UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Wendell Fioravante da Silva Diniz

Acionamento de Dispositivos Robóticos através de Interface Natural em Realidade Aumentada

Campinas, 2012

26/2012

Wendell Fioravante da Silva Diniz

Acionamento de Dispositivos Robóticos através de Interface Natural em Realidade Aumentada

Dissertação apresentada ao Curso de Mestrado da Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas, como requisito para a obtenção do título de Mestre em Engenharia Mecânica.

Área de Concentração: Mecânica dos Sólidos e Projeto Mecânico

Orientador: Prof. Dr. Eurípedes Guilherme de Oliveira Nóbrega

Campinas
2012

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

D615a

Diniz, Wendell Fioravante da Silva
Acionamento de dispositivos robóticos através de interface natural em realidade aumentada / Wendell Fioravante da Silva Diniz – Campinas, SP: [s.n.], 2012.

Orientador: Eurípedes Guilherme de Oliveira Nóbrega.
Dissertação de Mestrado – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Interfaces de usuário (Sistema de computador). 2. Realidade aumentada. 3. Robótica. I. Nóbrega, Eurípedes Guilherme de Oliveira. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

Título em Inglês:	Robotic device actuation through natural interface in augmented reality
Palavras-chave em Inglês:	User interfaces (Computer systems), Augmented reality, Robotics.
Área de concentração:	Mecânica dos Sólidos e Projeto Mecânico
Titulação:	Mestre em Engenharia Mecânica
Banca examinadora:	Cairo Lúcio Nascimento Junior, Janito Vaqueiro Ferreira.
Data da defesa:	29-02-2012
Programa de Pós Graduação:	Engenharia Mecânica

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE MECÂNICA COMPUTACIONAL

DISSERTAÇÃO DE MESTRADO

Acionamento de Dispositivos Robóticos através de Interface Natural em Realidade Aumentada

Autor: Wendell Fioravante da Silva Diniz
Orientador: Prof. Dr. Eurípedes Guilherme de Oliveira Nóbrega

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

Prof. Dr. Eurípedes Guilherme de Oliveira Nóbrega, Presidente
DMC/FEM/Unicamp

Prof. Dr. Cairo Lúcio Nascimento Junior
Departamento de Sistemas e Controle - Instituto Tecnológico de Aeronáutica

Prof. Dr. Janito Vaqueiro Ferreira
DMC/FEM/Unicamp

Campinas, 29 de fevereiro de 2012

Dedico este trabalho à minha família, que são sempre as pessoas com quem podemos contar em dificuldades. E as pessoas queridas que mesmo distantes, têm sempre um lugar especial em meu coração.

AGRADECIMENTOS

Este trabalho é fruto de um esforço que foi possível apenas com a colaboração de várias pessoas, às quais presto neste momento uma pequena homenagem.

Em primeiro lugar, ao professor, agora Doutor Alessandro Ferreira Alves, que ainda durante a graduação foi quem apresentou a possibilidade de cursar o mestrado na Unicamp e que com sua indicação, contribuiu para que eu conhecesse meu futuro orientador.

Em seguida, agradeço ao meu orientador, o Prof. Doutor Eurípedes Nóbrega, que abraçou a minha ideia e através de sua orientação, permitiu que este trabalho pudesse ser realizado. Também agradeço a todos os professores que, com seus cursos, contribuiram para a construção do conhecimento necessário para a conclusão deste trabalho.

Estre trabalho foi realizado ao longo de dois anos, o que permitiu que eu conhecesse muitas pessoas e, principalmente, fizesse novos amigos. Cada um deles contribuiu de alguma forma para o trabalho, destacando-se os meus "irmãos" de orientação, Danilo Pagano, Nilson Inocente, Pablo Souza, Helói Genari e Alexandre Tomazatti. Também os colegas de outros orientadores, como Giovani Bernardes, Josué Labaki, Alessandro Piveta e todos os outros que compartilham o laboratório, cuja lista completa seria demasiado extensa, mas que certamente serão lembrados. A convivência com todos contribuiu para que estes dois anos fossem bastante divertidos, afinal, não só de trabalho vive o homem.

Agradeço também à CAPES e ao povo brasileiro, que através dos recursos financeiros disponibilizados permitiu que eu pudesse dedicar-me integralmente à pesquisa.

Também merecem meu agradecimento todos os funcionários da Faculdade de Engenharia Mecânica e da Unicamp, que sempre atenderam com presteza todas as solicitações que precisei fazer.

A todos o meu obrigado.

"Sei que a minha missão e a de minhas irmãs representa uma gota no oceano. Mas sem essa gota, o oceano seria menor."

Madre Teresa de Calcutá

RESUMO

Desde o início da História e particularmente da Revolução Industrial, o homem tem buscado substituir ou complementar sua força de trabalho com dispositivos e máquinas capazes de ampliar a capacidade produtiva ou resolver tarefas para as quais o emprego de força humana não é satisfatório ou é perigoso. Esta necessidade impulsionou, no último século, o desenvolvimento de dispositivos robóticos, que foram se tornando cada vez mais complexos, à medida que a tecnologia avançava. No entanto, quanto mais complexos, mais difícil se torna sua operação. Reduzir a complexidade de operação destes dispositivos sem comprometer sua eficácia é um objetivo desejável e que tem impulsionado significativos esforços de pesquisa ultimamente. Neste trabalho, estudam-se duas tecnologias diferentes que podem ser aplicadas de forma complementar para tornar a operação de dispositivos robóticos mais simplificada. A área de Interfaces Naturais estuda a criação de mecanismos de operação baseados nas formas naturais de interação entre os seres humanos e os computadores, como gestos e fala, visando diminuir a curva de aprendizado para a operação de sistemas complexos, tornando esta uma tarefa intuitiva. A Realidade Aumentada tem entre seus objetivos ampliar o conhecimento sobre uma certa situação ao apresentar a um usuário informações adicionais fornecidas por instrumentação. Para isso, faz uso de técnicas de Computação Gráfica. Também permite criar situações simuladas para avaliar o comportamento de sistemas, ou treinamento para a operação destes sistemas de maneira virtual. Neste trabalho, montou-se um experimento em que um braço robótico será acionado de forma natural, através de gestos do usuário, em um ambiente de Realidade Aumentada. Como transdutor para a captura do movimento do usuário, foi o usado o sensor Kinect™ , que é um equipamento disponível comercialmente com custo relativamente baixo. Foi desenvolvida uma arquitetura que permite o acionamento da plataforma robótica de forma remota, através de comunicação em rede.

Palavras-Chave: Interfaces de usuário (Sistema de computador); Realidade aumentada; Robótica.

ABSTRACT

Since the beginning of History and particularly of the Industrial Revolution, human beings have sought to replace or supplement their workforce with devices and machines, in order to expand their productive capacity or to solve tasks for which the use of human power is not satisfactory or it is dangerous. This need pushed the development of robotic devices in the last century, which have become increasingly complex as technology advances. However, the more complex it is, more difficult its operation becomes. Reducing these devices operational complexity without compromising its effectiveness is a desirable goal that has driven significant research efforts lately. In this work, two different technologies are studied which can be applied in a complementary way to simplify robotic operation. The area of Natural Interfaces studies the creation of software mechanisms, based on natural forms of interaction between humans and computers, such as gestures and speech, in order to reduce the learning curve for the operation of complex systems, making it more intuitive. Augmented Reality has among its objectives to expand the knowledge about certain situations, presenting to the user additional instrumentation provided information. To accomplish this objective, Augmented Reality makes use of Computer Graphics techniques. It also allows the creation of simulated scenarios to evaluate the behavior of systems, or for virtually training the operators of these systems. In this work, an experiment was set up in which a robotic arm is activated through user gestures in an Augmented Reality environment. As a low cost commercially available equipment, the Kinect™ sensor is adopted to capture the user's movement. An architecture was developed and tested with auspicious results, implementing the proposed natural interface to interprete the human gestures and to provoke the respective remote activation of the robotic platform through network communication.

Keywords: User interfaces (Computer systems); Augmented reality; Robotics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

2D	duas dimensões ou bidimensional
3D	três dimensões ou tridimensional
AMS	American Mathematical Society
API	Application Programming Interface
ARPA	Advanced Research Projects Agency
ARPANET	ARPA Network
CAD	Computer Aided Design
DLL	Dynamic Link Library
DNS	Domain Name System
DoD	Department of Defense
e-mail	Electronic Mail
EDO	Equação Diferencial Ordinária
FPGA	Field-Programmable Gate Arrays
fps	Frames Per Second
FTP	File Transfer Protocol
GPU	Graphics Processing Unit
HTTP	Hyper-Text Transfer Protocol
IP	Internet Protocol
IR	Infra Red
ISO	International Standards Organization
MIT	Massachusetts Institute of Technology
NUI	Natural User Interface
OGRE	Object-oriented Graphics Rendering Engine
OSI	Open Systems Interconection Reference Model
POO	Programação Orientada a Objetos
PWM	Pulse Width Modulation
RTT	Round Trip Time
SCARA	Selectively Compliant Assembly Robot Arm
SDK	Software Development Kit
SMTP	Simple Mail Transfer Protocol

SQL	Search Query Language
TCP	Transfer Control Protocol
TELNET	Terminal Emulation Link Network
UDP	User Datagram Protocol
VANT	Veículo Aéreo Não Tripulado
WWW	World Wide Web

SUMÁRIO

1 INTRODUÇÃO

Desde o início da Revolução Industrial, o homem busca construir máquinas que substituam ou complementem a força de trabalho humano. Uma unidade industrial moderna certamente possuirá autômatos realizando variadas tarefas. A construção de tais sistemas robóticos, bem como as estratégias de operação destes equipamentos é uma tarefa desafiadora. Quanto maior a complexidade dos movimentos, maior a dificuldade de construir e operar esses equipamentos.

Os robôs podem auxiliar o homem a realizar tarefas difíceis, ampliando sua força, ou realizando tarefas repetitivas. Também podem executar tarefas em ambientes hostis ou contaminados, nos quais a exposição de um ser humano poderia acarretar riscos à sua saúde ou sua vida.

A operação destes equipamentos é tão mais difícil conforme a complexidade do sistema, o que pode exigir muitas horas de treinamento, até que o operador domine totalmente a interface utilizada e conheça as capacidades do equipamento.

As Interfaces Naturais de Usuário surgem como alternativa para acelerar o processo de aprendizagem de operação de sistemas complexos. Tais interfaces têm o objetivo de aproximar a operação de um dado equipamento à forma natural de interação humana. As interfaces controladas por gestos e os interpretadores de linguagem natural são exemplos da aplicação desta tecnologia. As interfaces por gestos têm recebido a atenção dos pesquisadores já há algum tempo, desde os trabalhos de Zimmerman et al. (1987), onde uma luva instrumentada especificamente projetada permitia reconhecimento de gestos, posição e orientação em tempo real e Fisher (1987), que faz uma análise da aplicação desta tecnologia em manipuladores robóticos. Inicialmente, o reconhecimento dos gestos do usuário requeria o uso de dispositivos específicos, como a luva citada anteriormente, que capturam o movimento das mãos, o que impede que a interação seja classificada como natural. Entretanto, com o avanço da tecnologia, novas técnicas baseadas em visão estão substituindo esses dispositivos por outros que não precisam estar em contato com o usuário. Destaca-se o sensor Kinect™ , desenvolvido para atuar como um mediador de interface natural para jogos eletrônicos. Sua aplicabilidade em outras áreas vem sendo sistematicamente estudada, como podemos notar no trabalho de Chang et al. (2011), que estudou a aplicação do sensor para auxiliar a reabilitação de

pessoas com disfunções motoras.

O uso de equipamentos reais para treinamento tem algumas consequências, tais como desgaste, consumo de combustível/energia, diminuição da disponibilidade e outras, o que seria desejável minimizar e/ou evitar. Uma forma de atingir este objetivo é através da simulação destes equipamentos, evitando o desgaste e o consumo de recursos por parte do equipamento que seria usado para se treinar. Por outro lado, nem sempre é possível cobrir todos os aspectos da utilização do equipamento apenas em simulação. Uma técnica que pode ser utilizada para tornar o treinamento simulado mais próximo do real é a Realidade Aumentada. A Realidade Aumentada é uma técnica que combina elementos virtuais e reais em uma única cena, que é percebida pelo usuário como uma única realidade (AZUMA, 1997). As pesquisas em Realidade Aumentada têm crescido nos últimos anos, em decorrência da evolução dos computadores pessoais, que atingiram uma capacidade de processamento grande o suficiente para permitir a aplicação de técnicas que antes só eram possíveis por detentores de grandes e caros sistemas computacionais. A Realidade Aumentada tem a capacidade de apresentar o ambiente percebido pelo usuário ao mesmo tempo que acrescenta novas informações, registradas através de instrumentação, que estão além da capacidade de percepção dos sentidos humanos. Estas informações ampliam o conhecimento do usuário sobre a situação e contribuem para melhorar a tomada de decisões. É possível aplicar as vantagens da Realidade Aumentada em várias áreas, tais como a manutenção de sistemas complexos (HENDERSON; FEINER, 2009) ou teleoperação de sistemas robóticos (PORTILLA; BASANEZ, 2007).

A combinação de Interfaces Naturais e Realidade Aumentada é uma ferramenta com enorme potencial, podendo auxiliar a reduzir tempo de treinamento e gerar economia de recursos. Acrescentando a capacidade de operação remota, é possívelo compartilhamento de recursos por usuários geograficamente distantes, o que amplia ainda mais o alcance das tecnologias envolvidas.

1.1 Proposta do Trabalho

1.1.1 Objetivos

O objetivo deste trabalho será construir um sistema composto por associação de *hardware* e *software* que permita acionar um dispositivo robótico em um ambiente de Realidade Aumentada através da captura dos movimentos do usuário. Como sistema de percepção do usuário, pretende-se utilizar o sensor Kinect™ , que é um equipamento disponível comercialmente com custo relativamente baixo. O braço robótico utilizado será o conjunto Rascal da Robix™ , com o objetivo de demonstrar a aplicabilidade da Interface de Usuário Natural em sistemas robóticos.

O sistema deverá ser construído de forma modular, visando o melhor aproveitamento dos recursos computacionais e flexibilidade para a reconfiguração para utilização em aplicações similares. Também será implementado o acionamento remoto do braço robótico, permitindo avaliar a eficiência deste método e a capacidade de se adaptar outras configurações de dispositivos.

1.1.2 Organização da Dissertação

A presente dissertação está organizada da seguinte forma: no Capítulo 1 serão introduzidos o panorama e as motivações que levaram ao desenvolvimento deste trabalho. No Capítulo 2 será feita uma revisão do conhecimento necessário e das técnicas utilizadas durante o desenvolvimento das propostas. No Capítulo 3 serão apresentados os dispositivos utilizados para atingir os objetivos propostos assim como a metodologia empregada. No Capítulo 4 serão apresentados e discutidos os resultados obtidos. Finalmente, no Capítulo 5 serão apresentadas as conclusões atingidas e as sugestões e recomendações para trabalhos futuros.

2 ESTADO DA ARTE

2.1 Computação Gráfica

Computação Gráfica é a área da Computação dedicada ao estudo dos métodos para sintetizar e manipular digitalmente conteúdo visual. Os estudos nessa área tiveram início quase que simultaneamente ao surgimento dos computadores, mas foi com o trabalho de um aluno do *Massachusetts Institute of Technology* (MIT) que a área começou a avançar (SUTHERLAND, 1964). Neste trabalho, foi desenvolvido o *Sketch Pad*, uma aplicação de desenho que utilizava uma caneta luminosa para traçar as linhas em uma tela. Muitas das soluções desenvolvidas por Sutherland (1964) foram tomadas como ponto de partida para trabalhos subsequentes, formando as bases da Computação Gráfica.

O termo Computação Gráfica é muitas vezes tomado como o estudo de gráficos em três dimensões, mas na verdade ele também contempla gráficos em duas dimensões e o processamento de imagens. A Computação Gráfica pode ser dividida em diversas subáreas conforme o objeto específico de estudo. Por exemplo a Geometria Computacional é o estudo da representação de dados geométricos de forma computacional e as operações que podem ser realizadas sobre estes dados, gerando modelos computacionais que descrevem a forma dos objetos. A Renderização trata das técnicas e algoritmos para a geração de imagens através dos modelos geométricos. A Animação estuda métodos para representar e aplicar movimentos e deformações nos modelos geométricos através do tempo. Todas as técnicas têm sido aplicadas com sucesso em diversas áreas, permitindo a visualização de fenômenos científicos diversos, contribuindo para o estudo de simulações destes fenômenos de forma visual. Diversos pacotes e bibliotecas gráficas estão disponíveis, permitindo a construção de elaboradas aplicações nas mais diversas áreas, desde entretenimento e jogos até Projeto Assistido por Computador (CAD, *Computer Aided Design*) e Visualização de Dados Científicos.

2.1.1 Transformações Geométricas

As transformações geométricas são um conjunto de operações que manipulam pontos em um espaço geométrico. As transformações de escala, rotação e translação são operações essenciais a qualquer aplicação gráfica, tanto em 2D (duas dimensões) quanto em 3D (três dimensões).

As Transformações Geométricas são operações essenciais a qualquer aplicação de Computação Gráfica. Elas são fornecidas em praticamente todos os pacotes gráficos e tem literatura extensa. Uma revisão destas operações foi incluída no Apêndice A.

2.1.2 Visualização Tridimensional

A visualização tridimensional é o conjunto de técnicas que permitem transpor uma cena tridimensional para um espaço de visualização. A complexidade do método reside no fato de que os dispositivos de visualização são bidimensionais, tais como a tela de um monitor onde a cena será exibida ou uma folha de papel onde a cena será impressa. Para cumprir esse objetivo, são usadas, predominantemente, as técnicas de projeção. As projeções são transformações que levam os objetos em um espaço n-dimensional para outro com dimensão menor que n. Em Computação Gráfica, geralmente, restringe-se o estudo às projeções que levam de um espaço 3D para 2D, embora a Computação Gráfica tenha sido aplicada com sucesso em casos de visualização de espaços n-dimensionais através da projeção para 2D (NOLL, 1967).

As imagens bidimensionais criadas a partir de projeções permitem preservar um certo grau de percepção de profundidade da cena original. A técnica consiste em transpor os raios de luz que partem de toda a extensão dos objetos da cena e convergem para um ponto. Os raios são chamados de projetores e o ponto onde se encontram é chamado de centro da projeção. A imagem será formada pelos pontos de um plano localizado a uma certa distância do centro de projeção que são interceptados pelos projetores. As técnicas de projeção já eram aplicadas desde a Idade Média,

mas foi durante o Renascimento que ela passou a ser usada com frequência, por artistas como Leonardo da Vinci. A Figura 2.1 mostra uma litogravura datada de 1525, onde um artista usa uma antiga máquina de perspectiva, na qual esses conceitos são aplicados. A Figura 2.2 mostra como Leonardo da Vinci utiliza as projeções para dar a noção de profundidade em uma cena no interior de um edifício.

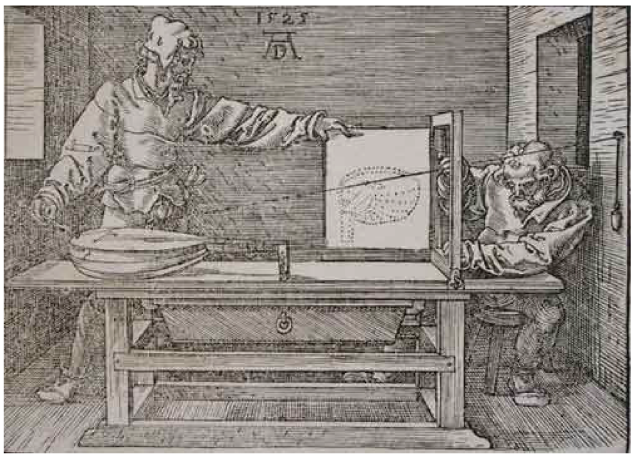


Figura 2.1 - An Artist Using a Perspective Machine, 1525, Albrecht Dürer (1471-1528). A máquina utiliza o conceito da projeção para permitir a transposição de uma cena 3D para uma superfície 2D.
Fonte: Victoria and Albert Museum, Londres, Inglaterra

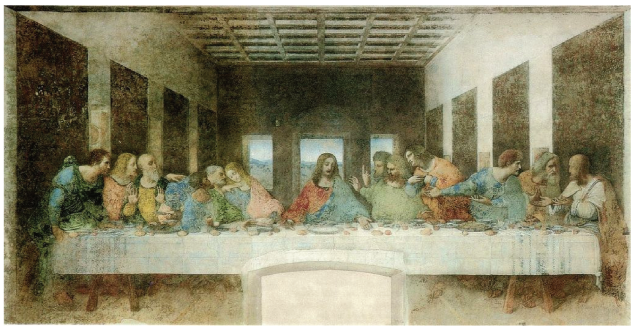


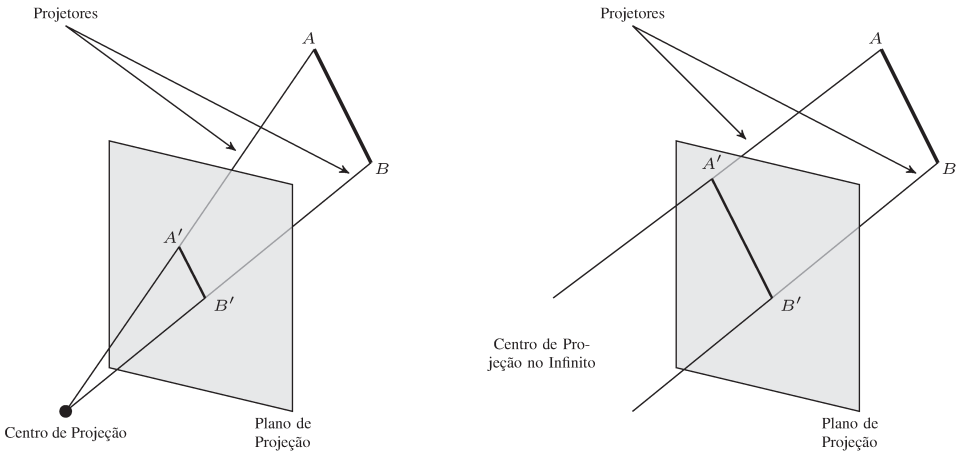
Figura 2.2 - A Última Ceia (*L'Ultima Cena* ou *Il Cenacolo*), 1495-1497, Leonardo da Vinci. Repare como as paredes estão projetadas, permitindo a percepção da profundidade do ambiente.
Fonte: Wikimedia Commons

Para realizar a Visualização Tridimensional, especifica-se um volume de visualização, que é a região do mundo que será visualizada; define-se o tipo da projeção e o plano de projeção e finalmente, um quadro ou janela de visualização (*viewport*) na superfície de visualização. Os objetos da cena são recortados pelo volume de visualização e então projetados sobre a superfície de visualização. O conteúdo é mapeado para a janela de visualização. O produto final são coordenadas

2D do dispositivo de exibição utilizado.

Tipos de Projeção

As projeções utilizadas em Computação Gráfica são conhecidas como Projeções Geométricas Planares, porque utilizam projeções retilíneas em um plano. Algumas projeções cartográficas são não-planares ou não-geométricas. As Projeções Geométricas Planares são dividas em duas classes básicas: Perspectivas e Paralelas. A distinção é feita através da distância do centro de projeção em relação ao plano de projeção. Na projeção paralela, o centro de projeção está localizado a uma distância infinita do plano de projeção. A Figura 2.3 ilustra esses casos.



Projeção em Perspectiva Projeção Paralela

Figura 2.3 - Tipos de Projeções Geométricas Planares.
Fonte: Adaptado de (FOLEY et al., 1996)

Quando definimos uma projeção em perspectiva, é necessário especificar as coordenadas do centro de projeção. Para uma projeção paralela, nós especificamos uma direção de projeção. Se pensarmos no centro de projeção como um ponto, definido em coordenadas homogêneas, ele será definido na forma $(x, y, z, 1)$. Uma direção é um vetor, ou seja, pode ser representado por uma diferença entre dois pontos, $d = (x, y, z, 1) - (x', y', z', 1) = (a, b, c, 0)$. Ou seja, direções são pontos localizados no infinito. Desta forma, podemos pensar na projeção paralela como um caso

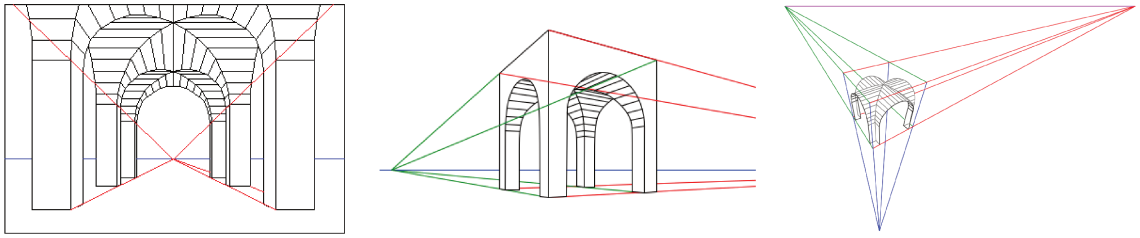
especial da projeção em perspectiva, onde a distância do centro de projeção ao plano de projeção é infinita.

A projeção em perspectiva produz resultados mais realistas, pois é similar em funcionamento a sistemas fotográficos e ao próprio olho humano. Objetos idênticos mais distantes aparecem menores que objetos mais próximos. Entretanto, ela não é útil para extrair a forma ou medidas dos objetos, pois os ângulos não são preservados, exceto nas faces que são paralelas ao plano de projeção, mas linhas paralelas mantém o paralelismo. Já a projeção paralela é menos realista, mas pode ser usada para medições exatas. Ela também mantêm o paralelismo das retas, mas os ângulos das faces que não são paralelas ao plano de projeção são alterados tal qual na projeção em perspectiva.

Projeção em Perspectiva

Na Projeção em Perspectiva, um conjunto de linhas paralelas, que não são paralelas ao plano de projeção, encontram-se em um ponto. Este ponto é chamado de ponto de fuga. Como as linhas paralelas convergem no infinito, um ponto de fuga pode ser pensado como a projeção de um ponto no infinito. Logicamente, existem infinitos pontos de fuga, um para cada uma das infinitas direções em que se pode orientar uma linha.

Cada um dos eixos da projeção possui um ponto de fuga correspondente. Todas as linhas paralelas a um dado eixo, convergem para este ponto. Podem existir até três desses pontos, de acordo com o número de eixos que são interceptados pelo plano de projeção. O número de eixos que o plano corta, serve para classificar uma dada projeção. A Figura 2.4 mostra as três classificações.



(a) (b) (c)

Figura 2.4 - Classes de Projeção em Perspectiva. (a) Um ponto de fuga. (b) Dois pontos de fuga. (c) Três pontos de fuga.
Fonte: Wikimedia Commons

Projeção Paralela

Existem duas classes de projeções paralelas, que se diferenciam quanto à direção da projeção. Na projeção paralela ortográfica, a direção da projeção é normal ao plano de projeção. Na projeção paralela oblíqua, a direção da projeção não é normal ao plano de projeção.

Os tipos mais comuns de projeção ortográfica utilizados são aqueles em que a direção da projeção é alinhada a um eixo principal. Elas comumente são chamadas de visão frontal, visão superior ou plano e visão lateral. A Figura 2.5 mostra esses três tipos de projeção. Elas são bastante usadas em desenhos técnicos para visualizar peças e edifícios, porque os ângulos e medidas podem ser extraídos diretamente. Entretanto, como mostram apenas uma face do objeto, não é possível distinguir a natureza tridimensional do objeto projetado.

Quando uma projeção ortográfica utiliza um plano de projeção que não é perpendicular a um eixo principal, a projeção resultante recebe o nome de axonométrica. Ela se parece com a projeção em perspectiva, mas difere quanto ao encurtamento da projeção, que é uniforme e não dependente da distância ao centro de projeção. O paralelismo das linhas é preservado, mas os ângulos do objeto não são. Medidas podem ser feitas ao longo do eixo principal. Quando os ângulos entre a normal ao plano de projeção com cada um dos eixos é igual, a projeção resultante é chamada isométrica. É o tipo mais usado de projeção axonométrica.

A projeção oblíqua utiliza planos de projeção cuja normal não é igual à direção da projeção.

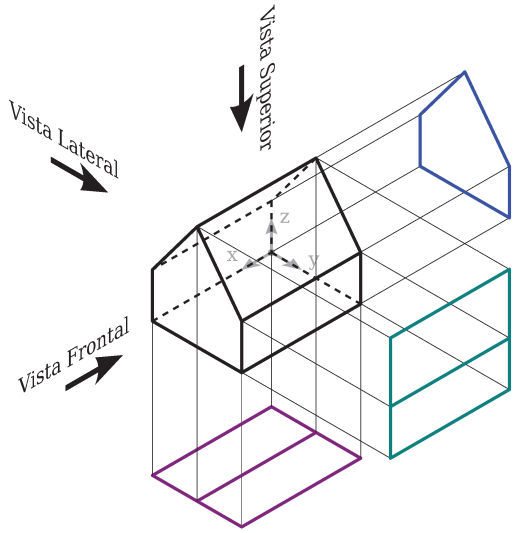


Figura 2.5 - Projeção ortográfica. Cada um dos planos corresponde às vistas frontal, lateral e superior.
Fonte: Wikimedia Commons

Ela tem a característica de combinar propriedades das projeções ortográficas axonométricas com as das visões superior, frontal e lateral. Permite a medida de distâncias ao longo de um eixo principal. Preserva também os ângulos de faces paralelas ao plano de projeção.

Os dois tipos de projeção oblíqua mais utilizados são a cavaleira (*cavalier*) e a gabinete (*cabinet*). Na projeção cavaleira, a direção da projeção forma um ângulo de $45°$ com o plano de projeção. Desta forma, a projeção de uma linha perpendicular ao plano de projeção tem o mesmo comprimento da linha original, ou seja, não há encurtamento. Na projeção gabinete, o ângulo entre a direção de projeção e o plano de projeção é de $\arctan(2) = 63{,}4°$. Isto faz com que as linhas perpendiculares sejam projetadas com metade de seu comprimento original. A projeção de gabinete é um pouco mais realista que a projeção cavaleira, uma vez que o encurtamento que produz é consistente com a visão humana.

Os diferentes tipos de projeção foram discutidos e ilustrados no trabalho de Carlbom e Paciorek (1978), onde são apresentados maiores detalhes.

2.2 Visão Computacional

Visão Computacional nomeia o conjunto de técnicas e métodos pelos quais é possível interpretar, extrair informações e tomar decisões sobre objetos físicos baseadas nas imagens destes objetos. Ou seja, é a ciência e a tecnologia que permite que máquinas vejam. A Visão Computacional evoluiu a partir dos estudos de Processamento de Imagens, sobretudo a partir de técnicas de reconhecimento de padrões combinadas a técnicas de Inteligência Artificial. É uma ciência multidisciplinar, que combina técnicas de várias áreas como meios para a construção de modelos semânticos da imagem. A Visão Computacional é uma área intensamente pesquisada, com aplicações em vários campos. Algumas aplicações incluem inspeção de processos industriais, reconhecimento de faces para identificação de indivíduos, exploração espacial.

2.2.1 Percepção Visual 3D

A percepção de uma cena tridimensional depende sobretudo de processos geométricos, como os discutidos na Seção 2.1. Conhecendo-se alguns parâmetros, é possível aferir distâncias e definir as coordenadas de objetos previamente reconhecidos. Em Visão Computacional, costuma-se considerar que a imagem foi produzida por uma câmera. Existem vários modelos propostos de câmeras, desde as mais simples, a câmera "buraco de agulha" (*pinhole*) até modelos mais realistas, que levam a ótica de lentes em consideração, como o modelo proposto por Kolb et al. (1995).

Câmera *Pinhole*

O modelo de câmera *pinhole* ideal considera a imagem formada pelos raios de luz que partem da cena e atravessam um orifício pontual e são projetados em um plano situado atrás do orifício. O tipo de projeção obtido é uma projeção em perspectiva. A Figura 2.6 mostra o diagrama desse

modelo.

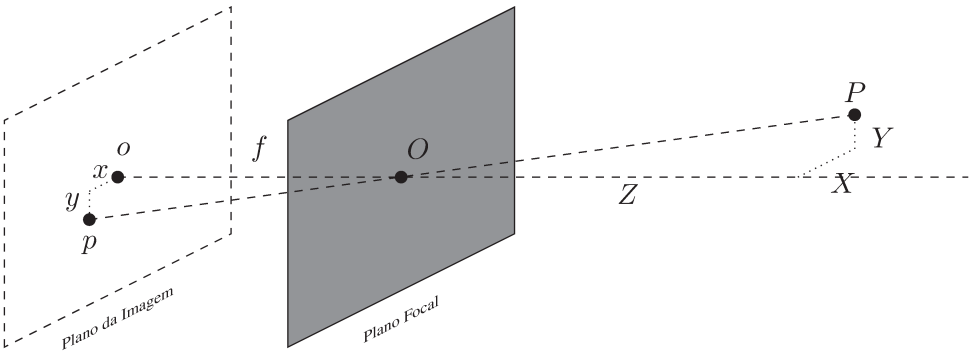


Figura 2.6 - Câmera buraco de agulha.
Fonte: Adaptado de (SHAPIRO; STOCKMAN, 2001)

O ponto bidimensional $p(x, y)$ na imagem corresponde à projeção do ponto tridimensional $P(X, Y, Z)$ no mundo. O ponto O é o ponto focal, correspondente ao centro de projeção. Neste ponto, está localizada a origem do sistema de coordenadas da câmera. O eixo que aponta em direção à cena a partir do ponto focal é o eixo principal, ou eixo óptico. Ele intercepta o plano da imagem no ponto o, que é chamado ponto principal. A distância entre o plano da imagem e o ponto focal, representada por f é a distância focal.

A relação dos pontos $p(x, y)$ e $P(X, Y, Z)$ é dada pela equação:

$$x = \frac{Xf}{Z}, \quad y = \frac{Yf}{Z} \tag{2.1}$$

Podemos evitar a divisão através de uma representação matricial em coordenadas homogêneas, dessa forma:

$$p \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

Câmera de Lentes Finas

O modelo de lentes finas utiliza uma lente para projetar a imagem da cena capturada no plano da imagem. Uma lente é considerada fina quando o diâmetro da lente é desprezível em relação à distância focal da mesma. A Figura 2.7 mostra esse modelo.

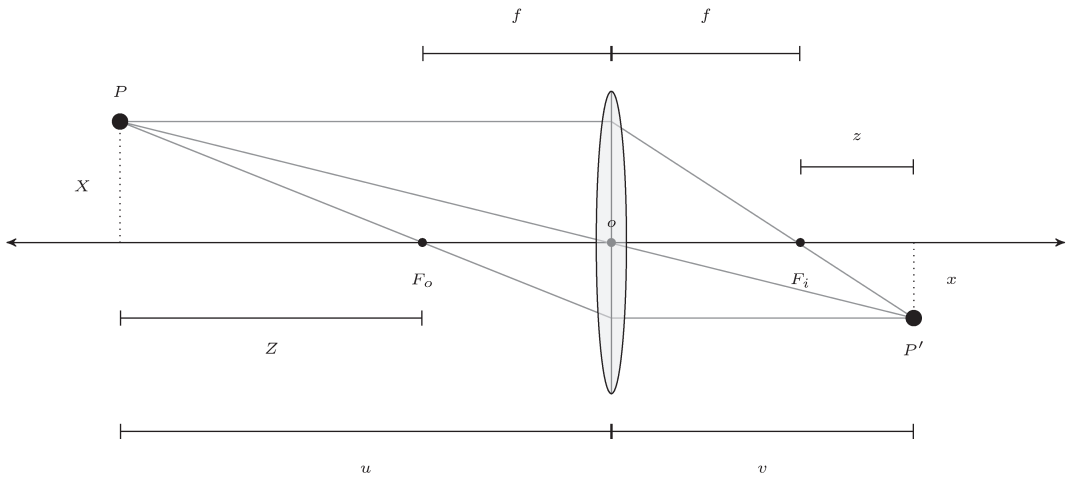


Figura 2.7 - Câmera de lentes finas.
Fonte: Adaptado de (SHAPIRO; STOCKMAN, 2001)

Um raio que entra pela lente paralelamente ao eixo óptico, sai passando pelo ponto focal do lado oposto. Um raio que passa pelo centro óptico O da lente não sofre refração. Finalmente, um raio que passe por um ponto focal emerge paralelo ao eixo óptico no lado oposto. A partir destas proposições, utilizando semelhança de triângulos, é possível extrair as relações:

$$\frac{X}{f} = \frac{x}{z} \tag{2.3}$$

$$\frac{X}{f+Z} = \frac{x}{f+z} \tag{2.4}$$

Substituindo então o valor de X na Equação 2.3 na Equação 2.4, teremos:

$$f^2 = Zz \tag{2.5}$$

Sendo que $Z = u - f$ e $z = v - f$, efetuando outra substituição, teremos:

$$uv = f(u + v) \tag{2.6}$$

Dividindo esta última equação por uvf, teremos a forma mais comum da equação das lentes finas:

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v} \tag{2.7}$$

Esta equação relaciona a distância u do objeto ao centro da lente com a distância v da imagem projetada ao centro da lente.

Calibração de Câmeras

É possível estimar as coordenadas 3D de uma cena a partir de um conjunto de equações que as relacionam com as coordenadas 2D da imagem obtida, para uma câmera particular. Esse processo é chamado de Calibração. A calibração de câmeras tem sido objeto de muitos trabalhos; dentre estes, destacam-se os métodos propostos por Tsai (1987) e Zhang (2000).

Parâmetros Intrínsecos e Parâmetros Extrínsecos de uma Câmera

Uma câmera pode ser caracterizada por dois conjuntos de parâmetros. Os Parâmetros Intrínsecos de uma câmera são aqueles que se relacionam diretamente com as características ópticas e geométricas da mesma. Variam de acordo com a câmera utilizada.

No método proposto por Tsai (1987), são considerados os seguintes parâmetros intrínsecos:

- Ponto Principal $[u_0, v_0]$: a intersecção do eixo óptico com o plano da imagem;
- Fatores de escala $[d_x, d_y]$: relativos ao tamanho do pixel nos eixos x e y;
- Fator de distorção de aspecto $[\tau_1]$: fator para corrigir o aspecto do pixel no sensor. Se o pixel for quadrado, este fator é de 1;
- Distância focal f: a distância do centro óptico ao plano da imagem;
- Fator de distorção da lente κ_1: modela a distorção radial da lente.

O ponto de referência da câmera é o seu centro óptico de sua lente. A origem do sistema de coordenadas da câmera está localizada neste ponto. O eixo óptico é perpendicular ao plano da imagem passando pelo centro óptico. O ponto principal geralmente está localizado no centro da imagem, mas pode haver algum deslocamento, devido à montagem do conjunto óptico em seu suporte. Os fatores de escala d_x e d_y representam as dimensões do pixel no sensor, nas direções horizontal e vertical. Seus valores são dados em uma unidade do mundo real, como por exemplo, em milímetros.

Os Parâmetros Extrínsecos guardam a posição e a orientação da câmera em relação ao sistema de coordenadas do mundo. São uma transformação de corpo rígido, ou seja, uma rotação seguida de uma translação.

- Translação:

$$t = [t_x \; t_y \; t_z]^T \tag{2.8}$$

- Rotação:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

Tsai propõe duas abordagens. A abordagem mais comum utiliza como padrão referência um conjunto de pontos coplanares cujas distâncias de centro a centro são conhecidas. São tomadas várias imagens do padrão em distâncias diferentes. A câmera não deve ser movida ou focada durante a tomada das imagens. Em seguida, calculam-se as coordenadas dos centros dos pontos de referência no espaço da imagem. Obtém-se dessa forma um conjunto de correspondências entre coordenadas 3D do mundo e coordenadas 2D da imagem. A partir de relações entre os parâmetros, é feito um

processo de otimização por mínimos quadrados para calcular os parâmetros desconhecidos.

Estereopsia

As imagens formadas nos olhos são bidimensionais. Mesmo assim, somos capazes de perceber a profundidade das cenas. Esse mecanismo é importante, pois permite avaliar corretamente distâncias. Podemos, por exemplo, pegar objetos e desviar de obstáculos facilmente. Mesmo possuindo dois olhos, percebemos o mundo como parte de uma única imagem. No cérebro, as imagens provenientes dos dois olhos são fundidas em uma única imagem. Através da mudança de posição dos objetos percebidos por cada olho, o cérebro é capaz de recuperar a informação de distância dos objetos percebidos e enxergar em três dimensões. Esse processo recebe o nome de estereopsia.

Percepção de Profundidade por Visão Estéreo

A forma mais comum de estereopsia artificial é a utilização de um par de câmeras estéreo. Através de uma aplicação simples de geometria, é possível recuperar a distância de um ponto comparando a projeção deste ponto em duas imagens geradas por um par de câmeras estéreo, como mostrado na Figura 2.8.

Assume-se que as câmeras estão montadas de um modo que seus planos de imagem são colineares no eixo X e paralelos nos eixos Y e Z. Os centros de projeção O_d e O_e das câmeras direita e esquerda estão afastados por uma distância conhecida b, chamada linha de base do par estéreo. Um ponto P irá projetar um ponto P_d na câmera direita e um ponto P_e na câmera esquerda. Geometricamente, deduz-se que o ponto P está na intersecção do prolongamento dos segmentos $\overline{O_d P_d}$ e $\overline{O_e P_e}$. Por semelhança de triângulos, podemos obter:

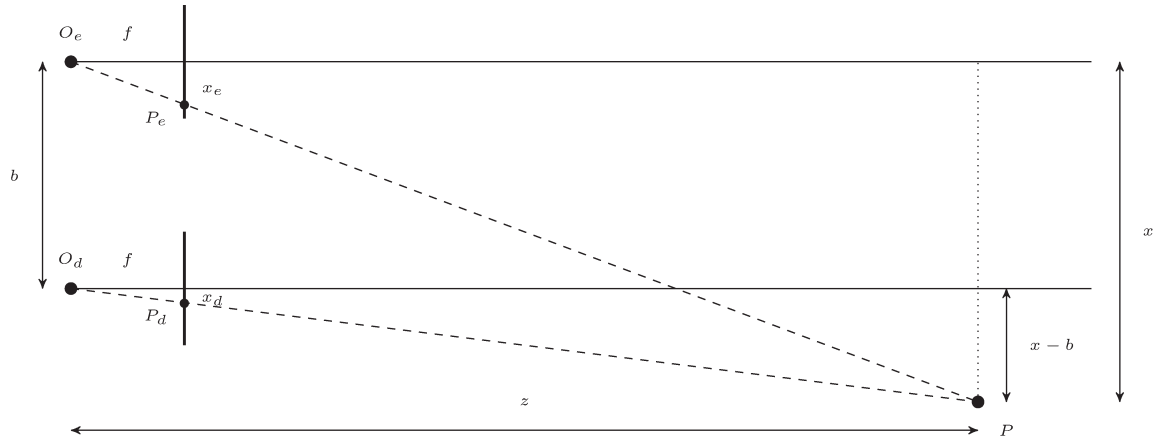


Figura 2.8 - Modelo geométrico de um par de câmeras estéreo.
Fonte: Adaptado de (SHAPIRO; STOCKMAN, 2001)

$$\begin{aligned} z/f &= x/x_e \\ z/f &= (x-b)/x_d \\ z/f &= y/y_e = y/y_d \end{aligned} \tag{2.10}$$

Como as câmeras estão alinhadas, podemos assumir que as coordenadas y_d e y_e são idênticas. O sistema de coordenadas para esta dedução foi tomado com origem no centro óptico da câmera esquerda. Após algumas substituições, podemos obter os valores das coordenadas x e z do ponto P:

$$\begin{aligned} z &= fb/(x_e - x_d) = fb/d \\ x &= x_e z/f = b + x_d z/f \\ y &= y_e z/f = y_d z/f \end{aligned} \tag{2.11}$$

Na Equação 2.11 introduzimos o termo d, a diferença entre as coordenadas x_e e x_d. Este termo é chamado de disparidade. Lembramos que não há disparidade entre as coordenadas y. É possível observar que a disparidade é inversamente proporcional à distância, aumentando para pontos mais próximos e tendendo a zero quando o ponto tende ao infinito.

A determinação da profundidade é bem simples. A maior carga de trabalho da técnica de

visão estéreo está na determinação dos pontos correspondentes nas duas imagens. Ou seja, para cada pixel na imagem esquerda, é preciso encontrar o pixel correspondente na imagem direita. Várias técnicas foram propostas para resolver este problema, uma das mais usadas é o operador de correlação cruzada (SHAPIRO; STOCKMAN, 2001).

O método de correlação cruzada trabalha da seguinte forma: toma-se um ponto P_1 da imagem de referência do par estéreo. Determina-se uma região da segunda imagem onde o ponto P_2, correspondente a P_1 deve ser encontrado. O tamanho da região de busca é determinado em função de características do par estéreo utilizado. Essas informações podem ser conseguidas através dos parâmetros da câmera ou por estimação através de treinamento com imagens conhecidas.

O resultado é o mapa de profundidade, um tipo especial de imagem onde os pixels guardam o valor da distância dos pontos da imagem ao sensor. As Figuras 2.9 e 2.10 mostram, respectivamente, um par estéreo e o mapa de profundidade esperado.

Figura 2.9 - Cones, exemplo de par de imagens estéreo.
Fonte: (MIDDLEBURY. . . , 2003)

Percepção de Profundidade por Luz Estruturada

Outra técnica usada para recuperação da informação de profundidade da cena é o uso de luz estruturada. Esta técnica consiste em projetar um padrão luminoso sobre a superfície a ser medida. Geralmente, mas nem sempre, esse padrão consiste de uma grade de linhas horizontais

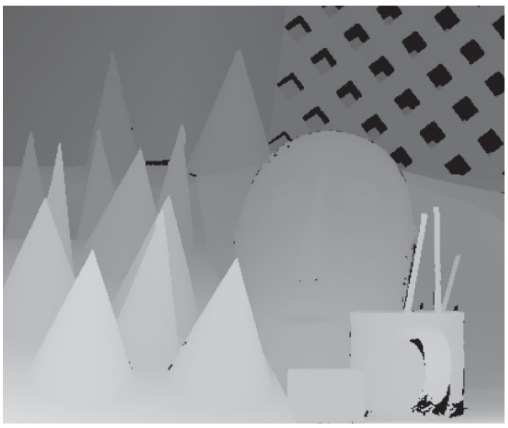


Figura 2.10 - Mapa de profundidade esperado da imagem Cones. Os pixels mais claros estão mais próximos do sensor.
Fonte: (MIDDLEBURY..., 2003)

e verticais perpendiculares. A distorção causada no padrão pode ser medida, através das matrizes de calibração da câmera e do projetor. As equações são as mesmas usadas para pares de câmeras estéreo generalizadas (HU; STOCKMAN, 1989; SHRIKHANDE; STOCKMAN, 1989).

Em suma, resolve-se o seguinte sistema:

$$\begin{aligned} D.{}^{W}P_{ij} &= {}^{G}P_{ij} \\ C.{}^{W}P_{ij} &= {}^{I}P_{uv} \end{aligned} \tag{2.12}$$

onde D e C são, respectivamente, as matrizes de calibração do projetor e da câmera, ${}^{W}P_{ij}$ é o ponto 3D onde uma determinada linha i e coluna j do padrão é projetado. ${}^{G}P_{ij}$ é o ponto correspondente do padrão no espaço do projetor, e ${}^{I}P_{uv}$ é o ponto correspondente na imagem tomada pela câmera associada ao projetor. A determinação de profundidade por luz estruturada vêm sendo estudada e aplicada em situações práticas como inspeção industrial e engenharia reversa.

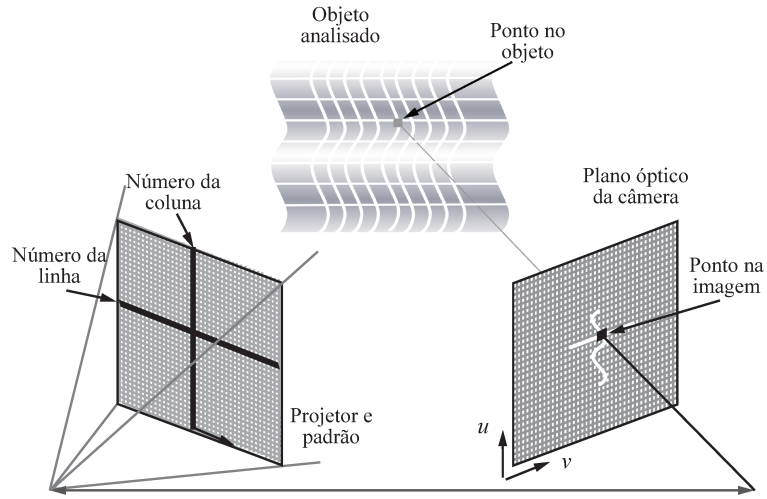


Figura 2.11 - Recuperação de informação 3D a partir de luz estruturada.
Fonte: Wikimedia Commons

2.3 Realidade Virtual e Realidade Aumentada

O conceito de Realidade Aumentada foi definido por Azuma (1997) como uma variação da técnica de Ambientes Virtuais, ou Realidade Virtual, como mais conhecida, onde elementos virtuais e elementos reais são combinados e percebidos pelo usuário como coexistentes no mesmo espaço. Na Realidade Virtual, o usuário é totalmente imerso em um mundo virtual. Enquanto imerso, o usuário não é capaz de ver o mundo real. A Realidade Aumentada, por sua vez, permite o usuário continuar a perceber o mundo real, com objetos virtuais superpostos a elementos reais. Seu objetivo é suplementar o mundo real com novas informações, em vez de substituí-lo completamente. Milgram e Kishino (1994) propuseram o "*Continuum* da Virtualidade", que posiciona os diversos tipos de realidade em uma linha, como mostrado na Figura 2.12. A Realidade Aumentada ocupa uma posição intermediária entre os Ambientes Virtuais (completamente sintéticos) e a Telepresença (completamente reais).

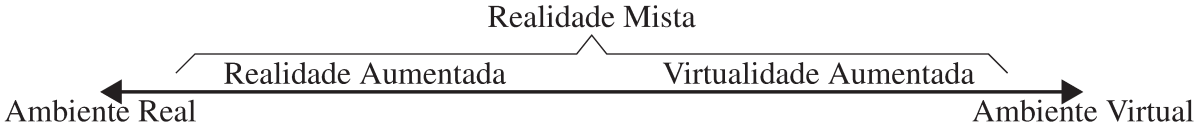


Figura 2.12 - O *Continuum* da Virtualidade.
Fonte: Adaptado de (MILGRAM; KISHINO, 1994)

Idealmente, o usuário deve perceber os objetos virtuais e reais coexistindo no mesmo espaço, como mostrado na Figura 2.13. Em uma única sentença, a Realidade Aumentada pode ser definida como a combinação de elementos reais e elementos virtuais em um único espaço, interativamente, em tempo real e em três dimensões.

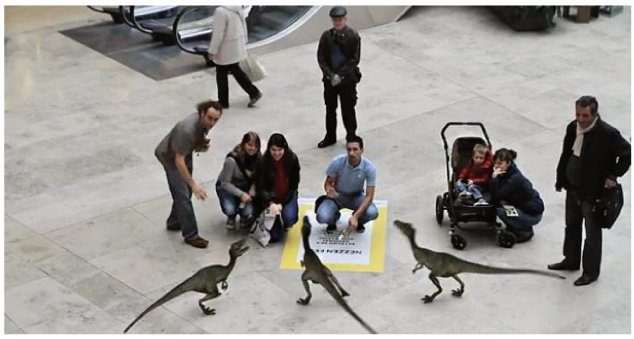


Figura 2.13 - Atividade de Realidade Aumentada promovida pelo canal de TV *National Geographic*. Os espectadores puderam interagir com uma série de personagens virtuais.
Fonte: (KIEFABER, 2011)

A realidade aumentada pode ser obtida através de um amplo conjunto de sistemas. Esses sistemas precisam preencher três requisitos:

- Combinar objetos reais e objetos virtuais;
- Permitir interatividade em tempo real;
- Registrar os objetos em 3D.

Ao se colocar esses requisitos, a quantidade de sistemas de Realidade Aumentada torna-se bastante ampla, englobando desde soluções montadas diretamente a frente dos olhos do usuário a soluções em que uma câmera capta o usuário e mostra sua imagem em um monitor juntamente com os objetos virtuais.

As capacidades de melhorar a percepção do usuário, prover novos meios de interação com o ambiente e proporcionar informações que não seriam percebidas apenas com os sentidos naturais do usuário, permitem aplicar a técnica de Realidade Aumentada para auxiliar a resolução de problemas que, de outra forma, seriam difíceis ou até mesmo impossíveis de realizar. Esta capacidade de utilizar a computação como ferramenta para tornar a realização de uma tarefa mais facilitada, foi definida por Brooks (1996) como Ampliação de Inteligência. A Realidade Aumentada é uma téc-

nica que se encaixa perfeitamente nesta descrição. Esses motivos justificam e endossam o interesse no estudo do tópico e a descoberta de novas aplicações.

O interesse no estudo das técnicas de Realidade Aumentada aumentou particularmente a partir dos anos noventa. O motivo é que as técnicas e algoritmos necessários têm custo computacional particularmente alto. Apesar destas técnicas serem conhecidas desde os anos sessenta, as pesquisas ficavam restritas aos grandes centros de pesquisa, dados os elevados requisitos computacionais então exigidos. A partir da segunda metade dos anos noventa, os sistemas computacionais acessíveis pela maioria das pessoas começaram a possuir o poder computacional equivalente àquele dos antigos supercomputadores. Pesquisadores independentes e centros de pesquisa menores puderam ter acesso a esses equipamentos e consequentemente, condições de realizar pesquisas nesta área. Esse cenário contribuiu imensamente para o aumento do volume de trabalhos.

A aplicabilidade da Realidade Aumentada estende-se a várias áreas. Podemos citar várias aplicações interessantes onde foram feitos estudos significativos. As áreas de visualização médica, manutenção e reparação, entretenimento e robótica têm explorado com sucesso a técnica.

2.4 Simulação de Física em Aplicações de Computação Gráfica

A Simulação de Física em aplicações de Computação Gráfica tem recebido bastante atenção. Sobretudo, a indústria do entretenimento tem impulsionado o desenvolvimento desse tipo de simulação, buscando jogos eletrônicos e efeitos visuais cada vez mais realistas. Na academia, as simulações também desempenham papel importante, uma vez que permitem, entre outras aplicações, a visualização de fenômenos inacessíveis aos sentidos humanos, como a dinâmica de moléculas ou o movimento de galáxias. Uma simulação de fenômenos físicos precisa e com tempo de processamento aceitável é desejável.

Os conjuntos de rotinas e bibliotecas que permitem implementar a simulação de física em uma aplicação são conhecidas como *motores de física* (do inglês *physics engine*. Existem muitos motores de física disponíveis ao desenvolvedor. A escolha de um motor de física deve levar vários

aspectos em consideração, como seu desempenho, os problemas que conseguem resolver, a forma como modelam os objetos e outras. Boeing e Bräunl (2007) realizaram um estudo comparativo de vários motores de física comerciais e de código aberto, mostrando que essa escolha não é trivial.

Tradicionalmente, os motores de física têm como principal tarefa a solução da seguinte proposição: dado um sistema mecânico e as forças que atuam sobre ele, qual será o movimento resultante do sistema? Este problema é chamado de dinâmica direta (do inglês *forward dynamics*). Como existem vários fatores que influenciam o projeto de um motor de física, é bastante comum que diferentes motores gerem respostas diferentes para um mesmo sistema excitado da mesma forma. Ao se escolher um motor de física para uma simulação, o desempenho e a precisão do motor na característica principal a ser simulada deve ser o fator determinante na escolha (BOEING; BRÄUNL, 2007; ERLEBEN, 2004).

De acordo com a arquitetura proposta por Erleben (2004), o núcleo da simulação é composto pelos seguintes componentes: unidade de controle de tempo, integrador ou resolvedor de movimento, resolvedor de colisões e resolvedor de restrições. A unidade de controle de tempo é responsável por controlar o andamento da simulação, acionando os demais componentes no momento correto. O resolvedor de movimento realiza a resolução dos sistemas de equações que modelam o movimento dos objetos. O resolvedor de colisões trata especificamente a restituição de movimento dos objetos que colidem uns com os outros e o resolvedor de restrições resolve casos especiais onde o movimento de um objeto é restrito de alguma forma.

2.4.1 Controle de Tempo

A simulação começa com a entrada do estado inicial do sistema e a quantidade de tempo que deve ser simulada. Ao final do processamento, o resultado será o estado resultante no tempo requisitado. Existem várias abordagens para o controle do tempo. Geralmente, os algoritmos de controle de tempo são classificados em duas classes, os de passo de tempo fixos e os de passo de tempo adaptativo.

Os algoritmos de passo de tempo fixo funcionam avançando a simulação em quantidades de tempo predeterminadas até que o tempo final requisitado seja atingido, arredondando para cima. Os passos de tempo utilizados pela unidade de controle devem estar pelo menos uma ordem de magnitude abaixo do tempo de simulação requisitado. A desvantagem deste método é que podem ocorrer falhas de penetração e ultrapassagem. A penetração é quando dois objetos se sobrepõem e a ultrapassagem é quando dois objetos passam "um por dentro do outro" sem colidir. Estas falhas ocorrem pela não detecção de uma colisão, o que é causado quando o passo de tempo adotado é muito grande em comparação com o tamanho ou a velocidade dos objetos simulados. Alguns métodos foram propostos para resolver estes problemas (STEWART; TRINKLE, 1996; MILENKOVIC; SCHMIDL, 2001).

Os algoritmos de passo de tempo adaptativos estão divididos em duas categorias: retrocesso de passo de tempo e os de mão única. Ambos funcionam procurando pelo tempo em que ocorre um contato. Este tempo é informado pelo resolvedor de tempo(*time solver*). Os algoritmos de retrocesso de passo de tempo funcionam requisitando ao resolvedor de tempo para simular um passo, seguido por uma consulta ao detector de colisões para verificar se houve alguma penetração. Caso tenha acontecido alguma penetração, requisita-se ao resolvedor de tempo que ele restaure o estado dos objetos até o momento anterior ao acontecimento da penetração. Em seguida, o movimento é recalculado a um passo de tempo menor, até que estejam separados ou em contato sem penetração. O detector de colisões trata então as ocorrências e calcula as forças resultantes da colisão. Este processo é repetido para todos os objetos que colidem, até que o avanço no tempo originalmente requisitado seja atingido. Os algoritmos de mão única, por sua vez, nunca permitem penetrações. Em vez disso, é calculada uma estimativa do Tempo de Impacto entre objetos que se aproximam e este tempo é usado para a simulação. Os tempos são estimados a partir das informações do detector de colisões, que informa as distâncias de todos os pares de objetos próximos. Dentre todos os Tempos de Impacto que foram calculados, é selecionado o menor deles. O resolvedor de tempo avança a simulação até este ponto. Em seguida, o resolvedor de colisões determina os impulsos resultantes da colisão deste par. Este padrão de estimar o Tempo de Impacto, selecionar o menor, resolver a colisão, é repetido até que seja atingido o tempo final da simulação previamente requisitado para todos os pares de objetos. Conforme os objetos se aproximam, o tempo de impacto é recalculado, o que faz com que a precisão aumente até que se atinja um limite preestabelecido.

Existem ainda algumas propostas diferentes, de sistemas híbridos, que escolhem o algoritmo mais indicado para cada caso de acordo com critérios específicos (BARAFF, 1990) e abordagens assíncronas (MIRTICH, 2000), onde os movimentos são dessincronizados e resolvidos independentemente, seguido da ressincronização global pela unidade de controle de tempo.

A simulação de um quadro representa, por fim, várias iterações do controle de tempo, seguidas das chamadas aos respectivos componentes auxiliares.

2.4.2 Resolvedor de Movimento

O resolvedor de movimento é responsável pela continuidade do movimento dos objetos através da integração numérica das equações de movimento. Cada vez que o módulo de controle de tempo necessita calcular a simulação até um certo ponto, ele faz uma chamada ao resolvedor de movimento.

Tabela 2.1 - Notação para EDO

Símbolo	Descrição
$Y(t)$	Função de estado
$\vec{r}$	Posição do centro de massa do objeto
q	Orientação do objeto em forma de quatérnion
$\vec{P}$	Momento linear
$\vec{L}$	Momento angular em relação ao centro de massa
$\vec{v}$	Velocidade linear do centro de massa
$[0,\vec{\omega}]$	Velocidade angular em forma de quatérnion
$\vec{F}$	Força total
$\vec{\tau}$	Torque total em relação ao centro de massa
$\vec{a}$	Aceleração linear do centro de massa
$\vec{\alpha}$	Aceleração angular

Fonte: (ERLEBEN, 2004)

As equações de movimento são dadas por um conjunto de Equações Diferenciais Ordinárias (EDO) ou por uma função de movimento predefinida (*scripted motion*) (ERLEBEN; HENRIKSEN, 2002). Tipicamente, a EDO de movimento de um objeto é dada dessa forma, seguindo a notação da Tabela 2.1:

$$\frac{d}{dt}\begin{bmatrix}\vec{r}\\ q\\ \vec{P}\\ \vec{L}\end{bmatrix} = \begin{bmatrix}\vec{v}\\ \frac{1}{2}[0,\vec{\omega}]q\\ \vec{F}\\ \vec{\tau}\end{bmatrix}, \tag{2.13}$$

Os objetos com movimento predefinido ou controlado pelo usuário têm seu movimento descrito dessa forma:

$$Y(t) = \{\vec{r}, q, \vec{v}, \vec{\omega}, \vec{a}, \vec{\alpha}\} \tag{2.14}$$

ou seja, uma lista de tempos e estados gerados tipicamente pela interpolação de pontos de controle sobre uma trajetória. Esta trajetória pode ser definida de várias maneiras, como curvas de Bézier ou *splines* (ERLEBEN, 2004; ERLEBEN; HENRIKSEN, 2002).

Ambas as equações determinam o estado de um único objeto. As configurações comumente utilizadas consistem em ambientes mistos com vários objetos com movimento controlado pelas EDOs e outros com movimento predeterminado. As EDOs dependem das forças de restrição e forças externas, enquanto os objetos com movimento predeterminado são completamente independentes das forças agindo na configuração. Isso permite que o animador tenha liberdade de produzir cenas em que um ou mais objetos podem se mover sem restrições quanto a Física, enquanto outros objetos com os quais venham a interagir tenham seu movimento simulado de forma fisicamente realista.

A resolução dos movimentos de cenas em configuração mista deve seguir uma certa ordem. O resolvedor de movimento usará um método de integração para resolver o movimento dos objetos governados por EDOs. Entretanto, a cada passo da integração no resolvedor de EDOs, o resolvedor de movimento calcula, em primeiro lugar, os estados dos objetos com movimento predefinido. Em seguida calcula as forças externas e as forças de restrição agindo sobre os objetos. Um resolvedor de restrições pode ser usado para resolver as forças de restrição a partir dos estados intermediários encontrados pelo resolvedor de movimento. Após o cálculo de todos estes parâmetros, o resolvedor de EDOs os aplica e gera o estado final dos objetos.

Vários métodos de integração podem ser usados para resolver o conjunto de equações de movimento. O mais comumente utilizado é o integrador de Euler, tanto em sua forma normal quanto em sua forma simplética (BOEING; BRÄUNL, 2007; BARAFF; WITKIN, 1997). O integrador de Euler é descrito dessa forma:

$$x(t_0 + h) = x_0 + h\dot{x}(t_0) \tag{2.15}$$

Na forma simplética, a diferença em relação à forma normal é que a velocidade atualizada é utilizada antes do cálculo da posição:

$$\begin{aligned} \dot{x}(t_0 + h) &= \dot{x_0} + h\ddot{x}(t_0) \\ x(t_0 + h) &= x_0 + h\dot{x}(t_0) \end{aligned} \tag{2.16}$$

Outros métodos podem ser utilizados, de acordo com a estabilidade numérica ou eficiência de processamento desejados para a aplicação planejada do motor de física. Geralmente, os motores destinados a jogos eletrônicos prezam a velocidade de processamento em restrição à precisão, em razão de seus requisitos de execução em tempo real. Os motores destinados a produção de efeitos visuais cinematográficos não têm grandes exigências de tempo ou de precisão, mas de liberdade para os animadores inserirem objetos com movimento predefinido. Portanto, podem utilizar-se dos motores para jogos sem maiores problemas. Já os motores destinados a simulações científicas podem necessitar de métodos mais robustos e precisos, muitas vezes sacrificando o tempo de processamento, quando não há requisitos de execução em tempo real.

2.4.3 Detecção de Colisão e Contato

A detecção de colisão é um problema puramente geométrico, de intersecção entre objetos. Também é a parte com maior custo computacional da simulação física.

O componente de detecção de colisões é chamado após o módulo de simulação ter calculado as posições de todos os objetos. O componente analisa a geometria dos objetos em busca de contatos ou penetrações entre eles. A detecção então, prossegue em fases. Na primeira fase, é feita uma análise ampla das distâncias entre os objetos. O objetivo desta fase é diminuir o espaço de busca, separando apenas objetos que estejam próximos o suficiente para haver um contato para serem analisados na próxima fase. Como resultado, o custo computacional é bastante reduzido. Os algoritmos usados podem basear-se em busca exaustiva, varredura e descarte entre outras(ERLEBEN, 2004; COHEN et al., 1995; BARAFF; WITKIN, 1997). A proximidade pode ser extraída através da aproximação do volume aparente ocupado pelo objeto. A informação do movimento do objeto pode ser útil para a construção de um volume do movimento total (MIRTICH, 1996). Fazendo o uso de uma previsão antecipada, uma colisão pode ser detectada antes que ela aconteça. Este método é predominantemente utilizado na abordagem de mão única, já que esta não permite que aconteçam penetrações.

A segunda fase da detecção de colisão é busca refinada nos pares de objetos selecionados na fase anterior para determinar se realmente aconteceu uma colisão. Os algoritmos utilizados nesta fase retornam informações mais ricas sobre a colisão, podendo retornar pontos de contato, informações de penetração e outras informações de proximidade. Este tema é bastante estudado, vários algoritmos podem ser encontrados na literatura. Não há grandes implicações na escolha do método de detecção de colisão refinado para a simulação como um todo. A maior influência está na determinação da escolha do algoritmo a ser usado na próxima fase, a da determinação de contato.

Na fase de determinação de contato, é extraída a região de contato entre os objetos. Este é um problema puramente geométrico, mas a determinação dessa região é importante para determinar as forças de contato. Sua solução, entretanto, não é trivial, devido à quantidade de incertezas e variabilidade das representações. Com o cálculo da região de suporte do contato, diminui-se o número de restrições ao movimento e o custo computacional é bastante reduzido.

Após a detecção das colisões, é necessário resolver os impulsos produzidos por elas. Uma colisão produz uma descontinuidade no movimento do objeto. Como o resolvedor de movimentos não lida com estas descontinuidades, ele precisa ser informado e o estado do objeto atualizado com os parâmetros resultantes da colisão. Um resolvedor de colisões pode ser usado para o cálculo dos

impulsos resultantes das colisões. Esses cálculos podem ser feitos de forma algébrica, incremental ou deformável.

A forma algébrica resolve um sistema de equações lineares e tem custo computacional baixo. Ela descreve a rede de colisões através de quantidades pré e pós colisão. A forma incremental usa um modelo de colisão microscópico que é integrado através do tempo de colisão. Tem custo computacional mais alto que a forma algébrica. Finalmente, os modelos deformáveis resolvem uma equação diferencial parcial que descreve as mudanças físicas que ocorrem durante a colisão. Esta forma não é utilizada em sistemas em tempo real por ser praticamente impossível determinar as condições iniciais da equação e também porque a resolução de equações diferenciais parciais é um problema computacionalmente muito caro.

Os impulsos podem ser aplicados de forma simultânea ou sequencial. O resolvedor de colisões é chamado depois da determinação das regiões de contato, tomando estas como argumento. Dessa forma, os impulsos são aplicados apenas nas regiões de contato.

2.5 Interfaces de Usuário Naturais

As interfaces desempenham um importante papel no uso de sistemas complexos. É através das interfaces que é feita a ponte homem-máquina; a construção de interfaces amigáveis, que permitam um rápido domínio por parte do usuário, é uma característica bastante desejada. Uma interface mal projetada aumenta o tempo dispensado com treinamentos e pode limitar o uso do equipamento ao causar fadiga prematura no usuário. A construção de interfaces é um tópico que vem recebendo especial atenção nos últimos anos.

Dentre os estudos desenvolvidos nesta área, destacam-se os estudos em Interfaces de Usuário Naturais (NUI - *Natural User Interfaces*), ou apenas Interfaces Naturais. Trata-se de uma técnica multidisciplinar que visa a criação de dispositivos de interação que sejam capazes de assimilar aspectos naturais de interação entre seres humanos e interpretá-los em comandos para a máquina. Os seres humanos interagem entre si e com o mundo exterior através de vários meios, sendo o mais

notável a linguagem. A interação baseada em linguagem engloba a fala, a escrita e também um importante componente gestual. As NUIs buscam construir meios de interação homem-máquina que utilizem as formas naturais de interação humana.

O uso de NUIs apresenta como principal vantagem, a redução do tempo dispensado em aprendizado e adaptação. Uma vez que a técnica utiliza-se de características naturais para sua operação, o aprendizado e o domínio da interface acontecem intuitivamente. Sua grande dificuldade, conforme observado por Buxton e Myers (1986), vem da necessidade de integrar dispositivos de entrada-saída que permitam a interação natural. Atualmente, esse cenário vem mudando, a medida que dispositivos como telas de toque, câmeras e microfones tornaram-se mais eficientes e com custos de produção baixos o suficiente para permitir a massificação do emprego de tais dispositivos. Esse contexto contribuiu diretamente para a formalização e o incentivo de pesquisas em NUI.

O primeiro dispositivo que permitiu o desenvolvimento das técnicas de NUI, foram as telas de toque. A interação com elementos de interface pelo toque é bem mais intuitiva que a interação por meio de um dispositivo apontador, como o mouse. A primeira tela de toque foi descrita por Johnson (1967). Tratava-se de uma tela resistiva projetada para uso em terminais de controle de tráfego aéreo. Os primeiros dispositivos desse tipo eram capazes de detectar as coordenadas do ponto onde o usuário estava tocando a tela e produzir a ação correspondente, mas apenas para um único ponto. Algumas aplicações, como visualização de dados tridimensionais, necessitam um número maior de graus de liberdade para os movimentos possíveis. Para permitir esta aplicação e aproveitar todo o potencial oferecido pela tecnologia, as telas evoluíram para dispositivos capazes de processar múltiplos toques. As vantagens dessa abordagem foram discutidas inicialmente por Buxton e Myers (1986). A partir destes conceitos, as aplicações foram evoluindo, a medida que novos dispositivos foram sendo criados e lançados no mercado. Atualmente, as telas de toque múltiplo estão tornando-se padrão de fato na indústria de telefones celulares inteligentes (*smartphones*) e dispositivos multimídia sem teclado (*tablets*). A maioria destes dispositivos segue os conceitos propostos por Buxton e Myers (1986) e aprimorados por Westerman (1999).

A interação por toque é interessante, mas ainda limita o usuário a estar fisicamente próximo ao dispositivo. Essa abordagem funciona bem em dispositivos pequenos e médios, que podem ser diretamente suportados nas mãos do usuário ou apoiados em algum móvel. Dispositivos maiores ou

que não possam estar diretamente em contato com o usuário não podem beneficiar-se desta tecnologia. O ser humano também possui uma classe de elementos de interação gestuais que não necessitam de contato físico. Para suprir essas necessidades desenvolveram-se as interfaces gestuais. Os dispositivos utilizados para possibilitar esse tipo de interação necessitam de alguma tecnologia que possibilite que o usuário seja percebido. Em sua maioria, tratam-se de câmeras que utilizam técnicas de Visão Computacional e Processamento de Imagens com o objetivo de detectar e registrar o usuário e rastrear seus movimentos.

Uma das principais técnicas utilizadas para recuperar a informação tridimensional que é perdida ao gerar a imagem de uma cena é a Visão Estéreo. A Visão Estéreo tradicionalmente é e continua a ser o tópico mais estudado em Visão Computacional (SCHARSTEIN; SZELISKI, 2002). Existem muitos métodos e algoritmos, com características próprias. Alguns concentram-se na precisão, porém têm custos computacionais elevados impedindo seu uso em tempo real. Alguns concentram-se em minimizar o tempo de execução, mas sacrificam a precisão. Um dos maiores desafios nessa área, é produzir algoritmos e métodos que tenham boa precisão e tempo de execução baixo o suficiente para permitir o uso em tempo real. Para atingir este objetivo, é necessário o uso de técnicas avançadas, como processamento massivamente paralelo utilizando unidades de processamento de vídeo, proposto por Sizintsev et al. (2010). Novas soluções em *hardware*, como câmeras equipadas com *Field-Programmable Gate Arrays* (FPGAs), têm também permitido o uso de técnicas de visão estéreo precisas em tempo real.

A indústria dos jogos eletrônicos é uma das que mais se beneficiam e investem nesta tecnologia. Nos últimos anos, vislumbramos o lançamento de dispositivos de interação natural nessas plataformas, recebendo ótimas críticas e grande aceitação por parte de seu público alvo. Esses dispositivos são aproveitados por pesquisadores independentes visando criar novas soluções. Estas soluções estão sendo adaptadas para uso em sistemas de computação pessoal que logo estarão disponíveis comercialmente, contribuindo para a universalização destas técnicas.

2.6 Redes de Computadores

Uma rede de computadores consiste em dois ou mais computadores e/ou outros dispositivos, interconectados por algum meio de acesso que permita que compartilhem recursos lógicos ou físicos. A ideia de trocar informações e compartilhar recursos surgiu praticamente de forma simultânea à criação dos computadores. Já em 1940, George Robert Stibitz demonstrou durante uma conferência da Sociedade Americana de Matemática (AMS, *American Mathematical Society*) no *Dartmouth College* o acionamento de seu recém construído Calculador de Números Complexos (*Complex Number Calculator*). O aparelho estava nos laboratórios da *Bell Labs* em Nova Iorque e foi acessado por uma máquina de teletipos, pela qual os problemas eram transmitidos e as respostas recebidas, utilizando a rede telefônica da época (HOLBROOK; BROWN, 1982). A conexão de equipamentos distantes para troca de informações chamou a atenção da *Advanced Research Projects Agency* (ARPA), uma agência de pesquisa de tecnologia para fins militares, que na década de 60 montou um grupo de trabalho para o desenvolvimento de uma rede de interconexão de computadores. Esse esforço resultou na *ARPANET*, uma rede de interconexão que foi o embrião da Internet como conhecemos hoje. Muitas das tecnologias e modelos utilizados hoje por milhões de pessoas, diariamente, foram concebidas neste momento.

Existem várias tecnologias para comunicação em redes. No campo do *hardware*, elas diferem principalmente quanto ao meio de acesso utilizado para comunicação (cabos metálicos, fibra óptica, ondas de rádio) e a velocidade de transmissão que são capazes de atingir. No campo do *software*, elas diferem quanto aos protocolos utilizados. Um protocolo é a camada de *software* responsável por gerenciar todo o processo da comunicação. Os protocolos são responsáveis por acessar o *hardware*, codificar e decodificar os dados a serem transmitidos, identificar os vários dispositivos que estejam conectados à rede, direcionar as transmissões para o destinatário correto, entre outros.

2.6.1 Protocolos de Comunicação

As primeiras redes de computadores foram projetadas com o *hardware* como principal preocupação, deixando o *software* em segundo plano. Essa estratégia não é mais utilizada. Hoje, o *software* de rede é altamente estruturado (TANENBAUM, 2003). As principais peças de *software* que regem a comunicação entre computadores são os protocolos. Um protocolo pode ser entendido como um acordo entre as partes de como a comunicação deve proceder. Se ocorre uma violação do protocolo, a comunicação torna-se difícil ou até mesmo impossível.

Para facilitar o projeto e reduzir a complexidade, o *software* de rede é organizado em forma de uma pilha de camadas construídas umas sobre as outras. O número de camadas e os seus nomes variam de rede para rede. Cada camada oferece um conjunto de serviços para a camada imediatamente acima, prevenindo a camada superior de conhecer detalhes do funcionamento e da implementação da camada inferior. Os dados então, são transmitidos das camadas superiores para as camadas inferiores até que a camada mais baixa seja atingida. Abaixo dessa camada, está o meio físico pelo qual a comunicação acontece. Entre cada camada do protocolo, existe uma interface que expõe os serviços oferecidos e permite a troca dos dados. A função desta interface é normalizar o modo de acesso à troca de dados, de forma que a implementação possa variar sem que o conjunto da operação seja afetado. Por exemplo, pode-se mudar o meio físico de cabos para fibras ópticas, alterando-se a implementação da primeira camada apenas. Mantendo a interface inalterada, as outras camadas não precisam ser alteradas.

Cada camada acrescenta um conjunto de informações à mensagem recebida. Esse conjunto de informações é chamado cabeçalho (*header*). Seu objetivo é permitir que, ao partir do remetente e atingir o destinatário, os dados serão corretamente interpretados pela camada equivalente. Se, por exemplo, uma camada utiliza um limite de tamanho nos dados que podem ser transmitidos, é necessário que estes sejam divididos em conjuntos menores. Os cabeçalhos guardam a informação de como os dados devem ser reconstruídos. Cada camada do emissor estabelece um circuito de comunicação virtual com a camada correspondente do receptor. Ou seja, cada protocolo de uma determinada camada percebe a comunicação como se fosse um processo horizontal entre protocolos iguais. O circuito real é vertical, passando por todas as camadas inferiores. A Figura 2.14

exemplifica esse processo.

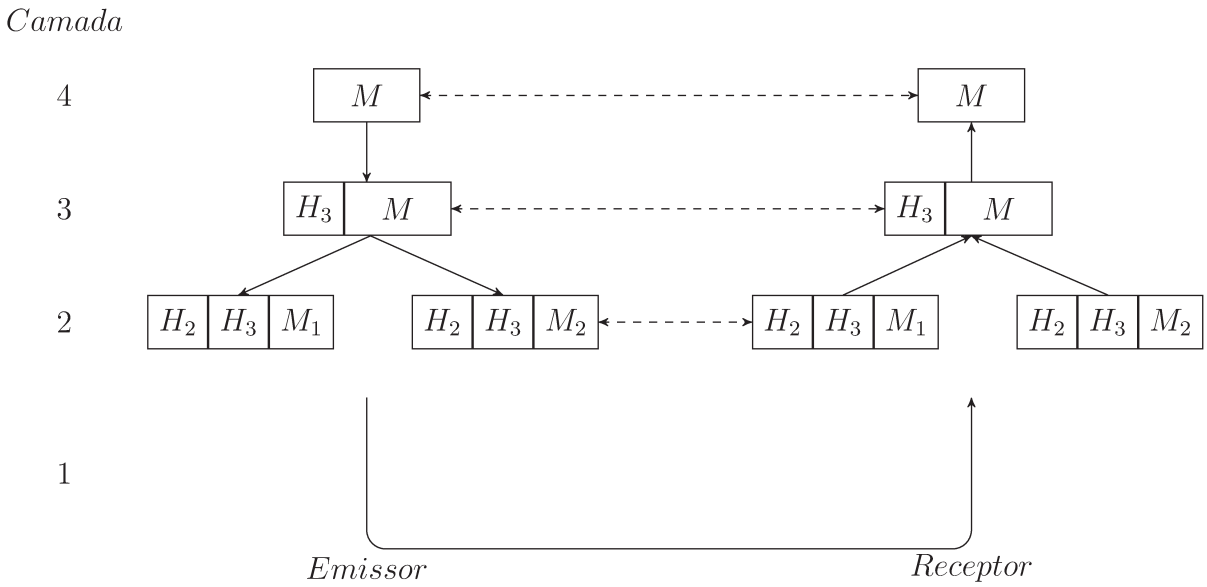


Figura 2.14 - Exemplo de transmissão de dados através de um protocolo de 4 camadas. O protocolo de cada camada do emissor acrescenta um cabeçalho (*header*) que é retirado pelo protocolo da camada superior no destinatário, a fim de reconstruir a mensagem original. O tracejado mostra um circuito virtual. O circuito real é mostrado em linhas sólidas.
Fonte: Adaptado de (TANENBAUM, 2003)

O Modelo de Referência OSI

No início da disseminação das tecnologias de rede, não existiu um esforço voltado à padronização dos sistemas. Cada fabricante adotava uma solução de protocolos proprietária, o que impedia que equipamentos de fabricantes diferentes se comunicassem entre si. Era necessário implantar toda a infraestrutura utilizando equipamentos de um único fabricante. O primeiro passo em direção a uma padronização internacional de protocolos de rede foi dado pela *International Standards Organization* (ISO). A organização propôs um modelo de referência que deveria ser seguido pelos fabricantes de equipamentos de redes, garantindo desse modo a interoperabilidade entre estes. Este modelo recebeu o nome de *Open Systems Interconection Reference Model* (OSI *Reference Model*, Modelo de Referência para Interconexão de Sistemas Abertos).

O modelo OSI é composto por sete camadas, cada camada sendo tratada por um protocolo diferente. Cada camada foi criada com base em um nível de abstração específico e deve prover uma função bem definida. As funções de cada camada devem ser definidas observando protocolos padronizados. As fronteiras de cada camada devem ser escolhidas de modo a minimizar o fluxo de informações entre interfaces e o número de camada foi pensado para ser grande o bastante para garantir que funções não sejam replicadas em camadas diferentes, mas pequenas o suficiente para não tornar a especificação muito extensa. Com base nestes princípios, chegou-se ao número de sete camadas.

Figura 2.15 - As sete camadas do modelo de referência OSI.
Fonte: Adaptado de (TANENBAUM, 2003)

- Camada Física - essa camada é responsável pelo tráfego dos *bits* através dos canais de comunicação. Ou seja, descreve como os dados devem ser representados da forma que o meio de acesso utilizado utilize, por exemplo sinais elétricos para cabos metálicos, sinais luminosos para fibras ópticas ou ondas de rádio para satélites.
- Camada de Enlace de Dados - esta camada tem a tarefa de transformar o fluxo de *bits* como uma linha contínua. Lida também com o controle e correção de erros. Para isso, ela quebra o fluxo em quadros compostos por algumas centenas ou milhares de *bits* e os envia sequencialmente. Caso o serviço seja do tipo confiável, o receptor envia de volta um quadro de confirmação (*acknowledgement*). Esta camada também faz o controle do fluxo, impedindo que um transmissor rápido sobrecarregue um receptor mais lento, danificando a comunicação. Essa camada ajusta a velocidade de transmissão dos equipamentos de modo que operem na mesma velocidade. Em redes que compartilham o canal de comunicação, chamadas *broadcasting networks*, essa camada também

controla o acesso ao meio, impedindo que dois ou mais equipamentos transmitam ao mesmo tempo, ocasionando uma colisão.

- Camada de Rede - esta camada cuida do direcionamento dos dados, fazendo com que as mensagens cheguem aos destinatários corretos. Seu principal princípio é o modo como os pacotes serão roteados entre as redes. As rotas podem ser armazenadas de modo dinâmico ou de forma estática, através de uma tabela de roteamento. Quando vários equipamentos fazem parte da mesma rede, o gerenciamento do tráfego também é função desta camada. Muitos dos indicadores usados para avaliar a qualidade do serviço, são preocupações desta camada. A inteconexão de redes diferentes faz parte desta camada e também a adequação de protocolos diferentes, para permitir que redes heterogêneas possam se comunicar. Nas redes de broadcasting, o problema do roteamento é mais simples, então essa camada é bem pequena ou pode até mesmo ser dispensada.
- Camada de Transporte - a função desta camada é aceitar os dados da camada superior, dividi-los em unidades menores, se necessário, passá-los à camada de rede e garantir que as partes sejam recebidas corretamente na outra ponta. Essa camada também é responsável por determinar quais os serviços serão providos às camadas superiores em razão do tipo de transporte utilizado. Este, pode ser uma conexão livre de erro ponto a ponto, que garante que os *bytes* da mensagem serão entregues na mesma ordem que foram enviados. Outros serviços transportam mensagens isoladas, sem garantia da ordem em que estas chegam, outros são especializados em entregar a mensagem para múltiplos destinatários. Esta camada atua de ponto a ponto, diferentemente das camadas inferiores. Ou seja, um programa no emissor comunica-se com outro programa no receptor, enquanto as camadas inferiores comunicam-se entre máquinas vizinhas e não diretamente entre o emissor e o receptor,sendo que estes muitas vezes estão separados por diversas rotas.
- Camada de Sessão - esta camada executa o controle do diálogo, mantendo um registro de qual máquina pode transmitir e cuida da sincronização, permitindo que uma comunicação continue do ponto em que foi interrompida, em caso de queda na conexão.
- Camada de Apresentação - as camadas inferiores a esta são responsáveis pelo movimento da informação entre os pontos. O que a camada de apresentação faz é recuperar o significado desses dados de forma que eles possam ser usados. Ela cuida, basicamente,

das estruturas de dados abstratas que permitem às aplicações utilizar as informações transmitidas e recebidas.

- Camada de Aplicação - esta é a camada mais superior e por isso é a que os usuários têm mais contato. Ela define a operação que será realizada na rede, por exemplo, transferência de arquivos, correio eletrônico (*e-mail*).

O Protocolo TCP/IP

O protocolo TCP/IP (*Transfer Control Protocol/Internet Protocol*) foi proposto como solução de requisitos da rede ARPANET. A ARPANET foi uma rede de pesquisa patrocinada pelo DoD (Departamento de Defesa dos Estados Unidos da América). Esta rede chegou a conectar centenas de universidades e agências do governo através de linhas telefônicas. Mais tarde, foram adicionadas a estas linhas redes de satélites e estações de rádio. Neste momento, os protocolos existentes até então tiveram dificuldade em interligar-se com estas novas redes. Para resolver este problema, foi proposta uma nova arquitetura que permitisse a interconexão de redes diferentes. Esta arquitetura foi nomeada mais tarde como Modelo de Referência TCP/IP, tomando o nome de seus dois protocolos mais destacados. O protocolo foi descrito por Cerf e Kahn (1974) e mais tarde consolidado por Leiner et al. (1985).

Um dos requerimentos da ARPANET foi formulado em decorrência do raciocínio que, em caso de guerra, poderiam ser perdidos clientes, roteadores ou outros elementos da rede em razão de um ataque inimigo. Por esta razão, a rede deveria ser capaz de sobreviver à perda de elementos sem a interrupção de comunicações em curso. Ou seja, uma conexão deveria manter-se aberta enquanto as máquinas emissora e receptora estivessem intactas e houvesse pelo menos uma rota disponível. Adicionalmente, a arquitetura deveria ser flexível o suficiente para permitir serviços diferentes com requisitos também diferentes, como transferência de arquivos, troca de mensagens e transmissão de voz.

O modelo TCP/IP difere quanto ao modelo OSI em número de camadas e funções. A Fi-

gura 2.16 mostra a diferença entre as camadas dos dois modelos.

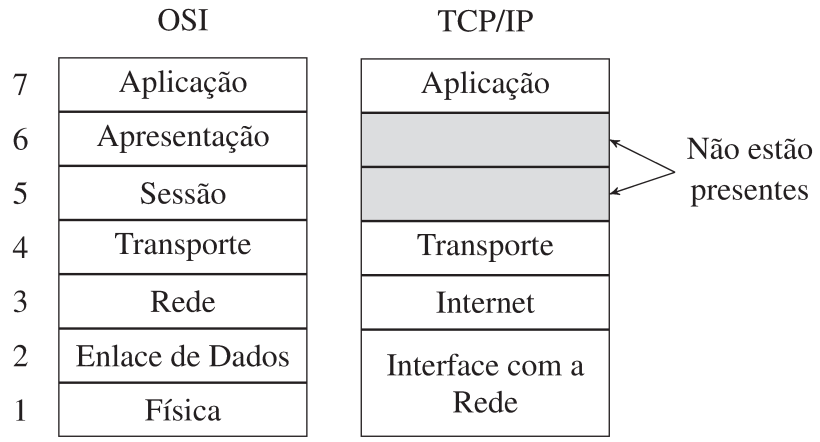


Figura 2.16 - Comparação entre os modelos OSI e TCP/IP.
Fonte: Adaptado de (TANENBAUM, 2003)

O modelo TCP/IP possui quatro camadas:

- Aplicação - O modelo TCP/IP não possui as camadas de Apresentação e de Sessão. Percebeu-se que estas camadas teriam pouca utilidade, então foram incorporadas à camada de aplicação. Aqui residem os protocolos de alto nível, que definem o tipo de comunicação que será realizada. Os primeiros protocolos desenvolvidos nesta camada ofereceram serviços de acesso remoto a terminais (TELNET, *Terminal Emulation Link Network*), transferência de arquivos (FTP, *File Transfer Protocol*) e correio eletrônico (SMTP, *Simple Mail Transfer Protocol*). Mais tarde, outros protocolos mais especializados foram adicionados, provendo uma nova série de serviços, como resolução de nomes de domínios em endereços de rede (DNS, *Domain Name System*) e o Protocolo de Transferência de Hipertexto (HTTP, *Hyper-Text Transfer Protocol*), que permite o acesso a páginas da *World Wide Web* (WWW), provavelmente o protocolo mais utilizado no mundo.
- Transporte - a camada de transporte do modelo TCP/IP tem funcionamento análogo ao do modelo OSI, permitir que dois elementos da rede mantenham uma conversação ponto a ponto. Nesta camada, existem dois protocolos definidos, o TCP (*Transfer Control Protocol*) e o UDP (*User Datagram Protocol*). O TCP é um protocolo orientado a conexão confiável que permite que um fluxo de *bytes* seja enviado, sem erros, do emis-

sor ao receptor. Ele fragmenta o fluxo em uma série de mensagens discretas e as passa para a camada de Internet. No destino, o protocolo TCP remonta o fluxo de *bytes*. É realizado um controle de erro baseado no reenvio de pacotes perdidos ou corrompidos. O TCP também realiza o controle do fluxo de dados, impedindo que um emissor sobrecarregue um receptor mais lento. O protocolo UDP, por outro lado, é não-confiável e sem conexão. Ele é usado em aplicações que não necessitem do sofisticado controle de sequenciamento e fluxo, ou que desejem implementar seu próprio controle. Também é amplamente usado em aplicações do tipo cliente-servidor ou em aplicações em que a agilidade na entrega seja mais importante que a integridade dos dados, por exemplo, transmissão de vídeo em tempo real.

- Internet - a camada de Internet (não confundir com a grande rede mundial) é a coluna vertebral do modelo TCP/IP. Para garantir o requisito de entrega de dados através de múltiplas rotas, a escolha caiu sobre um protocolo de troca de pacotes, sem conexão, que fosse capaz de transportá-los entre redes diferentes de forma independente. Os pacotes selecionam a rota de acordo com vários critérios, então a ordem com que chegam nem sempre é a mesma em que foram enviados. Rearranjá-los na ordem correta é deixado como tarefa para as camadas superiores, caso necessário. O protocolo dessa camada é o IP (*Internet Protocol*). Ele define um formato próprio de pacote e uma forma de endereçamento, para garantir que os pacotes cheguem aos dispositivos corretos. A maior preocupação nessa camada é o roteamento dos pacotes e controle do fluxo para evitar congestionamentos. Seu funcionamento é similar à camada de Rede do modelo OSI.
- Interface com a rede - assim como as camadas inferiores do modelo OSI, essa camada tem a função de adequar os fluxos de *bits* ao meio utilizado. Na realidade, o modelo TCP/IP não define muitos requisitos nessa camada, exceto que o protocolo utilizado aqui seja capaz de se conectar a uma rede e enviar pacotes IP através dela.

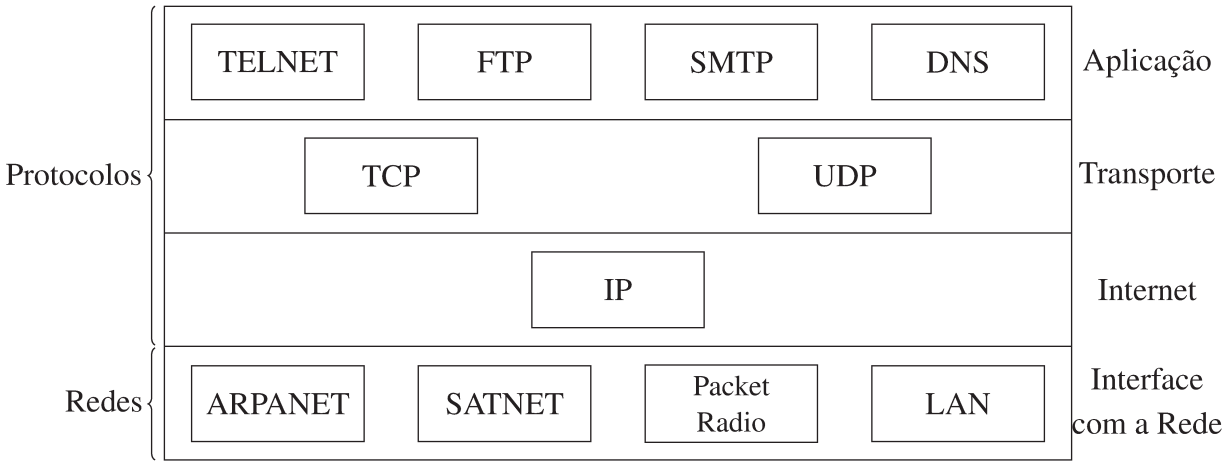


Figura 2.17 - Pilha de protocolos no modelo TCP/IP.
Fonte: Adaptado de (TANENBAUM, 2003)

2.6.2 Modelo Cliente-Servidor

O modelo cliente servidor é um paradigma de projeto de sistema no qual um ou mais Clientes em conjunto com um ou mais Servidores formam um sistema composto que permite computação distribuída (SINHA, 1992). Os clientes e servidores comunicam-se através de um sistema de comunicação inter-processos. O servidor e o cliente, geralmente, residem em máquinas diferentes interligadas através de uma rede, mas também podem coexistir em uma única máquina. A Figura 2.18 mostra uma arquitetura do tipo cliente-servidor.

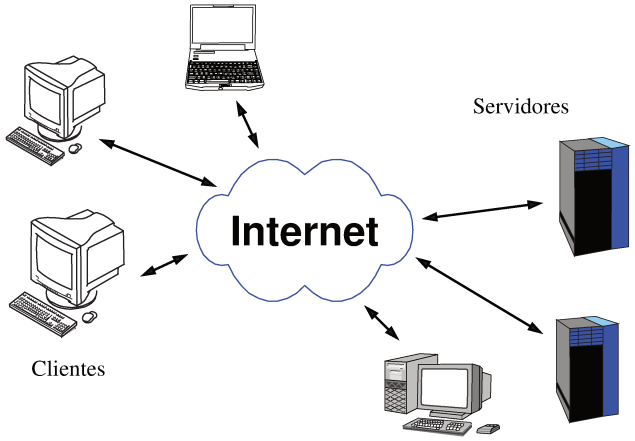


Figura 2.18 - O Modelo Cliente-Servidor.

Neste paradigma, cada uma das entidades tem comportamentos distintos. O servidor é um processo que provê um serviço para os clientes. A natureza e a extensão dos serviços são defi-

nidas de acordo com a função do sistema. O servidor responde a comandos dos clientes, logo, um servidor nunca inicia uma comunicação, ele aguarda a requisição de um cliente. Ao receber essa requisição, ele interpreta o comando e responde fornecendo o serviço requisitado ao cliente. O servidor pode agir como um repositório de dados (servidor de arquivos), conhecimento (banco de dados) ou meramente como um provedor de serviços (servidor de impressão). Os comandos são passados através de uma linguagem que pode ser padronizada, como a SQL (*Search Query Language*) para bancos de dados, ou de uma linguagem própria. A natureza computacional dos serviços fornecidos pelo servidor variam, podendo ser de baixa intensidade computacional (servidores de arquivos, servidores de impressão) ou de alta capacidade (servidores de banco de dados, ou de processamento).

O cliente é o processo que interage com o usuário. Ele apresenta a interface ao usuário, onde serão formados os comandos para o servidor. Também deve iniciar e gerenciar a conexão com o servidor e analisar as respostas recebidas. O funcionamento ideal de um cliente deve ocultar o processo de comunicação do usuário, apresentando os resultados como se tivessem sido realizados pelo próprio cliente. Devem exigir pouca ou até mesmo nenhuma configuração.

Muitos sistemas distribuídos utilizam o modelo Cliente-Servidor como paradigma de funcionamento. Na verdade, muitos dos serviços encontrados na *World Wide Web*, são dessa natureza. O navegador que usamos para acessar a *Web* é um exemplo de aplicativo cliente. Ele faz requisições utilizando o protocolo HTTP para servidores, que respondem enviando de volta o documento requisitado, uma página da *Web*. A arquitetura distribuída também tem sido usada como forma de compartilhar trabalho computacional por redes de pesquisa.

2.7 Robótica

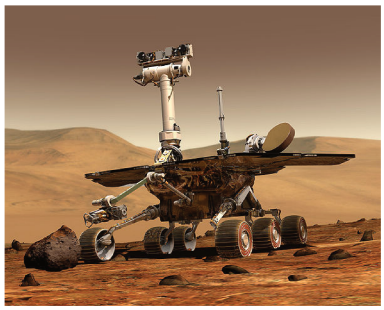
A palavra Robótica refere-se a uma área do conhecimento multidisciplinar que dedica-se ao estudo, projeto e construção de robôs e dos sistemas que os controlam. Engloba conceitos da Engenharia Mecânica, Engenharia Elétrica e da Computação. Os robôs, por sua vez, são agentes

virtuais ou mecânicos, capazes de executar ações de maneira autônoma ou guiada. Os robôs podem ser autônomos, quando são capazes de tomar suas próprias decisões, semi-autônomos, quando são auxiliados por um operador humano, ou controlados, quando todas as suas ações são controladas pelo operador. A palavra robô tem origem na palavra checa *robota*, que significa "trabalho forçado". O robô presente no imaginário mundial teve origem na peça R.U.R. (*Rossum's Universal Robots*) do dramaturgo checo Karel Čapek, na qual existia um autômato com forma humana, capaz de fazer tudo em lugar do homem (ČAPEK, 2010).

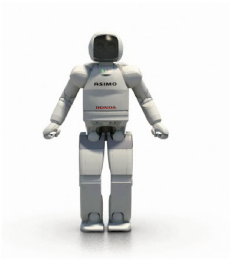
No mundo moderno, os robôs desempenham importante papel como força de trabalho industrial, realizando tarefas com precisão, velocidade e eficiência inigualáveis por operários humanos. Também podem ser utilizados para substituir os humanos em tarefas perigosas ou em ambientes hostis. Nos últimos anos, observamos também um crescimento do uso de veículos robóticos de uso militar, com destaque os veículos aéreos não tripulados. No início do ano de 2012, os Veículos Aéreos Não Tripulados (VANT) já perfaziam 31% da frota da Força Aérea Americana (ACKERMAN; SHACHTMAN, 2012), desempenhando tarefas que vão desde a espionagem, comando e controle e até mesmo como plataforma de lançamento de armas. Esta grande presença tem impulsionado pesquisas e o desenvolvimento de novas soluções.

Os robôs podem ser classificados em diversas categorias, de acordo com sua forma e/ou função. Os robôs móveis têm a capacidade de locomoverem-se livremente. Estão nesta categoria os veículos autônomos. Os robôs industriais realizam tarefas de fabricação em linhas de produção. Os manipuladores são aqueles usados para movimentar cargas ou realizar trabalhos. Normalmente, consistem de braços articulados, com um ou mais eixos de movimento, com um atuador fixado em sua extremidade. O atuador pode ser um dispositivo que permita segurar objetos ou uma ferramenta. Os robôs de serviço são os que realizam tarefas de auxílio a atividades que não sejam a fabricação.

(a)



(b)



(c)

(d)

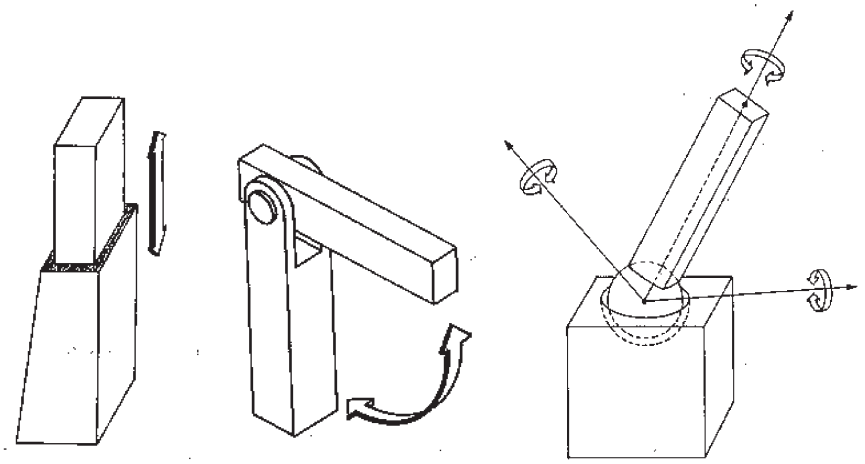
Figura 2.19 - Classes de Robôs. (a) Um manipulador robótico de uso industrial. (b) Sonda Spirit, um exemplo de robô móvel autônomo. (c) Asimo, um robô de serviço humanoide. (d) Um exemplo de robô de serviço não humanoide, o aspirador de pó robótico Roomba.
Fonte: (a) ABB Group, Inc. (b) Jet Propulsion Lab, Mars Rovers Mission. (c) Honda Inc. (d) iRobot Corp.

2.7.1 Manipuladores Robóticos

Os manipuladores robóticos são os robôs de uso mais difundido, sobretudo nas indústrias. Praticamente todas as linhas de produção modernas contam com algum manipulador exercendo alguma função produtiva. Um manipulador é constituído de um conjunto de juntas ligadas em série por elos ou vínculos, como na Figura 2.19(a), cada junta ligando dois vínculos. Uma das extremidades é fixada em uma base e a outra é livre para se movimentar e posicionar. A extremidade livre é referenciada como o atuador do robô. No atuador, podem ser fixadas ferramentas variadas, de acordo com o tipo de trabalho que o robô deve realizar. As juntas, por convenção, são numeradas a partir da base em direção ao atuador. A configuração mais comum é com seis eixos de movimento, ou graus de liberdade. Os três primeiros posicionam o robô no espaço. Sua configuração determina

o espaço de trabalho do robô, ou seja, os pontos que este poderá alcançar. Os três últimos eixos, geralmente, fornecem movimentos de rotação para posicionar o atuador. Este conjunto costuma ser chamado de pulso do robô (LEWIS et al., 2004).

Três tipos de juntas podem ser usadas na construção dos robôs manipuladores: deslizantes, rotativas ou esféricas (bola e encaixe), conforme mostrado na Figura 2.20. A juntas deslizantes permitem um movimento linear, onde um vínculo desliza sobre o outro. As juntas rotativas permitem um movimento de rotação de um vínculo sobre outro, como em uma dobradiça. As juntas esféricas permitem movimentos de rotação em três eixos simultaneamente, mas são mais raramente usadas devido a sua complexidade. Seu funcionamento pode ser conseguido com a associação de três juntas rotativas, com os eixos encontrando-se em um ponto.



(a) (b) (c)

Figura 2.20 - Tipos de juntas utilizadas em manipuladores robóticos. (a) Deslizante. (b) Rotativa. (c) Esférica.

Os manipuladores podem ser classificados de acordo com o tipo de juntas utilizadas nos três eixos mais próximos à base. As classificações mais comumente encontradas entre os robôs disponíveis comercialmente são:

- Cartesianos - são compostos por três juntas deslizantes montadas ortogonalmente entre si, chamados de eixos x, y e z. O espaço de trabalho é um paralelepípedo com as dimensões da extensão do movimento das juntas. São muito precisos e fáceis de construir. O atuador pode ter um conjunto de um, dois ou três eixos para sua orientação. São encontrados em grande variedade de tamanho e capacidade de carga, desde poucos

gramas até algumas toneladas.

- Braços Articulados - são construídos com três juntas rotativas para posicionamento do braço. Podem incluir mais três eixos para orientação do atuador. Dois eixos são co-planares e produzem movimento no plano vertical. O eixo da base é vertical e permite girar o robô, aumentando o espaço de trabalho. Tem precisão menor que outras configurações, já que os erros de posição de cada junta são multiplicados, então exigem recalibração em frequências maiores. Porém, são muito versáteis podendo atingir um variado número de posições. Um dos problemas em sua construção está no fato que a segunda junta deve suportar o peso dos eixos subsequentes somado ao da carga. O tamanho e potência dos motores pode limitar a construção e a capacidade de carga do braço.
- Articulações Horizontais - também conhecidos como SCARA (*Selectively Compliant Assembly Robot Arm*), possuem duas juntas rotativas paralelas, produzindo movimento no plano horizontal. Uma terceira articulação deslizante adiciona movimento vertical, o que adiciona volume ao espaço de trabalho do robô. Um quarto eixo de rotação pode adicionar orientação ao atuador. Raramente têm mais de quatro eixos. São muito usados para inserir componentes eletrônicos em placas de circuito e na montagem de peças de tamanho médio. Um outro tipo de configuração adiciona a articulação deslizante na base, em vez de usá-la próxima ao atuador.
- Esféricos - nesta configuração, as duas primeiras articulações são rotativas, montadas próximas à base e ortogonais entre si. A terceira é deslizante permitindo a extensão do atuador. O resultado é um espaço de trabalho esférico centrado no ponto de encontro dos eixos das articulações rotativas. São usados mais raramente que os braços articulados.
- Cilíndricos - a primeira articulação desse robô é rotativa e localizada na base. As outras duas são deslizantes, o que resulta em um espaço de trabalho cilíndrico. Juntamente com os robôs esféricos foram bastante populares no início do surgimento de linhas de produção robotizadas, mas atualmente, poucos modelos comerciais ainda existem.
- Vínculos Paralelos - são formados por uma série de três a seis vínculos paralelos ligados a uma base fixa e a uma plataforma móvel. Dependendo do projeto, podem atingir seis graus de liberdade em relação à base. São mais apropriados em algumas situações do

que robôs de vínculos em série. Podem ser bastante leves e ainda assim movimentar grandes cargas com precisão. Tem sido usados com sucesso na indústria aeroespacial. Entretanto, usam um modelo de cinemática fechada, que é bem complexo de analisar (LIU et al., 1993) e seu circuito de controle também é difícil de se projetar.

A Figura 2.21 mostra um exemplo de cada uma dessas classes.

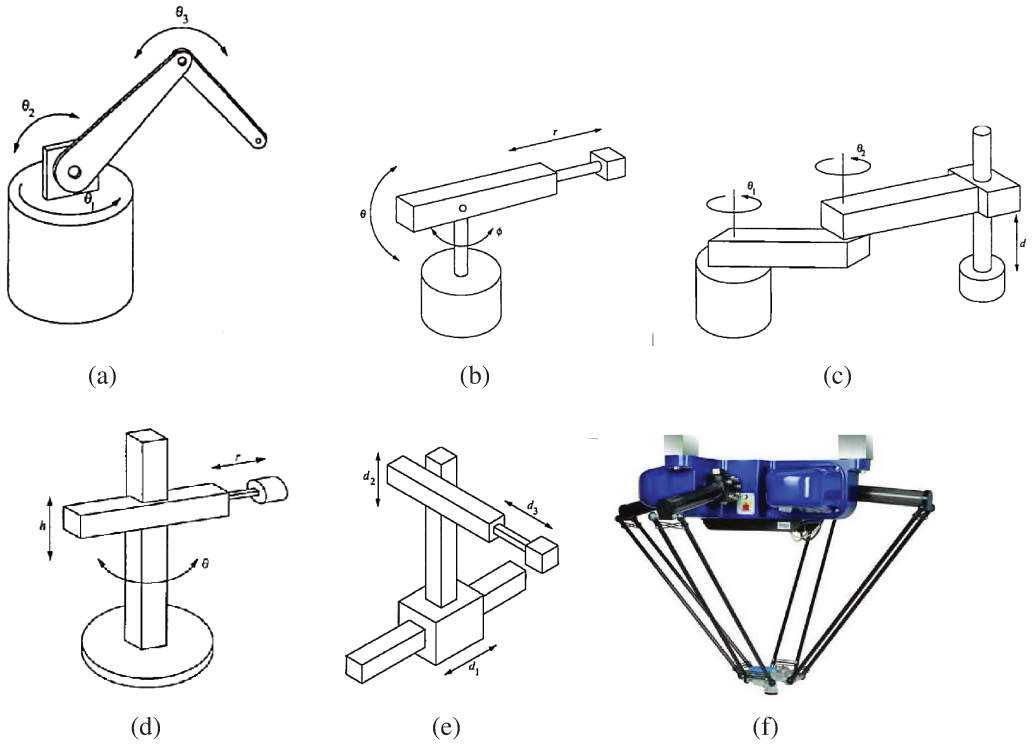


Figura 2.21 - Classes de robôs. (a) Robô de braços articulados. (b) Robô de coordenadas esféricas. (c) Robô SCARA. (d) Robô de coordenadas cilíndricas. (e) Robô de coordenadas cartesianas. (f) Robô de vínculos paralelos.
Fonte: (a) a (e) Adaptado de (LEWIS et al., 2004). (f) Adept Quattro s650H, Adept Technology, INC.

3 IMPLEMENTAÇÃO

A proposta deste trabalho é construir uma arquitetura de sistema composta por *hardware* e *software* que permita o estudo da integração de elementos de Interfaces de Usuário Naturais ao acionamento de dispositivos robóticos. Através do uso de Realidade Aumentada em associação à Interface Natural, pretende-se aumentar a consciência situacional do usuário, permitindo que através de uma representação virtual de um ambiente, um usuário seja capaz de operar um sistema robótico remotamente. Para a Interface Natural, decidiu-se utilizar uma abordagem baseada em Visão Computacional, utilizando como sensor uma câmera disponível comercialmente.

O conjunto de equipamentos utilizado foi montado como mostrado na Figura 3.1. O sensor Kinect™ foi ligado em um microcomputador, que será identificado como Servidor, onde estão instalados os *drivers* e as bibliotecas de desenvolvimento. O braço robótico foi ligado em outro microcomputador, também com os respectivos *drivers* instalados, que será identificado como Cliente. As máquinas comunicam-se através da rede local. O movimento do usuário será captado pelo Kinect™ e processado pelo Servidor. Este realizará a extração da pose do usuário e a exibição do ambiente em Realidade Aumentada. A partir da pose do usuário, são gerados os comandos que deverão acionar o robô. Os comandos são enviados pela rede para o Cliente que então acionará o robô de acordo com a posição do braço direito do usuário. O robô, então, deve reproduzir o movimento do braço do usuário.

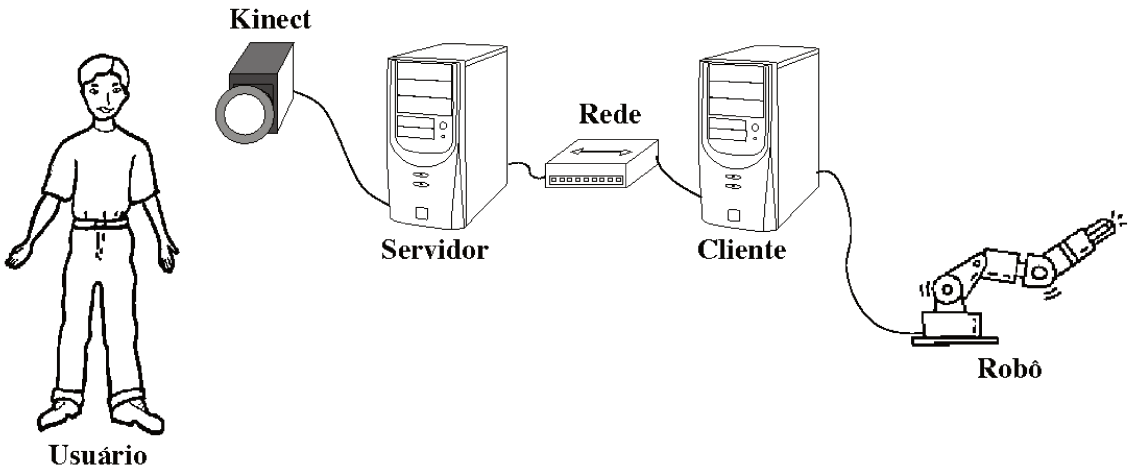


Figura 3.1 - Montagem do conjunto de equipamentos utilizados.

Como demonstração da tecnologia, será montada uma cena composta por caixas com as quais o usuário deverá interagir. O usuário será representado no mundo virtual por marcadores nas articulações correspondentes às posições do modelo gerado pelo sensor Kinect™ . Esses marcadores farão a ligação entre o mundo virtual e o mundo real. Enquanto o usuário interage com o mundo virtual, o braço robótico reproduzirá os movimentos do braço direito do usuário.

3.1 Arquitetura de *Hardware*

3.1.1 Sensor Kinect™

Para a construção de um sistema que utilize Interface Natural, é necessário utilizar alguns elementos de *hardware* específicos. Como ponto principal, é necessário utilizar um transdutor que seja capaz de reconhecer os elementos de interação que serão utilizados para acionamento, como microfones para controle por voz ou sistemas de Visão Computacional para sistemas baseados por gestos. Este último foi o escolhido para aplicação neste trabalho. Como transdutor, decidiu-se por usar um sensor que estivesse disponível comercialmente a baixo custo. O sensor Kinect™ , lançado em novembro de 2010 pela Microsoft© se encaixa nesse perfil.

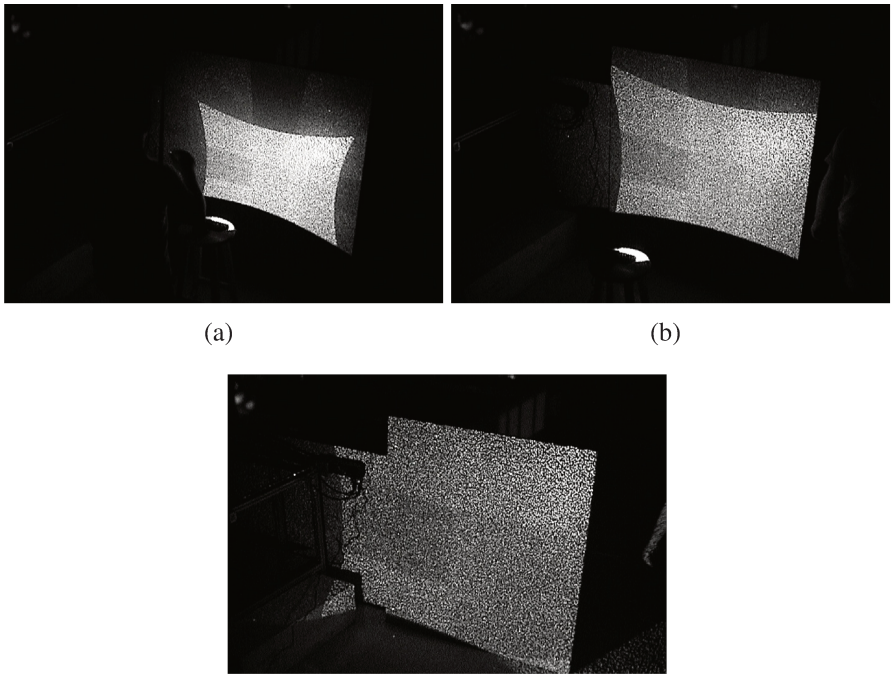
O sensor Kinect™ é um periférico para o console de jogos eletrônicos Xbox 360© da Microsoft© e foi lançado em novembro de 2010. O objetivo do periférico é realizar o rastreamento do movimento do jogador e permitir o controle de elementos de interface e dos jogos através de gestos. Também possui um conjunto de microfones para permitir o controle por voz.

O sensor funciona aplicando o método de reconstrução tridimensional por luz estruturada. O *hardware* do sensor é formado por um projetor a laser de classe 1 operando na banda do infravermelho-próximo (*near-IR*) e duas câmeras, sendo uma câmera monocromática ajustada com um filtro para a banda do infravermelho-próximo correspondente à do projetor e a outra uma câmera de vídeo a cores tradicional. O laser é projetado através de uma placa polarizadora, formando um

Figura 3.2 - Sensor Kinect™ .
Fonte: Microsoft© Corporation.

padrão de projeção em formato de uma matriz de pontos aleatoriamente distribuídos sobre o campo de visão do sensor. O padrão repete-se tanto na direção horizontal como na vertical, formando uma grade de 3×3 padrões. A projeção é captada pela câmera infravermelha e através da comparação da distorção induzida pelos objetos contidos no campo de visão do projetor com o padrão previamente calibrado, é possível obter o mapa de disparidade da cena. O mapa de disparidade é gerado através da busca dos padrões por um operador de correlação utilizando uma janela de 8×8 pixels. A partir do mapa de disparidade, é encontrada a distância de cada pixel da imagem ao sensor, produzindo um mapa de profundidade da cena. A câmera infravermelha opera em uma resolução de 320×240 pixels. Através de um processo de interpolação numérica, o mapa de profundidade resultante tem a sua resolução ampliada para 640×480. É possível observar que as câmeras têm razão de aspecto de 4×3 (horizontal x vertical) (ZALEVSKY et al., 2007; GEISS, 2010). A Figura 3.3 mostra esse padrão projetado sobre um anteparo, para melhor visualização.

O algoritmo que gera o mapa de profundidade é executado por uma solução em *hardware* especializado, tecnologia desenvolvida pela empresa israelense PrimeSense™ e licenciada pela Microsoft© . Através do mapa de profundidade, o *hardware* é capaz de gerar uma representação do usuário. Um conjunto de vinte articulações é reconhecido e tem calculada sua posição em relação ao sensor. Combinando as vinte articulações, é possível gerar um esqueleto que representa o corpo do usuário. O arranjo de microfones é capaz de determinar a posição do emissor de um som e na plataforma Xbox 360© , é usado para habilitar o uso de comandos por voz. Junto com um algoritmo de Rastreamento de Corpo desenvolvido pelos grupos de pesquisa *Microsoft Research Cambridge* e *Xbox Incubation* especificamente para emprego com o *hardware* desenvolvido para o sensor, o Kinect™ é capaz de rastrear a posição do corpo do jogador em diversas poses. O



(a) (b)

(c)

Figura 3.3 - Padrão de pontos aleatórios projetado pelo sensor Kinect™ . Os padrões foram fotografados usando uma câmera com filtro IR, com o Kinect™ posicionado a distâncias variadas do anteparo.

método de classificação utilizado é o da classificação por floresta de ótimos caminhos. Os dados de treinamento foram gerados por um conjunto de mais de 900.000 poses de pessoas com biótipos variados. Esses dados foram analisados e as árvores de decisão geradas por um *cluster* computacional contando com mil núcleos (SHOTTON et al., 2011). Este tipo de classificação tem maior custo no treinamento que na classificação. Uma vez que o treinamento é gerado, a classificação é rápida. Também foi desenvolvido um método que torna a classificação um processo por pixel, extremamente paralelizável. Assim, o processador de vídeo (GPU, *Graphics Processing Unit*) do console Xbox 360© pode ser utilizado para acelerar a estimação da pose. O resultado final é um modelo de um esqueleto humano, composto por vinte articulações com as respectivas distâncias de cada uma delas até o sensor. Um exemplo do resultado da estimação da pose e rotulação dos *pixels* pode ser observada na Figura 3.4. A partir das poses reconhecidas, uma outra camada de *software* pode, por exemplo, reconhecer gestos específicos que podem ser interpretados como comandos para controlar as interfaces do console e principalmente, seus jogos.

O conjunto de microfones embarcado no sensor Kinect™ pode ser utilizado para tarefas como reconhecimento de comandos por voz. A montagem conta com quatro microfones dispostos em

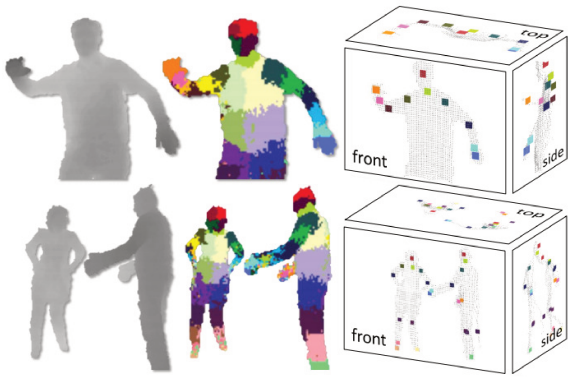


Figura 3.4 - Imagem de profundidade gerado pelo sensor Kinect™ , seguido da rotulação dos pixels e estimação das posições das juntas.
Fonte: Adaptado de (SHOTTON et al., 2011).

uma linha, a certa distância uns dos outros. Desta forma, o conjunto pode ser usado para a determinação da direção de uma fonte sonora. No console, em conjunto com os comandos por voz, a determinação da direção do som é usada para identificar se um comando foi disparado pelo jogador que está no controle do jogo. Isto evita a ativação acidental de um comando por terceiros. O poder de reconhecimento de posições do corpo do usuário aliado ao conjunto de microfones torna o Kinect™ uma ferramenta adequada para o desenvolvimento de aplicações em Interface Natural.

3.1.2 Robix™ Rascal

O braço robótico usado neste trabalho foi construído utilizando a plataforma Robix™ Rascal. O Robix™ Rascal é um robô programável, customizável, baseado em servomotores. Ele possui um conjunto de componentes intercambiáveis que permitem a montagem de diversas configurações.

O acionamento das juntas se dá por meio de servomotores. Servomotores são dispositivos eletromecânicos que, dependendo de um sinal de controle, têm o eixo posicionado em determinada posição angular. O servomotor é constituído de um motor, redução mecânica e um sistema de controle e realimentação interno, que serve para controlar a sua posição.

Os motores utilizados funcionam por meio de controle em PWM (*Pulse Width Modulation*),

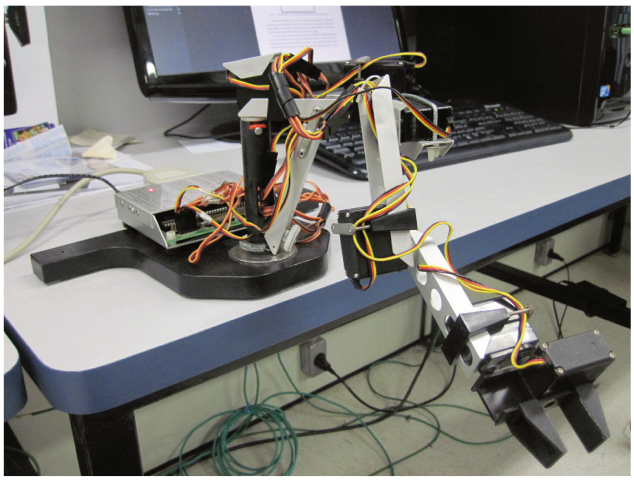


Figura 3.5 - Plataforma Robix™ Rascal. Sua estrutura modular permite várias configurações.

onde o sinal de controle deve ser um pulso, atualizado com uma taxa de 50Hz, onde a largura do pulso determina a posição do motor. Para o caso do servomotor Hitec HS-422, modelo utilizado no Rascal, o pulso deve ter a largura entre 1ms e 2ms, larguras que fazem o motor se posicionar na posição mínima e máxima, respectivamente.

O controlador que acompanha o conjunto conecta-se ao microcomputador através de uma interface paralela. Ele permite o controle simultâneo de até seis servomotores e também possui sete entradas digitais, oito entradas analógicas e duas saídas digitais. Através do uso do controlador e de uma API (*Application Programming Interface*) fornecida com o conjunto, é possível utilizar a combinação de entradas e saídas para a programação de movimentos complexos.

A configuração utilizada foi a de braço articulado, com dois movimentos, no plano vertical. A escolha desta configuração deve-se à similaridade de movimento do braço humano, com um movimento correspondendo ao cotovelo e o outro ao ombro. Não foi considerado o movimento completo do ombro, pois seria necessário uma articulação em três eixos, que não foi possível modelar com o robô utilizado.

3.2 Arquitetura de *Software*

O conjunto de *software* do sistema proposto foi desenvolvido utilizando arquitetura modular. Esta arquitetura consiste na divisão das diferentes tarefas em módulos individuais que cooperam para a produção do resultado final. Desta forma, consegue-se uma melhor divisão da carga de trabalho, permitindo tirar vantagem dos modernos processadores multinúcleos. Os módulos que executam tarefas críticas, são alocados em uma *thread* própria, executada concorrentemente com os outros módulos, diminuindo a incidência de latências e elevados tempos de espera. Por outro lado, o desenvolvimento de aplicações *multithreading* é complexo e segue um paradigma diferente da programação sequencial tradicional, além de introduzir uma série de fatores que devem ser observados para que o *software* funcione conforme esperado, como controle de concorrência e sincronização (KLEIMAN et al., 1996).

As tarefas de captura da imagem e rastreamento do usuário e a tarefa de movimentação do braço robótico foram separadas, de modo a permitir o acionamento remoto do robô. O modelo de comunicação utilizado foi o modelo Cliente-Servidor, através de comunicação em rede utilizando um protocolo personalizado especialmente desenvolvido para este fim. O desenvolvimento do protocolo possibilitou maior controle sobre a organização e formato dos dados relevantes, além de atender a capacidades específicas que não estão presentes em protocolos padronizados.

Todo os módulos do sistema foram desenvolvidos seguindo o paradigma de Programação Orientada a Objetos (POO). A Orientação a Objetos é um paradigma não só de programação, mas também de análise e projeto de *software*. Este paradigma baseia-se na composição e interação de peças de *software* chamadas objetos, que são descritos por um conjunto de atributos e que podem realizar operações definidas. Um objeto é uma instância de uma classe; a classe é definida como um modelo para a produção de objetos, que irão diferenciar-se de acordo com a variação de seus atributos. Este modelo foi escolhido por ser um padrão estabelecido e cujas características se adequam bem às exigências de programação do sistema proposto. Cada módulo do sistema foi construído como uma classe distinta.

A escolha do modelo de programação leva à escolha da linguagem, uma vez que a linguagem

adotada deve suportar o paradigma escolhido. Foi adotada a linguagem C++, que além de cumprir os requisitos do paradigma de Orientação a Objetos, também tem as vantagens de ser bastante documentada e consolidada.

Foram desenvolvidos cinco módulos, cada um cumprindo uma série de tarefas específicas. A Figura 3.6 mostra os subsistemas Servidor e Cliente, os módulos pertinentes a cada subsistema e também a colaboração e fluxo de dados entre eles.

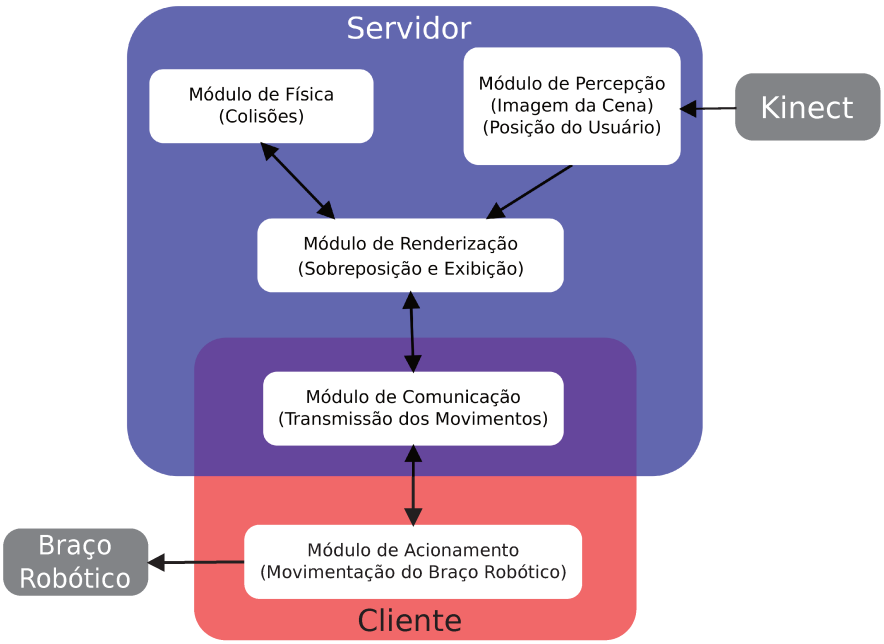


Figura 3.6 - Diagrama de colaboração e fluxo de dados do sistema proposto.

A seguir, uma breve descrição de cada módulo.

- **Módulo de Renderização** - este é o módulo central do sistema, já que o ponto de entrada e de saída do processamento bem como o laço principal são executados neste módulo. Sua principal finalidade é a geração dos quadros de vídeo (renderização). Outras atribuições são gerenciar e consolidar as informações dos módulos de Física e Percepção e compor a cena em Realidade Aumentada para a renderização final. Através do Módulo de Comunicação, os dados do posicionamento do usuário rastreado gerados pelo Módulo de Percepção são transferidos ao Módulo de Acionamento, que se encarrega de traduzir as posições em comandos para o braço robótico.
- **Módulo de Percepção** - este módulo tem como tarefa gerenciar o dispositivo de cap-

tura de imagens, o sensor Kinect™ . Com os dados gerados pelo sensor, é possível reconstruir tridimensionalmente a cena captada pelo aparelho. Através de um conjunto de *driver* e biblioteca de funções, o Kinect™ for Windows© SDK (*Software Development Kit*), é possível acessar os fluxos de dados e utilizá-los. O SDK também provê um conjunto de funções capaz de reconhecer um ser humano como usuário e rastrear seus movimentos, gerando uma representação deste. A partir desta representação, o módulo extrai os dados do posicionamento do usuário e os passa para o Módulo de Renderização, que realiza a formatação destes dados e os repassa para o Módulo de Acionamento através do Módulo de Comunicação.

- **Módulo de Física** - este módulo é responsável por calcular os estados de movimento dos objetos presentes na cena que irão interagir com o usuário. Dadas a massas dos objetos e as forças que atuam sobre eles, o módulo é capaz de calcular os estados de movimento (rotação e deslocamento) de cada objeto da cena, a cada quadro. O Módulo de Renderização então usará estas informações para desenhar o objeto na posição e com as rotações adequadas. O uso deste módulo permite a criação de uma cena em Realidade Aumentada com maior nível de interatividade.
- **Módulo de Comunicação** - este módulo tem a tarefa de gerenciar a comunicação entre os módulos que rodem em máquinas diferentes. A comunicação é feita através da infra-estrutura de rede utilizando um protocolo personalizado construído sobre UDP. Um sistema composto pelos módulos de Renderização, Percepção, Física e Comunicação atua como servidor e outro sistema constituído pelos módulos de Comunicação e Acionamento atua como cliente.
- **Módulo de Acionamento** - este módulo é responsável por gerar os comandos para a movimentação do braço robótico. O braço robótico utilizado possui uma biblioteca de funções capazes de enviar comandos para a unidade de controle. As funções estão disponíveis em uma DLL (*Dynamic Link Library*). O sistema comunica-se com o controlador através de uma interface paralela com conector DB-25 padrão. As informações do usuário produzidas pelo Módulo de Percepção são recebidas e processadas, gerando os comandos correspondentes que irão mover o braço robótico.

3.2.1 Módulo de Renderização

Concepção

A tarefa principal do Módulo de Renderização é criar, através de Computação Gráfica, um fluxo de vídeo através da descrição de uma cena. A cena será construída com o vídeo captado do usuário, ao qual serão sobrepostos elementos virtuais com os quais ele deverá interagir. Para agilizar o desenvolvimento, decidiu-se usar um *framework* de renderização que permitisse a construção de aplicações 3D. Como o foco do trabalho não está na renderização, mas na integração de tecnologias, o uso do *framework* evita o trabalho de desenvolver rotinas de Computação Gráfica, que já são bastante conhecidas. Adicionalmente, um *framework* provê um nível de abstração maior, já que oculta as operações fundamentais, o que permite que a programação seja feita de forma mais fluida. O programador concentra-se na descrição e manipulação da cena, todas as operações gráficas ficam sob responsabilidade da biblioteca.

Framework OGRE

Para a construção do módulo de renderização foi empregado o *framework* de computação gráfica de código aberto OGRE (*Object-oriented Graphics Rendering Engine*). O OGRE surgiu em 2001 com o objetivo de prover um conjunto de rotinas computacionais que permitissem o desenvolvimento ágil de aplicações de computação gráfica, tanto em 2D quanto em 3D. O OGRE é escrito na linguagem C++ seguindo o paradigma de Programação Orientada à Objetos e tem como características ser compatível com os dois maiores motores de renderização em uso atualmente (OpenGL© e DirectX©), bastante robusto e bem documentado. Suporta uma grande variedade de construções e efeitos, além de ser extensível através de uma estrutura de suporte a *plugins*. Por ser publicado como código aberto, o OGRE pode ser modificado de acordo com as necessidades do programador.

Funcionamento

Estrutura de Controle Aplicação

Outra tarefa importante do Módulo de Renderização, uma vez que este é o módulo central do sistema, é a organização do fluxo de tarefas do processo principal. O fluxo de tarefas foi organizado como uma máquina de estados finitos, mostrada na Figura 3.7.

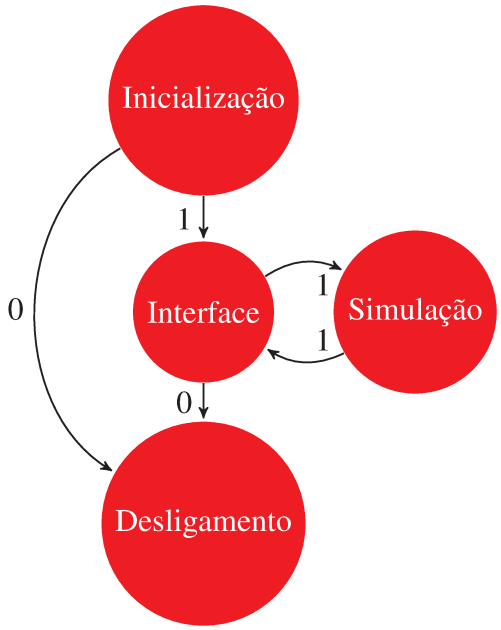


Figura 3.7 - Máquina de estados do processo principal.

Ao ser ativado, o sistema entra no estado de Inicialização. Neste estado, é realizada a alocação da memória para as estruturas estáticas e a inicialização dos módulos auxiliares. O grafo de cena utilizado pelo sistema de renderização do OGRE também é inicializado. Caso a inicialização seja bem sucedida, a aplicação muda para o estado de Interface. Caso algum erro impeça o funcionamento do sistema, a inicialização é interrompida e a aplicação passa para o estado de Desligamento, chamando as rotinas de limpeza e saindo da função principal. No estado de Interface é exibida uma tela com comandos. A partir desta tela, o usuário pode encerrar a aplicação ou passar para o estado de Simulação. Os comandos podem ser dados pelo dispositivo apontador ou por atalhos de teclado. No estado de Simulação, acontece a atualização das informações produzidas

pelos módulos auxiliares e renderização do resultado.

A transição entre os estados é gerenciada através de requisições. Ao se fazer uma requisição de mudança de estado, o mecanismo gerenciador verifica as regras de transição e realiza ou nega a mudança requisitada. Todas as transições de estado são ativadas por uma ação do usuário, exceto a transição do estado de Inicialização para o estado de Interface ou Desligamento. Esta transição ocorre automaticamente para o estado correspondente de acordo com o sucesso ou a falha das rotinas de inicialização do sistema.

Laço Principal

O processamento é controlado por um laço, onde são requisitadas as tarefas para cada módulo assim que são necessárias. Primeiro, é realizada a captura dos eventos de entrada do usuário no teclado e no dispositivo apontador. Em seguida é verificado no mecanismo gerenciador qual o estado ativo. Se o estado de Interface estiver ativo, o quadro de renderização exibe a página de interface, onde o usuário pode selecionar os comandos disponíveis. Caso seja o estado de Simulação, é realizada a mudança para a exibição de vídeo. A seguir, é verificada se houve uma notificação de novos dados do sistema de vídeo, do gerador de profundidade e do sistema gerador de esqueleto no Módulo de Percepção. Em caso de notificação positiva, é realizada a cópia dos dados, seguida da atualização do grafo de cena. Caso um novo esqueleto seja produzido, o Módulo de Comunicação é notificado da existência de novos dados e o *buffer* de envio é atualizado. O próximo passo, é a atualização dos estados de movimento dos elementos da cena. O tempo decorrido desde o último quadro é calculado e passado ao Módulo de Física, que realiza a atualização dos movimentos e repassa as posições resultantes para o Módulo de Renderização. Por último, o quadro da cena é gerado e a cena é renderizada para o vídeo. O laço retorna ao início e executa novamente a sequência, caso o estado de Desligamento não tenha sido solicitado. Durante a transição do estado de Simulação para o estado de Interface, a contagem do tempo de execução é pausado. Isso garante que a simulação não continue a ser executada e o estado seja salvo. O Algoritmo 3.1 exemplifica este processo.

Algoritmo 3.1 Laço principal da aplicação.

Inserir: $estado, tempo$

```
1:  enquanto estado != DESLIGAMENTO faça
2:      se Janela fechada então
3:          Requisita estado(DESLIGAMENTO)
4:      fim se
5:      Captura estado de entrada;
6:      se estado == INTERFACE então
7:          Apresenta a tela de interface;
8:      senão se estado == SIMULAÇÃO então
9:          Apresenta a tela de simulação;
10:         se Nova captura de vídeo então
11:             Atualiza a textura do vídeo;
12:         fim se
13:         se Nova captura de profundidade então
14:             Atualiza a textura de profundidade;
15:         fim se
16:         se Nova captura de esqueleto então
17:             Atualiza a textura do esqueleto;
18:             Extrai posições das juntas;
19:             Envia posições para o cliente;
20:         fim se
21:         tempo ⇐ tempo desde o último quadro;
22:         Atualiza o estado da simulação de Física(tempo);
23:     fim se
24:     Renderiza o quadro de vídeo;
25:     tempo ⇐ 0;
26: fim enquanto
27: Desliga o sistema;
```

3.2.2 Módulo de Percepção

O Módulo de Percepção tem o objetivo de capturar as imagens do mundo real através da câmera de vídeo do sensor e fornecer estas imagens ao módulo de Renderização para a composição da cena em Realidade Aumentada. Este módulo também fornece as informações do posicionamento do usuário através do mapa de profundidade gerado pelo sensor. O Módulo de Renderização recebe as coordenadas e realiza a transformação necessária para alinhar o sistema de coordenadas do esqueleto com o sistema de coordenadas do mundo.

Concepção

Um dos maiores problemas em aplicações de Realidade Aumentada é reconstruir a informação de profundidade da cena do mundo real. Essa informação é importante porque permite estimar o posicionamento dos objetos captados na cena e utilizando estas posições, justapor os objetos virtuais, fundindo o mundo real e o mundo virtual. Esta informação é necessária para o processo de *clipping*, ou seja, o recorte de partes do objeto virtual que estejam encobertos por objetos reais na cena composta. As aplicações mais comuns de Realidade Aumentada realizam a estimação da pose (posição e rotações) através de marcadores conhecidos através dos parâmetros de calibração da câmera. Os objetos virtuais são posicionados sobre a cena na posição e com a orientação encontrada. Com essa técnica, não se obtém a informação de profundidade da cena real e portanto os objetos virtuais são sempre renderizados no primeiro plano, encobrindo partes da cena real.

Existem algumas técnicas que possibilitam a reconstrução tridimensional de uma cena sendo a mais estudada a da Visão Estéreo. Entretanto, a técnica de Visão Estéreo apresenta alguns desafios significativos para aplicações de Realidade Aumentada. Em primeiro lugar, ela possui um elevado custo computacional, sendo muito difícil atingir resultados satisfatórios em tempo real. Também é dependente dos parâmetros intrínsecos da câmera utilizada, sendo muito sensível à variação destes. Para contornar este problema, sugere-se o uso de câmeras paramétricas, que têm parâmetros

conhecidos e estáveis, porém custo elevado. A montagem da câmera (ou câmeras) deve garantir que um dos eixos óticos estejam alinhados. Isso garante que pixels correspondentes estejam na mesma linha. Caso contrário, o espaço de busca dos pixels correspondentes torna-se bidimensional, o que faz a complexidade do algoritmo aumentar consideravelmente e consequentemente, o tempo de processamento também torna-se muito alto. O problema do espaço de busca bidimensional pode ser resolvido utilizando-se de Geometria Epipolar, mas ainda assim o tempo resultante é alto demais para permitir uso em aplicações de Realidade Aumentada. Esse é o principal motivo que inviabiliza o uso da visão estéreo em tempo real.

O uso de câmeras de profundidade resolve esta questão, mas até o lançamento do Kinect™ , não existia uma câmera de profundidade capaz de entregar dados em tempo real ou de lidar com formas humanas variadas em movimentos complexos (SHOTTON et al., 2011). Por esse motivo, decidiu-se usar o sensor Kinect™ como meio de captura dos movimentos do usuário.

Kinect™ SDK

O principal interesse em usar o Kinect™ está em sua capacidade de processar a cena e entregar simultaneamente ao quadro de vídeo, o mapa de profundidade da cena. Essa capacidade permite construção de aplicações em tempo real. Como o mapa é gerado em *hardware*, o tempo de processamento que seria utilizado para gerar o mapa de profundidade por visão estéreo ou outra técnica, fica livre para outras aplicações. O baixo custo do sensor aliado à boa precisão de seus dados motivaram diversos grupos de pesquisa a utilizá-lo em seus trabalhos. Tais pesquisas vão desde desenvolvimento de interfaces naturais baseadas em gestos a navegação robótica autônoma, passando por mapeamento e reconstrução tridimensional e até mesmo aplicações médicas (SANTOS et al., 2011; ENGELHARD et al., 2011; RAMEY et al., 2011; CHANG et al., 2011).

Neste trabalho, o acesso às funções do dispositivo é feito através do Kinect™ SDK, um conjunto de bibliotecas e *drivers* fornecidos pelo próprio fabricante para uso do Kinect™ no computador. Existem bibliotecas para as linguagens C/C++, C# e Visual Basic através do *framework* .NET.

O lançamento de uma biblioteca pelo próprio fabricante do dispositivo é um fator que contribui para a confiabilidade na plataforma. A biblioteca é de uso gratuito para fins não-comerciais. Há a previsão de lançamento de uma versão comercial desta biblioteca em data ainda indefinida, assim como o lançamento de uma versão do Kinect™ adaptada para uso em microcomputadores. Existem também outras alternativas de bibliotecas de código aberto, como a *OpenNI*, mantida pela fabricante do sistema de visão utilizado no Kinect™ , a Prime Sense™ e a biblioteca *freenect* desenvolvida pela comunidade *Open Source*. A biblioteca oficial foi escolhida por oferecer acesso a algumas funções específicas que são de propriedade intelectual da Microsoft© e não estão acessíveis a outras bibliotecas.

Funcionamento

A Figura 3.8 mostra o diagrama de blocos do fluxo de dados do Módulo de Percepção.

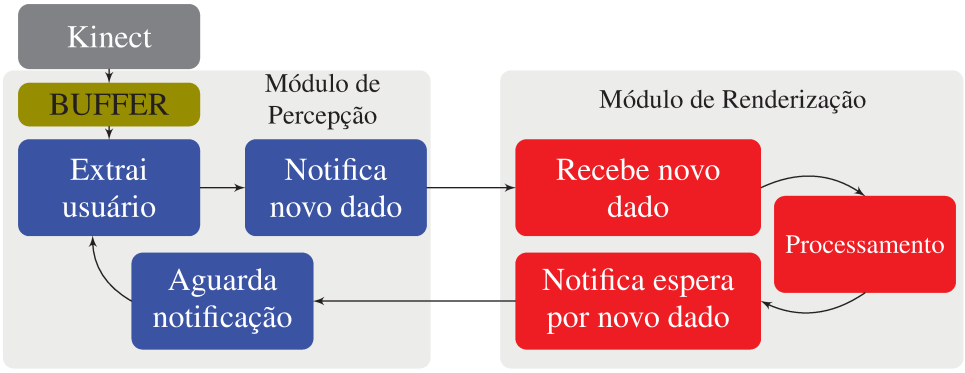
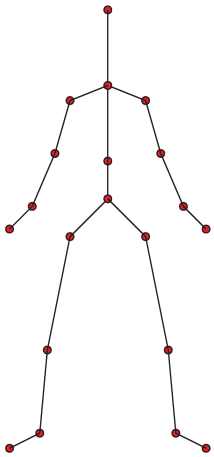
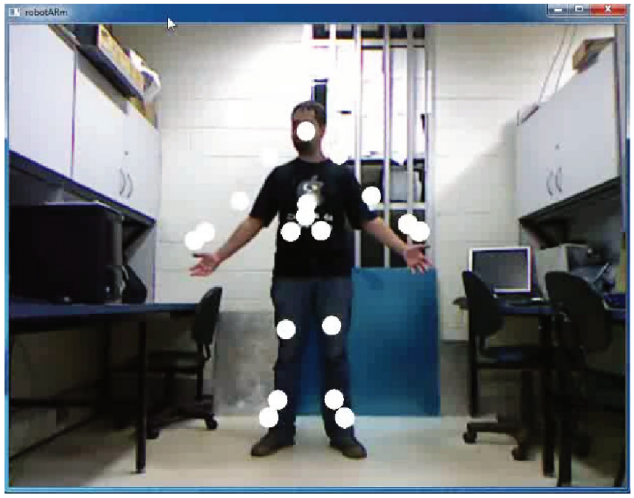


Figura 3.8 - Fluxo de dados do Módulo de Percepção.

O sensor notifica ao Módulo de Percepção a disponibilidade de novos dados. O módulo recebe estes dados e notifica o Módulo de Renderização, que então atualiza suas estruturas de dados. Cada processo de captura é monitorado por uma *thread* diferente, o que garante o paralelismo e a independência dos fluxos de dados. Desta forma, o processo de renderização é mantido independente da realimentação dos dados, o que permite que a renderização seja executada na maior velocidade possível, mantendo taxas de quadros por segundo (fps, *frames per second*) independentes. Para representar o usuário no mundo virtual foram utilizadas esferas como marcadores, que



(a)



(b)

Figura 3.9 - Modelo do esqueleto, mostrando as vinte articulações em sobreposição com a imagem capturada.

foram posicionadas nas posições correspondentes às vinte articulações do esqueleto modelado. As posições são calculadas pelo sensor e passadas ao Módulo de Renderização que transforma as coordenadas para o espaço da aplicação e renderiza a animação. A Figura 3.9 mostra o modelo do esqueleto e as esferas que marcam suas posições na imagem capturada.

3.2.3 Módulo de Simulação de Física

O Módulo de Física tem a tarefa de calcular os estados de movimento dos objetos virtuais e verificar as colisões entre os objetos e a representação do usuário e dos objetos entre si. Para atingir este objetivo, foi utilizado um motor de física. Um motor de física pode ser definido como uma ou mais bibliotecas de *software* capazes de prover simulação de certos sistemas físicos, tradicionalmente dinâmica de corpos rígidos (incluindo detecção de colisão), dinâmica de corpos não-rígidos e dinâmica dos fluidos.

Atualmente, existem muitas opções de motores de Física, cuja precisão e robustez da simulação variam de acordo com a aplicação para qual serão usados. Alguns têm foco na precisão das simulações enquanto outros focam em tempo de processamento. Geralmente, os motores físicos usados em jogos eletrônicos têm modelos mais simples, mas são capazes de processar os dados no curto espaço de tempo exigido para este tipo de simulação. Os motores de física usados pelo meio acadêmico, pela indústria e pelo cinema têm modelos mais elaborados e precisos, porém uma simulação pode demorar várias horas para ser processada. A escolha do motor de física deve levar essas exigências em consideração.

Aliado à capacidade de recuperar as informações do posicionamento do usuário na cena real, é possível criar uma experiência de Realidade Aumentada mais interativa, onde um usuário pode tocar e manipular objetos virtuais de forma natural.

Motor de Física Bullet

Neste trabalho, o objetivo é construir uma aplicação de Realidade Aumentada. As exigências são bastante próximas das de um jogo eletrônico; ambos lidam com um fluxo de vídeo no qual vários quadros devem ser gerados a cada segundo. Caso o tempo dispensado no processamento da simulação de cada quadro seja maior que o tempo gasto para renderizá-lo, a taxa de quadros

por segundo cai drasticamente, resultando em uma experiência de visualização ruim. Uma boa experiência visual se dá quando os quadros são atualizados a uma taxa de no mínimo, 30 fps. A escolha de um motor de física utilizado para jogos parece então, bastante natural. Eles oferecem boa precisão na simulação de colisões e movimentos dinâmicos, a taxas interativas.

O motor de física utilizado neste trabalho foi a biblioteca *Bullet Physics Library*. A Bullet é um motor de física profissional com funções de detecção de colisões e dinâmica de corpos rígidos e não-rígidos, desenvolvida em C++ e de código aberto. É uma biblioteca bastante robusta e capaz de atender os requisitos de simulação em tempo real com boa precisão. Está disponível para várias plataformas e é capaz de realizar diversos tipos de simulações. A biblioteca é facilmente integrável com diversos renderizadores. Suas funções foram projetadas para funcionar de forma independente do mecanismo de renderização. Desta forma, não há transferência de custo de processamento para o renderizador. A biblioteca também incorpora técnicas modernas de programação para processadores multinúcleos. Desta forma, ela consegue aproveitar a capacidade de processamento paralelo destes sistemas para otimizar o desempenho dos resolvedores que ela utiliza para o cálculo dos estados de movimento através de *multithreading*. De acordo com Boeing e Bräunl (2007), o motor de física Bullet possui o melhor desempenho entre as opções de código aberto, inclusive mostrando-se superior a alguns similares comerciais.

O motor de renderização utilizado, o OGRE, possui capacidade de extensão através de um sistema de *plugins*. Os *plugins* podem ser usados para adicionar novas funcionalidades ou permitir integração com bibliotecas externas. Este sistema tem o objetivo de possibilitar a utilização e integração de outras funções que estendam as capacidades do renderizador. A comunidade de desenvolvedores e utilizadores do OGRE preocupa-se com essa característica e provê interfaces para vários projetos afins. A integração do motor de física Bullet com o motor de renderização OGRE foi bastante facilitada pelo uso do *plugin OgreBullet*. Este *plugin* provê uma interface para acesso às funções da Bullet através do encapsulamento destas através das estruturas e objetos nativos do OGRE, tornando muito prático o desenvolvimento de aplicações que utilizem os dois projetos.

Funcionamento

O fluxo de dados do Módulo de Física é mostrado na Figura 3.10. A Bullet classifica os objetos em Dinâmicos, Cinemáticos e Estáticos. Os objetos dinâmicos são aqueles cujo movimento é controlado pelo motor de física, como resultado das forças que interagem sobre eles. Os objetos Cinemáticos são aqueles cujo movimento não será calculado pela biblioteca, mas gerados pelo usuário. Entretanto, estes objetos influenciam o movimento dos corpos dinâmicos, gerando colisões e forças atuantes. Por fim, os objetos Estáticos são os que não se movimentam em nenhum momento, mas produzem forças resultante da colisão com objetos dinâmicos, como exemplo, o chão.

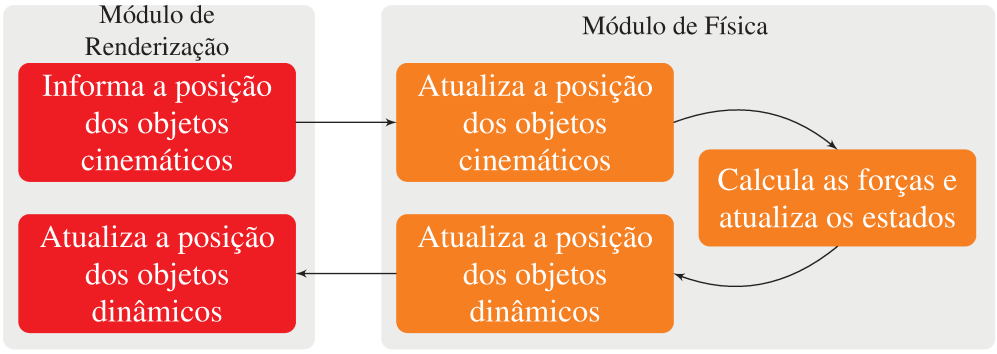


Figura 3.10 - Fluxo de dados do Módulo de Simulação de Física.

Na cena construída, as esferas que marcam os pontos das articulações são classificados como objetos cinemáticos. Então, o Módulo de Renderização deve informar as posições destas para o Módulo de Física. As posições são fornecidas pelo Módulo de Percepção através do sensor.

O Módulo de Renderização também informa o tempo decorrido desde a exibição do último quadro, para que o Módulo de Física possa atualizar o estado dos objetos dinâmicos. A estratégia de atualização utilizada foi a de incrementos de passo de tempo fixo. Os passos de tempo foram fixados a $1/60$ segundos. Como a taxa de renderização não é constante, uma estratégia de controle de tempo adaptativo sobre passos fixos foi implementada. Nesta estratégia, a renderização atua como produtor de tempo e a atualização do motor de física atua como consumidor de tempo. A cada passagem do laço de renderização, o tempo gasto até atingir a etapa de atualização do motor de física é acumulado em uma variável. Um laço interno cuida de atualizar o motor de física em

"pacotes" de $1/60$ segundos. O tempo acumulado é decrementado até que seja menor que o passo de tempo definido. Esta pequena "sobra" de tempo é guardada e acumulada com o tempo gasto na próxima passagem do laço. Caso o tempo gasto na renderização seja menor que o passo de tempo definido para atualização do motor de física, não é feita a atualização e o tempo é acumulado com as próximas passagens até que se "produza" tempo suficiente para ser "consumido" pelo motor de física. Essa estratégia garante a independência das rotinas de renderização e simulação de física, mantendo seus funcionamentos sincronizados e evitando o fenômeno de *aliasing* no tempo. Este fenômeno acontece quando o tempo gasto na simulação é maior que a taxa de renderização, o que faz com que ocorra um desalinhamento entre a simulação e a renderização. Usar passos de tempo variáveis também prejudica a fluidez da renderização, já que um quadro que demore mais para ser renderizado causará saltos nos movimentos dos objetos. A estratégia de tempo adaptativo com atualização a passos de tempo fixo garante que o movimento seja simulado de forma fluida, mesmo com a variação da taxa de quadros. É importante destacar que o passo de tempo deve ser definido em função do tempo médio de renderização, de forma empírica ou estimada (FIEDLER, 2006).

Internamente, o motor de física divide o tempo de simulação requisitado em mais passos, com a finalidade de aumentar a resolução da simulação. As forças são resolvidas primeiro para os objetos cinemáticos que tenham se movimentado. Em seguida são resolvidos os objetos dinâmicos aplicando as forças resultantes dos movimentos dos objetos cinemáticos, caso existam. O resultado é devolvido para o Módulo de Renderização, que então atualiza as posições e orientações de cada objeto dinâmico existente na cena.

3.2.4 Módulo de Comunicação

O Módulo de Comunicação tem a tarefa de gerenciar as comunicações entre os subsistemas. Na arquitetura desenvolvida, o modelo de comunicação utilizado foi o modelo Cliente-Servidor, onde uma aplicação composta pelos módulos de Comunicação e Acionamento atuará como cliente e outra aplicação composta pelos módulos de Percepção, Física, Renderização e Comunicação atuará como servidor. O cliente solicita uma conexão com o servidor. Caso a conexão seja estabelecida,

o servidor passará a enviar os pacotes de dados com as posições absolutas das vinte articulações informadas pelo Kinect™ . Ao receber os dados, o cliente então deverá decodificá-los e em seguida, comandar o movimento do braço robótico.

Para a comunicação entre os subsistemas foi utilizado um protocolo personalizado, que pudesse atender as demandas de atualização rápida do cliente, mas que oferecesse algum controle de fluxo e que fosse orientado à conexão.

Modelo Cliente-Servidor

O modelo de comunicação utilizado foi o modelo Cliente-Servidor. O modelo Cliente-Servidor é um modelo computacional que classifica as camadas de *software* que irão se comunicar em clientes e servidores. Os clientes e os servidores são interligados entre si através de uma interface de comunicação, que pode ser, por exemplo, uma rede de computadores. Cada instância de um cliente envia requisições de dados para algum dos servidores conectados e espera pela resposta. O servidor que estiver disponível, aceita essa requisição e após realizar algum processo, retorna ao cliente os dados requisitados.

Este modelo é bastante utilizado em sistemas de computação distribuída, onde vários computadores autônomos compartilham recursos através de uma rede para completar uma tarefa comum. Esta abordagem permite a colaboração de sistemas heterogêneos. Os servidores se encarregam de consolidar os resultados a medida que os clientes vão concluindo as tarefas previamente designadas.

Várias razões contribuem para a decisão de dividir o sistemas em dois subsistemas em modelo Cliente-Servidor. Uma razão foi garantir a distribuição da carga computacional entre os subsistemas, mantendo módulos criticamente conectados no mesmo subsistema, garantindo baixo tempo de resposta na comunicação entre eles. Outra razão foi deixar o sistema flexível, uma vez que as informações do servidor podem ser utilizadas por clientes diferentes, permitindo o uso do sistema como mediador do sensor Kinect™ . Para variar a aplicação, basta implementar um novo cliente

que receba as informações e as interprete da forma adequada, não sendo necessária a reimplementação de todo o sistema. Outra razão, decorrente da anterior, é que deixa-se o sistema com melhor manutenibilidade e com possibilidade de expansão, uma vez que novas capacidades podem ser acrescentadas sem grande dificuldade. Caso seja necessária uma mudança no protocolo, a arquitetura modular permite flexibilidade, uma vez que apenas a reimplementação do Módulo de Comunicação é suficiente, desde que as informações passadas aos módulos aos quais ele se liga não sofram alterações.

Arquitetura do Servidor

A aplicação-servidor desenvolvida foi projetada para aceitar a conexão de apenas um cliente. Essa decisão foi tomada para manter a simplicidade da implementação, uma vez que não serão utilizados mais de um cliente nos testes. Seu funcionamento é exemplificado na Figura 3.11.

A aplicação-servidor também tem a capacidade de encerrar uma conexão que deixe de responder. Desta forma, evita-se a necessidade de reiniciar o programa caso o cliente desconecte-se por problemas na rede ou desligamento. O envio dos dados ocorre cada vez que o quadro do esqueleto é atualizado. Quando não existem novos dados, o servidor envia uma mensagem de controle, para manter a conexão aberta. Uma queda na conexão é representada pelo não recebimento da confirmação de recebimento pelo cliente, dentro de um certo limite. Decorrido este limite, o monitor de conexão encerra a comunicação e o servidor é colocado novamente no estado de espera por conexões.

Arquitetura do Cliente

A aplicação-cliente, ao ser inicializada, tem como primeira ação tentar conectar-se ao servidor. Caso não exista servidor disponível, o cliente espera um tempo predeterminado e então tenta

Figura 3.11 - Fluxograma do Servidor.

uma nova conexão. A aplicação permanece nessa rotina até que consiga uma conexão ou até o programa ser encerrado. Quando ocorre uma conexão, ele passa a responder às mensagens de controle do servidor, até que um novo dado seja enviado. Quando um novo dado chega, é feita uma notificação ao Módulo de Acionamento, que então copia estes dados para sua utilização. O fluxograma mostrado na Figura 3.12 mostra como o processo do cliente foi modelado.

Caso ocorra uma queda na conexão, um monitor de conexão reinicia o processo e o cliente passa a tentar conectar-se novamente a um cliente. Da mesma forma como é feito com o servidor, o cliente permanece em um desses dois estados, tentando conectar-se ou recebendo dados, durante toda a execução do programa. O monitoramento da conexão é feito através do recebimento dos pacotes enviados pelo servidor. Cada pacote recebido é confirmado. A queda de conexão é determinada pelo não recebimento de nenhum dado dentro de um tempo limite.

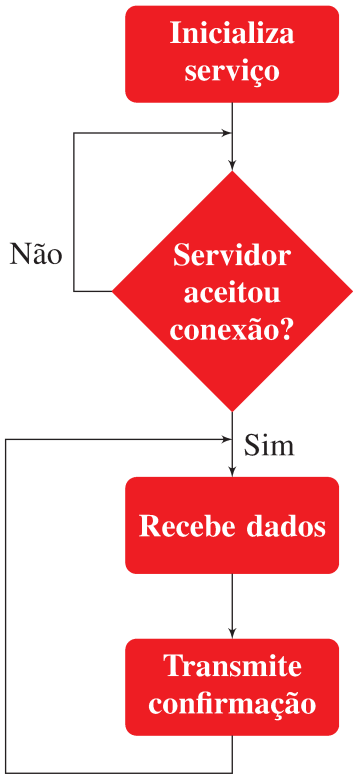


Figura 3.12 - Fluxograma do Cliente.

Protocolo Desenvolvido

Como esse trabalho é uma aplicação que se baseia em visão para movimentar o braço robótico, é mais importante obter o dado mais recente do que obter todos os dados. Com esse requisito em mente, foi escolhido como base o protocolo UDP. Ele não provê algumas funcionalidades desejáveis, como controle de fluxo ou de conexão além de não prover nenhum controle da integridade dos dados. Por isso, decidiu-se implementar um protocolo próprio, tendo o UDP como base, que provesse um mecanismo de controle de conexão e de fluxo mais simplificado que o oferecido pelo TCP, que garantisse maior robustez à comunicação porém mantendo o desempenho do UDP. O controle de perda de pacotes e reenvio não foi implementado por não ser estritamente necessário. Devido a taxa de atualização dos dados ser muito rápida, um pacote perdido pode ser descartado e o próximo utilizado sem prejuízo ao funcionamento do sistema. A Figura 3.13 mostra a pilha de protocolos utilizada.

O protocolo desenvolvido pertence à camada de Aplicação do modelo de referência TCP/IP.

Figura 3.13 - Pilha de Protocolos utilizada.

Ele provê dois serviços básicos, enviar e receber dados. Um sistema de controle de fluxo foi desenvolvido, de modo a evitar que uma deterioração nas condições da rede afete a comunicação. Este controle é feito através de uma métrica simples, mas bem significativa, que a medida do tempo decorrido entre o envio de um pacote e o recebimento de sua confirmação. Este tempo é denominado de *Round Trip Time* (RTT). O monitor de fluxo observa o RTT médio. Quando este aumenta, um fator de penalidade é adicionado à conexão. Quando este fator ultrapassa um certo limite, as condições da rede são consideradas ruins. Se o RTT começa a baixar, a penalidade da conexão é diminuída, até atingir outro valor limite. Este valor marca o ponto onde as condições da rede são consideradas boas. O monitor de fluxo seleciona entre duas taxas de transmissão diferentes, uma taxa mais lenta para as condições ruins e outra mais rápida para condições boas.

O protocolo controla o recebimento dos pacotes através de um identificador que fica registrado em seu cabeçalho. O protocolo UDP, na camada de Transporte, não faz distinção entre os datagramas que envia. O identificador de cabeçalho evita que o sistema processe pacotes de outros protocolos que coincidentemente sejam endereçados ao mesmo destinatário. O cabeçalho também identifica os dois tipos de pacotes utilizados, um pacote de dados e um pacote de controle. O pacote de controle tem a função de manter a conexão aberta. O pacote de dados tem a função de transportar os dados das posições das articulações entre os subsistemas. O pacote de dados foi modelado como uma cadeia de caracteres, utilizando quatro caracteres para cada articulação. O primeiro caractere informa o identificador da articulação. Os três próximos conjuntos de caracteres guardam as coordenadas x, y e z de cada coordenada, com um caractere de quebra de linha (“\n”) separando cada dado. A Figura 3.14 mostra como é organizado o pacote de dados e a Tabela 3.1 mostra a lista de identificadores e a articulação correspondente.

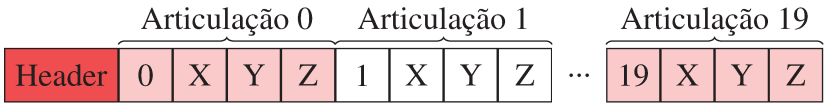


Figura 3.14 - Pacote de dados do protocolo desenvolvido.

Tabela 3.1 - Identificadores e articulações utilizado no modelo de esqueleto.

Identificador	Articulação
0	Centro do quadril
1	Centro da coluna
2	Centro dos ombros
3	Cabeça
4	Ombro Esquerdo
5	Cotovelo Esquerdo
6	Pulso Esquerdo
7	Mão Esquerda
8	Ombro Direito
9	Cotovelo Direito
10	Pulso Direito
11	Mão Direita
12	Quadril Esquerdo
13	Joelho Esquerdo
14	Tornozelo Esquerdo
15	Pé Esquerdo
16	Quadril Direito
17	Joelho Direito
18	Tornozelo Direito
19	Pé Direito

3.2.5 Módulo de Acionamento

O Módulo de Acionamento cumpre a tarefa de gerar os comandos que irão movimentar o braço robótico. As posições das vinte articulações reconhecidas são geradas pelo sensor Kinect™ e transmitidas pelo Módulo de Percepção até o Módulo de Renderização. Este as repassa para seu Módulo de Comunicação que as envia para o Módulo Comunicação conectado ao Módulo de Acionamento. O Módulo de Acionamento recebe os dados e realiza o cálculo dos movimentos que devem ser executados pelo braço robótico. O Módulo de Acionamento comunica-se com o braço robótico através do controlador do robô. Este, por sua vez, é ligado ao sistema através da porta paralela do microcomputador.

Interface de Acionamento

O robô utilizado possui uma linguagem de *scripts* própria, que pode ser utilizada através de um *software* fornecido pelo fabricante. Este modo de utilização é útil para o aprendizado e reconhecimento das características e capacidades do robô. Alternativamente, é possível acessar o controlador diretamente utilizando uma API acessada através de uma DLL também fornecida pelo fabricante. Através da inclusão da biblioteca, é possível utilizar um conjunto de funções nas linguagens C ou Visual Basic. Estas funções permitem enviar comandos para o controlador do robô, permitindo o uso da plataforma com *software* personalizado. Esta foi a forma utilizada neste trabalho.

Determinação dos Movimentos

Foi utilizado o movimento do braço direito do usuário para movimentar o braço robótico. O robô deverá reproduzir os movimentos de elevação do cotovelo e do ombro do usuário, apenas em um plano. Esses dois movimentos foram utilizados em função dos movimentos possíveis de serem efetuados pela plataforma robótica disponível. Como o robô foi modelado como braço articulado, são necessários dois ângulos, o do cotovelo e do ombro para o acionamento do robô.

As posições das juntas são representadas por um conjunto de coordenadas na forma (x,y,z). Essas coordenadas representam um vetor entre a origem e o ponto da articulação. Para encontrar os segmentos de reta que representam o braço e o antebraço do usuário, devemos realizar uma operação de diferença vetorial. Para o braço, o segmento será dado pela diferença entre as articulações Cotovelo Direito-Pulso Direito. O antebraço será dado pela diferença entre os pontos Ombro Direito-Cotovelo Direito. De posse desses vetores, podemos calcular o ângulo entre o braço e o antebraço através do produto interno entre estes. O mesmo cálculo é realizado para o vetor do antebraço em relação a um vetor vertical, normal ao solo, com origem no ombro direito. Este é o ângulo tomado como elevação do antebraço. Esse processo é mostrado na Figura 3.15.

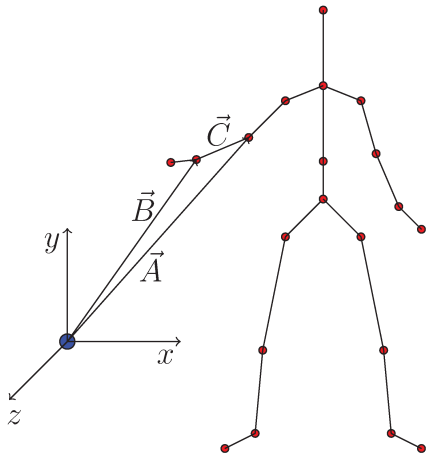


Figura 3.15 - Cálculo do ângulo de abertura da articulação Braço-Antebraço. O vetor $\vec{C}$ é dado pela diferença de $\vec{A} - \vec{B}$.

Os ângulos encontrados são normalizados para a escala utilizada pelo controlador do robô, que movimenta as articulações correspondentes no braço robótico. Os movimentos do braço direito do usuário são rastreados e atualizados a uma taxa de 10Hz. Com isso, o braço robótico movimenta-se espelhando o movimento rastreado do braço do usuário.

4 RESULTADOS E DISCUSSÃO

4.1 Captura dos Movimentos do Usuário e Representação no Mundo Virtual

O sistema é capaz de detectar quando uma ou mais pessoas estão diante do sensor. Uma delas é registrada como o usuário e a partir deste momento, passa a ter seus movimentos rastreados em tempo real. O Kinect™ foi projetado para conseguir rastrear até seis pessoas simultaneamente e registrar duas como usuários. Não existe um mecanismo de controle de usuário rígido, ou seja, caso uma pessoa que esteja no controle da aplicação saia do campo visual do sensor, o controle passa para a próxima pessoa que for detectada. Este comportamento visa permitir a fácil transição entre os usuários e também visa economia de recursos computacionais, uma vez que não é necessária nenhuma rotina de calibração para trocar de usuário. No sistema proposto, observou-se que o tempo necessário para o registro do usuário é pequeno e a precisão é muito boa. A única exigência para a correta detecção do usuário é que ele esteja totalmente no campo de visão do sensor e na distância mínima exigida, cerca de 1,2 metro. A distância máxima recomendada para que o sensor consiga seguir com precisão o usuário é de 4 metros. Entretanto, o mapa de profundidade pode ser tomado a distâncias de até 10 metros, mantendo ainda uma precisão razoável. A partir deste ponto, a precisão já começa a degradar-se rapidamente.

A taxa média de geração de quadros de vídeo e de profundidade conseguida pelo sensor Kinect™ foi de 25 fps. A utilização de programação *multithreading* pelo Módulo de Percepção foi projetada para executar os processos de geração dos fluxos de dados em *threads* distintas. Dessa forma, não há degradação visível na taxa de quadros da renderização. O sistema de eventos que foi implementado para sinalizar a ocorrência de novos dados, funcionou como esperado e garantiu a independência das tarefas. A Figura 4.1 mostra um usuário e sua representação virtual correspondente.

O mecanismo de atualização da textura e das posições baseado em evento contribuiu para a fluidez da renderização. A velocidade da renderização não foi prejudicada, já que o mecanismo de

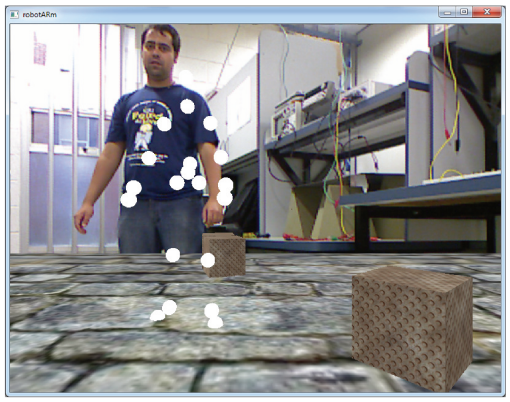


Figura 4.1 - Representação do Usuário no Mundo Virtual. As esferas brancas marcam as posições das articulações detectadas pelo sensor Kinect™ .

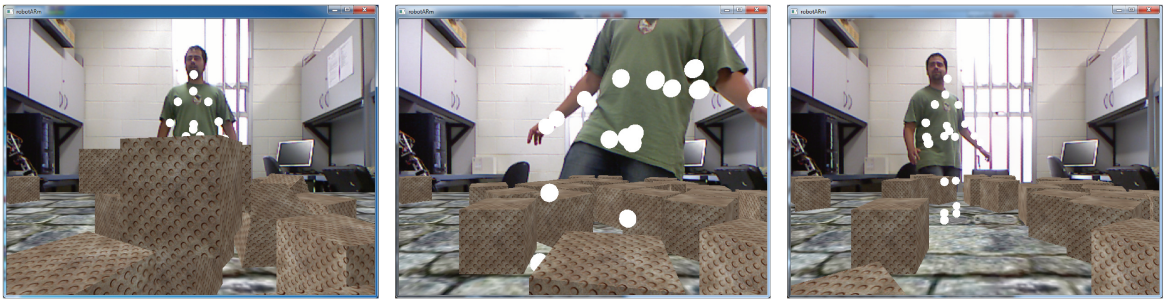
eventos associado à programação *multithreading* garante que a tarefa de atualização da textura da câmera e das posições das articulações serão feitas apenas no momento em que novas informações são disponibilizadas pelo sensor. A taxa média da renderização manteve-se em cerca de 60 fps, mesmo com a taxa de atualização do sensor mantendo-se em 25 fps, não mostrando-se computacionalmente sensível à queda na taxa de produção dos quadros pelo sensor. O que se observa quando a taxa de atualização cai abaixo dos 30 fps, é uma experiência de vídeo degradada, mas sem prejuízo das tarefas de renderização. Entretanto, a carga de trabalho nos testes não foi suficiente para causar degradação perceptível no vídeo.

4.2 Interação do Usuário com Elementos Virtuais

A interação do usuário com os elementos virtuais ocorreu de forma satisfatória. O motor de Física Bullet também é construído com programação *multithreading*, garantindo a independência e o aproveitamento dos recursos do processador, deixando a simulação bem ágil. O sensor Kinect™ é capaz de medir a sua altura em relação ao plano do solo, e o valor informado mostrou-se bastante preciso, o que contribuiu para o correto posicionamento da representação do usuário e do plano do mundo virtual utilizado pela simulação de Física. Tanto a detecção de colisões quanto à dinâmica dos objetos livres mostraram-se com precisão muito boa.

A câmera de vídeo do sensor é de boa qualidade e apresenta parâmetros estáveis, não sendo

necessário recalibrar o sistema após a determinação inicial destes parâmetros. Assim, a sobreposição manteve-se estável. A Figura 4.2 mostra a representação do usuário interagindo com caixas virtuais.



(a) (b) (c)

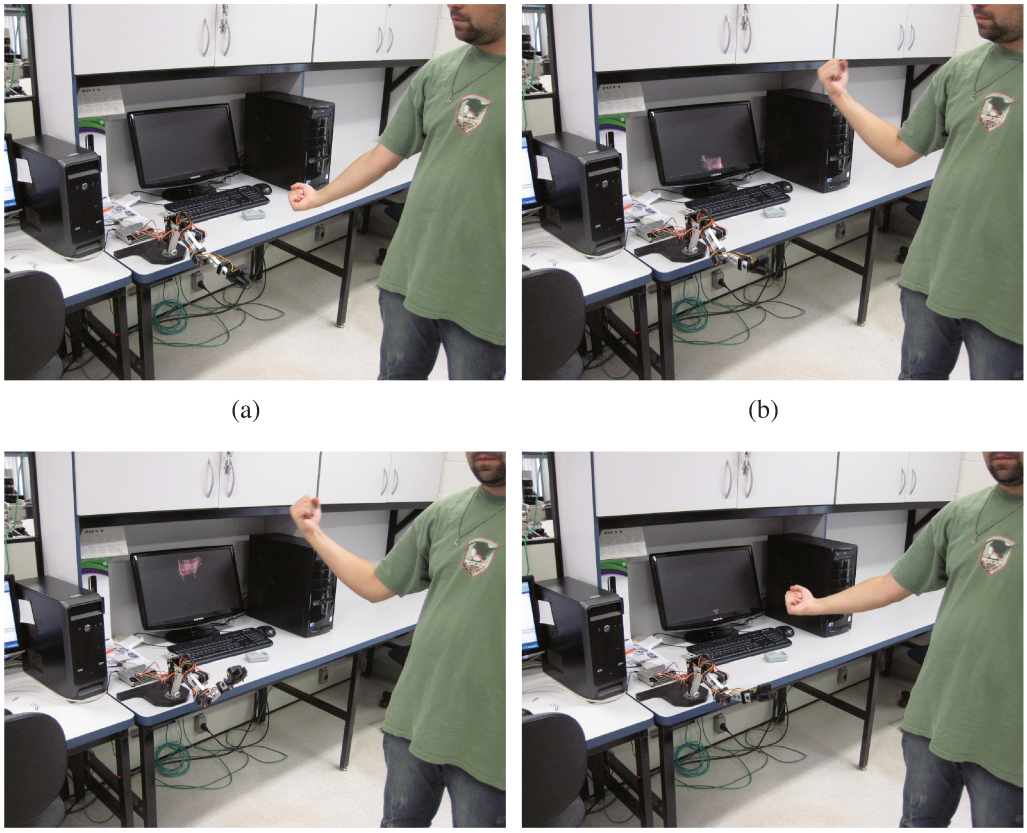
Figura 4.2 - Interação Usuário-Mundo Virtual. As caixas virtuais podem ser manipuladas pelo usuário.

4.3 Acionamento do Robô através da Extração dos Dados do Usuário

O acionamento do robô utilizando a interface paralela transcorreu sem maiores problemas. A dificuldade mais evidente foi a necessidade do uso de um sistema operacional com arquitetura de 32 *bits*, já que os sistemas mais modernos, com arquitetura de 64 *bits*, não suportam mais o formato do driver e da biblioteca de acionamento da interface. Resolvida esta questão, a integração da biblioteca de acionamento do robô ao projeto não apresentou dificuldades significativas. A documentação disponibilizada pelo fabricante é bastante completa e constituiu-se em grande auxílio à correta utilização da biblioteca.

Todos os movimentos mapeados para o robô foram executados corretamente, porém as características mecânicas do robô foram um pequeno fator limitante. Os servos originais já contam vários anos de uso, apresentando desgaste significativo. O torque dos servos também é subdimensionado, causando atraso na execução de alguns movimentos e perda da sincronia entre o movimento do usuário e o movimento do robô. Esse problema foi resolvido com a substituição dos servos desgastados por novas unidades, com maior torque. Após a substituição o desempenho do braço robótico melhorou consideravelmente. Apesar dos novos servos apresentarem um consumo de corrente

maior que os originais, o controlador original não precisou ser substituído. A sequência mostrada na Figura 4.3 mostra o braço robótico replicando os movimentos do braço direito do usuário.



(a) (b)

(c) (d)

Figura 4.3 - Braço robótico replicando os movimentos do usuário.

A taxa de transmissão de dados manteve-se estável, não prejudicando a movimentação do robô. Não foi analisado o impacto em redes lentas, entretanto a taxa de atualização do movimento do robô não exige grandes velocidades, portanto o tráfego normal na rede utilizada não foi grande o suficiente para causar alterações observáveis. Tanto o servidor quanto o cliente estão na mesma rede, o que também contribuiu para a qualidade da conexão. O aumento do número de saltos pode causar latência na rede e consequentemente prejudicar a operação remota a grandes distâncias. A Figura 4.4 mostra os dois microcomputadores utilizados.

(a) Servidor (b) Cliente

Figura 4.4 - Microcomputadores utilizados para a montagem do sistema.

5 Conclusões e Perspectivas

5.1 Sumário do Trabalho Realizado

Neste trabalho, foi desenvolvida uma interface natural, com base nos gestos do braço humano para o acionamento remoto de um braço robótico. De forma complementar, o usuário foi inserido em um ambiente de Realidade Aumentada. Para isto, foi desenvolvido um conjunto de programas que integrasse as diversas tecnologias necessárias. Para possibilitar a interação baseada em gestos, foi usado um sensor disponível comercialmente a baixo custo.

Para a realização deste trabalho, foi necessário o estudo de técnicas bastante diversas. Uma extensa revisão bibliográfica permitiu o conhecimento do estado da arte destas técnicas, bem como serviu de inspiração para a construção do sistema de demonstração. Um desafio adicional foi o uso de uma tecnologia recém lançada. Quando do início da realização deste trabalho, o sensor utilizado ainda era um projeto. O lançamento comercial do mesmo aconteceu após o início das pesquisas. Porém, o seu desempenho e características apresentaram-se bastante aplicáveis, o que motivou seu uso em detrimento de técnicas mais tradicionais. Como toda nova tecnologia, sua integração apresentou algumas dificuldades, que foram resolvidas através de constantes retrabalhos, à medida que as bibliotecas que permitiam seu uso foram sendo atualizadas ou modificadas.

A construção do conjunto de *software* ocupou uma parte significativa do trabalho. Apesar do conceito do trabalho ser aparentemente simples, sua implementação envolveu a integração de vários sistemas e técnicas relativamente complexas. Entretanto, utilização de bibliotecas já implementadas auxiliou a agilidade da integração das tecnologias. O fato das bibliotecas utilizadas serem em sua maioria de código aberto permitiu flexibilidade para a implementação de modificações necessárias. Como destaque, a biblioteca de renderização utilizada não tem suporte nativo a dispositivos de captura de vídeo, como câmeras. Tal funcionalidade foi implementada especificamente para o trabalho desenvolvido.

A construção do sistema de forma modular o deixou bastante flexível, permitindo a adaptação

para diferentes cenários de acordo com a necessidade apresentada. A decisão de separar o acionamento robótico da camada de percepção do usuário permitiu avaliar o impacto do acionamento remoto. O acionamento remoto permite o uso da técnica sem exigir a proximidade dos sistemas, o que confere grande liberdade de configurações possíveis.

A utilização dos conceitos Interface Natural mostrou-se uma solução aplicável para o acionamento de dispositivos robóticos. O fato de utilizar aspectos naturais de interação contribui para acelerar o aprendizado da operação do mecanismo.

Constatou-se através do somatório dos experimentos, que combinação de Realidade Aumentada e Interfaces Naturais constitui-se em uma ferramenta poderosa capaz de tornar intuitiva a operação de sistemas complexos, reduzindo o tempo de treinamento e adaptação do usuário, o que pode gerar uma grande economia tanto de tempo quanto de recursos financeiros. A flexibilidade mostrada por essa combinação, associada ao acionamento remoto, permite a adaptação para diversos tipos de sistemas, não limitando-se apenas aos sistemas robóticos.

5.2 Perspectivas de Trabalhos Futuros

A arquitetura desenvolvida pode ser aplicada em uma série de situações de interesse particular. A construção modular permite a adaptação do sistema de forma ágil. Alguns tópicos interessantes serão listados abaixo e brevemente discutidos nas seções seguintes.

- Sistemas de Treinamento;
- Operação Remota de Dispositivos Robóticos em Ambiente Hostil;

5.2.1 Sistemas de Treinamento

O conjunto de Realidade Aumentada e Interface Natural pode ser explorado na construção de sistemas de treinamento virtuais. A Realidade Aumentada permite uma percepção do ambiente bem próximo ao ambiente real. Com isso, pode-se diminuir o custo e o tempo dispendido no aprendizado para a operação de variados tipos de sistemas. Para atingir este objetivo, será necessário a construção de modelos virtuais adequados ao ambiente de trabalho, que reproduzam consistentemente os equipamentos a serem operados e integrar estes modelos à rotina de renderização. Já existem e estão em uso diversas ferramentas capazes de gerar estes modelos, portanto esta integração não apresentará grandes dificuldades.

5.2.2 Operação Remota de Dispositivos Robóticos em Ambiente Hostil

A arquitetura de operação remota pode ser aplicada para a operação de dispositivos robóticos capazes de operar em ambientes onde seres humanos não podem atuar. Desta forma, os robôs podem desempenhar tarefas críticas, como manutenção de equipamentos contaminados, operações em águas profundas, exploração espacial. Para estas aplicações, devem ser adaptadas tecnologias de comunicação mais robustas e confiáveis, capazes de operar em velocidades e com qualidade de serviço adequadas à finalidade pretendida. Novamente, a arquitetura modular permite a integração de diversas tecnologias de transmissão de forma transparente.

Como continuidade no desenvolvimento deste trabalho, pode-se citar:

- Construção de um modelo de representação do usuário mais realista, utilizando um modelo articulado com volume similar ao de um ser humano, em vez de marcadores apenas nos pontos de interesse;
- Integração com outros tipos de sensores capazes de reconstruir tridimensionalmente uma cena, como sonares e sensores a laser;

- Adaptação do sistema para trabalho com outros tipos de plataformas robóticas, tais como veículos autônomos exploradores, terrestre ou aéreos;
- Desenvolvimento de um protocolo de comunicação mais genérico, que permita integração com diferentes tipos de clientes.
- Inclusão de um robô de dois braços, o que daria suporte a tarefas mais complexas, que necessitem do uso simultâneo dos dois braços.

REFERÊNCIAS

ACKERMAN, S.; SHACHTMAN, N. **Almost 1 In 3 U.S. Warplanes Is a Robot**. jan. 2012. Wired - Danger Room, acessado em jan/2012. Disponível em: <http://www.wired.com/dangerroom/2012/01/drone-report/>.

AZUMA, R. T. A survey of augmented reality. **Presence: Teleoperators & Virtual Environments**, v. 6, n. 4, p. 355, 1997. Disponível em: <https://nzdis.org/projects/projects/berlin/repository/revisions/22/raw/trunk/Master's%20Docs/Papers/A%20Survey%20of%20Augmented%20Reality.pdf>.

BARAFF, D. Curved surfaces and coherence for non-penetrating rigid body simulation. In: **Proceedings of the 17th annual conference on Computer graphics and interactive techniques**. New York, NY, USA: ACM, 1990. (SIGGRAPH '90), p. 19–28. ISBN 0-89791-344-2. Disponível em: <http://doi.acm.org/10.1145/97879.97881>.

BARAFF, D.; WITKIN, A. **Physical based modeling: Rigid body simulation**. 1997. ONLINE SIGGRAPH 2001 COURSE NOTES. Disponível em: <http://www-2.cs.cmu.edu/~baraff/sigcourse>.

BLINN, J. A homogeneous formulation for lines in 3 space. In: ACM. **ACM SIGGRAPH Computer Graphics**. [S.l.], 1977. v. 11, n. 2, p. 237–241.

BLINN, J.; NEWELL, M. Clipping using homogeneous coordinates. **ACM SIGGRAPH Computer Graphics**, ACM, v. 12, n. 3, p. 245–251, 1978.

BOEING, A.; BRÄUNL, T. Evaluation of real-time physics simulation systems. In: **Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia**. New York, NY, USA: ACM, 2007. (GRAPHITE '07), p. 281–288. ISBN 978-1-59593-912-8. Disponível em:

<http://doi.acm.org/10.1145/1321261.1321312>.

BROOKS, J. F. P. The computer scientist as toolsmith ii. **Commun. ACM**, ACM, New York, NY, USA, v. 39, p. 61–68, March 1996. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/227234.227243>.

BUXTON, W.; MYERS, B. A study in two-handed input. **SIGCHI Bull.**, v. 17, p. 321–326, April 1986.

ČAPEK, K. **R.U.R. (Rossum's Universal Robots)**. 2010. EBooks@Adelaide, acessado em jan/2012. Disponível em: <http://ebooks.adelaide.edu.au/c/capek/karel/rur/>.

CARLBOM, I.; PACIOREK, J. Planar geometric projections and viewing transformations. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 10, p. 465–502, December 1978. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/356744.356750>.

CERF, V.; KAHN, R. A protocol for packet network intercommunication. **Communications, IEEE Transactions on**, v. 22, n. 5, p. 637–648, may 1974. ISSN 0090-6778.

CHANG, Y.; CHEN, S.; HUANG, J. A kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. **Research in Developmental Disabilities**, Elsevier, 2011.

COHEN, J. D. et al. I-collide: an interactive and exact collision detection system for large-scale environments. In: **Proceedings of the 1995 symposium on Interactive 3D graphics**. New York, NY, USA: ACM, 1995. (I3D '95), p. 189–ff. ISBN 0-89791-736-7. Disponível em: <http://doi.acm.org/10.1145/199404.199437>.

COUMANS, E. **Bullet Manual**. 2010. Acessado em janeiro de 2012. Disponível em: <http://bulletphysics.com/ftp/pub/test/physics/Bullet_User_Manual.pdf>.

ENGELHARD, N. et al. Real-time 3d visual slam with a hand-held rgb-d camera. In: **Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum**. Vasteras, Sweden: [s.n.], 2011.

ERLEBEN, K. **Stable, robust, and versatile multibody dynamics animation**. Tese (Doutorado) — University of Copenhagen, 2004. Disponível em: <http://image.diku.dk/kenny/download/erleben.05.thesis.pdf>.

ERLEBEN, K.; HENRIKSEN, K. **Scripted bodies and spline driven animation**. [S.l.], ago. 2002. Disponível em: <http://www.diku.dk/OLD/publikationer/tekniske.rapporter/rapporter/02-18.pdf>.

FIEDLER, G. **Fix Your Timestep!** set. 2006. Acessado em janeiro de 2012. Disponível em: <http://gafferongames.com/game-physics/fix-your-timestep/>.

FISHER, S. Telepresence master glove controller for dexterous robotic end-effectors. **Intelligent robots and computer vision**, p. 396–401, 1987.

FOLEY, J. D. et al. **Computer Graphics Principles and Practice**. [S.l.]: Addison-Wesley, 1996.

MICROSOFT CORPORATION. Ryan M. Geiss. **Visual Target Tracking**. WO/2010/088032, August 2010. Published by World Intellectual Property Organization. Disponível em: <http://www.sumobrain.com/patents/WO2010088032.html>.

HENDERSON, S.; FEINER, S. Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. In: **Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on**. [S.l.: s.n.], 2009. p. 135–144.

HOLBROOK, B.; BROWN, W. **Computing Science Technical Report No. 99 A History of Computing Research* at Bell Laboratories (1937-1975)**. [S.l.], 1982. Disponível em: <http://cm.bell-labs.com/cm/cs/cstr/99.pdf>.

HU, G.; STOCKMAN, G. 3-d surface solution using structured light and constraint propagation. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, v. 11, n. 4, p. 390–402, apr 1989. ISSN 0162-8828.

JOHNSON, E. Touch displays: a programmed man-machine interface. **Ergonomics**, Taylor & Francis, v. 10, n. 2, p. 271–277, 1967. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/00140136708930868>.

KIEFABER, D. **National Geographic Lets You Pet Dinosaurs at the Mall. Augmented reality goes Jurassic**. nov. 2011. Adweek, acessado em jan/2012. Disponível em: <http://www.adweek.com/adfreak/national-geographic-lets-you-pet-dinosaurs-mall-136591>.

KLEIMAN, S.; SHAH, D.; SMAALDERS, B. **Programming with threads**. [S.l.]: SunSoft Press, 1996.

KOLB, C.; MITCHELL, D.; HANRAHAN, P. A realistic camera model for computer graphics. In: ACM. **Proceedings of the 22nd annual conference on Computer graphics and interactive techniques**. [S.l.], 1995. p. 317–324.

LEINER, B. et al. The darpa internet protocol suite. **Communications Magazine, IEEE**, v. 23, n. 3, p. 29–34, march 1985. ISSN 0163-6804.

LEWIS, F. L.; DAWSON, D. M.; ABDALLAH, C. T. **Robot Manipulator Control: Theory and Practice**. 2. ed. [S.l.]: Marcel Dekker, 2004. 1. ed.: Control of Robot Manipulators, F.L. Lewis, C.T. Abdallah, D.M. Dawson, 1993. Prentice-Hall, Inc.

LIU, K. et al. The singularities and dynamics of a stewart platform manipulator. **Journal of Intelligent & Robotic Systems**, Springer Netherlands, v. 8, p. 287–308, 1993. ISSN 0921-0296. 10.1007/BF01257946. Disponível em: <http://dx.doi.org/10.1007/BF01257946>.

MAXWELL, E. **The methods of plane projective geometry based on the use of general homogeneous coordinates**. [S.l.]: University Press, 1946.

____. **General homogeneous coordinates in space of three dimensions**. [S.l.]: University Press, 1961.

MIDDLEBURY Stereo Vision Dataset, The. 2003. The Middlebury Computer Vision Pages. Acessado em janeiro de 2012. Disponível em: <http://vision.middlebury.edu//stereo/>.

MILENKOVIC, V.; SCHMIDL, H. Optimization-based animation. In: ACM. **Proceedings of the 28th annual conference on Computer graphics and interactive techniques**. [S.l.], 2001. p. 37–46.

MILGRAM, P.; KISHINO, F. A taxonomy of mixed reality visual displays. **IEICE Transactions on Information and Systems E series D**, Citeseer, v. 77, p. 1321–1321, 1994.

MIRTICH, B. V. **Impulse-based dynamic simulation of rigid body systems**. Tese (Doutorado) — University of California, 1996. Disponível em: <http://www.kuffner.org/james/software/dynamics/mirtich/mirtichThesis.pdf>.

MIRTICH, B. V. Timewarp rigid body simulation. In: **Proceedings of the 27th annual conference on Computer graphics and interactive techniques**. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000. (SIGGRAPH '00), p. 193–200. ISBN 1-58113-208-5. Disponível em: <http://dx.doi.org/10.1145/344779.344866>.

MÖBIUS, A. **Der barycentrische Calcul: ein neues Hülfsmittel zur analytischen Behandlung der Geometrie: mit 4 Kupfertafeln**. [S.l.]: Barth, 1827.

NOLL, A. M. A computer technique for displaying n-dimensional hyperobjects. **Communications of ACM**, ACM, New York, NY, USA, v. 10, p. 469–473, August 1967. ISSN

0001-0782. Disponível em: <http://doi.acm.org/10.1145/363534.363544>.

OGRE Manual v1.7 (“Cthugha”). Vários autores, acessado em janeiro de 2012. Disponível em: <http://www.ogre3d.org/docs/manual/>.

PORTILLA, H.; BASANEZ, L. Augmented reality tools for enhanced robotics teleoperation systems. In: **3DTV Conference, 2007**. [S.l.: s.n.], 2007. p. 1–4.

RAMEY, A.; GONZÁLEZ-PACHECO, V.; SALICHS, M. Integration of a low-cost rgb-d sensor in a social robot for gesture recognition. In: ACM. **Proceedings of the 6th international conference on Human-robot interaction**. [S.l.], 2011. p. 229–230.

ROBERTS, L. **Homogeneous matrix representation and manipulation of n-dimensional constructs**. [S.l.]: S.l. : S.n., 1966?, 1966.

SANTOS, E.; LAMOUNIER, E.; CARDOSO, A. Interaction in augmented reality environments using kinect. In: **Virtual Reality (SVR), 2011 XIII Symposium on**. [S.l.: s.n.], 2011. p. 112–121.

SCHARSTEIN, D.; SZELISKI, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. **International journal of computer vision**, Springer, v. 47, n. 1, p. 7–42, 2002.

SHAPIRO, L. G.; STOCKMAN, G. C. **Computer Vision**. [S.l.]: Prentice Hall, 2001.

SHOTTON, J. et al. Real-time human pose recognition in parts from single depth images. In: **Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on**. [S.l.: s.n.], 2011. p. 1297–1304. ISSN 1063-6919.

SHRIKHANDE, N.; STOCKMAN, G. Surface orientation from a projected grid. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, v. 11, n. 6, p. 650–655, jun 1989. ISSN 0162-8828.

SINHA, A. Client-server computing. **Commun. ACM**, ACM, New York, NY, USA, v. 35, p. 77–98, July 1992. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/129902.129908>.

SIZINTSEV, M. et al. Gpu accelerated realtime stereo for augmented reality. In: **Proceedings of the 5th International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)**. [S.l.: s.n.], 2010.

STEWART, D.; TRINKLE, J. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. **International Journal for Numerical Methods in Engineering**, Chichester, New York, Wiley [etc.] 1969-, v. 39, n. 15, p. 2673–2691, 1996.

SUTHERLAND, I. E. Sketch pad a man-machine graphical communication system. In: **Proceedings of the SHARE design automation workshop**. New York, NY, USA: ACM, 1964. (DAC '64), p. 6.329–6.346. Disponível em: <http://doi.acm.org/10.1145/800265.810742>.

TANENBAUM, A. S. **Computer Networks**. 4. ed. [S.l.]: Pearson Education, 2003.

TSAI, R. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. **Robotics and Automation, IEEE Journal of**, v. 3, n. 4, p. 323–344, august 1987. ISSN 0882-4967.

VICTORIA and Albert Museum. 2012. Disponível em: <http://www.vam.ac.uk/>.

WESTERMAN, W. **Hand Tracking, Finger Identification and Chordic Manipulation on a Multi-touch Surface**. Tese (Doutorado) — University of Delaware, 1999. Disponível em: <http://www.eecis.udel.edu/~westerma/main.pdf>.

WIKIMEDIA Commons. 2012. Open media repository, acessado em jan/2012. Disponível em: <http://commons.wikimedia.org/wiki/Main_Page>.

PRIME SENSE LTD. Zeev Zalevsky, Alexander Shpunt, Aviad Maizels e Javier Garcia. **Method and System for Object Reconstruction**. WO 2007/043036 A1, abr. 2007. Published by World Intellectual Property Organization. Disponível em: <http://www.sumobrain.com/patents/WO2007043036.html>.

ZHANG, Z. A flexible new technique for camera calibration. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, v. 22, n. 11, p. 1330–1334, nov 2000. ISSN 0162-8828.

ZIMMERMAN, T. G. et al. A hand gesture interface device. In: **Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface**. New York, NY, USA: ACM, 1987. (CHI '87), p. 189–192. ISBN 0-89791-213-6. Disponível em: <http://doi.acm.org/10.1145/29933.275628>.

APÊNDICE A - OPERAÇÕES FUNDAMENTAIS EM COMPUTAÇÃO GRÁFICA

Transformações 2D

Translação

A translação de um ponto $P(x, y)$ para um ponto $P'(x', y')$, no plano, é feita adicionando-se d_x unidades de deslocamento paralelo ao eixo x e d_y unidades de deslocamento paralelo ao eixo y. Assim, podemos escrever:

$$x' = x + d_x, y' = y + d_y \tag{A.1}$$

Essa operação pode ser representada desse modo:

$$P' = P + T, \tag{A.2}$$

onde P, P' e T são vetores-coluna nesta forma:

$$P = \begin{bmatrix} x \\ y \end{bmatrix}, P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \tag{A.3}$$

Para realizar a translação de um objeto, podemos aplicar a Equação A.1 a todos os pontos do objeto. Como cada linha em de um objeto possui um número infinito de pontos, essa operação seria demasiadamente cara. Entretanto, a translação de uma linha pode ser feita apenas em seus pontos de início e fim, e desenhando-se uma nova linha entre os pontos transladados. Essa propriedade

também é válida para as operações de escala e rotação. A Figura A.1 mostra o efeito de uma translação de um objeto por $(2, 2)$.

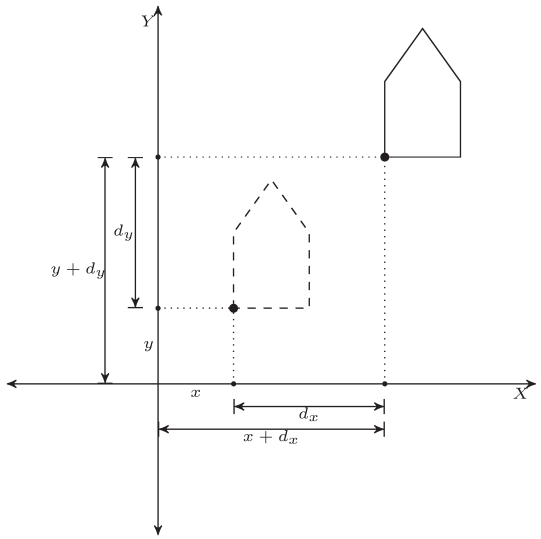


Figura A.1 - Translação de um objeto no plano. O objeto tracejado representa a posição inicial. Note o deslocamento do ponto de referência, na base do objeto.

Escala

A transformação de escala é uma operação que altera o tamanho do objeto. Similarmente à translação, a escala é aplicada multiplicando-se os pontos por um fator de escala s_x no eixo horizontal e por um fator de escala s_y no eixo vertical, ou seja:

$$x' = s_x \cdot x, \; y' = s_y \cdot y \tag{A.4}$$

Ou, na forma matricial:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}, \text{ ou } P' = S \cdot P \tag{A.5}$$

A Equação A.5 realiza a operação de escala centrada na origem. Caso o fator de escala seja

maior que 1, teremos como resultado da transformação um objeto maior e mais distante da origem que o original. Caso o fator de escala seja menor que 1 e maior que 0, teremos como resultado um objeto menor e mais próximo da origem que o original. Um caso especial da escala é o espelhamento, que consiste em aplicar uma escala com fator -1 em um ou ambos os eixos. O resultado será um objeto espelhado em relação ao eixo com o fator negativo ou em relação à origem, caso ambos os fatores sejam negativos.

A Figura A.2 mostra o resultado de uma operação de escala pelo fator $(2.5, 1.5)$. Note que como os fatores de escalas nos eixos x e y são diferentes, as proporções do objeto são alteradas. Esse tipo de escala é chamado de escala diferencial. Quando os fatores são iguais, a escala é chamada uniforme. A escala uniforme preserva as proporções do objeto. A Figura A.3 mostra uma operação de espelhamento.

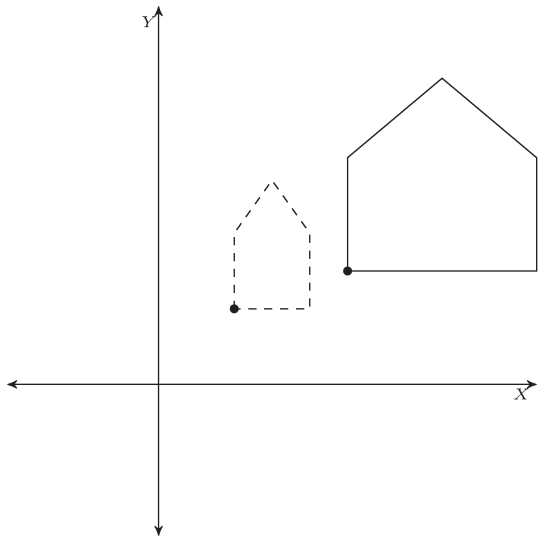


Figura A.2 - Aplicação da transformação de escala. Note o deslocamento do ponto de referência, na base do objeto.

Rotação

A operação de rotação transforma um ponto P em um ponto P' deslocando-o em um ângulo θ em torno da origem, preservando a distância do ponto original à origem. Matematicamente, a rotação é definida dessa forma:

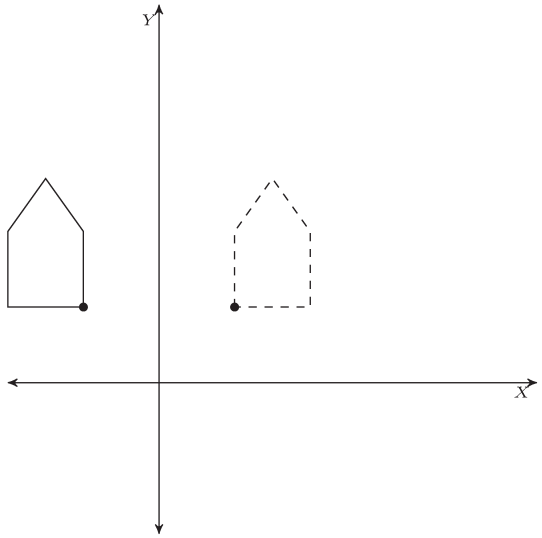


Figura A.3 - Aplicação da transformação de espelhamento no eixo vertical. Note a posição do ponto de referência.

$$x' = x \cdot \cos\theta - y \cdot \sin\theta, \quad y' = x \cdot \sin\theta + y \cdot \cos\theta \tag{A.6}$$

Na forma matricial, temos:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}, \text{ ou } P' = R \cdot P \tag{A.7}$$

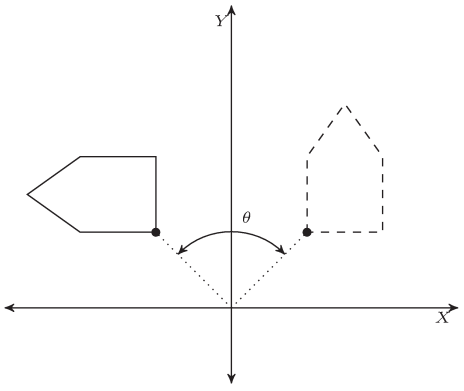


Figura A.4 - Rotação do objeto por $\theta = 90°$.

Ângulos positivos rotacionam em direção anti-horária. Para ângulos negativos, utilizam-se as identidades $\cos(-\theta) = \cos\theta$ e $\sin(-\theta) = -\sin\theta$ para modificar as Equações A.6 e A.7.

A Equação A.6 pode ser derivada com o auxílio da Figura A.5, onde um ponto P é transfor-

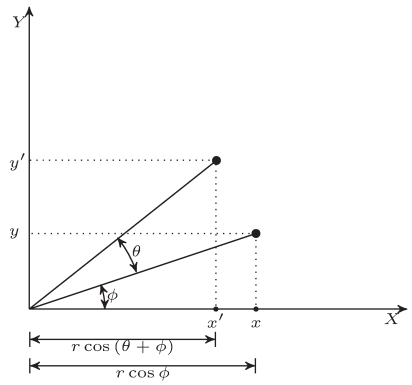


Figura A.5 - Derivação da equação da rotação.
Fonte: Adaptado de (FOLEY et al., 1996)

mado por rotação de θ no ponto P'. Como a rotação é aplicada em relação à origem, a distância r não se altera. Por trigonometria, sabemos que:

$$x = r \cdot \cos\phi, \quad y = r \cdot \sin\phi \tag{A.8}$$

Da mesma forma, sabe-se que:

$$\begin{aligned} x' &= r \cdot \cos(\theta + \phi) = r \cdot \cos\phi \cdot \cos\theta - r \cdot \sin\phi \cdot \sin\theta, \\ y' &= r \cdot \sin(\theta + \phi) = r \cdot \cos\phi \cdot \sin\theta + r \cdot \sin\phi \cdot \cos\theta \end{aligned} \tag{A.9}$$

Substituindo a Equação A.8 na Equação A.9, chegamos à Equação A.6.

Coordenadas Homogêneas e Representação Matricial das Transformações 2D

As Coordenadas Homogêneas foram introduzidas e desenvolvidas na geometria (MÖBIUS, 1827; MAXWELL, 1946; MAXWELL, 1961) e posteriormente aplicadas à computação gráfica (BLINN, 1977; BLINN; NEWELL, 1978; ROBERTS, 1966). As coordenadas homogêneas são utilizadas por amplo número de bibliotecas e pacotes gráficos.

Para representação de pontos do espaço bidimensional em coordenadas homogêneas, acrescenta-se uma terceira coordenada ao ponto. Assim, um ponto (x, y) passa a ser representado por um tripla (x, y, w). Se duas triplas (x, y, w) e (x', y', w') forem múltiplas entre si, elas representam o mesmo ponto. Esta propriedade permite que um único ponto tenha infinitas representações em coordenadas homogêneas.

Pelo menos uma das coordenadas homogêneas precisa ser diferente de 0. A representação $(0, 0, 0)$, portanto, não é um ponto válido. Se a coordenada w for diferente de 0, podemos normalizar a representação dividindo todas as coordenadas por w, resultando em $(\frac{x}{w}, \frac{y}{w}, 1)$. Com isso, obtemos as coordenadas cartesianas do ponto homogêneo como sendo o par $(\frac{x}{w}, \frac{y}{w})$. As coordenadas homogêneas também permitem um modo eficiente de representar pontos no infinito. Estes pontos são representados por conjuntos de coordenadas onde $w = 0$. Se considerarmos o ponto no infinito $(x, y, 0)$ e tomarmos um vetor partindo da origem em direção a um ponto $(x, y, 1)$, obteremos a direção do ponto no infinito.

Como agora representamos os pontos por vetores-coluna de três elementos, as matrizes das transformações devem ser 3×3 para que o resultado também seja um vetor de três elementos. A representação homogênea da equação da translação, Equação A.1 é:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{A.10}$$

Alguns autores preferem representar as matrizes de transformação através da multiplicação de um vetor-linha pela matriz, ao invés de multiplicar a matriz por um vetor coluna. Para mudar a representação, basta transpor a matriz e inverter a ordem da operação.

Para simplificar, podemos representar a Equação A.10 por:

$$P' = T(d_x, d_y) \cdot P, \tag{A.11}$$

onde

$$T(d_x, d_y) = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \tag{A.12}$$

A representação em coordenadas homogêneas permite a aplicação de sucessivas transformações através da concatenação das respectivas matrizes, permitindo aninhar um conjunto de transformações através de uma única matriz. Por exemplo, um ponto P é transladado por $T(d_{x1}, d_{y1})$ para P' e depois transladado novamente por $T(d_{x2}, d_{y2})$ para P''. Teremos:

$$P' = T(d_{x1}, d_{y1}) \cdot P, \tag{A.13}$$

$$P'' = T(d_{x2}, d_{y2}) \cdot P' \tag{A.14}$$

Substituindo a Equação A.13 na Equação A.14, obtemos:

$$P'' = T(d_{x2}, d_{y2}) \cdot (T(d_{x1}, d_{y1}) \cdot P) = (T(d_{x2}, d_{y2}) \cdot T(d_{x1}, d_{y1})) \cdot P \tag{A.15}$$

Representando matricialmente, a Equação A.15 fica assim:

$$\begin{bmatrix} 1 & 0 & d_{x2} \\ 0 & 1 & d_{y2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & d_{x1} \\ 0 & 1 & d_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_{x1} + d_{x2} \\ 0 & 1 & d_{y1} + d_{y2} \\ 0 & 0 & 1 \end{bmatrix} \tag{A.16}$$

Finalmente, podemos concluir que a transformação final pode ser representada por:

$$P' = T(d_{x1} + d_{x2}, d_{y1} + d_{y2}) \cdot P \tag{A.17}$$

Essa propriedade também vale para as transformações de rotação e escala. O uso da representação em coordenadas homogêneas permite combinar as três transformações multiplicando-se as respectivas matrizes. Essa propriedade permite economizar tempo de processamento, ao combinar várias transformações em uma única operação.

A operação de escala em coordenadas homogêneas fica na seguinte forma:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{A.18}$$

Ou, de forma mais simples:

$$\begin{aligned} P' &= S(s_x, s_y) \cdot P, \quad \text{where} \\ S(s_x, s_y) &= \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{A.19}$$

A representação em coordenadas homogêneas da transformação de rotação fica dessa forma:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{A.20}$$

Ou então, fazendo:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.21}$$

Temos:

$$P' = R(\theta) \cdot P \tag{A.22}$$

As transformações de rotação e translação, quando combinadas em qualquer sequência arbitrária, recebem o nome de transformações de corpo rígido, porque preservam ângulos e distâncias,

alterando apenas a posição do objeto.

A combinação das transformações de rotação, translação e escala, em qualquer sequência arbitrária tem a propriedade de preservar o paralelismo das linhas, mas não ângulos e distância. Elas recebem o nome de transformações afins. Tanto as transformações de rotação, escala e translação quanto o produto de suas combinações são transformações afins. Um outro tipo de transformação afim é a transformação de cisalhamento. A transformação de cisalhamento transporta as faces do objeto ao longo de um dos eixos. A Figura A.6 mostra como um objeto é afetado por essa transformação.

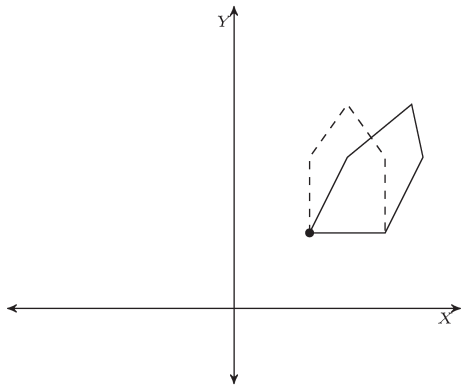


Figura A.6 - Operação de cisalhamento ao longo do eixo X, com fator $a = 0.5$.

A operação de cisalhamento ao longo do eixo x é representada pela matriz

$$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.23}$$

onde o termo a é a constante de proporcionalidade. A transformação aplicada é uma mudança proporcional de x como função de y pela constante a. Do mesmo modo, a operação de cisalhamento no eixo y opera como uma mudança proporcional de y em função de x e é representada pela matriz

$$SH_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.24}$$

Representação Matricial de Transformações 3D

Similarmente à representação de transformações 2D através de matrizes 3×3 utilizando coordenadas homogêneas, as transformações 3D podem ser representadas com a mesma eficiência, por matrizes 4×4. Os pontos no espaço 3D são representados então, por quádruplas na forma (x, y, z, w). Qualquer conjunto de quádruplas que sejam múltiplas entre si, representam o mesmo ponto. Isso permite a homogeneização de qualquer quádrupla (x, y, z, w) para $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1)$, onde a tripla $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$ representa o ponto cartesiano correspondente. Assim como cada ponto no espaço 2D é representado por uma linha no espaço 3D, um ponto no espaço 3D é representado por uma reta através da origem no espaço 4D. O conjunto de todos os pontos 3D, representados homogeneamente, forma um subconjunto 3D do espaço 4D. Esse espaço é definido pela equação $w = 1$.

A orientação do espaço 3D é um aspecto importante que deve ser levado em consideração. Por convenção matemática, a orientação utilizada é a orientação de mão direita ou dextrógira. Nesta orientação, uma rotação positiva, observada de uma posição positiva em direção à origem, é orientada em sentido anti-horário. Porém, muitos sistemas gráficos utilizam a orientação de mão esquerda ou levógira, o que faz com que a mesma rotação se dê em sentido horário. Esta representação faz mais sentido, porque quando superposta à face do monitor, como mostrado na Figura A.7, produz a interpretação de que maiores valores na coordenada z parecem afastar-se do observador para "dentro" da tela.

A transformação de translação em 3D, pode ser representada por uma simples extensão da translação 2D, representada na Equação A.10:

$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.25}$$

A transformação de escala, também é representada pela extensão da Equação A.18:

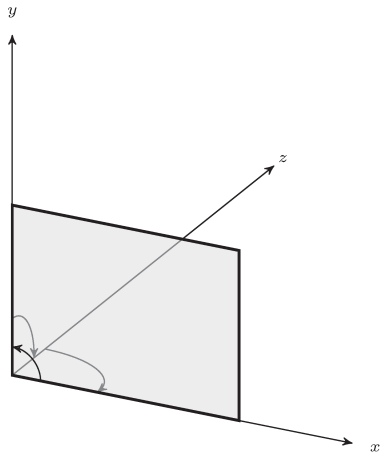


Figura A.7 - Orientação de um sistema de coordenadas 3D levógiro. As setas mostram o sentido positivo de orientação das rotações e translações.
Fonte: Adaptado de (FOLEY et al., 1996)

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.26}$$

A transfomação de rotação 2D (Equação A.7) pode ser entendida como uma rotação 3D em torno do eixo z. Então, a representação em coordenadas homogêneas em 3D fica dessa forma:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.27}$$

Para rotações nos eixos x e y, é necessário uma matriz de rotação diferente para cada eixo. Para o eixo x:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.28}$$

Para o eixo y:

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.29}$$

As transformações 3D também podem ser combinadas tais quais as transformações 2D. O resultado será sempre obtido nessa forma:

$$M = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.30}$$

onde a submatriz 3×3 superior esquerda R é a composição das rotações e escalas, enquanto o subvetor $T = [t_x\ t_y\ t_z]'$ representa a translação agregada. Podemos melhorar a eficiência computacional desta operação fazendo a transfomação desta forma:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + T, \tag{A.31}$$

onde R e T são as submatrizes da Equação A.31.