. Como a função objetivo é não-linear decidimos usar o algoritmo de Levenberg–Marquardt [25] para encontrar seu mínimo. É um algoritmo simples de implementar e de rápida convergência, no entanto precisa de uma boa estimativa inicial e na sua forma mais simples não consegue lidar com *outliers*.

Fizemos testes com três parametrizações da matriz $\boldsymbol{E}$. Descreveremos cada uma delas a seguir.

A primeira, e mais simples, é dada por

$$\boldsymbol{\gamma}_1 = [r_{11}, r_{12}, r_{13}, r_{21}, \ldots, r_{33}, t_1, t_2, t_3],$$

onde $\boldsymbol{R} = [r_{ij}]$ e $\boldsymbol{t} = [t_i]$. Ela não impõe restrição aos elementos de $\boldsymbol{E}$, o que poder ser problemático para pontos iniciais muito longe do ponto ótimo. A matriz $\boldsymbol{R}$ rapidamente pode deixar de ser uma matriz de rotação e em muitos casos a norma do vetor $\boldsymbol{t}$ se aproxima de zero, o que é uma resposta que de fato minimiza a função objetivo (i.e. se $\boldsymbol{t} = \boldsymbol{0}$, então $\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R} = [\boldsymbol{0}]_\times \boldsymbol{R} = \boldsymbol{0}$ e $\boldsymbol{x}_{t_i}^T \boldsymbol{M}^{-T} \boldsymbol{E} \boldsymbol{M}^{-1} \boldsymbol{x}_{t_{i+1}} = \boldsymbol{x}_{t_i}^T \boldsymbol{M}^{-T} \boldsymbol{0} \boldsymbol{M}^{-1} \boldsymbol{x}_{t_{i+1}} = \boldsymbol{0}$).

A segunda parametrização tenta obrigar a matriz $\boldsymbol{R}$ a se manter uma matriz de rotação ao quebrá-la em ângulo de rotação θ e vetor unitário normal ao plano de rotação $\boldsymbol{n}_{rot}$

$$\boldsymbol{\gamma}_2 = [\theta, r_1, r_2, r_3, t_1, t_2, t_3],$$

com $\boldsymbol{n}_{rot} = [r_i]$. Para encontrar a matriz $\boldsymbol{R}$ usamos a fórmula de Rodrigues

$$\boldsymbol{R} = \boldsymbol{I} \cos\theta + \sin\theta [\boldsymbol{n}_{rot}]_\times + (1 - \cos\theta) \boldsymbol{n}_{rot} \boldsymbol{n}_{rot}^T.$$

Mesmo assim, o vetor $\boldsymbol{n}_{rot}$ é livre e sua norma pode se afastar de 1 à medida que o algoritmo avança. Outro problema que ainda persiste é que o vetor de translação pode se tornar muito pequeno.

Para restringir a norma dos vetores $\boldsymbol{t}$ e $\boldsymbol{r}$ a 1 usamos uma parametrização com duas variáveis, os dois ângulos usados para expressar pontos usando coordenadas esféricas. Um vetor unitário $\boldsymbol{x}$ pode ser escrito da seguinte forma

$$\boldsymbol{x} = \begin{bmatrix} \sin\alpha_{\boldsymbol{x}} \cos\beta_{\boldsymbol{x}} \\ \sin\alpha_{\boldsymbol{x}} \sin\beta_{\boldsymbol{x}} \\ \cos\alpha_{\boldsymbol{x}} \end{bmatrix}, \qquad \alpha_{\boldsymbol{x}} \in [0, \pi], \beta_{\boldsymbol{x}} \in [0, 2\pi]$$

e a nova parametrização se torna

$$\boldsymbol{\gamma}_3 = [\theta, \alpha_{\boldsymbol{n}_{rot}}, \beta_{\boldsymbol{n}_{rot}}, \alpha_{\boldsymbol{t}}, \beta_{\boldsymbol{t}}].$$

Note que todas as parametrizações usadas nos permitem recuperar $\boldsymbol{R}$ e $\boldsymbol{t}$ prontamente, ao invés de recuperarmos $\boldsymbol{E}$ e a partir daí obter a rotação e translação. Embora esta escolha torne as parametrizações mais complexas usamos ela pois os algoritmos para recuperar $\boldsymbol{R}$ e $\boldsymbol{t}$ são não triviais. Além disso esta escolha nos permite restringir os valores dos elementos de $\boldsymbol{E}$ mais facilmente.

7.5.4.1 Testes

Para entender melhor o impacto das diferentes parametrizações e também a influência dos diferentes erros na hora de estimar o movimento de câmera, realizamos testes com dados artificiais. A vantagem é que conseguimos modelar cada uma das fontes de erro da fase de estimativa isoladamente e analisar seu impacto na recuperação dos parâmetros reais.

Fonte de erro	Unidade	Desvio Padrão	Observação
Projeção dos pontos	*pixels*	0.2	
Centro da fruta	1 raio da fruta	0.16	Erro de até meio raio da fruta
Eixo de rotação da fruta	radianos	0.26	Erro de até 45°
Ângulo de rotação da fruta	radianos	0.13	Erro de até 22.5°

Tabela 7.1: Fontes de erro modeladas.

Outra vantagem é que criamos testes livres de *outliers*, anomalia que não foi prevista na formulação presente.

A cena emula o que é capturado dentro de uma máquina classificadora, com uma câmera na parte superior observando uma fruta girar embaixo, a uma de distância de 35cm. A matriz de parâmetros intrínsecos é

$$\begin{bmatrix} 1800 & 0 & 638 \\ & 1800 & 403 \\ & & 1 \end{bmatrix}.$$

Como objeto sendo rastreado usamos o modelo de uma maçã.

Para cada teste colocamos a fruta em duas posições simulando o movimento da esteira (Figura 7.29). A orientação na primeira é sempre igual. A orientação da segunda emula o giro da fruta na esteira e o ângulo de rotação segue uma distribuição gaussiana, de modo a fornecer em média 7.5 fotos com um desvio padrão de 0.8. Para cada teste sorteamos 40 pontos que são visíveis nas duas imagens. Escolhidas as posições da fruta determinamos a matriz $\boldsymbol{P}_{t_i t_{i+1}}$ que descreve o movimento da câmera com relação à fruta. Também criamos $\tilde{\boldsymbol{P}}_{t_i t_{i+1}}$, uma versão corrompida por diversas fontes de ruído. A matriz $\tilde{\boldsymbol{P}}_{t_i t_{i+1}}$ junto com a projeção dos pontos é passada à rotina de otimização para obter $\boldsymbol{P}'_{t_i t_{i+1}}$. Usamos como erro $\|\boldsymbol{P}_{t_i t_{i+1}} - \boldsymbol{P}'_{t_i t_{i+1}}\|$.

A tabela 7.1 lista os tipos de erro inserido e a Figura 7.28 oferece uma ilustração de cada um deles. O primeiro, "Projeção dos pontos", se refere à coordenada do ponto na imagem (medida em *pixels*). "Centro da fruta" se refere à posição da fruta no espaço tridimensional, para os dois instantes de tempo. Todos as fontes de erro foram modeladas como erros gaussianos aditivos com média zero.

Rodamos três conjuntos de testes, uma para cada parametrização do movimento de câmera. Para cada conjunto rodamos uma bateria de testes para cada tipo de erro isolado e mais duas, uma com todos os tipos de erro e outra com todos os tipos de erro menos o erro na projeção dos pontos. Cada bateria consistia de 10000 testes.

Os gráficos das Figuras 7.30, 7.31 e 7.32 mostram os histogramas de erro para as três parametrizações. Podemos ver uma melhora gradativa seguindo a ordem: $\boldsymbol{\gamma}_1$, $\boldsymbol{\gamma}_2$

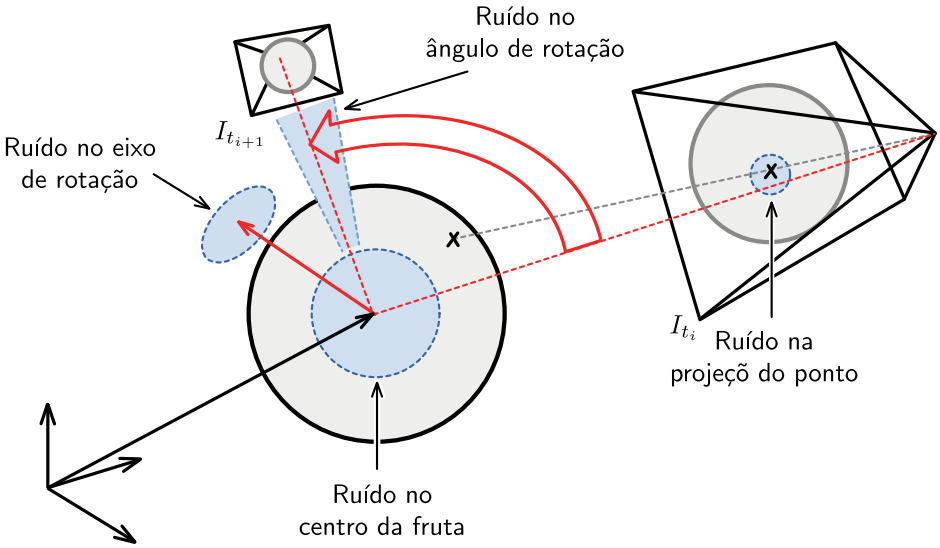


Figura 7.28: Diversos tipos de erro na estimativa do movimento de câmera.

e γ_3. O erro que mais prejudica a otimização é o erro na projeção dos pontos, o que fica evidenciado comparando os gráficos das Figuras 7.33 e 7.34. O pico observado para γ_1 no gráfico com todos os tipos de ruído é causada por soluções em que a componente da translação de $\boldsymbol{P}'_{t_i t_{i+1}}$ se iguala a zero (lembrando que esta parametrização não impõe nenhuma restrição à translação). A parametrização γ_3 consegue praticamente eliminar o erro.

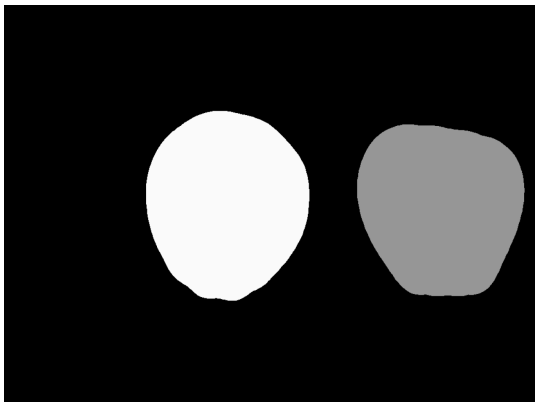


Figura 7.29: Exemplo de movimento típico entre dois quadros usados para teste.

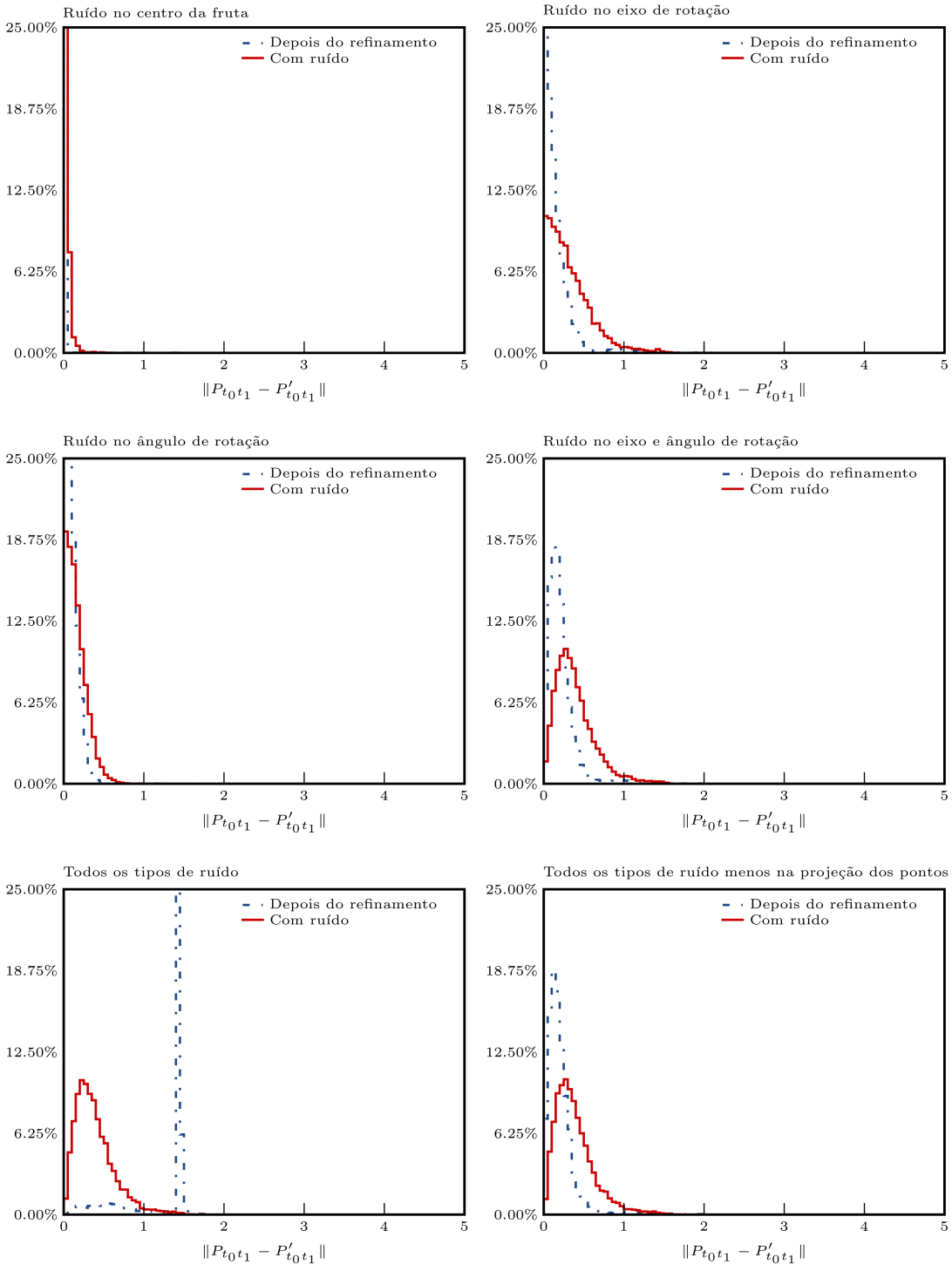


Figura 7.30: Efeito dos diversos tipos de ruído para $\boldsymbol{\gamma}_1$.

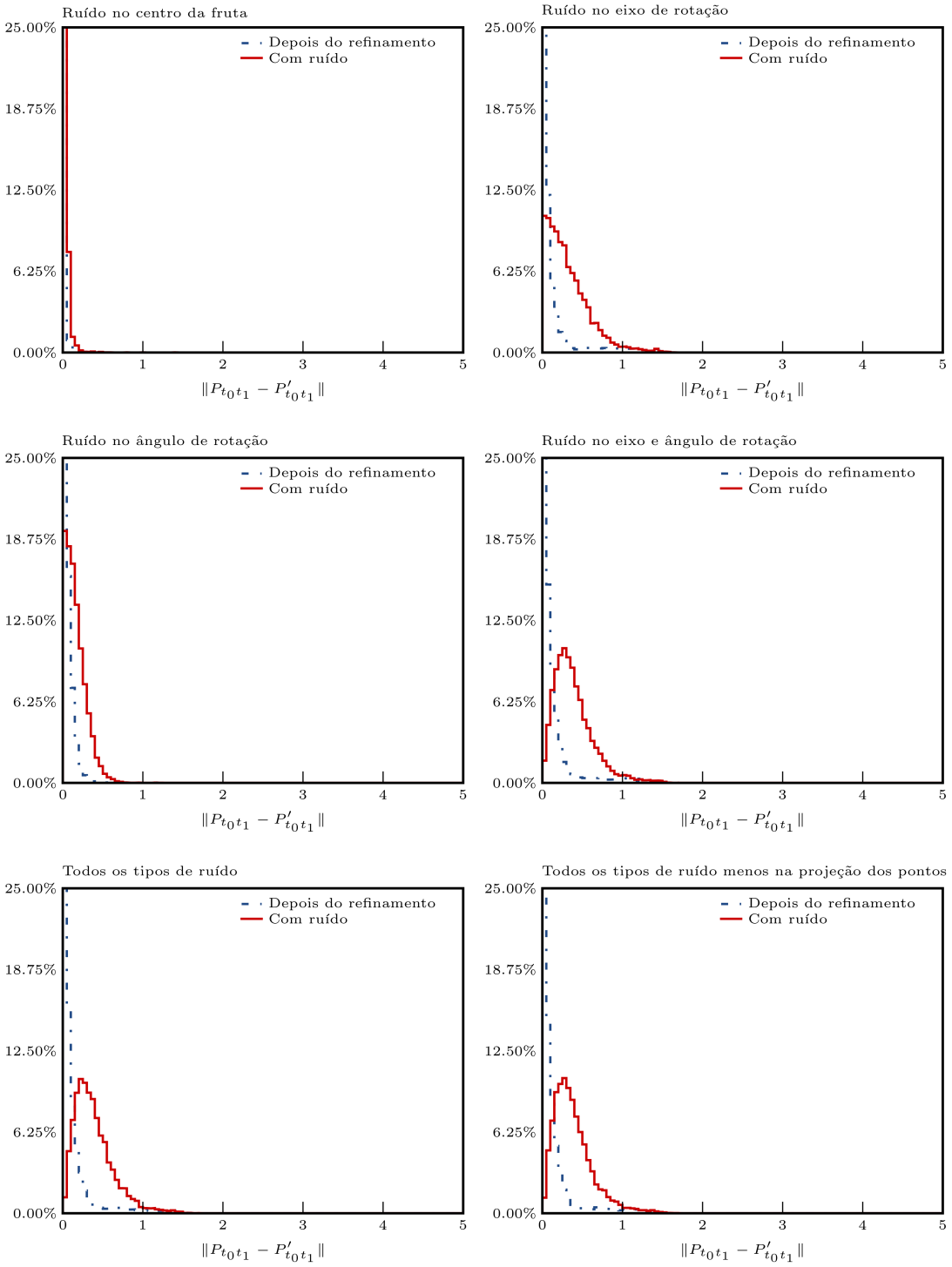


Figura 7.31: Efeito dos diversos tipos de ruído para $\boldsymbol{\gamma}_2$.

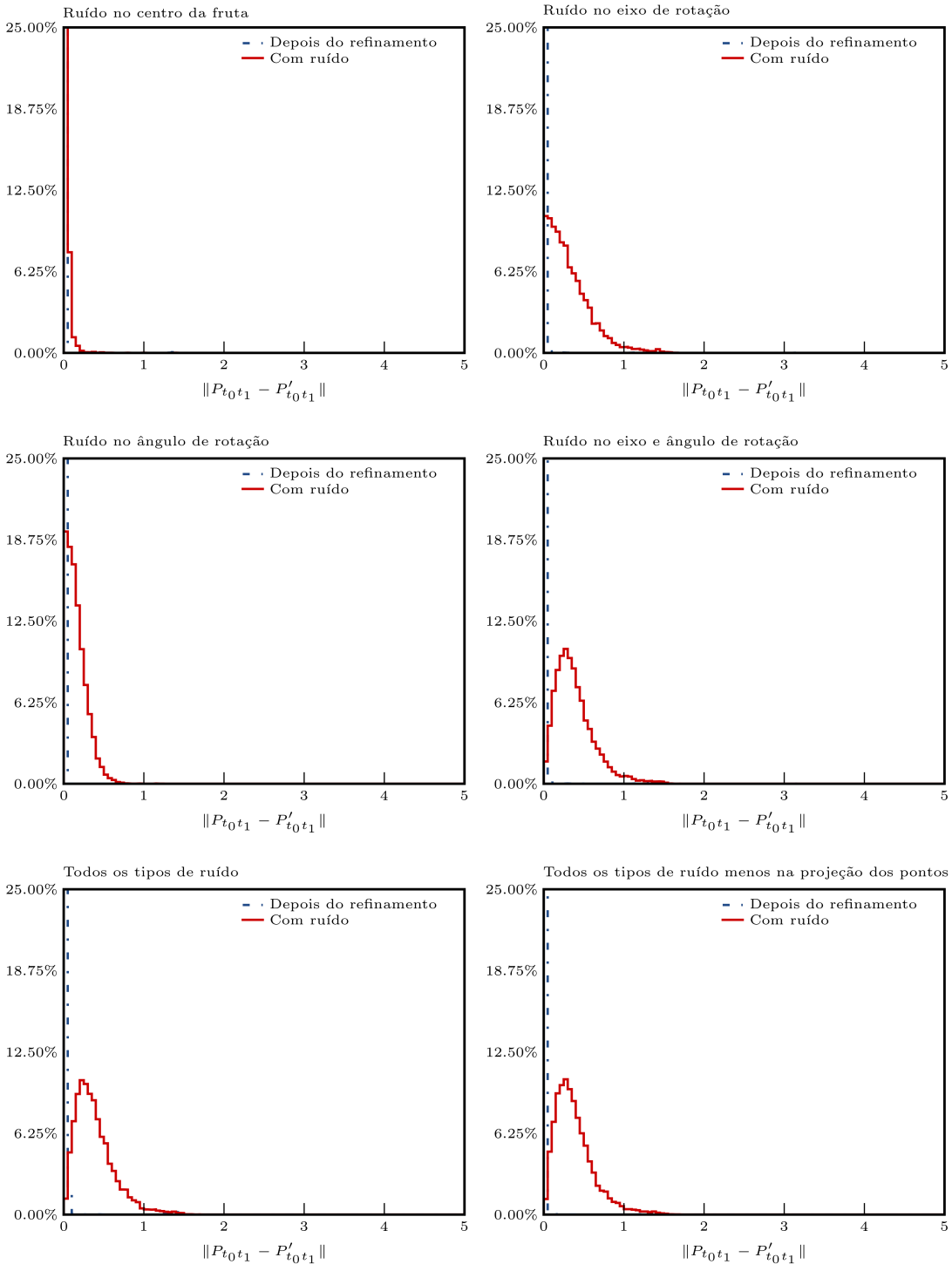


Figura 7.32: Efeito dos diversos tipos de ruído para $\boldsymbol{\gamma}_3$.

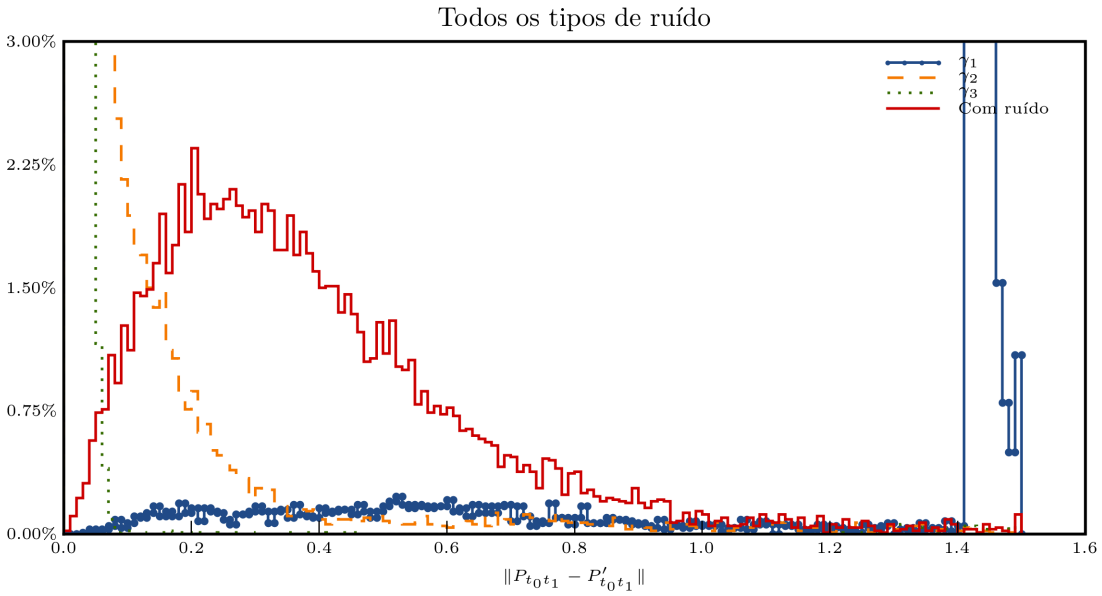


Figura 7.33: Performance das três parametrizações sob o efeito de todos os tipos de ruído.

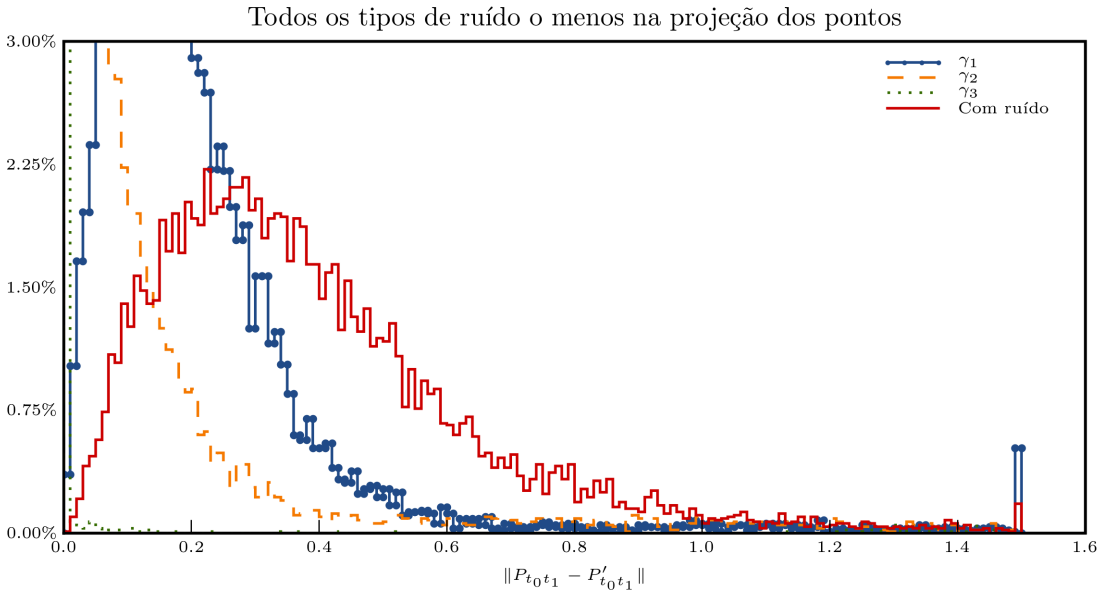


Figura 7.34: Performance das três parametrizações sob o efeito de todos os tipos de ruído exceto o de projeção dos pontos.

7.5.5 Reposicionamento das Silhuetas

O passo final para o algoritmo de reconstrução da fruta é o reposicionamento das silhuetas encontradas pela subtração de fundo. Fizemos testes com dois algoritmos. Explicaremos primeiro o algoritmo mais simples.

Para encontrar a posição da silhueta no espaço tridimensional olhamos para o cone formado pelas retas que partem do centro de projeção e passam pela silhueta na imagem. Estes pontos tangenciam o objeto em apenas um ponto, chamamos a curva que passa por todos estes pontos de curva geradora, pois é a imagem dela que gera a silhueta. Se o objeto for uma esfera (existem outros sólidos que apresentam a mesma propriedade mas não entraremos nestes detalhes aqui) estes pontos de tangência são coplanares. Encontrando a distância deste plano até o centro de projeção podemos transportar os pontos da silhueta ao longo deste cone até a distância encontrada. No primeiro algoritmo que propomos aproximamos a fruta por uma esfera para calcular a distância dos pontos de tangência até o centro de projeção, veja a Figura 7.22.

O segundo algoritmo proposto tenta melhorar a estimativa da profundidade dos pontos usando pontos especiais do contorno chamados pontos de fronteira (*frontier points* ou *epipolar tangency points*) [16]. Estes pontos ocorrem na intersecção de duas curvas geradoras (Figura 7.35). A importância deles é que, embora sejam pontos da silhueta que geralmente não oferecem informação sobre sua profundidade, são pontos reais da cena e através de triangulação conseguimos determinar sua distância até a câmera.

Para encontrar os pontos de fronteira observamos que as tangentes às curvas geradoras nas duas imagens formam um plano e que este plano contém o ponto de fronteira e os dois centros de projeção. Além disso, a imagem deste plano é uma reta tangente às silhuetas nas duas imagens. Desta forma, a imagem do plano deve ser uma linha epipolar que tangencia a silhueta. Se encontrarmos todos os pontos que satisfazem esta restrição nas duas imagens, ainda precisamos determinar a correspondência entre eles. Determinamos esta correspondência encontrando os pares de pontos que satisfazem a Equação (7.31).

7.5.5.1 Resultados

Um exemplo de reconstrução usando a primeira abordagem pode ser vista na Figura 7.37. Algumas das imagens que foram usadas por esta reconstrução podem ser vistas na Figura 7.36. Esta abordagem obteve bons resultados para situações em que o eixo de rotação era próximo do eixo de simetria da fruta, já para o caso em que estes dois eixos eram perpendiculares os resultados não foram satisfatórios. O principal problema nestes casos é que nós aproximamos a fruta por uma esfera, e no caso em que a fruta gira com o eixo de rotação perpendicular ao eixo de simetria, o raio da esfera aproximada varia de imagem para imagem. Como exemplo vamos considerar uma fruta de formato elongado,

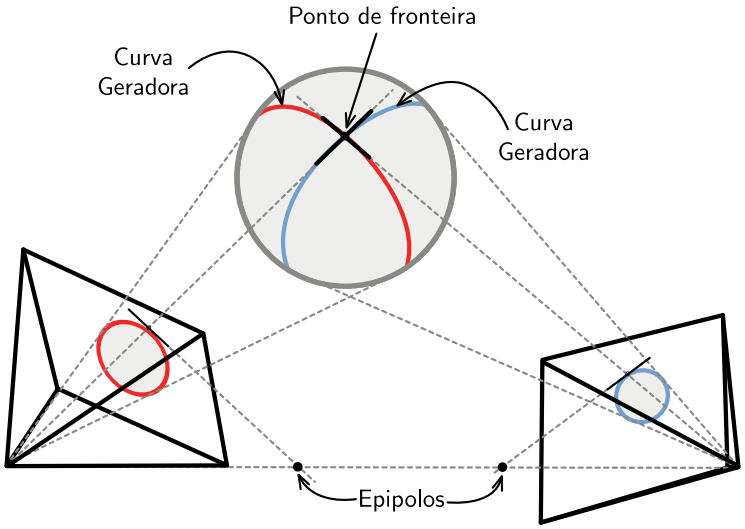


Figura 7.35: Pontos de fronteira.

quando o eixo de simetria aponta em direção à câmera a área da sua imagem é mínima, o que corresponde a uma esfera pequena e uma distância entre a silhueta e o centro de projeção máximo (justamente o contrário do que deveria ser). Quando o eixo de simetria é perpendicular ao plano da imagem a situação se inverte. Nos dois casos a distância entre a silhueta e o centro de projeção são erroneamente calculados, o que resulta em erros de reconstrução. O conceito pode ser entendido através da Figura 7.38.

O segundo algoritmo, usando pontos de fronteira, obteve bons resultados em simulações mas não com imagens reais. O maior problema é a determinação precisa da silhueta e depois da reta tangente, o que não foi possível. Outra dificuldade foi que a determinação dos pontos de fronteira se torna muito instável para movimentos pequenos de câmera.

7.6 Localização do eixo Pedúnculo Cálice

O resultado final do algoritmo de reconstrução é a nuvem de pontos no espaço tridimensional e a posição da câmera, em relação a esta nuvem de pontos, em cada uma das imagens. Uma informação bastante importante para a classificação da fruta que podemos extrair destes pontos é a localização do pedúnculo e do cálice. Para encontrar estes dois elementos localizamos o eixo de simetria da fruta, o pedúnculo e o cálice se encontram nas extremidades deste eixo.

Para encontrar o eixo de simetria dos pontos calculamos a matriz de co-variância da localização dos pontos e seu centro de massa. Os autovetores da matriz de co-variância nos dão as principais direções de variação enquanto que seus autovalores nos informam

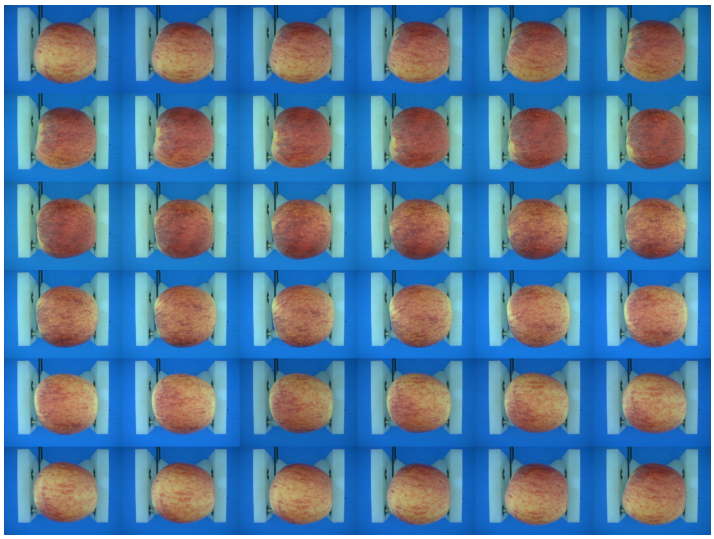


Figura 7.36: Imagens usadas para a reconstrução.

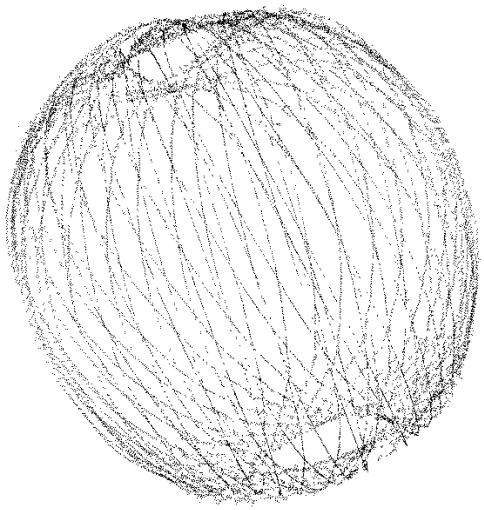


Figura 7.37: Reconstrução 3D da fruta.

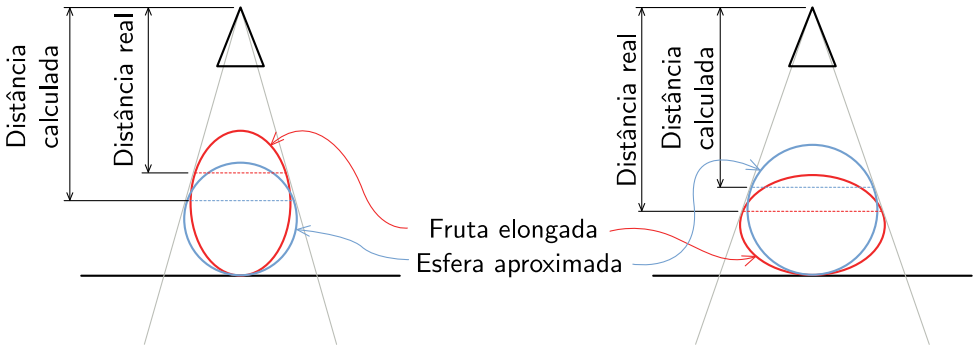


Figura 7.38: Fonte de erro durante a reconstrução.

a magnitude desta variação. Como a fruta é um objeto radialmente simétrico, podemos esperar que dois dos autovalores sejam iguais (a intersecção do objeto com um plano perpendicular ao eixo de simetria resulta em um círculo). Desta forma, localizamos o eixo de simetria ao determinar quais dos autovalores são iguais e escolhemos o autovetor correspondente ao autovalor que é diferente. O eixo de simetria é a reta que contém o centro de massa e é paralela ao autovetor escolhido.

Uma vantagem deste método com relação aos outros propostos na literatura é que, junto com a informação sobre a posição das câmeras, este método indica a posição do eixo de simetria na imagem mesmo quando o pedúnculo ou cálice não são visíveis.

Esperamos que o método falhe quando a fruta se aproxima muito de uma esfera. Neste caso os autovalores se tornam muito próximos e pequenas oscilações em seus valores nos levam a escolher o autovalor incorreto.

Após a localização do eixo de simetria passamos ao problema de determinar a extensão da fruta ao longo deste eixo. Esta informação é importante para podermos marcar a posição do pedúnculo e cálice nas imagens. Sabendo sua localização podemos desenvolver algoritmos específicos para a localização de defeitos fora e dentro destas regiões.

Para determinar a extensão deste eixo projetamos todos os pontos no eixo de simetria, para cada ponto $\boldsymbol{x}_i$ da nuvem calculamos a distância d_i da sua projeção no eixo até o centro de massa dos pontos . Acumulamos um histograma a partir dos valores de d_i e após aplicar um limiar para eliminar *bins* com poucos elementos, encontramos a extensão do eixo como sendo os dois *bins* mais externos que têm algum elemento (Figura 7.39).

7.6.1 Resultados

A Figura 7.40 mostra o resultado da localização para uma fruta girando com o eixo de simetria aproximadamente paralelo ao eixo de rotação. A Figura 7.41 mostra o resultado para o eixo de simetria perpendicular ao eixo de rotação.

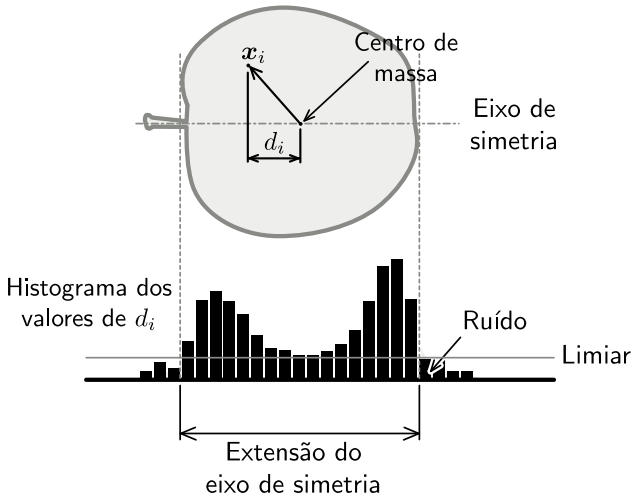


Figura 7.39: Histograma de d_i.

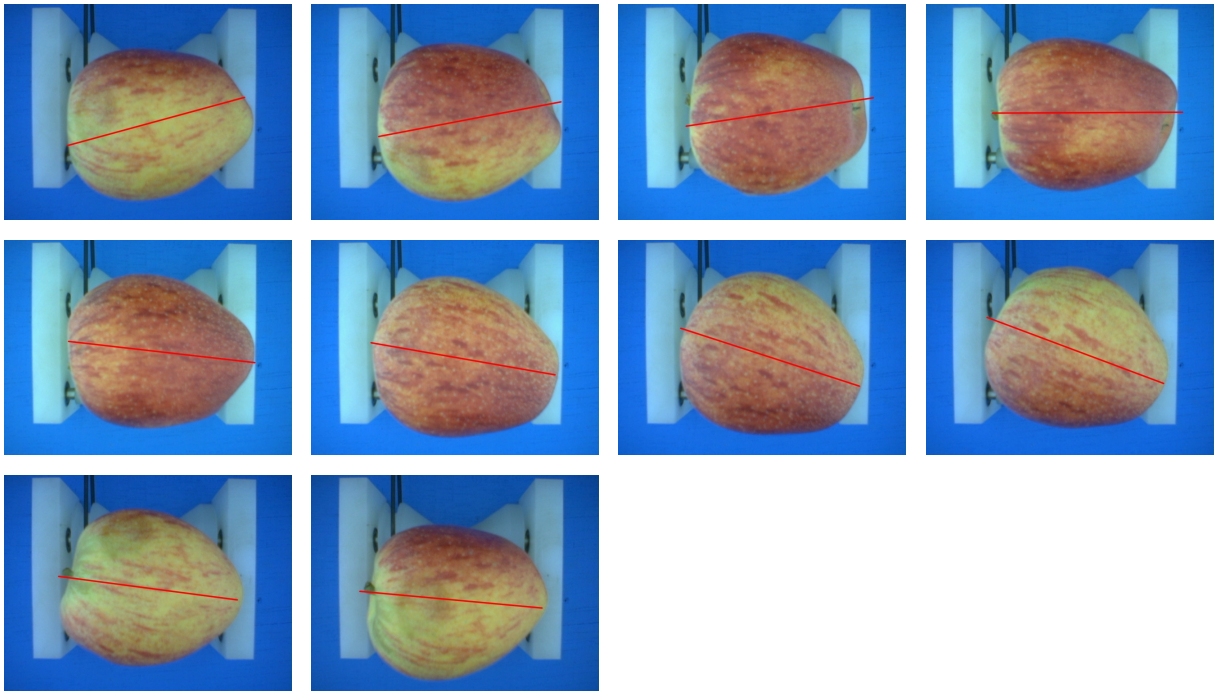


Figura 7.40: Resultado da localização do cálice e pedúnculo para eixo de simetria próximo ao eixo rotação.

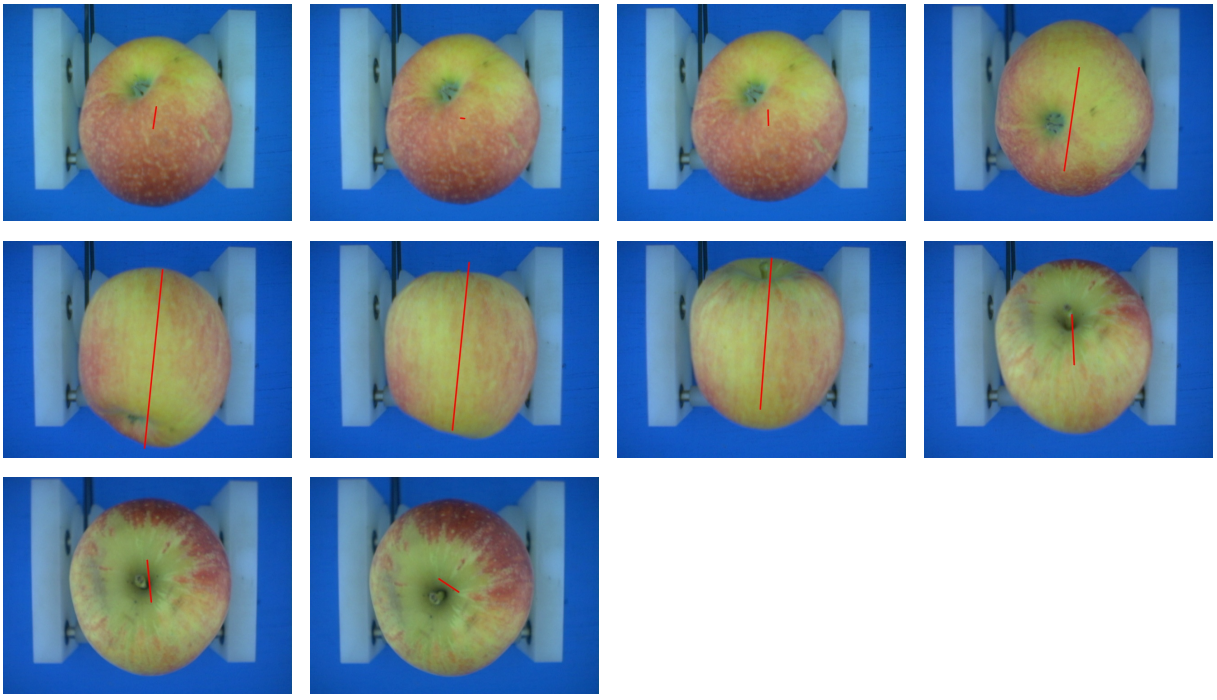


Figura 7.41: Resultado da localização do cálice e pedúnculo para eixo de simetria perpendicular ao eixo rotação.

Capítulo 8

Conclusões

Construímos um aparato de captura que simula o ambiente real de uma máquina de classificação. Desenvolvemos um aplicativo para permitir uma captura rápida das bases de dados e em uma viagem ao sul do Brasil criamos quatro bases significativas com uma gama variada de defeitos e as principais variedades cultivadas no país.

Desenvolvemos um sistema simples e efetivo de rastreamento de frutas em uma esteira mecânica. O sistema foi validado e os resultados mostraram uma performance satisfatória.

Também desenvolvemos um algoritmo de reconstrução tridimensional da fruta. Embora os resultados do sistema como um todo não são ainda suficientes para a aplicação na indústria, obtivemos bons resultados em algumas das etapas do processo: fluxo óptico, estimativa inicial do movimento de câmera e refinamento da estimativa de movimento.

Desenvolvemos e testamos variantes do fluxo óptico diferencial otimizados para o nosso problema. Nos testes que realizamos pudemos ver que a variação da iluminação era o fator que mais influenciava o rastreamento de pontos. A adaptação esférica, embora tenha permitido ganhos em imagens sintéticas, mostrou-se pouco efetiva para imagens reais. Acreditamos que a principal razão para isso seja que nas imagens reais a câmera usada fugia bastante do modelo ortográfico de projeção, o que reduzia a distorção da imagem nas bordas da fruta. Além disso, o fluxo óptico com modelo esférico de deformação torna o algoritmo consideravelmente mais complexo e sua implementação é da ordem de 10 vezes mais lenta. Por esta razão decidimos usar apenas a variante com robustez a iluminação e modelo translacional de deformação para desenvolvimentos futuros.

Criamos um método para fazer uma primeira estimativa da rotação da fruta. O método aproxima a fruta por uma esfera e se mostrou sensível o suficiente para discriminar depressões causadas pelas extremidades da fruta. Através de simulações verificamos que esta estimativa inicial do movimento é precisa o suficiente para que o método de otimização não-linear encontre o movimento correto.

Testamos três parametrizações para o movimento de câmera para a otimização não-

linear e verificamos a eficácia de cada uma delas através de simulações que levavam em consideração os erros cometidos pelo algoritmos de estimativa de movimento.

Apresentamos duas maneiras para a reconstrução da fruta através de sua silhueta. A primeira aproxima a fruta por uma esfera para determinar a distância da silhueta até o centro de projeção, a segunda usa pontos de fronteira para encontrar a distância de maneira mais precisa. A primeira abordagem mostrou resultados razoáveis para frutas que giravam com o eixo de simetria paralelo ao eixo de rotação, mas falhou na grande maioria dos casos em o eixo de rotação era perpendicular ao eixo de simetria.

A segunda abordagem mostrou bons resultados nas simulações porém mais trabalho precisa ser feito para que possamos aplicá-lo a imagens reais. O principal problema é a determinação precisa da silhueta, que é necessária para o cálculo da reta tangente.

Desenvolvemos um método para localizar as extremidades da fruta baseado no eixo de simetria, a partir dos pontos obtidos pela reconstrução tridimensional. Em testes com imagens reais obtivemos resultados bons para frutas que giravam com seu eixo de simetria próximo do eixo de rotação mas performance inferior para os casos em que os eixos eram perpendiculares. O principal limitador nestes casos era a má qualidade da reconstrução. Esperamos que o desempenho do método melhore quando conseguirmos melhorar a qualidade da reconstrução.

8.1 Trabalhos futuros

O intuito inicial deste mestrado era o desenvolvimento de um sistema completo para a classificação de maçãs, por isso a construção das grandes bases de dados. Um trabalho futuro seria a rotulação destas bases e o desenvolvimento de algoritmos de classificação.

Para o algoritmo de rastreamento de frutos seria interessante o desenvolvimento de um método para a determinação automática de parâmetros como o limiar do subtrator de fundo e o tamanho das frutas. Outro desenvolvimento desejável seria reduzir o tempo gasto na convolução para a separação das frutas, uma máscara separável iria ajudar neste sentido. Uma máscara baseada na diferença de gaussianas seria uma solução a ser explorada. Poderíamos também investigar qual o tamanho mínimo de imagem, quando comparado com o raio da fruta, que rende bons resultados. Desta maneira poderíamos reduzir as imagens e acelerar o algoritmo sem perda de precisão na localização dos frutos.

Sobre o algoritmo de fluxo ótico, gostaríamos de aumentar movimento entre imagens, para isso poderíamos recorrer a algum algoritmo mais simples (como a correlação) para obter uma estimativa inicial para o movimento da fruta. Usando esta estimativa inicial no fluxo óptico para obter a correspondência de maneira mais precisa. Também poderíamos supor uma velocidade constante de rotação da fruta.

Seria interessante investigar outras maneiras de obter a estimativa inicial de rotação da

fruta, que não impusessem a aproximação por esfera. Isto permitiria o uso do algoritmo para frutas não esféricas, como a pêra.

O refinamento da estimativa do movimento de câmera se beneficiaria de uma formulação em que *outliers* pudessem fazer parte das correspondências [35]. Uma formulação de máxima verosimilhança seria uma abordagem interessante.

Para tornar o algoritmo de reconstrução baseado em pontos de fronteira viável teríamos que investir em métodos mais precisos para encontrar a silhueta da fruta.

Referências Bibliográficas

[1] Agricultura em números 2005. Publicação eletrônica, novembro 2006. `http://www.agricultura.gov.br`. 2

[2] Major food and agricultural commodities and producers. Publicação eletrônica, 2008. `http://www.fao.org`. (document), 2.1

[3] BS Bennedsen, DL Peterson, and A. Tabb. Identifying defects in images of rotating apples. *Computers and Electronics in Agriculture*, 48(2):92–102, 2005. 3.2

[4] Z. Bin, J. Lu, and T. Yang. Three-dimensional shape enhanced transform for automatic apple stem-end/calyx identification. *Optical Engineering*, 46:017201, 2007. 3.2, 7, 7.3

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 5

[6] J. Blasco, N. Aleixos, and E. Moltó. Machine vision system for automatic quality grading of fruit. *Biosystems Engineering*, 85(4):415–423, June 2003. 3.1, 3.3

[7] Jean-Yves Bouguet. *Visual methods for three-dimensional modeling*. PhD thesis, California Institute of Technology, 1999. 7.3

[8] J.Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. *Intel Corporation, Microprocessor Research Labs, OpenCV Documents*, 1999. 7.5.1.2

[9] X. Cheng, Y. Tao, YR Chen, and Y. Luo. Nir/mir dual–sensor machine vision system for online apple stem–end/calyx recognition. *Transactions of the ASAE*, 46(2):551–558, 2003. 3.1, 3.2

[10] Marcus Vinicius Pratini de Moraes. Instrução normativa n° 50, 9 2002. 2.1, 2.5

[11] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 5, 7.5.3.1

[12] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 7.5.3

[13] D.A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002. 5, 7.5.3, 7.5.3.1

[14] Y. Furukawa, A. Sethi, J. Ponce, and D.J. Kriegman. Robust Structure and Motion from Outlines of Smooth Curved Surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(2):302–315, February 2006. 7.3

[15] Gregory D. Hager and Peter N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(10):1025–1039, 1998. 7.5.1.1, 7.5.1.2

[16] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. 7.5.5

[17] B.K.P. Horn and B.G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17(1-3):185–203, 1981. 7.5.1.2

[18] Hailin Jin, Paolo Favaro, and Stefano Soatto. Real-time feature tracking and outlier rejection with changes in illumination. *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, pages 684–689, 2001. 7.5.1.1

[19] K. Kang, J.P. Tarel, R. Fishman, and B.D. Cooper. A Linear Dual-Space Approach to 3D Surface Reconstruction from Occluding Contours Using Algebraic Surface. In *ICCV*, pages 198–204, 2001. 7.3

[20] V. Leemans and M.-F. Destain. A real-time grading method of apples based on features extracted from defects. *Journal of Food Engineering*, 61:83–89, 2004. 3.1, 3.2

[21] V. Leemans, H. Magein, and M.-F. Destain. Defects segmentation on 'golden delicious' apples by using colour machine vision. *Computers and Electronics in Agriculture*, 20(2):117–130, July 1998. 3.1, 3.3

[22] V. Leemans, H. Magein, and M.-F. Destain. Defect segmentation on 'jonagold' apples using colour vision and a bayesian classification method. *Computers and Electronics in Agriculture*, 23:43–53, April 1999. 3.1

[23] V. Leemans, H. Mageinb, and M.-F. Destain. On-line fruit grading according to their external quality using machine vision. *Biosystems Engineering*, 83(4):397–404, 2002. 3.1, 3.3

[24] Qingzhong Li, Maohua Wang, and Weikang Gu. Computer vision based system for apple surface defect detection. *Computers and Electronics in Agriculture*, 36:215–223, 2002. 3.1, 3.2

[25] M.I.A. Lourakis and A.A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. 6, 7.5.4

[26] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *International Joint Conference on Artificial Intelligence*, 1981. 7.5.1, 7.5.1.1, 7.5.1.1, 7.5.1.2

[27] W. Matusik, C. Buehler, and L. McMillan. Polyhedral Visual Hulls for Real-Time Rendering. *Rendering Techniques 2001: Proceedings of the Eurographics Workshop in London, United Kingdom, June 25-27, 2001*, 2001. 7.3

[28] The Commission of the European Communities. Commission regulation (ec) no. 85/2004 laying down the marketing standard for apples. Comission Regulation, 1 2004. 2.1

[29] I. Paulus, R. De Busscher, and E. Schrevens. Use of image analysis to investigate human quality classification of apples. *Journal of Agricultural Engineering Research*, 68:341–353, September 1997. 2.2

[30] DW Penman. Determination of stem and calyx location on apples using automatic visual inspection. *Computers and Electronics in Agriculture*, 33(1):7–18, 2001. 3.2

[31] M. Piccardi. Background subtraction techniques: a review. *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, 4, 2004. 5

[32] USDA's Agricultural Marketing Service. United states standards for grades of apples, December 2002. 2.1

[33] J. Shi and C. Tomasi. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994. 7.5.1.2, 7.5.1.3

[34] Olli Silvén, Matti Niskanen, and Hannu Kauppinen. Wood inspection with non-supervised clustering. *Machine Vision and Applications*, 13(5-6):275 – 285, 2003. 3.3

[35] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment–a modern synthesis. *Vision Algorithms: Theory and Practice*, 1883:298–372, 2000. 8.1

[36] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1998. 7.5.4

[37] B. Vijayakumar, D.J. Kriegman, and J. Ponce. Structure and motion of curved 3d objects from monocular silhouettes. *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, pages 327–334, 1996. 7.3

[38] Qingsheng Yang. Apple stem and calyx identification with machine vision. *Journal of Agricultural Engineering Research*, 63(3):229–236, March 1996. 3.2, 7

[39] Z. Zhang. A Flexible New Technique for Camera Calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 1330–1334, 2000. 7.5