O Uso de Realidade Virtual em Sistemas Telerobóticos Baseados na Internet

Luiz Cláudio Alencar da Silva

Dissertação de Mestrado submetida à Coordenação do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande - Campus de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação

Antônio Marcus Nogueira Lima, Dr.
Orientador
Vicente Ferreira de Lucena Junior, Dr-Ing.
Orientador

Campina Grande, Paraíba, Brasil

O Uso de Realidade Virtual em Sistemas Telerobóticos Baseados na Internet

Luiz Cláudio Alencar da Silva

Dissertação de Mestrado apresentada em Junho de 2007

Antônio Marcus Nogueira Lima, Dr.
Orientador
Vicente Ferreira de Lucena Junior, Dr-Ing.
Orientador

João Marques de Carvalho, Ph.D.
Componente da Banca
José Sérgio da Rocha Neto, D.Sc.
Componente da Banca

Campina Grande, Paraíba, Brasil, Junho de 2007

Dedicatória

A minha esposa Leocília Benitah Printes da Silva, aos meus filhos Ester Benitah Printes da Silva e Daniel Benitah Printes da Silva, e aos meus irmãos da fé (Comunidade Evangélica Ebenezer), pelas inúmeras orações e apoio nas minhas dificuldades, e estímulo que me ofereceram. Este trabalho é dedicado especialmente ao Senhor Jesus, meu Salvador e Senhor, que tem me sustentado durante todo esse tempo, dando-me sabedoria e perseverança nas dificuldades, a Ele dedico esta conquista como gratidão.

Agradecimentos

- À Deus que tem me dado sabedoria, direção, fé e certeza de minha vitória;
- Aos meus grandes amigos do mestrado e os conquistados no laboratório de elétrica e tantos outros, pelo o apoio, incentivo e sobretudo, por estarem sempre presentes diante da menor solicitação de suas presenças, em especial aos alunos de mestrado Roberto Higino e Walter Valenzuela (Waltinho);
- Ao meu orientador Antônio Marcus Nogueira Lima pela paciência que sempre soube me encaminhar nos estudos;
- Ao Centro de Tecnologia de Eletrônica e da Informação (CETELI) da Universidade Federal do Amazonas, na pessoa do professor Vicente Ferreira de Lucena Junior por fornecer toda a infra-estrutura de estudo sem a qual este trabalho não seria possível;
- A Fundação Centro de Análise Pesquisa e Inovação Tecnológica (FUCAPI) por ter me dado a oportunidade de aprimorar meus conhecimentos, e também pela liberação e financiamento do curso;
- Agradecimento especial ao aluno de doutorado de Engenharia Elétrica da UFCG, José Renato de Brito Sousa, pelas inúmeras dúvidas esclarecidas sobre Redes de Petri.

Resumo

Um sistema telerobótico baseado na Internet foi desenvolvido neste trabalho de pesquisa. O sistema consiste de uma interface gráfica de usuário com representação virtual do robô e do ambiente de trabalho, e um painel para programação *off-line* de tarefas do robô. As tarefas podem ser enviadas via Internet ao ambiente remoto, onde são executados, e o resultado, pode ser visualizado de forma *on-line* em um ambiente virtual no lado de cliente. O ambiente virtual recebe atualizações via Internet de posições do robô e dos objetos da área de trabalho, permitindo eliminar a necessidade de uma transmissão de imagens de vídeo. O uso do ambiente virtual neste sistema permite que: (1) atrasos de tempo inerentes à Internet sejam suprimidos; (2) o trabalho do operador seja simplificado e acelerado, se comparado com sistemas que usam imagens de vídeo. Entretanto, os ambientes virtual e físico devem se manter sincronizados de modo que o ambiente virtual possa mapear de forma mais realista o ambiente físico, principalmente se este for dinâmico. Para tratar com essa questão, neste trabalho é apresentada uma solução baseada no uso de redes de Petri para fazer a integração dos ambientes virtual e físico através de técnicas de processamento de imagens, de modo que os mesmos se mantenham sincronizados garantindo atualizações do ambiente virtual para operações subseqüentes.

Abstract

An Internet-Based-Telerobotic system has been developed in the research here reported. The system consists of a graphic user interface with virtual representation of the robot and of the worksite, and a panel for off-line planning robot tasks. The tasks can be sent via Internet to the remote environment, where they are executed, and the result, can be visualized on-line in a virtual environment at the client side. The virtual environment receives updates of robot and objects positions at the robot worksite via Internet, thus eliminating the need for a full image transmission. The use of the virtual environment in this system allows: (1) time delays inherent to Internet to be suppressed; (2) work of the operator to be simplified and accelerated, compared to sistems that use TV images. However, the virtual and physical environments have to remain synchronized so that the virtual environment can fit the physical environment in a more realistic way, mainly if it is a dynamic one. To deal with this issue, this work proposes a solution based on Petri nets to do the integration of virtual and physical environments through the image processing techniques, so that they remain synchronized and assuring updates of virtual environment for subsequent operations.

Sumário

Lista de Símbolos e Abreviaturas

AV	Ambiente Virtual. Espaço tridimensional gerado por computador que simula um ambiente físico.
CPNTools	Ferramenta para editar, simular e analisar Redes de Petri Coloridas, temporizadas, não temporizadas e hierárquicas.
HSI	Modelo de cor, especificado com valores de matiz, saturação, e intensidade, respectivamente.
Internet	Rede de computadores de alcance mundial, conectado através do protocolo IP.
Matlab	MATrix LABoratory. Um software interativo de alta performance voltado para o cálculo numérico.
PDI	Processamento Digital de Imagem. Processamento de dados no qual a entrada e saída são imagens tais como fotografias ou quadros de vídeo.
RGB	Modelo no qual as cores são produzidas através da combinação de três cores básicas (vermelho, verde, azul).
RP	Redes de Petri Lugar/Transição. Representação matemática para sistemas distribuídos discretos.
RPC	Redes de Petri Coloridas. Redes utilizadas na modelagem de aplicações complexas, onde caracterizam-se pela incorporação de tipo de dados complexos manipulados pelo uso de uma linguagem.
RPCH	Redes de Petri Coloridas Hierárquicas. Redes que possibilitam a construção de um modelo através da combinação de um conjunto de redes relativamente menores.
RV	Realidade Virtual. Interface avançada entre um usuário e um sistema computacional.
SAI	Interface que permite uma aplicação Java manipular *nós* de objetos virtuais de um ambiente virtual.

SCR	Sistema de Controle do Robô. Sistema desenvolvido pela Universidade Federal do Amazonas para fornecer um laboratório remoto baseado na Internet.
SSL	Secure Sockets Layer - Camada de Soquete Segura. Protocolos criptográfico que provém comunicação segura na Internet para serviços, tais como, e-mail (SMTP), navegação por páginas (HTTP) e outros tipos de transferência de dados.
STI	Sistema Telerobótico Baseado na Internet.
TCP	Transmission Control Protocol (Protocolo de Controle de Transmissão). Uma norma que define o processo de transmissão de pacotes de dados em redes de telecomunicações, garantindo que eles sejam recebidos na mesma ordem em que foram emitidos.
TCP/IP	Transmission Control Protocol/Internet Protocol. Um conjunto de protocolos que permite o compartilhamento de aplicações entre computadores heterogêneos em uma rede de comunicação.
Teleoperação	Operação à distância de um veículo ou um sistema.
Telepresença	Tecnologia que permite ao ser humano ter a sensação de estar presente em tempo real em outro lugar, sendo este capaz de interagir com um ambiente remoto ou virtual.
Telerobótica	Operação que envolve interação entre um operador e um robô localizado remotamente.
VPN	Virtual Private Network - Rede Privada Virtual. É uma rede de comunicações privada normalmente utilizada por uma empresa ou um conjunto de empresas e/ou instituições, construída em cima de uma rede de comunicações pública.
VRML	Virtual Reality Modeling Language (Linguagem para Modelagem de Ambientes Virtuais). Um padrão que permite a modelagem e a navegação de informações através um ambiente 3D.
X3D	Um padrão aberto para distribuir conteúdo 3D. Ela é a revisão da especificação ISO do VRML97.
Xj3D	Navegador padrão aberto utilizado para navegação do ambiente X3D.

Lista de Tabelas

Lista de Figuras

Introdução

Atualmente, existem mais de um milhão de robôs em uso em todo o mundo (YANG et al., 2004). Esses robôs, com diferentes níveis de inteligência artificial, são controlados de alguma forma por seres humanos e empregados na execução de tarefas precisas, que podem ser realizadas em lugares próximos, ou em ambientes remotos de difícil acesso, em ambientes relativamente seguros, ou em localizações altamente perigosas. Uma das vantagens do emprego de robôs é que eles podem executar tarefas em ambientes onde o acesso humano é difícil e/ou perigoso.

Embora robôs possuam diferentes níveis de inteligência e possam, em alguns casos, executar algumas tarefas automaticamente, a intervenção e inteligência humana são ainda indispensáveis, especialmente situações em que os robôs não conseguem tratar sozinhos, tais como, em explorações em lugares inóspitos. A telerobótica é o modo de combinar a inteligência do ser humano com a ação de robôs. A telerobótica, ou teleoperação de robôs pode ser definida como sendo uma operação que envolve interação entre um operador (operador humano) e um robô remoto via canais de comunicação.

A Internet tem se tornado a mais importante rede para comunicação e a maior em armazenamento de dados. Ela conecta milhões de computadores em todo mundo, dando acesso a informação, dados, imagens, vídeos e, até mesmo, ao controle de dispositivos remotamente. A Internet tem sido utilizada como um meio para transmitir comandos em sistemas telerobóticos; pelo qual o controle pode ser enviado para sistemas remotos e o retorno pode ser obtido. O uso da Internet em aplicações de telerobótica oferece a vantagem de baixo custo, dispensando a necessidade de montar equipamentos caros no lado do operador. Qualquer computador conectado à Internet pode ser usado para controlar um dispositivo teleoperado. Com essas características, a Internet é um dos meios mais convenientes de transmissão de dados para tarefas de teleoperação e portanto, o uso da Internet para telerobótica em ambientes remotos tem se tornado um dos tópicos mais investigados em robótica e automação. Nos últimos anos, Sistemas Telerobóticos baseados na Internet (STIs) tem sido usados em aplicações industriais (LUO; LEE, 1999), e também, em ambientes inóspitos, tais como, sítios radioativos (REDD et al., 2006), exploração espacial (BACKES; TSO, 2004) e operação em águas profundas (ZHANG et al., 2003).

Entretanto, o progresso nessa área tem sido impedido pela largura de banda limitada e atrasos de tempo imprevisíveis da Internet, que influenciam diretamente no controle de STIs. Em teleoperações com atrasos de tempo, o operador não pode observar mudanças em tempo real do robô e do ambiente através da transmissão das imagens de vídeo devido a atrasos da rede. De fato, STIs com controle baseado em imagens de vídeo tem apresentado grandes atrasos de tempo (de alguns segundos a vários minutos). Isso pode ser constatado nos STIs desenvolvidos por Goldberg, Maschna e Gentner (1995), Goldberg e Santarromana (2006) e Taylor e Trevelyan (1995), que revelaram retardos na ordem de minutos no tempo de resposta de seus sistemas.

Para evitar essa limitação, alguns STIs tem utilizado a Realidade Virtual (RV) para reduzir o volume de dados que são transmitidos entre o local de controle (operador) e a área de trabalho remota (robô), e com isso, permitir que as respostas no controle desses sistemas se tornem mais rápidas (ZAKARIA; AMIN; MAMAT, 2000). O retorno necessário para controlar o robô é essencialmente visual. A transmissão de vídeo em tempo real requer uma alta largura de banda (a partir de 128 Kbps) que atualmente a Internet padrão não pode oferecer (SAFARIC; SORGO, 2003); como uma alternativa, o Ambiente Virtual (AV) local é atualizado com pouco volume de informações - os dados de posição do robô e objetos do ambiente físico são transmitidos rapidamente para permitir atualizações rápidas do AV local.

Neste trabalho de pesquisa é desenvolvido um STI baseado em RV para tratar com a questão dos atrasos de tempo imprevisíveis da Internet. Um AV é utilizado no lugar de imagens de vídeo, comumente utilizadas em STIs, para resolver o problema de transferência de imagens reais (a transferência é reduzida a dados de posicionamento). Dessa forma, o tempo de resposta do sistema é reduzido, resultando assim, em uma resposta mais rápida e fornecendo um melhor controle ao operador. Além disso, o AV permite ao operador simular as tarefas de operação no robô virtual, observando estados futuros do robô real, salvando estas tarefas e, quando quiser, enviando-as ao local de execução. O resultado é visualizado no AV de forma *on-line*, mais rapidamente que imagens de vídeo sendo transmitidas do local da operação. Essa característica da RV, facilita uma melhor interação usuário-sistema, pois o operador interage com o robô virtual obtendo respostas instantâneas, sem os atrasos de tempo indesejáveis da Internet.

Entretanto, os ambientes virtual e físico devem se manter sincronizados de modo que o ambiente virtual possa mapear de forma mais realista o ambiente físico, principalmente se este for dinâmico. Pesquisas anteriores (TAN; CLAPWORTHY, 2003), (BELOUSOV; CHELLALI; CLAPWORTHY, 2001) mostram que ao efetuar o tratamento das incertezas associadas às mudanças que podem acontecer no ambiente físico durante a teleoperação é possível aproveitar o máximo do potencial de AVs em STIs. Essas mudanças geralmente dizem

respeito ao surgimento de novos objetos na cena de trabalho ou variações nas localizações correntes de objetos conhecidos.

Para tratar com a questão supracitada, neste trabalho é apresentada uma solução baseada no uso de Redes de Petri (RPs) para fazer a integração dos ambientes virtual e físico através de técnicas de Processamento Digital de Imagens (PDI), de modo que, os mesmos se mantenham sincronizados, garantindo atualizações do AV para operações subseqüentes. Neste aspecto, esta abordagem juntamente com a interface de operação, compõe um conjunto completo de ferramentas para o controle de STIs.

Motivação

Os principais motivos de estudo deste trabalho são:

- Tornar os STIs uma realidade prática em aplicações industriais;
- Tornar os STIs funcionais à conexões de baixa velocidade (conexões a partir de linha discada);
- Tornar os STIs acessíveis ao público menos experiente.

Objetivos e Contribuição

O objetivo deste trabalho é projetar e construir um STI em arquitetura cliente/servidor, utilizando a RV para tratar com a questão dos atrasos de tempo imprevisíveis da Internet.

A solução desenvolvida compõe de uma arquitetura genérica e modular, capaz de suportar robôs de maior complexidade pelo motivo de a mesma não utilizar sensores para fazer a captura das posições do robô. A utilização de imagens independe do tipo de robô e também não interfere no desempenho do sistema. Além disso, essa solução é mais barata e simples, podendo evitar problemas oriundos do mal sensoriamento do ambiente físico.

Neste STI é utilizado um robô industrial Plotter (traçador) de 3 graus de liberdade capaz de desenhar retas, retângulos e círculos a partir de linhas retas em uma área de $40cm^2$. A estratégia de controle adotada neste trabalho se baseia em especificar as tarefas do robô em modo *off-line* utilizando AV com o objetivo de melhorar a interação usuário-sistema.

São metas do presente trabalho de pesquisa atender os seguintes requisitos do sistema:

- Modelar o AV contendo o robô e área de trabalho virtuais para representar o ambiente físico (ambiente robótico);

- Desenvolver uma solução baseada em RPs e PDI para integrar os ambientes virtual e físico e tratar com a questão de detecção de colisão do robô e os objetos do ambiente físico;

- Desenvolver a parte servidor do STI para enviar comandos ao controlador do robô e processar as imagens capturadas do ambiente físico para se extrair as posições do robô e objetos com o intuito de atualizar os AVs;

- Desenvolver a parte cliente do STI composta de duas interfaces gráficas de usuário e AVs para especificação *off-line* das tarefas do robô e visualização *on-line* do resultado das operações do mesmo.

As principais contribuições esperadas do presente projeto de pesquisa são:

- Utilizar a RV como fator redutor do tempo de resposta de STIs e, com isso, permitir uma melhor interação usuário-sistema;

- Implementar uma solução baseada em RPs e PDI (RP-PDI) para integrar os ambientes físico e virtual.

Organização dos Capítulos

O conteúdo do presente trabalho está dividido em seis capítulos, apresentando os conceitos básicos que suportam a abordagem do trabalho, uma visão geral da arquitetura do STI, a descrição detalhada dos subsistemas cliente e servidor do sistema, a demonstração dos casos de teste do sistema, e por fim, as considerações finais do trabalho.

Apresenta-se no Capítulo 1, os fundamentos básicos dos formalismos a serem introduzidos neste trabalho, tais como, noções de Telerobótica, de Realidade Virtual, de Processamento Digital de Imagens e de Redes de Petri, necessários para um entendimento dos conceitos introduzidos neste trabalho.

No Capítulo 2, descreve-se o trabalho apresentando de forma geral a estrutura do sistema e os requisitos necessários para funcionamento do mesmo. Neste capítulo também é abordado o estado da arte, fazendo-se uma revisão bibliográfica, a fim de avaliar outras soluções existentes com o trabalho proposto.

No Capítulo 3, é apresentado de forma detalhada o projeto do subsistema servidor, descrevendo cada módulo da arquitetura de *software*, enfatizando o módulo de imagem utilizado na manutenção do ambiente virtual. Este capítulo também descreve o módulo de comunicação servidor/cliente para receber conexão do cliente.

No Capítulo 4, descreve-se o projeto do subsistema cliente e apresenta-se os módulos de ambiente virtual de controle e de resultado, bem como, a modelagem do ambiente

virtual utilizado nestes módulos e o módulo de comunicação cliente/servidor para fazer conexão com o servidor do robô.

No Capítulo 5, por sua vez, são tratados os casos de teste do STI e apresentados os resultados adquiridos através da execução do modelo de rede de Petri que controla as etapas de processamento digital de imagens no processo de extração dos parâmetros das imagens.

Finalmente, no Capítulo 6 apresenta-se as discussões finais e resume-se as principais conclusões e contribuições obtidas a partir dos resultados obtidos no estudo. Aqui, são também sugeridas algumas futuras investigações que poderão dar prosseguimento às pesquisas na área de STIs.

Capítulo 1

Fundamentação Teórica

Neste capítulo são introduzidos os conceitos básicos dos formalismos que são utilizados neste trabalho. Na seção 1.1 é introduzido o conceito de Telerobótica, apresentando sua arquitetura típica, áreas de aplicação, questões atuais e o uso da Internet em Telerobótica. Na seção 1.2 são apresentadas as noções básicas sobre a RV, objetivando esboçar sua estrutura, modelagem, aplicações, linguagens para modelagem de AV e exemplos de AVs. Na seção 1.3 são apresentadas as noções básicas sobre PDI, introduzindo o conceito de amostragem e quantização de imagens, etapas de PDI, Processamento de Imagem Colorida, modelos de cores e segmentação por modelo de cor. Na seção 1.4 são introduzidas as RPs, apresentando os conceitos de RPs Coloridas e RPs Coloridas Hierárquicas.

1.1 Telerobótica

Sistemas telerobóticos têm sido utilizados em muitas áreas, tais como, aplicações cirúrgicas (ROVETTA; BEJCZY; SALA, 1997), automação de manufatura (LUO; LEE, 1999), exploração espacial (BACKES; TSO, 2004) e operação em águas profundas (ZHANG et al., 2003). Uma das vantagens, é permitir a execução de tarefas sem a presença do ser humano no local de trabalho.

A telerobótica pode ser definida como sendo uma operação que envolve interação entre um operador e um telerobô via canais de comunicação. Sheridan define telerobô como uma máquina com sensores de ambiente e dispositivos para executar trabalho mecânico remotamente (SHERIDAN, 1992). O operador supervisiona o telerobô através de um computador intermediário. Com este computador, o operador faz a programação de tarefas, envia essas programações e visualiza o retorno do resultado da execução do robô. O telerobô executa uma tarefa baseado na informação recebida do operador e dos dados capturados de seus sensores.

1.1.1 Arquitetura Típica de um Sistema Telerobótico

Um projeto típico de arquitetura de Sistema Telerobótico inclui três partes principais, conforme ilustrado na Figura 1.1: cliente, servidor e robô.

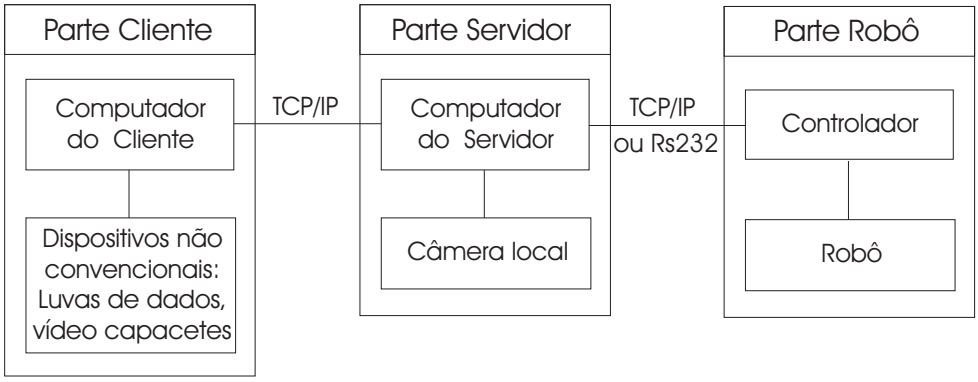

Figura 1.1: Arquitetura Típica de um Sistema Telerobótico

A parte cliente é a interface de operação. Alguns clientes também incluem dispositivos de entrada não convencionais, tais como, luvas de dados (datagloves) e vídeo-capacete (Head Mounted Display - HMD). O *software* na parte cliente inclui a interface de usuário que pode conter imagens de vídeo e outros *feedbacks* transmitidos do servidor. Alguns sistemas também podem incluir modelos de AVs para facilitar a interatividade com o usuário (YANG; CHEN, 2004).

A parte servidor contém um computador servidor que é conectado ao controlador do robô. Alguns sistemas também incluem uma câmera com uma placa de aquisição de imagem e diferentes sensores para diferentes propósitos. O *software* do servidor inclui diferentes sistemas de aquisição e processamento de dados, tais como, dados sensoriais e imagens contendo também a conexão e modelo de controle do robô (YANG; CHEN, 2004).

A parte robô, por sua vez, inclui o robô propriamente dito e o controlador do robô.

1.1.2 Alocação de Tarefas entre Humanos e Robôs

Robôs Industriais utilizam um alto grau da automação e reduzem a necessidade do trabalho humano. Em contraste, os telerobôs necessitam de um operador para realizarem a tarefa.

As diferenças essenciais entre telerobôs e robôs industriais estão na natureza da tarefa e da necessidade do controle por um operador. Os robôs industriais executam repetidamente tarefas na sua maior parte conhecidas em ambientes controlados com velocidade e exatidão. Ao passo que, em telerobôs, há muito pouco ou nada de repetição da mesma tarefa.

Com o avanço da tecnologia de telerobótica, os telerobôs podem executar automaticamente sub-tarefas de crescente complexidade, afim de reduzir o número de interações com o operador (FRIZ, 1998).

1.1.3 Áreas de Aplicação

Telerobôs podem ser usados em áreas de aplicação que necessitem da flexibilidade humana, mas que não podem ser executadas pelos seres humanos, por uma ou mais das razões seguintes:

- O ambiente é arriscado à saúde humana ou à sua sobrevivência. Exemplos: águas profundas, espaço exterior, ambientes tóxicos, minas, construções, incêndios, operações militares ou policiais;
- A tarefa está fora da escala humana. Exemplos: perfuração de ossos de implantes de união de quadril artificiais (necessita de alta precisão), escavadores e gindastes em construção (grandes forças e escala);
- Telerobôs podem estender a capacidade humana. Exemplos: membros artificiais, dispositivos de assistência para desabilitados;
- Transporte de seres humanos ao ambiente seria muito caro ou despenderia muito tempo. Exemplos: exploração em outros planetas (Marte) ou em águas profundas, manutenção de cabos debaixo d'água, telediagnóstico e telecirurgia em que especialistas não possam estar presentes;
- A presença de seres humanos prejudicaria o ambiente: Exemplos: exploração de ambientes sensíveis ou sítios arqueológicos. Telerobôs podem ser construídos bem menores que veículos de transporte de seres humanos.

1.1.4 Telerobótica baseada na Internet

Os custos de sistemas telerobóticos podem ser consideravelmente reduzidos com o uso de PCs e *software* padrão predominante na maior parte das tarefas computacionais. Uma outra forma de reduzir custos nesses sistemas é usar a Internet como meio de comunicação entre o computador do operador (cliente) e o computador que controla o robô (servidor). Não existe mais a necessidade de montar equipamentos caros no lado do operador. Quase todo computador conectado a Internet pode ser usado para controlar um dispositivo teleoperado. A Internet oferece a infra-estrutura para comunicação de sistemas telerobóticos.

STIs devem contar com protocolos de comunicação disponíveis para trocar dados em tempo real entre as partes local e remota. A Internet fornece um suporte transparente e

confiável para a troca de dados entre computadores usando o Protocolo de Controle de Transmissão (TCP - Transmission Control Protocol). Este protocolo fornece um serviço de fluxo *full duplex* [1], com tratamento automático de erro, retransmissão, reordenação de pacotes, e garantia de entrega. Em sistemas telerobóticos, o TCP é freqüentemente usado na transmissão de comandos de controle e outros dados importantes. Para um estudo mais detalhado sobre a teoria e aplicação de protocolos TCP/IP, o leitor deve se reportar a Tanenbaum (2003).

Por outro lado, a baixa taxa de transmissão da Internet restringe o controle em tempo real e o retorno das tarefas de operação (geralmente retorno visual). Como exemplo, o operador não pode observar mudanças em tempo real do robô e do ambiente através da transmissão de imagens de vídeo devido a atrasos da rede. De fato, o uso de vídeo na Internet é difícil, pois a mesma trabalha com a filosofia de melhor esforço, sem prover garantias na largura de banda, além de apresentar atrasos de tempo variáveis e perda de pacotes (APOSTOLOPOULOS; TAN; WEE, 2006).

A largura de banda disponível entre dois pontos na Internet é geralmente desconhecida e variante no tempo. Se um remetente transmitir mais rápido que a largura de banda disponível, então ocorre congestionamento, pacotes são perdidos, e há uma severa queda na qualidade de vídeo.

Os atrasos de tempo variáveis (randômicos) ocorrem porque os dados enviados a uma taxa constante na Internet, chegam a seu destino com um tempo variável. Esta variação, é devido aos efeitos combinados de armazenamento (*buffering*) nos roteadores e de diferentes políticas de roteamento.

A perda de pacotes, por sua vez, pode acontecer quando os *buffers* dos roteadores não conseguem armazenar as quantidades de dados requeridas devido ao congestionamento na rede.

Atrasos de Tempo em STIs

O atraso de tempo é o problema mais crítico em STIs, pois afeta diretamente o resultado das operações telerobóticas. Em teleoperações com atrasos de tempo, o operador não pode observar mudanças em tempo real do robô e do ambiente através da transmissão das imagens de vídeo. Enquanto o operador está esperando por uma resposta do robô, ele não será capaz de submeter o próximo pedido. Para minimizar este período de espera, o tempo de resposta deveria ser o menor possível. O tempo de resposta (tr) é definido pela equação 1.1 apresentada por Zakaria, Amin e Mamat (2000):

$$tr = tp + \frac{De + Dr}{vt} + tc \tag{1.1}$$

[1] Técnica de comunicação de dados capaz de transmitir e receber dados em paralelo.

Onde tp é o tempo de processamento requisitado em segundos, tc é o tempo gasto para iniciar a comunicação (aproximadamente 1 segundo), De e Dr são dados de envio e de retorno (em bytes), respectivamente, e vt é a velocidade de transmissão da rede (em bytes/s). Como não se pode controlar a velocidade de transmissão da Internet, o tempo mínimo de resposta pode ser alcançado através da transmissão mínima de dados (De e Dr). Uma forma de fazer isso é reduzir o tamanho das imagens enviadas ao operador. Assim, uma forma de otimizar a transmissão na rede é reduzir o volume de dados que nela trafega.

STIs com controle baseado em imagens de vídeo tem apresentado grandes atrasos de tempo (de alguns segundos a vários minutos). Isso pode ser constatado nos STIs desenvolvidos por Goldberg, Maschna e Gentner (1995), Goldberg e Santarromana (2006) e Taylor e Trevelyan (1995), que revelaram retardos na ordem de minutos no tempo de resposta de seus sistemas. Como exemplo do uso de imagens de vídeo, considera-se uma imagem de vídeo em preto e branco com resolução de 256 x 256 pixels, com 256 níveis de cinza. Com essas características, são necessários transmitir $65 Kbytes$ de dados da referida imagem na rede a uma taxa mínima de 5 a 6 quadros por segundo para se manter a imagem em movimento sem trazer desconforto ao usuário (TAN; CLAPWORTHY, 2003).

1.2 Realidade Virtual

De forma geral, a RV ou AV, é uma tecnologia de interface avançada entre um usuário e um sistema computacional. O objetivo dessa tecnologia é recriar ao máximo a sensação de realidade para um indivíduo, levando-o a adotar essa interação como uma de suas realidades temporais. Para isso, essa interação é realizada em tempo real, com o uso de técnicas e de equipamentos computacionais que ajudem na ampliação do sentimento de presença do usuário.

A RV também pode ser considerada como a junção de três idéias básicas: imersão, interação e envolvimento (MORIE, 1994). A idéia de imersão está ligada com o sentimento de se estar dentro do ambiente. A idéia de interação está ligada com a capacidade do computador detectar as entradas do usuário e modificar instantaneamente o AV e as ações sobre ele. A idéia de envolvimento, por sua vez, está ligada com o grau de motivação para o engajamento de uma pessoa com determinada atividade.

1.2.1 Sistemas de Telepresença e de Realidade Virtual

Os sistemas de telepresença e de RV são semelhantes na parte em que envolvem os usuários e as interfaces de *hardware* e *software* muito elaboradas. Eles diferem na atuação sobre o ambiente. Enquanto a telepresença faz com que a interface atue sobre o telerobô no mundo real, o sistema de RV faz com que a interface atue diretamente sobre o computador no AV.

Telepresença estende as capacidades motoras e sensoriais de um operador, bem como a suas habilidades de solução de problemas, para um ambiente remoto (KIRNER, 2006). A telepresença que é implementada por mecanismos de teleoperação, consiste de um usuário, uma interface humano-máquina, um telerobô e um ambiente remoto, conforme a Figura 1.2.

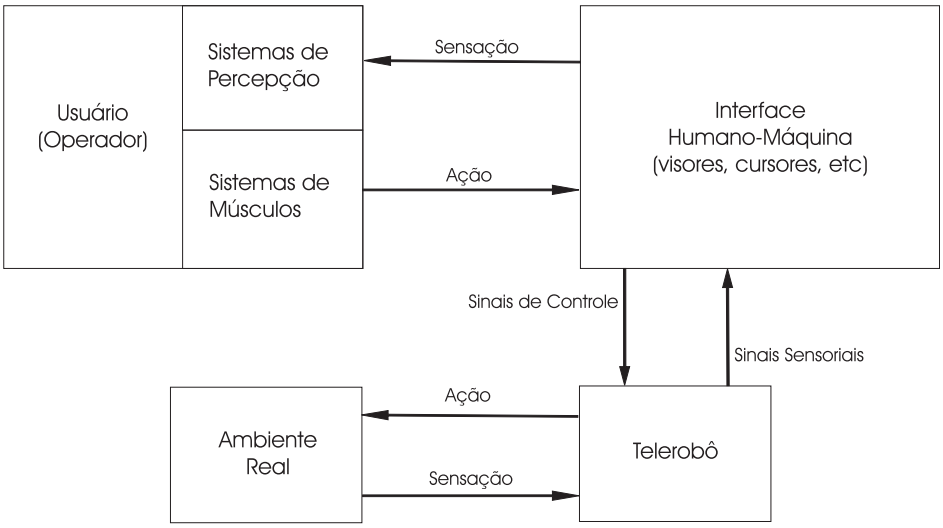

Figura 1.2: Sistema de Telepresença

Os sistemas de RV, por sua vez, consistem de um usuário, uma interface humano-máquina, e um computador, conforme a Figura 1.3. O usuário participa de um AV gerado no computador, usando dispositivos sensoriais de percepção e controle. Um AV pode ser projetado para simular tanto um ambiente imaginário quanto um ambiente real.

Em telepresença e em outros casos, onde possa haver dificuldades de transferência ou tratamento em tempo real de imagens reais complexas, a substituição do mundo real por um AV equivalente pode resolver o problema, na medida em que as imagens podem ser geradas localmente. As transferências de informações podem ser reduzidas a dados de posicionamento.

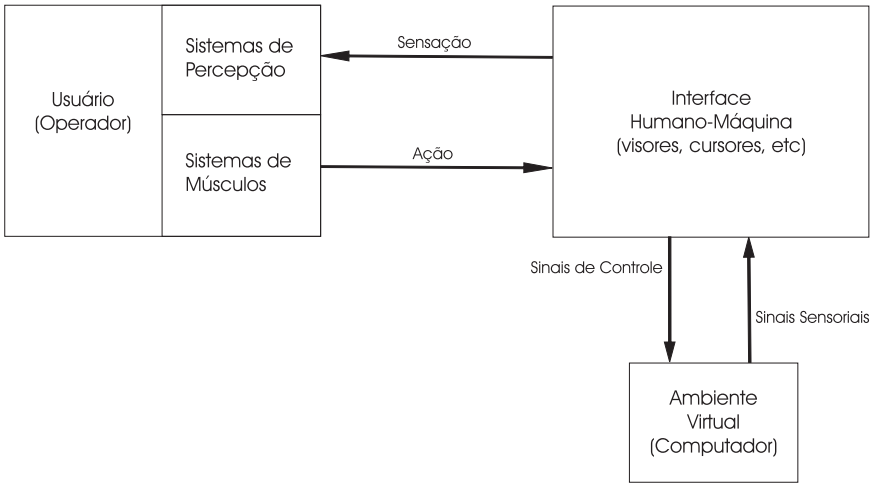

Figura 1.3: Sistema de Ambiente Virtual (Realidade Virtual)

Genericamente, os sistemas de telepresença e de RV podem enquadrar-se conforme a Figura 1.4. Geralmente, o usuário é projetado dentro de um ambiente novo e interativo, através de dispositivos eletrônicos não convencionais (Luvas de Dados, Vídeo-Capacete). Tanto o desempenho, quanto a experiência do usuário no novo ambiente, dependem fortemente da interface homem-máquina e das características de interação com o ambiente real ou virtual (KIRNER, 2006).

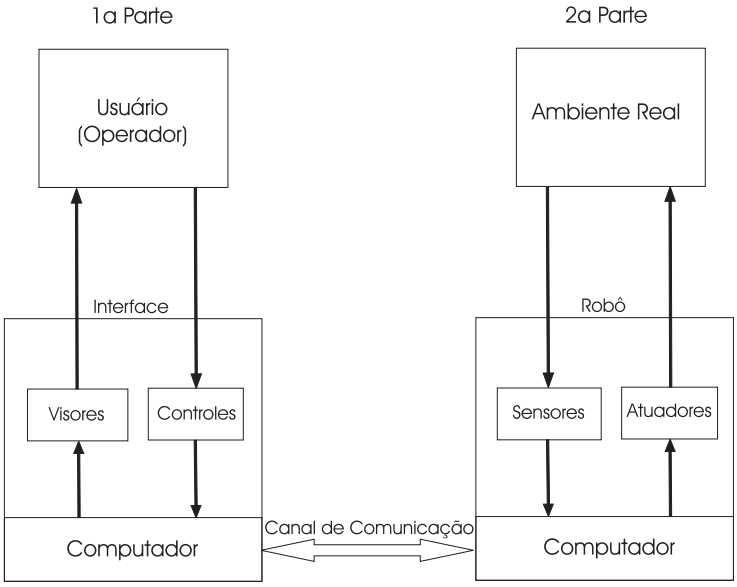

Figura 1.4: Configuração Genérica dos Sistemas de RV e de Telepresença

A particularização da configuração pode ser feita através de quatro maneiras:

a) Se a primeira parte for desprezada, a segunda parte sozinha poderá transformar-se num robô autônomo, considerando-se que o computador seja utilizado para realizar seu controle;

b) Se a segunda parte for desprezada e o computador for usado para gerar um AV, o sistema será visto como um sistema de RV;

c) Se as duas partes forem consideradas, mas o computador da primeira parte não for usado para gerar AVs, limitando-se a repassar os sinais de sensação e controle, o sistema funcionará como um sistema de telepresença;

d) Se as duas partes forem consideradas, e o computador da primeira parte for usado para gerar AVs, o sistema funcionará como um sistema simulado de telepresença.

1.2.2 Estrutura de um Sistema de Realidade Virtual

A estrutura de um sistema de RV pode ser mostrada sob diferentes pontos de vista e graus de detalhamento. O diagrama de blocos da Figura 1.5 fornece uma visão geral de um sistema de realidade virtual.

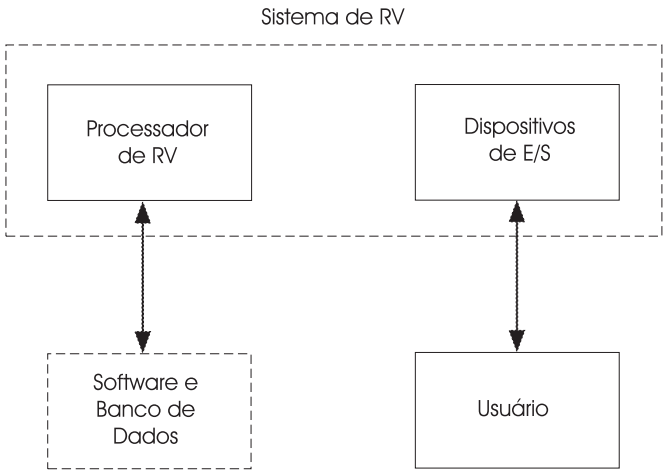

Figura 1.5: Diagrama de Blocos de um Sistema de RV

A interação do usuário com o processador de RV é intermediada pelos dispositivos de E/S. O processador de RV lê primeiramente a entrada do usuário e acessa o banco de dados para calcular as instâncias do mundo que correspondem aos quadros a serem mostrados em seqüência. Como não é possível prever as ações do usuário, os quadros devem ser criados e distribuídos em tempo real. Este banco de dados contém a descrição dos objetos do AV junto com a descrição dos movimentos dos objetos, seus comportamentos, efeitos de colisões, etc. Devido a necessidade de acesso e operação em tempo real, é necessário dispor-se da quantidade de memória suficiente, bem como usar técnicas de compressão de informação que não prejudiquem as restrições de tempo. As imagens devem ser geradas com um atraso aceitável para não provocar desconforto ao usuário, sendo o mínimo aceitável da ordem de 8 a 10 quadros por segundo.

1.2.3 Modelagem de Ambientes Virtuais

A modelagem de AVs é de fundamental importância num sistema de RV, definindo as características dos objetos como: forma; aparência; comportamento; restrições; e mapeamento de dispositivos de Entrada/Saída. Para isto, os sistemas de desenvolvimento de RV levam em conta os diversos aspectos de modelagem, mapeamento e simulação, conforme a Figura 1.6.

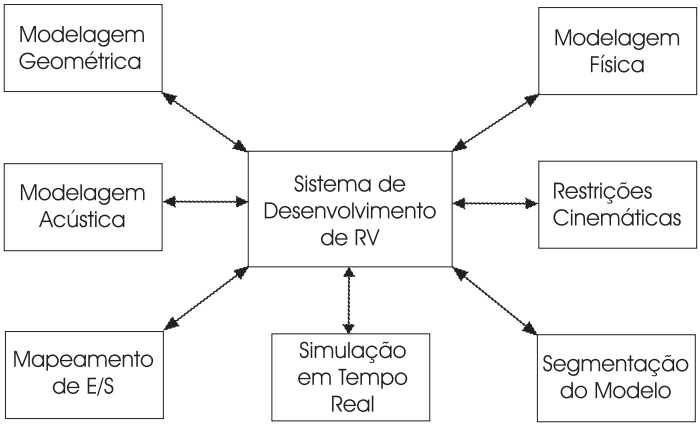

Figura 1.6: Sistema de Desenvolvimento de RV

A modelagem geométrica abrange a descrição da forma dos objetos virtuais através de polígonos, triângulos ou vértices, e sua aparência, usando textura, reflexão da superfície, cores, etc. A forma poligonal dos objetos pode ser criada, usando-se bibliotecas gráficas, como a biblioteca OpenGL (OPENGL, 2007) [2], ou usando-se modelos prontos de bancos de dados comerciais ou digitalizadores tridimensionais. Os objetos também podem ser criados por programas de Desenho Auxiliado por Computador (CAD - Computer Aided Design), como o AutoCAD© ou 3-D Studio©, ou com o uso de editores de RV, tais como,

A aparência dos objetos está relacionada principalmente com as características de reflexão da superfície e com sua textura. A textura dos objetos é obtida a partir do mapeamento de um padrão de textura do espaço bidimensional sobre os objetos tridimensionais. A textura oferece várias vantagens para a RV, uma vez que aumenta o nível de detalhe e de realismo de cena. Ela fornece melhor visão de profundidade, e permite a redução substancial do número de polígonos da cena, propiciando o aumento da taxa de quadros por segundo (AMES; NADEAU; MORELAND, 1997).

A modelagem cinemática trata com a animação de objetos, no que diz respeito, a alteração de posição, mudança de escala, detecção de colisões, etc. A utilização de coordenadas locais dos objetos e de coordenadas gerais, juntamente com matrizes de transformação, permitem a alteração das posições e as mudanças de escala.

[2] Biblioteca de rotinas gráficas para trabalhar em duas e três dimensões.

Visando a obtenção de realismo nos AVs, é necessário se fazer a modelagem física dos objetos virtuais. No mínimo, os objetos sólidos não poderão passar um pelo outro e as coisas deverão mover-se de acordo com o esperado, quando puxadas, empurradas, agarradas, etc. Nesse sentido, os objetos virtuais também deverão ser modelados fisicamente pela especificação de suas massas, pesos, inércia, texturas (lisas ou ásperas), deformações (elásticas ou plásticas), etc. Essas características, juntas com a modelagem geométrica e com as leis de comportamento, determinam uma modelagem virtual próxima da realidade.

A modelagem geométrica e física de AVs com muitos objetos deverá resultar em um modelo muito complexo, difícil e caro de ser mostrado. Normalmente, esses ambientes possuem vários espaços específicos, distâncias razoáveis e objetos móveis com velocidades diferentes. O problema da complexidade pode ser contornado por segmentação do ambiente (AMES; NADEAU; MORELAND, 1997), alteração do nível de detalhe dos objetos, alteração de resolução de imagens, pré-computação, etc. A segmentação do ambiente baseia-se na divisão do ambiente geral em ambientes menores, de forma que somente os objetos do ambiente menor sejam mostrados. É o caso de uma casa com diversas salas, onde cada sala é um ambiente menor. Embora o ambiente geral seja muito complexo, a visão do usuário sempre será a mais simples.

1.2.4 Aplicações de Realidade Virtual

A RV pode ser aplicada nas mais variadas áreas do conhecimento e de maneira bastante diversificada. A todo momento surgem novas aplicações, em função da demanda e da capacidade criativa das pessoas. Em muitos casos, a RV vem revolucionando a forma de interação das pessoas com sistemas complexos tratados com o uso de computadores, propiciando maior desempenho e economizando custos.

Dentre as várias áreas, no qual a RV vem sendo aplicada, pode-se citar as seguintes extraídas da referência de Netto, Machado e Oliveira (2007):

a) Visualização Científica: Visualização de superfícies planetárias; Túnel de vento virtual; Síntese molecular; etc.

b) Aplicações Médicas e em Saúde: Simulação cirúrgica; Planejamento de radioterapia; Saúde Virtual; Ensino de anatomia; Visualização médica; Tratamento de deficientes; etc.

c) Arquitetura e Projeto: CAD; Projeto de artefatos; Planejamento; Decoração; Avaliação acústica; etc.

d) Educação: Laboratórios virtuais; Exploração planetária; Educação à distância; Educação de excepcionais; etc.

e) Entretenimento: Turismo virtual; Passeio ciclístico virtual; Jogos; Cinema virtual; etc.

f) Treinamento: Simuladores de vôo; Planejamento de operações militares; Treinamento de astronautas; etc.

g) Artes: Pintura; Escultura virtual; Música; Museu virtual; etc.

h) Controle da Informação: Visualização financeira; Visualização da informação; Informação virtual; etc.

i) Telepresença: Telerobótica; Teleconferência; Professor virtual; Espectador remoto; etc.

1.2.5 Linguagens para Modelagem de Ambientes Virtuais

Hoje em dia existem diferentes linguagens para modelagem de AVs. Nesta seção, esboça-se algumas dessas tecnologias, expondo suas principais características.

VRML é a abreviação de *Virtual Reality Modeling Language*, ou Linguagem para Modelagem de AVs. É uma linguagem independente de plataforma que permite a criação de cenários 3D, por onde se pode passear, visualizar objetos por ângulos diferentes e interagir com eles. A linguagem tem como objetivo dar o suporte necessário para o desenvolvimento de AV tridimensionais multi-usuários na Internet, sem precisar de redes de alta velocidade (VRML, 2006).

O VRML, se tornou de fato, uma ferramenta no fornecimento de conteúdo 3D na *web*. Além disso, ferramentas de modelagem 3D são capazes de exportar arquivos VRML, de forma que profissionais sem nenhuma experiência na linguagem podem exportar seus modelos 3D para a *web*. Esta característica tem permitido o uso de VRML nas áreas acadêmica e industrial nas mais diversas aplicações, tais como, visualização 3D de estruturas químicas moleculares (CASHER et al., 2007), sistemas de manufatura (SILVA; PERKUSICH, 2003), simulação robótica e teleoperação (BURDEA, 1999).

De uma forma geral, um ambiente VRML é uma estrutura tipo árvore, como ilustrado na Figura 1.7. Quase tudo em VRML é visto como um *nó* (node). Cilindros, esferas, luz e mesmo cores. Contudo, esses *nós* não tem comportamentos associados a eles, sendo assim, *nós* estáticos. Por isso, afim de incluir alguma dinâmica em tais cenas, foram criados três novos mecanismos: *sensors*(sensores), *routes* (rotas) e *scripts*. Um *sensor*, como o próprio nome diz, detecta a sensação de mudanças de uma determinada ação específica, como por exemplo, o tempo e o clique de *mouse*. A informação adquirida por um sensor pode ser re-direcionada a outros *nós* no ambiente 3D, através de *routes* (rotas). Estes *routes*, entretanto, agem como um evento de mecanismo de despacho. Finalmente, *scripts*, também considerados *nós*, são a forma pela qual o desenvolvedor manipula os

objetos de uma cena de VRML usando uma *API*(Application Programming Interface - Interface de Programação de Aplicativos) de *script* interna.

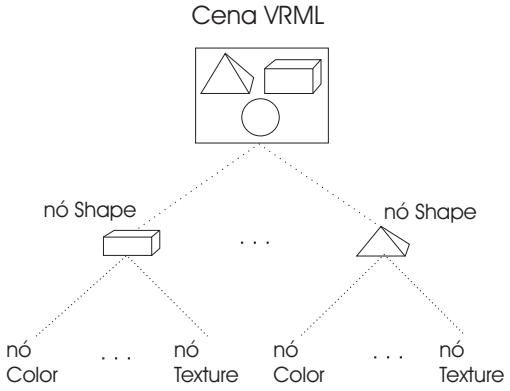

Figura 1.7: Estrutura Gráfica do VRML

A *API* Java3D faz parte do pacote *JavaMedia* da Sun© (SUN, 2006) e é dedicado a construção de conteúdo 3D em aplicações *applets* [3] java. Como em VRML, cenas em Java3D também são estruturas tipo árvore. Contudo, cada objeto em uma cena é visto como um *nó* da árvore. Através do Java3D é possível controlar a forma, cor e transparência dos objetos. Além disso, ela permite que o desenvolvedor defina o comportamento dos objetos (como se movem, rotacionam, encolhem, esticam, etc). Manipulações de ambientes também são possíveis, definindo imagens de fundo, modos de iluminação, e até afeitos de nevoeiro (fog).

A linguagem X3D é um padrão aberto para a entrega de conteúdo 3D (X3D, 2006). Ela é a revisão da especificação ISO do VRML97 (VRML, 2006), incorporando os últimos avanços em características de *hardware* de gráficos comerciais, bem como, melhorias arquitetônicas baseadas em anos de *feedback* da comunidade de desenvolvimento de VRML97. Como X3D é compatível com a especificação VRML, arquivos X3D podem ser modelados usando tanto a sintaxe X3D como a sintaxe VRML.

Diferentemente das linguagens anteriores, o X3D emprega uma arquitetura modular que provê maior extensibilidade e flexibilidade. X3D se baseia na noção de perfis (profiles), cada um fornecendo um conjunto específico de características. Os perfis são baseados em uma arquitetura componentizada, permitindo a inserção de novos componentes quando necessários. Isto permite a definição de novos perfis voltados a certas aplicações específicas, como por exemplo, visualização médica.

A estrutura gráfica do X3D é baseada também no modelo tipo árvore. De uma perspectiva funcional, o X3D fornece muitas das funcionalidades do VRML e Java3D, tais como, *scripts*, *lighting* (iluminação), *materials* (materiais), e *animations* (animações).

[3] Programa Java que executa em um navegador Web.

Todavia, uma característica interessante do X3D é a possibilidade de construção de AVs compostos inteiramente de objetos de outras cenas ou de localizações da *web*.

1.2.6 Exemplos de Ambientes Virtuais

Nesta subseção são apresentados dois exemplos de AV na linguagem X3D. O primeiro exemplo é baseado na forma primitiva cilindro e seu resultado é ilustrado na Figura 1.8(a). Outras formas primitivas (caixa, cone e esfera) seguem a mesma analogia do exemplo anterior. No segundo exemplo é mostrado como a sobreposição de formas primitivas podem criar figuras complexas e interessantes, como por exemplo, uma estação espacial ilustrada na Figura 1.8(b).

Basicamente a estrutura de um arquivo X3D é composta pelos seguintes componentes:

- Cabeçalho do Arquivo - $\#X3DV3.0utf8$;
- Perfil - PROFILE (Interchange, Interactive, Immersive e Full);
- Comentários - Inicia com o símbolo #;
- nó - Descreve formas, tipo de iluminação, som, etc;
- Campos - Atributos do nó;
- Valores - Valores dos atributos;

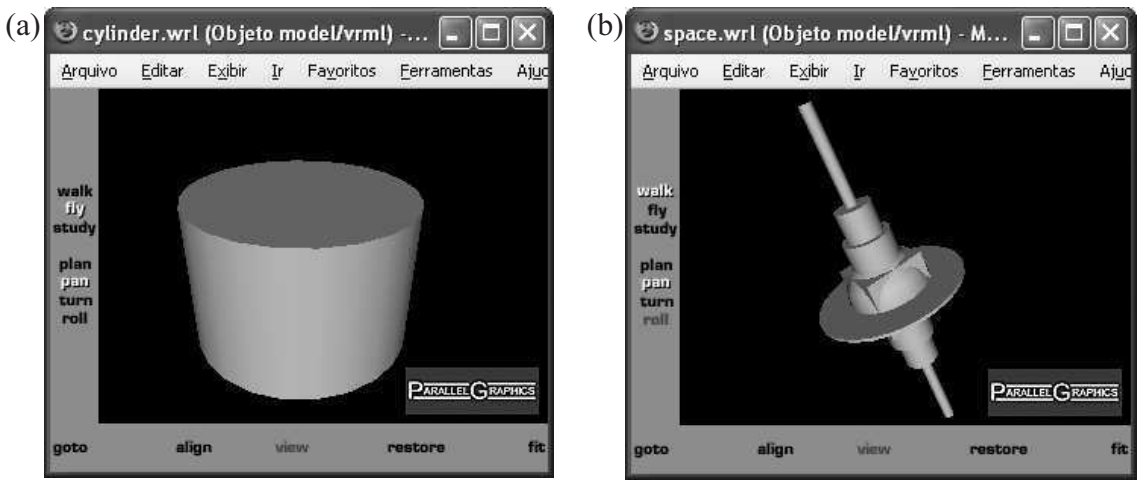

Figura 1.8: Exemplo de AV: (a)Forma primitiva cilindro; (b)Forma complexa como resultado da sobreposição de figuras primitivas.

O código em X3D abaixo, na sintaxe VRML, se refere ao exemplo da Figura 1.8(a):

```
#X3D V3.0 utf8
PROFILE Interactive
# Exemplo de Cilindro
Shape {
    appearance Appearance {
        material Material { }
    }
    geometry Cylinder {
        height 2.0
        radius 1.5
    }
}
```

1.3 Processamento Digital de Imagem

Os métodos de Processamento Digital de Imagem (PDI) são orientados para duas áreas de aplicação: melhoramento de imagens objetivando aumentar o nível de informação presente para posterior interpretação humana; e processamento de dados de uma cena para percepção autônoma de máquinas.

A primeira área surgiu no inicio dos anos 20 com a necessidade de se enviar fotos jornalísticas via cabo submarino de Nova York a Londres e hoje é aplicada para a resolução de uma variedade de problemas em diversos campos, tais como, medicina, arqueologia, física, astronomia, entre outros.

A segunda área, tem como foco de interesse procedimentos para extração de informações de imagens em uma forma adequada ao processamento computacional. Alguns problemas típicos desta área são reconhecimento automático de caracteres, máquinas industriais com visão para acompanhamento da montagem de produtos, reconhecimento militar, entre outros.

Nesta seção são abordados assuntos sobre PDI das referências de Gonzalez e Woods (1992), Oliveira (2007), Filho e Neto (1999) e Britto et al. (2007).

1.3.1 Representação de Imagens Digitais

Uma imagem monocromática pode ser descrita por uma função bidimensional de intensidade da luz *f(x,y)*, onde x e y denotam as coordenadas espaciais e o valor de f em qualquer ponto *(x,y)* é proporcional ao brilho (ou níveis de cinza) da imagem naquele ponto. Na Figura 1.9 é ilustrado a convenção dos eixos utilizada ao longo deste trabalho.

A função *f(x,y)* representa o produto da interação entre a iluminância *i(x,y)* e as propriedades de reflectância *r(x,y)*. A iluminância exprime a quantidade de luz que incide

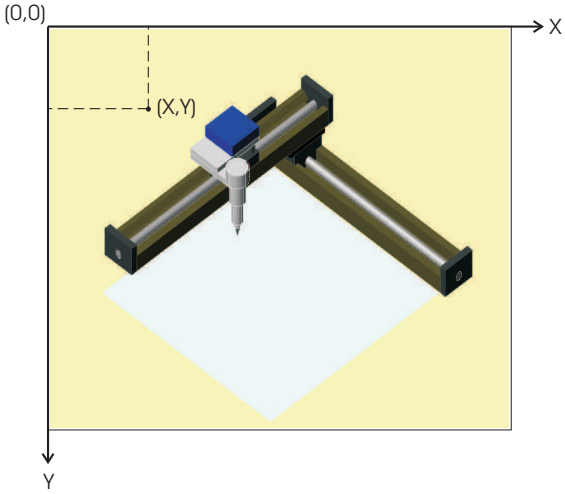

Figura 1.9: Convenção dos eixos para representação de Imagens Digitais

sobre o objeto, ao passo que, a reflectância exprime a fração de luz incidente que o objeto vai transmitir (refletir) no ponto *(x,y)*. Matematicamente, *f(x,y)* pode ser representada pela seguinte equação:

$$f(x,y) = i(x,y) \cdot r(x,y) \tag{1.2}$$

com:
$0 < i(x,y) < \infty$ e
$0 < r(x,y) < 1$

A situação onde *r(x,y)* é nulo indica que o objeto iluminado absorveu toda a luz incidente, ocorrendo justamente o contrário quando *r(x,y)* vale 1 (um), isto é, o objeto reflete toda luz que nele incide. O valor de *i(x,y)* é determinado principalmente pela fonte de luz, que na grande maioria dos casos trata-se da luz do sol. Nas Tabelas 1.1 e 1.2 são apresentados valores típicos de iluminância e reflectância.

Tabela 1.1: Exemplos de valores para *i(x,y)* em lux

i(x,y)	Exemplos
900	*dia ensolarado*
100	*dia nublado*
10	*iluminação média de escritório*
0,001	*noite clara de lua cheia*

A intensidade de uma imagem monocromática f nas coordenadas *(x,y)* é denominada nível de cinza (ou tom de cinza) (L) da imagem naquele ponto. Este valor está no intervalo:

Tabela 1.2: Exemplos de valores para *r(x,y)*

r(x,y)	Exemplos
0,93	*neve*
0,80	*parede branca fosca*
0,65	*aço inoxidável*
0,01	*veludo preto*

$$L_{min} \leq L \leq L_{max}$$

sendo L_{min} e L_{max} valores positivos e finitos. O intervalo [L_{min},L_{max}] é denominado escala de cinza da imagem.

No caso de uma imagem colorida é necessário uma função *f(x,y)* para cada banda de freqüência. As imagens coloridas padrão RGB são formadas pela informação de cores primárias aditivas, como o vermelho (R - Red), o verde (G - Green) e o azul (B - Blue).

1.3.2 Amostragem e Quantização de Imagens

Para converter uma imagem real em imagem digitalizada, duas etapas são imprescindíveis: a aquisição da imagem e sua digitalização.

O primeiro passo na conversão de uma cena tridimensional em uma imagem eletrônica bidimensional é a redução de dimensionalidade. Os dispositivos usados nessa conversão são as câmeras fotográficas e câmeras de vídeo. O dispositivo de aquisição de vídeo consiste de uma matriz de células semicondutoras fotossensíveis, que atuam como capacitores, armazenando carga elétrica proporcional a energia luminosa incidente.

O passo de digitalização compreende a discretização espacial e em amplitude do sinal analógico de vídeo obtido do dispositivo de aquisição. O processo de discretização espacial é chamado de amostragem e o processo de discretização em amplitude é chamado de quantização.

Basicamente a amostragem converte a imagem analógica em uma matriz de M por N pontos, cada qual denominado *pixel* [4]:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

[4]Elementos da imagem. Abreviação de "picture elements".

Maiores valores de M e N implicam em uma imagem de maior resolução.

Por outro lado, a quantização faz com que cada um destes *pixels* assuma um valor inteiro, na faixa de 0 a $2^n - 1$. Quanto maior o valor de n, maior o número de níveis de cinza presentes na imagem digitalizada.

1.3.3 Etapas do Processamento Digital de Imagens

Pode-se entender uma imagem como uma forma compacta de representar muitas informações. Em um sistema de processamento de imagens estas informações podem passar por diversas formas de representação. Portanto, as etapas do processamento de imagens descrevem o fluxo destas informações com um dado objetivo definido pela aplicação. Na Figura 1.10 é ilustrado este fluxo de forma geral.

Figura 1.10: Etapas do Processamento de Imagens Digitais

O primeiro passo no processo é a aquisição da imagem. Para tanto, são necessários um sensor e um digitalizador. O sensor converterá a informação óptica em sinal elétrico e o digitalizador transformará a imagem analógica em imagem digital. O sensor pode ser uma câmera de vídeo monocromática ou colorida que produza uma imagem inteira do domínio do problema a cada 1/30 s. Esta etapa produz na saída uma imagem digitalizada da cena.

A imagem resultante do passo anterior pode apresentar diversas imperfeições, tais como, presença de *pixels* ruidosos, contraste e/ou brilho inadequado, etc. A função da etapa de pré-processamento é aprimorar a qualidade da imagem para as etapas subseqüentes. As operações efetuadas nessa etapa são ditas de baixo nível porque trabalham diretamente com os valores de intensidade dos *pixels*. A imagem resultante desta etapa é uma imagem digitalizada de melhor qualidade que a original.

O próximo estágio trata da segmentação. Definida em termos gerais, a segmentação de imagens particiona a imagem em regiões disjuntas com algum significado para a aplicação. Por exemplo, pode-se querer separar um objeto de interesse do resto dos *pixels* da imagem particionando-a em duas regiões. A saída da segmentação pode ser a fronteira do objeto com seu exterior ou os pontos de seu interior. Isto define duas formas de representação para o objeto. A representação consiste, portanto, das várias formas de armazenar a fronteira e o interior de objetos segmentados.

Em seguida, o processo de representação e descrição, consiste das várias formas de armazenar a fronteira e o interior de objetos segmentados. Esta nova representação da imagem contém informações sobre a forma e a topologia dos objetos. A descrição quantitativa destas informações através da extração de características estruturais complementa o sentido de representação.

No último estágio, com base na descrição, o reconhecimento associa um rótulo a cada objeto segmentado enquanto a interpretação associa um significado ao conjunto de objetos segmentados. Um exemplo em uma aplicação de leitura automática de endereços é o reconhecimento e a interpretação de um conjunto de caracteres como o código de endereçamento postal.

1.3.4 Processamento de Imagem Colorida

O uso de cor em processamento de imagens é motivado por dois principais fatores:

- Na análise automática de imagens (reconhecimento de padrões), a cor é um poderoso descritor das propriedades de um objeto, que pode simplificar sua identificação e segmentação;
- Na análise de imagens com intervenção humana, o ser humano pode discernir milhares de tonalidades de cores de diferentes matizes e intensidades, enquanto sua capacidade de distinguir diferentes tons de cinza não passa de algumas poucas dúzias de tons diferentes.

Embora o processo psicofisiológico de percepção de cor pelo cérebro humano ainda não seja totalmente compreendido, os aspectos físicos da cor vêm sendo estudados há muitos anos por inúmeros pesquisadores e engenheiros, constituindo hoje uma base formal de resultados experimentais e teóricos.

As cores que os humanos percebem em objetos são determinadas pela natureza da luz refletida dos mesmos. Se por exemplo, um objeto parece verde à luz do dia é porque ele reflete somente a parte verde da luz; o restante do espectro é absorvido. A teoria da percepção cromática pelo olho humano baseia-se em hipótese formulada por Thomas Young em 1801, que estabelece que os cones (6 a 7 milhões de células foto-sensíveis responsáveis pela visão de cor) se subdividem em três classes que correspondem ao vermelho, verde e azul (GONZALEZ; WOODS, 1992). Aproximadamente 65% de todos os cones são sensíveis a luz vermelha, 33% são sensíveis a luz verde, e apenas 2% são sensíveis ao azul (mas os cones deste tipo são os mais sensíveis). Desta forma, todas as sensações de cor percebidas pelo olho humano são na verdade combinações das intensidades dos estímulos recebidos por cada um destes tipos de cones. Estas três cores são denominadas cores primárias

aditivas, pois é possível obter qualquer outra cor a partir de uma combinação aditiva de uma ou mais delas, em diferentes proporções.

Um exemplo clássico de dispositivo que opera sobre o princípio da combinação aditiva de cores é o monitor de vídeo, que possui em sua superfície pontos triangulares compostos de fósforos sensíveis a cada uma das cores primárias. Cada tipo de fósforo de cada ponto da tela é bombardeado por um feixe eletrônico cuja intensidade é proporcional à quantidade de vermelho, verde ou azul naquele ponto da imagem. O efeito, visto pelo monitor de vídeo, é que as três cores primárias de cada tríade de fósforo são "adicionadas"e recebidas pelos cones do olho humano e a imagem colorida correspondente é então percebida.

A mistura das cores primárias, duas a duas, produz as chamadas cores secundárias, que são: magenta (vermelho e azul), amarelo (vermelho e verde) e ciano (verde e azul).

1.3.5 Modelos de Cores: RGB e HSI

O objetivo dos modelos de cores é permitir a especificação de cores em um formato padronizado aceito por todos. Essencialmente, um modelo de cor é uma especificação de um sistema de coordenadas tridimensionais e um subespaço dentro deste sistema onde cada cor é representada por um único ponto.

Modelo de Cor RGB

O modelo RGB (R - Vermelho, G - Verde e B - Azul) é baseado em um sistema de coordenadas cartesianas, que pode ser visto como um cubo onde três de seus vértices são as cores primárias, outros três as cores secundárias, o vértice junto a origem é o preto e o mais afastado da origem corresponde à cor branca, conforme ilustrado na Figura 1.11. Nesse modelo, a escala de cinza estende-se do preto até o branco ao longo da linha juntando estes dois pontos, e as cores são pontos sobre ou dentro do cubo, definidas por vetores estendendo-se a partir da origem. Por conveniência, assume-se que os valores máximos de R, G e B estão normalizados no intervalo [0,1].

Imagens no modelo de cor RGB consistem em três planos de imagem independentes, um para cada cor primária. Quando alimentadas num monitor RGB, essas três imagens combinam-se sobre a tela fosfórea para produzir uma imagem de cores compostas. Assim, o uso do modelo RGB para processamento de imagens faz sentido quando as imagens são naturalmente expressas em termos de planos de três cores. Este modelo é ideal para geração de imagem colorida (como na captura de imagem por câmera ou na exibição de imagem em monitor de vídeo), mas o seu uso na descrição de cor é muito limitado.

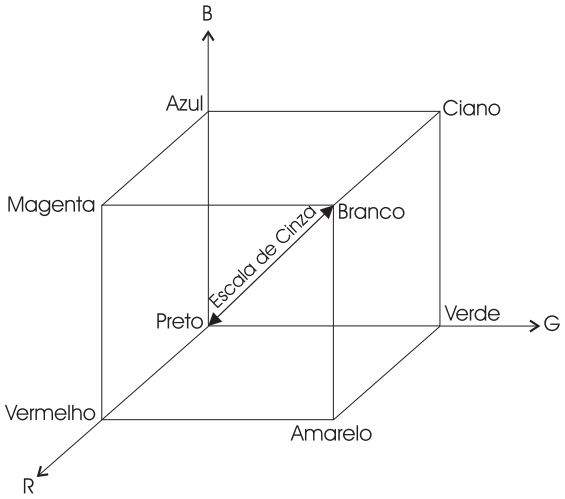

Figura 1.11: Modelo RGB

Modelo de Cor HSI

O modelo HSI (Hue, Saturation, Intensity - Matiz, Saturação, Intensidade) é de grande utilidade, uma vez que, permite separar as componentes de matiz, saturação e intensidade da informação de cor em uma imagem, da forma como o ser humano as percebe. Quando uma pessoa vê a cor de um objeto, ela tende a descrevê-la por meio de sua matiz, saturação e brilho (intensidade).

O componente matiz é um atributo associado com o comprimento de onda predominante em uma mistura de ondas de luz. Assim o matiz representa a cor dominante como percebida por um observador.

A componente saturação expressa a pureza do matiz ou, em outras palavras, o grau de mistura do matiz original com a luz branca. As cores do espectro puro são completamente saturadas. Cores como rosa e o vermelho, por exemplo, têm o mesmo matiz, mas apresentam diferentes graus de saturação. O grau de saturação é inversamente proporcional à quantidade de luz branca adicionada.

O componente brilho é um descritor subjetivo que é praticamente impossível de se mensurar. Ele incorpora a noção cromática da intensidade e é um dos fatores principais na descrição da sensação de cor.

Geometricamente, o modelo HSI pode ser visto como uma estrutura piramidal de três lados ilustrada na Figura 1.12. Qualquer ponto na superfície dessa estrutura representa uma cor puramente saturada. O matiz dessa cor é determinado por seu ângulo com respeito ao eixo vermelho e sua intensidade por sua distância perpendicular a partir do ponto preto (isto é, quanto maior a distância do preto, maior será a intensidade da cor). A situação é análoga aos pontos dentro da estrutura, com a única diferença, de que, as cores tornam-se menos saturadas conforme se aproximam do eixo vertical de intensidade

Preto-Branco.

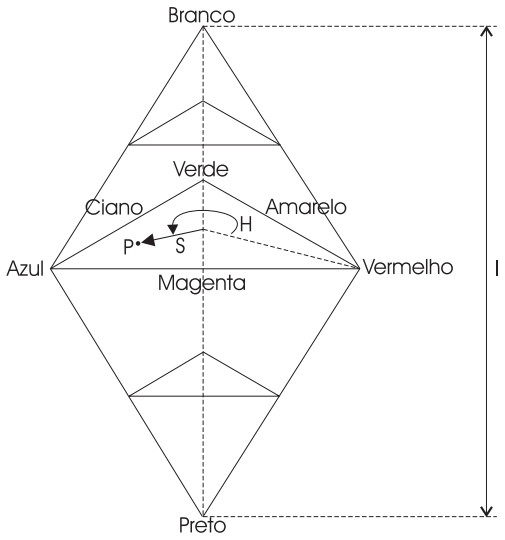

Figura 1.12: Modelo HSI

Os componentes matiz (H) e saturação (S) são definidos respectivamente ao triângulo central ilustrado na Figura 1.12. Nota-se, que o matiz H do ponto de cor P é o ângulo do vetor mostrado com respeito ao eixo vermelho. Assim, quando $H = 0^o$, a cor é vermelha, quando $H = 60^o$, a cor é amarela, e assim por diante. A saturação S, do ponto de cor P é o grau em que a cor não está diluída pelo branco e é proporcional à distância de P ao centro do triângulo. Quanto mais distante P for do centro do triângulo, mais saturada será a cor.

1.3.6 A Cor no Processo de Segmentação da Imagem

Em PDI, segmentar consiste em identificar e extrair estruturas homogêneas presentes em uma cena. Essas estruturas podem ser identificadas a partir de características como forma, geometria, topologia, textura, cor ou brilho, sendo escolhidas aquelas que possibilitam melhor distinção. A maior dificuldade normalmente encontrada está no fato de não haver conhecimento *a priori* do número e tipo de estruturas presentes na imagem. Portanto não é possível uma teoria completa sobre segmentação de imagens. A grande maioria das técnicas disponíveis são *ad hoc* e diferem entre si nas propriedades que enfatizam, como contornos e regiões fechadas (HARALICK; SHAPIRO, 1992).

O atributo cor possui um grande poder discriminatório, sendo por isso freqüentemente utilizado na identificação de estruturas ou objetos. Em diversas aplicações o homem faz uso da informação cor. Isto se deve ao aumento da capacidade de discernimento visual, quando a cor está presente.

Ohta, Kanade e Sakai (1980) demonstraram que no modelo HSI, a informação cor é

mais descorrelacionada que no caso do modelo RGB. Isso acontece, porque o componente de intensidade I é desacoplado da informação de cor (Matiz) da imagem, deixando assim, a segmentação menos sensível às mudanças de iluminação. O modelo RGB não permite tal flexibilidade.

De forma geral, os modelos de representação de cor mais adequados para o processo de segmentação são aqueles cujos componentes mostram-se menos correlacionados. A adequação do modelo HSI, durante a segmentação, deve-se à baixa correlação de suas componentes matiz, saturação e intensidade. Isto não acontece com o modelo RGB, que apresenta facilidades para a computação da informação cor, mas tem o inconveniente de possuir componentes altamente correlacionados, responsáveis, na maioria das vezes, pela segmentação espúria da imagem.

Na prática, o modelo HSI é deduzido do modelo RGB pelas seguintes relações (GONZALEZ; WOODS, 1992):

$$I = \frac{1}{3}(R + G + B) \tag{1.3}$$

$$S = 1 - \frac{3}{R + G + B}[min(R, G, B)] \tag{1.4}$$

$$H = \arccos \frac{\frac{1}{2}[(R - G) + (R - B)]}{[(R - G)^2 + (R - B)(G - B)]^{1/2}} \tag{1.5}$$

onde, se $(B/I) > (G/I)$, deve-se subtrair 360^o de H. A fim de normalizar a faixa de matiz, deve-se dividir H por 360^o.

1.4 Redes de Petri

As Redes de Petri (RPs) são uma ferramenta matemática com uma representação gráfica. Estas podem ser utilizadas para descrever e estudar sistemas de processamento da informação caracterizados como sendo concorrentes, assíncronos, distribuídos, paralelos, não determinísticos e/ou estocásticos (MURATA, 1989). Formalmente, uma RP é tripla *(P,T,F)* onde:

- P é um conjunto finito de lugares;
- T é um conjunto finito de transições;
- $F \subseteq (P \times T) \bigcup (T \times P)$ é uma relação de fluxo;

- $P \cap T = \varnothing$.

Uma RP é composta por uma *estrutura de rede*, *inscrições* associadas a essa estrutura e uma *marcação*. A estrutura de uma RP é representada por uma grafo bipartido direcionado, cujos *nós* são de dois tipos: lugares e transições. Os lugares podem representar estados ou recursos e as transições ações ou eventos.

Nota-se que, de acordo com a relação de fluxo F, os arcos da estrutura sempre conectam *nós* de tipos diferentes. Graficamente, os lugares da estrutura de uma RP são representados por círculos, as transições por retângulos e a relação de fluxo por arcos direcionados.

A cada arco pode ser associado um número inteiro positivo que é denominado peso ou ponderação do arco. A marcação de uma RP determina seu estado e corresponde à associação de elementos, denominados fichas, aos lugares. A marcação atribui a cada lugar um inteiro não negativo. Se a marcação atribui a um lugar p um inteiro não negativo k, então p é marcado com k fichas. Graficamente, estas fichas são representadas por k pontos localizados dentro do lugar p.

A semântica de uma RP é definida por sua regra de disparo. Para uma RP Lugar/Transição tem-se que:

1. uma transição está habilitada se cada um de seus lugares de entrada contém um número de fichas maior ou igual ao especificado no peso do respectivo arco de entrada;

2. uma transição habilitada pode disparar ou não;

3. quando uma transição habilitada dispara são removidas de seus lugares de entrada a quantidade de fichas indicada em seus respectivos arcos de entrada e são adicionadas fichas nos lugares de saída observando os pesos dos respectivos arcos de saída.

Por exemplo, a RP Lugar/Transição ilustrada na Figura 1.13 modela um sistema de alocação de recursos na qual dois tipos de processos, **A** e **B**, utilizam recursos do tipo **R**. O processo **A** precisa de dois recursos **R** para executar, enquanto que o processo **B** necessita apenas de um recurso **R**. A transição **T1** tem como lugares de entrada **P1** e **P4** e, como lugar de saída **P3**. Na **situação 1** duas transições estão habilitadas: **T1** e **T2**. Quando ocorrer o disparo de **T1**, duas fichas são removidas de **P4** e uma ficha de **P1**, ao passo que, uma ficha é incluída em **P3**, sendo atingida a **Situação 2**.

Nas seções subsequentes são abordados as Redes de Petri Coloridas e Redes de Petri Coloridas Hierárquicas a partir da referência do trabalho de dissertação de GORGÔNIO (2001).

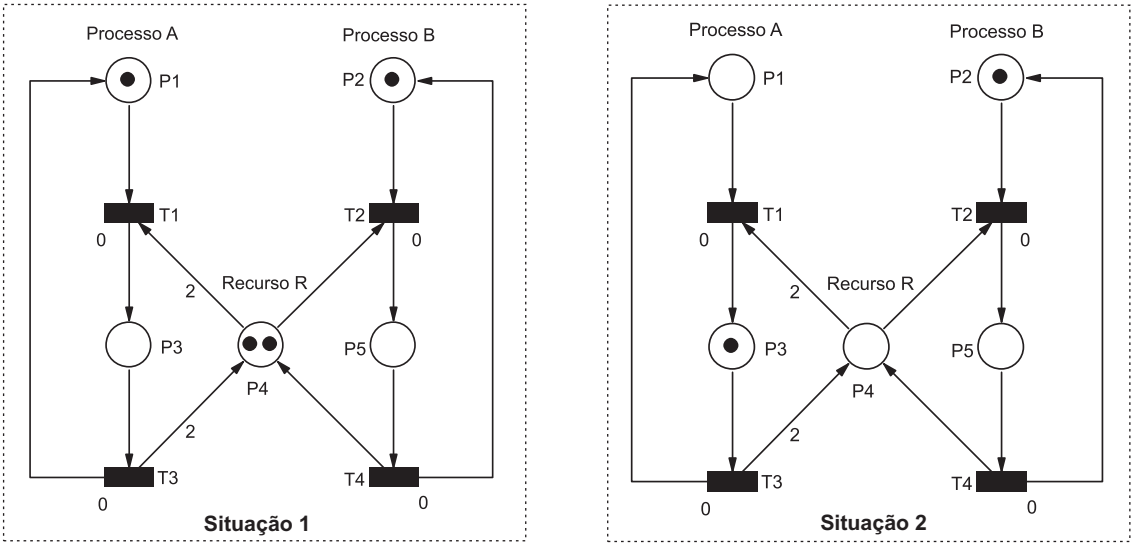

Figura 1.13: Exemplo de Rede de Petri Lugar/Transição

1.4.1 Redes de Petri Coloridas

No exemplo da Figura 1.13, apresenta-se dois tipos de processos que compartilham o mesmo recurso de forma similar. Para uma situação em que se tenha mais processos e recursos, o modelo poderia se tornar complexo e ilegível, apesar de possuir partes semelhantes. Este tipo de problema é típico em modelos de RP de Baixo Nível. Em sistemas do mundo real, as partes que compõem o sistema são geralmente similares, mas não idênticas. Isto significa que estas redes se tornam muitos grandes, devido a necessidade de duplicação da estrutura de rede para modelar processos semelhantes. Como exemplo, observa-se que na rede Lugar/Transição da Figura 1.13 a sub-rede para o processo **B** tem a mesma estrutura da sub-rede para o processo **A**. Essa replicação decorre do fato de ser impossível diferenciar os processos por meio de fichas.

Uma outra limitação das redes de baixo nível é a carência de elementos para estudo de desempenho dos sistemas modelados, pois a ocorrência de suas transições é instantânea.

Dessa forma, visando suprir tais limitações e facilitar a modelagem de sistemas complexos, extensões foram propostas para as redes da classe de Baixo Nível. Dentre elas, tem-se a classe das RPs de Alto Nível.

As RPs de Alto Nível caracterizam-se sobretudo pela incorporação da teoria de tipo de dados. As fichas para essas redes podem carregar informação complexa, que é manipulada pelo uso de uma linguagem.

O fato das fichas expressarem informação complexa aumenta o poder de descrição dessas redes e, conseqüentemente, modelos mais compactos podem ser obtidos. Essa flexibilidade na manipulação da informação permite ao projetista distribuir a complexidade do modelo de um sistema entre as inscrições e a estrutura da rede.

Dentre as RPs de Alto Nível, as mais notáveis são as RPs Coloridas (RPCs) (JENSEN, 1992). Estas redes são muito utilizadas na modelagem de aplicações complexas. Uma definição formal detalhada para as RPCs pode ser encontrada em (JENSEN, 1992).

Uma RPC compõe-se de três partes distintas: *estrutura*, *declarações* e *inscrições*. A estrutura é formada por lugares, transições e arcos direcionados, de maneira similar à definida para as RPs Lugar/Transição. As declarações definem conjuntos de cores (domínios), variáveis e operações (funções) usadas nas inscrições. As inscrições, por sua vez, podem ser de quatro tipos: cores dos lugares, guardas das transições, expressões dos arcos e inicializações.

As cores dos lugares determinam a cor (domínio) associada ao lugar. Um lugar só pode comportar fichas cujos valores respeitem sua cor. As guardas são expressões lógicas (booleanas) que restringem a ocorrência das transições. As expressões dos arcos servem para manipular a informação contida nas fichas. E as inicializações são associadas aos lugares para estabelecer a marcação inicial da rede.

Um exemplo de uma RPC é apresentado na Figura 1.14. As declarações estão expressas em uma caixa no canto superior esquerdo. Os textos próximos aos lugares indicam suas cores, e as expressões suas inicializações. As expressões dos arcos localizam-se junto aos arcos direcionados e não existem guardas associados as transições.

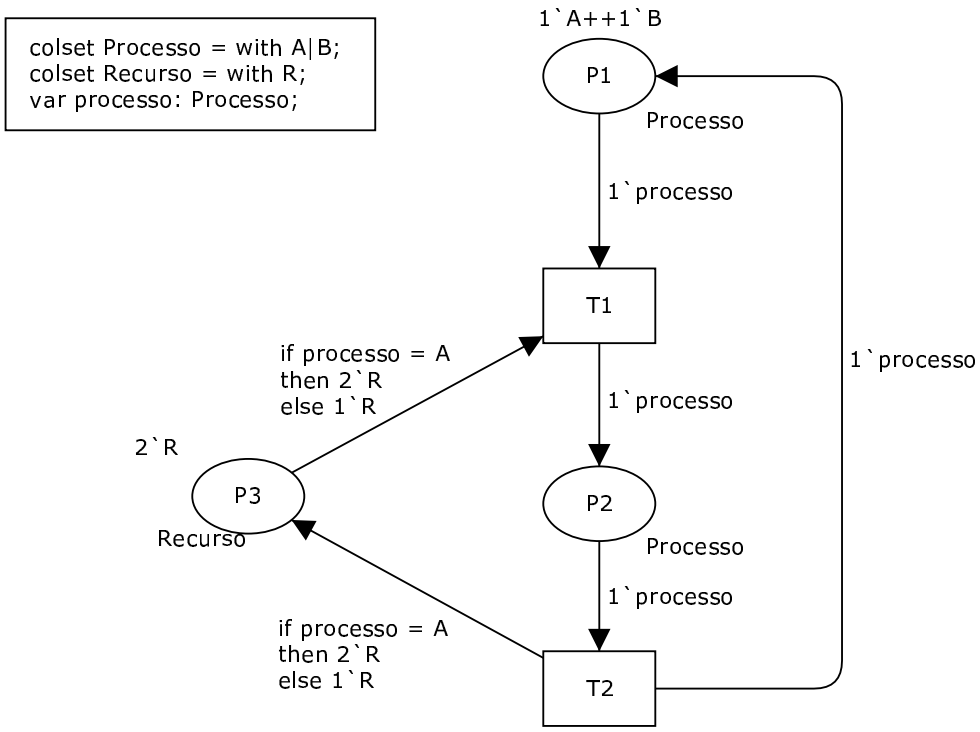

Figura 1.14: Exemplo de Rede de Petri Colorida

Para compreender a regra de ocorrência para RPC, é essencial, primeiramente, compreender os conceitos de *variável de transição*, *ligação* (binding) e *elemento de ligação*. As va-

riáveis de transição são aquelas presentes nos arcos direcionados. Por exemplo, na Figura 1.14, a transição **T1** tem **processo** como sua variável de transição. Uma ligação corresponde a associação de uma variável de transição a um valor da sua cor. Por exemplo, para a mesma variável de transição **processo**, são possíveis as ligações *b1* =<**processo**=**A**> e *b2* =<**processo**=**B**>. Um elemento de ligação é um par (transição,ligação). Como ilustração, considerando os exemplos anteriores, tem-se *be1* =(**T1**,*b1*=<**processo**=**A**>) e *be2* =(**T1**,*b2*=<**processo**=**B**>) como elementos de ligação.

A RPC da Figura 1.14 modela uma situação idêntica àquela da Figura 1.13. Entretanto, uma mesma estrutura foi usada para modelar os dois processos **A** e **B**, pois eles podem ser diferenciados através das fichas. As inscrições de inicialização dessa rede estabelecem que sua marcação inicial contém um processo do tipo **A**, um processo do tipo **B** e dois recursos tipo **R**. Conseqüentemente, apenas a transição **T1** está habilitada a ocorrer, pois ela é a única a apresentar algum elemento de ligação satisfeito pela marcação inicial. Nessa situação, pode ocorrer *be1* ou *be2*. Se *be1* ocorrer, uma ficha **A** é removida de **P1**, duas fichas **R** de **P3** e uma ficha **A** é adicionada a **P2** (pois em *be1* **processo** está ligada a **A**). Vale ressaltar que um elemento de ligação só está habilitado se satisfizer a guarda da transição. Assim, se para a transição **T1** fosse associada a guarda [**processo**=**A**], apenas *be1* estaria habilitado a disparar.

1.4.2 Redes de Petri Coloridas Hierárquicas

A idéia básica das RPCs Hierárquicas (RPCHs) é possibilitar a construção de um modelo através da combinação de um conjunto de redes relativamente menores denominadas *páginas*, de forma análoga à construção de um programa a partir de um conjunto de módulos e funções (JENSEN, 1992). O poder de modelagem de uma RP Lugar/Transição, de uma RPC e de uma RPCH são equivalentes. É sempre possível traduzir uma RPCH para uma RPC, que por sua vez, pode ser traduzida para uma RP Lugar/Transição. Em termos de linguagem de programação, pode-se comparar as RPs Lugar/Transição com as linguagens de máquina, as RPCs com as linguagens de alto nível e as RPCHs com o uso de módulos e funções (JENSEN, 1992).

As RPCHs são construídas utilizando-se o conceito de *lugares de fusão* e *transições de substituição*. Lugares de fusão são estruturas que permitem especificar um conjunto de lugares como funcionalmente um único lugar, isto é, se uma ficha é removida ou adicionada de um dos lugares, uma ficha idêntica é adicionada ou removida de todos os outros lugares pertencentes ao conjunto. Um conjunto de lugares de fusão é denominado conjunto de fusão (fusion set). Uma transição de substituição pode ser vista como uma transição de mais alto nível que se relaciona a uma rede mais complexa e que fornece maiores detalhes das atividades representadas pela transição de substituição. A página que

contém a transição de substituição é denominada *super-página* e a que contém uma visão mais detalhada é denominada *sub-página*. Cada transição de substituição é denominada *super-nó* da sub-página correspondente. Uma transição de substituição se relaciona com sua sub-página através da utilização de um tipo de conjunto de fusão de dois membros denominados portas e soquetes (sockets). Estas estruturas descrevem a interface entre a transição de substituição e a *sub-página*. Soquetes são atribuídos aos lugares conectados à transição de substituição, e portas são associadas a determinados lugares na sub-página tal que um par soquete/porta forma um conjunto de fusão. Dessa forma, quando uma ficha é depositada num soquete, ela aparece também na porta associada aquele soquete, permitindo assim, a conexão entre a super-página e a sub-página. É sempre possível traduzir uma RPCH para sua correspondente não hierárquica. Para isso, basta substituir cada transição de substituição e arcos conectados, por sua respectiva sub-página "colando"cada soquete com sua respectiva porta. A definição formal de RPCH pode ser encontrada em (JENSEN, 1992).

1.5 Conclusão

Neste capítulo foram apresentados os fundamentos básicos dos formalismos necessários para o entendimento dos conceitos utilizados neste trabalho. Foram dadas noções de telerobótica, RV, RPs, e PDI.

Capítulo 2

Uma Solução para Sistemas Telerobóticos Baseados na Internet

Neste capítulo é introduzido o trabalho proposto apresentando de forma geral a estrutura do sistema e descrevendo os requisitos necessários para o desenvolvimento do mesmo. Na seção 2.1 são discutidos alguns trabalhos desenvolvidos na área de STIs, apresentado um estudo comparativo destes com a solução proposta deste trabalho. Na seção 2.2, é discutido o projeto do sistema e apresentado a metodologia para desenvolvimento do sistema. Na seção 2.3 é apresentado o diagrama de casos de uso do sistema, mostrando as funcionalidades do mesmo. Na seção 2.4 é apresentada a arquitetura do sistema, mostrando os seus componentes de *hardware*. Na seção 2.5 é apresentada a arquitetura de *software* do sistema consistindo dos subsistemas cliente e servidor.

2.1 Trabalhos correlatos

No final de 1994 surgiram os primeiros STIs. O primeiro sistema denominado como projeto Mercury usava um robô com 3 graus de liberdade para cavar objetos enterrados dentro de uma área de trabalho bem definida (GOLDBERG; MASCHNA; GENTNER, 1995). Os usuários eram capazes de controlar a posição do braço robótico e visualizar as cenas como uma série de imagens estáticas atualizadas periodicamente. Poucas semanas depois, Taylor e Trevelyan (1995) desenvolveram um sistema na *University of Western Australia* que fornecia um controle baseado na Internet de um robô industrial ASEA IRB-6, no qual o operador usava o braço robótico para empilhar blocos de madeira. Como no projeto Mercury, a visualização da área de trabalho era limitada a uma seqüência de imagens estáticas capturadas por câmeras fixas localizadas ao redor da mesma.

Problemas de imagens estáticas podem ser evitados utilizando a tecnologia de *streaming video* que consiste em particionar e comprimir o vídeo em partes, transmitir estas

partes em sucessão, e habilitar o recebedor para decodificar e reproduzir o vídeo enquanto estas partes estão sendo recebidas. Ultimamente, aplicações de *streaming video*, tais como, Real Video©, QuickTime Movie©, e Microsoft Media Player© tem sido largamente utilizadas na Internet. Entretanto, *streaming video* via Internet padrão é difícil, pois esta trabalha com serviços de melhor esforço. Não existem critérios de prioridade na transmissão de dados. Quando há congestionamento na rede, pacotes (dados e multimídia) são descartados sem distinção. Especificamente estas características são desconhecidas e dinâmicas (APOSTOLOPOULOS; TAN; WEE, 2006).

No tratamento do problema de atraso de tempo indeterminado da Internet, alguns STIs tem utilizado a RV para reduzir o volume de dados que são transmitidos entre o local de controle (operador) e a área de trabalho remota (robô), e com isso permitir que as respostas no controle desses sistemas se tornem mais rápidas (ZAKARIA; AMIN; MAMAT, 2000). Além disso, a RV provê um ambiente tridimensional navegável para visualização da área de trabalho, além de possuir atributos que o ambiente físico não possui, tais como, simulação, planejamento, ensaio, e retificação de tarefas antes de serem submetidas à execução. Estas características tem sido exploradas nos STIs de Yang e Chen (2004), Belousov, Chellali e Clapworthy (2001), Tan e Clapworthy (2003) e Safaric e Sorgo (2003).

Yang e Chen (2004) desenvolveram um projeto de telerobótica baseado em técnicas de RV para controlar em tempo real um manipulador PUMA 560 via Internet. O AV é utilizado neste projeto para simular o ambiente físico, fornecendo respostas rápidas às ações do operador. Esta técnica reduz consideravelmente o tráfego do sistema se comparado com transmissão de imagens de vídeo. Ela permite que robôs sejam controlados com sucesso mesmo em baixas taxas de comunicação (0,1-0,5 KB/seg). Porém, neste trabalho não é apresentado uma técnica precisa para manter o AV sincronizado com o ambiente físico, principalmente se este for dinâmico.

Belousov, Chellali e Clapworthy (2001) desenvolveram ferramentas de RV para teleoperação robótica. O sistema compreende de uma representação virtual em tempo real baseada em Java3D do robô e área de trabalho, painel de controle e um ambiente para programação remota de robô. Para validar o trabalho foram utilizados os sistemas de controle dos manipuladores PUMA e CRS. Experimentos em tempo real com esses sistemas à longas distâncias e baixa conexão de Internet, revelaram que apenas AVs podem ser utilizados no controle eficiente dos robôs. O uso de AV simplifica o desempenho das operações (permite mudanças de pontos de visão, *zoom*, o uso de imagens semi-transparentes, etc.) e, o mais importante, suprime os atrasos de tempo por um fator de 20 (vinte), se comparado com imagens de vídeo. O sistema foi desenvolvido com tecnologia Java e Java3D que fornecem a portabilidade do sistema entre as diferentes plataformas de *hardware*, *softwares* e diferentes tipos de robôs. Entretanto, esse sistema não possui um método automático

para atualizar o AV, necessitando de ajustes manuais para manter a sincronização com o ambiente físico.

Para tratar com a questão de deficiência na representação virtual de ambientes operacionais, Tan e Clapworthy (2003) desenvolveram um método para modelagem dinâmica de AV. Neste método, as mudanças de características principais da cena são rastreadas e atualizadas manualmente de forma que o AV mapeie fielmente o ambiente físico. Ele se baseia na aquisição de parâmetros de imagem singular para modelagem de AVs. Experimentos em STIs utilizando esse método tem apresentado resultados significativos à baixas conexões da Internet (tipicamente um modem de 56 kb/s), sem o uso de imagens de vídeo no retorno. Entretanto, ele pode apresentar problemas na modelagem de AVs, no que diz respeito a falta de definição da posição de profundidade de objetos que estejam suspensos (sem base) ou que não estejam relacionados com outros objetos na sua redondeza. Esses problemas são difíceis de se tratar, pois esse método utiliza apenas uma câmera de vídeo para capturar a imagem da área de trabalho.

Os projetos anteriores (YANG; CHEN, 2004), (BELOUSOV; CHELLALI; CLAPWORTHY, 2001) e (TAN; CLAPWORTHY, 2003) utilizam a estratégia de controle direto[1] para a interação com o ambiente físico em tempo real. Porém, para fins de operações telerobóticas à longas distâncias (distâncias continentais e espaciais), esse tipo de controle não é adequado, devido a grandes atrasos de tempo na resposta de execução da operação. Existem situações em que a teleoperação leva segundos para o robô remoto receber as entradas do operador e para a confirmação visual retornar ao operador.

Nessas condições, uma forma de evitar a interação direta entre o operador e o ambiente remoto é permitir que o operador faça a configuração *off-line* do experimento usando uma representação simulada do ambiente físico. Safaric e Sorgo (2003) desenvolveram um sistema baseado em AV utilizando a técnica de visualização antecipada (Predictive Display)[2] para tratar com a questão de atrasos de tempo devido a operações à longas distâncias. O usuário interage diretamente com o robô virtual sem atrasos indesejáveis da Internet. O resultado da operação é visualizado em formato de imagens de vídeo ao usuário após ter sido concluído pelo robô. Apesar desta abordagem oferecer respostas imediatas as ações do operador através do AV, ela pode apresentar os seguintes pontos negativos:

- Não oferece ao usuário retorno imediato da operação. O resultado é visualizado após ter sido concluído pelo robô;

[1] Controle em malha fechada manual onde o operador forma parte da malha de controle.

[2] Técnica que consiste de uma interface gráfica de usuário projetada para permitir ao operador "predizer"os movimentos do robô antes de enviar os comandos de controle ao robô real remoto (SHERIDAN, 1992).

- Imagens de vídeo no retorno gerando lentidão ao sistema;
- Como a estratégia de controle adotada não fornece retorno imediato, é desejável que algum tipo de detecção de colisão entre o robô virtual e o ambiente virtual seja criado para prevenir colisões do robô no ambiente físico.

2.1.1 Estudo comparativo

Os trabalhos mencionados na seção anterior se restringem a robôs manipuladores, que utilizam protocolo TCP/IP para comunicação, e que não possuam automação (robôs sem inteligência).

Neste trabalho de pesquisa, a partir da proposta de Tan e Clapworthy (2003) combinada com a de Safaric e Sorgo (2003), é projetado e desenvolvido um STI, utilizando as tecnologias de RV, RPs e PDI com o intuito de reduzir o tempo de resposta no controle telerobótico e manter a sincronização entre os ambientes físico e virtual. Na Tabela 2.1 é apresentado um comparativo entre os STIs pesquisados, bem como, a própria solução proposta nesta dissertação, face aos seguintes critérios:

1. Visualização Antecipada: A visualização antecipada é uma técnica que ajuda a diminuir a latência da Internet e efeitos de atrasos de tempo. Esse método permite ao usuário especificar as tarefas do robô em um ambiente simulado, sem precisar controlar fisicamente o manipulador robótico (SHERIDAN, 1992);

2. RV: A partir de técnicas de modelagem baseadas em imagens de vídeo é possível se construir o AV 3D, onde o usuário é capaz de interagir com o sistema de qualquer ponto de visão. Além disso, essa técnica é muito útil para implementar a funcionalidade de visualização antecipada, além de servir como um ambiente para especificação de tarefas;

3. Controle de acesso: O controle de acesso é utilizado com o intuito de assegurar que o sistema só poderá ser acessado por pessoas previamente habilitadas mantendo a restrição de acesso. Cada usuário deve possuir um *login* e senha para poder acessar o sistema, que são cadastrados pelo administrador do sistema;

4. Banco de tarefas: Tarefas de operações do robô podem ser salvas em um banco de dados, onde poderão ser abertas em um momento posterior para serem enviadas ao servidor do robô. Para isso se utiliza um banco de tarefas, com o intuito de agilizar o trabalho do operador, economizando assim, tempo de trabalho;

5. Manutenção do AV: Com o intuito de tratar com a questão de sincronização entre os ambientes físico e virtual, vários sistemas tem utilizado métodos para manter o

AV atualizado com o ambiente físico. Este trabalho de pesquisa desenvolveu uma solução baseada no uso de RPs e PDI para fazer a sincronização do AV com o ambiente físico;

6. Vídeo no Retorno: O uso de imagens de vídeo no retorno é importante em telerobótica, pois são usadas para monitorar a operação robótica. Porém, quando se trata de STIs, elas produzem lentidão no sistema, devido à Internet padrão possuir restrição na transmissão de grandes volumes de dados na rede.

Na última coluna da Tabela 2.1 são apresentados os critérios satisfeitos pela solução proposta neste trabalho de dissertação. Conforme descrito na tabela, a solução proposta satisfaz os principais critérios apresentados, mais especificamente o critério de manutenção do AV que é o foco de pesquisa deste trabalho. Essa solução utiliza modelagem a eventos discretos e processamento de imagens para tratar com a questão de sincronização entre os ambientes físico e virtual e reduzir o tempo de resposta do sistema. Além disso, a abordagem possui a vantagem do uso de banco de tarefas e controle de acesso que são importantes para agilizar e dar a segurança a operação robótica.

Tabela 2.1: Estudo comparativo entre diferentes STIs e a solução proposta

Critérios	Goldberg	Taylor	Yang	Belousov	Tan	Safaric	**Solução Proposta**
Visualização Antecipada	Não	Não	Não	Não	Não	Necessita estar conectado	Não Necessita estar conectado
Realidade Virtual	Não	Não	Sim	Sim	Sim	Sim	Sim
Banco de Tarefas	Não	Não	Não	Não	Não	Não	Sim
Controle de Acesso	Não	Não	Não	Não	Não	Não	Sim
Manutenção do AV	Não	Não	Não	Ajuste manual necessário. (1 câmera)	Deficiência na definição de localização. (1 câmera)	Não	Solução RP-PDI (2 câmeras)
Vídeo no Retorno	Lento no resultado em tempo real	Lento no Resultado em tempo real	Não	Lento no Resultado em tempo real	Não	Resultado é enviado depois da execução	Não. O AV substitui o vídeo p/ reduzir o tempo de resposta

2.2 Casos de Uso do Sistema

A funcionalidade do sistema se baseia no diagrama de casos de uso da Figura 2.1.

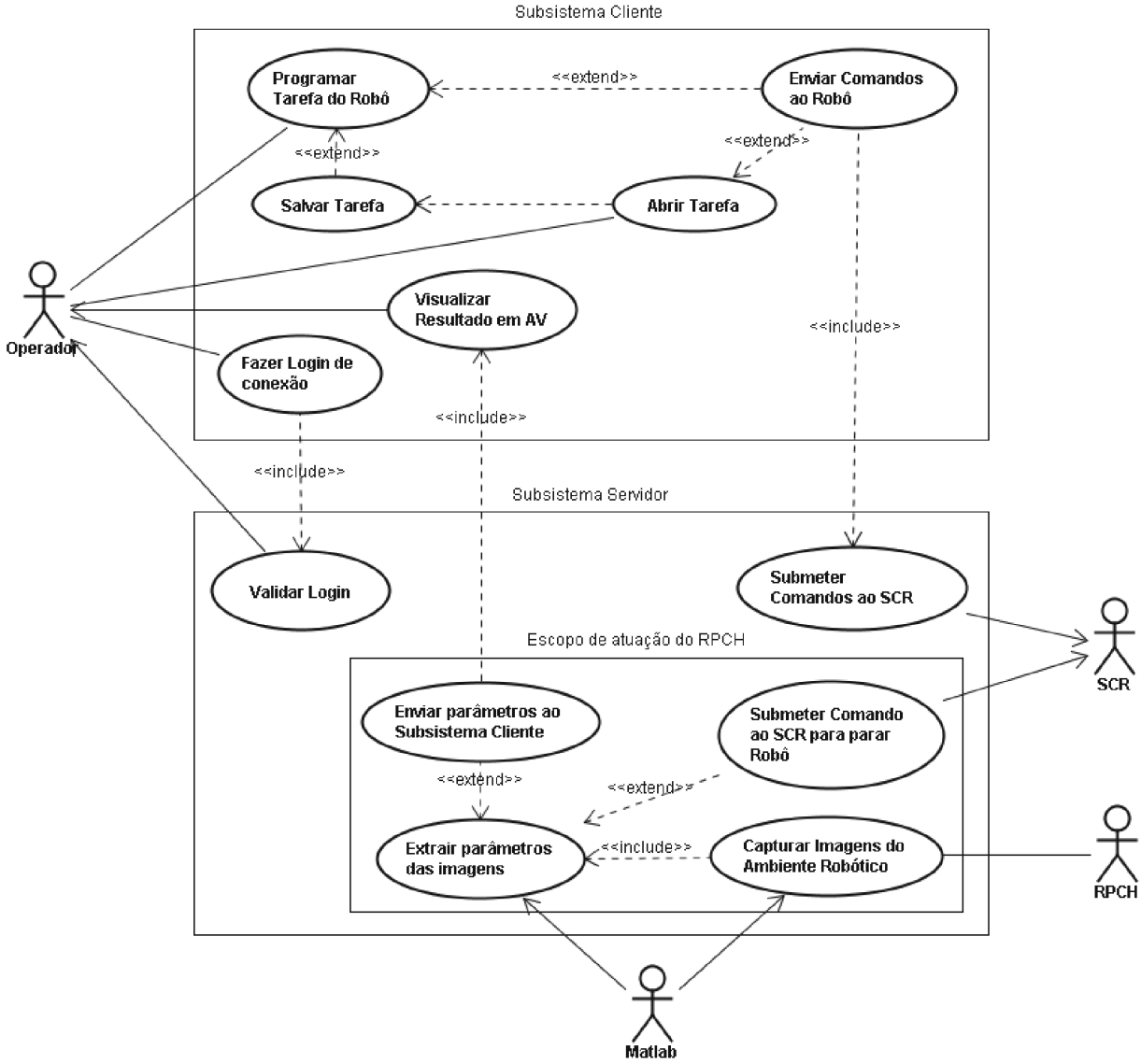

Figura 2.1: Diagrama de Casos de Uso do Sistema

Neste diagrama são modelados os subsistemas cliente e servidor do STI proposto neste trabalho. O subsistema cliente contém os casos de uso referentes as funcionalidades de interface gráfica de usuário, tais como, programar tarefa do robô, salvar tarefa, abrir tarefa, fazer *login* de conexão com o servidor, enviar comandos ao robô, etc. O subsistema servidor, por sua vez, consiste dos casos de uso referentes às funcionalidades de comunicação com o sistema robótico e processamento de imagens do ambiente robótico para ser transmitidas ao subsistema cliente.

Nos capítulos posteriores são apresentados com mais detalhes os casos de uso mais relevantes, abordando a fase de projeto de cada subsistema.

2.3 Projeto do Sistema

Como já definido, o presente trabalho de dissertação tem por objetivo a construção de um STI, utilizando as tecnologias de RV, RPs e PDI.

A RV é utilizada para substituir o retorno de imagens de vídeo, comumente utilizadas em STIs, por um AV local resolvendo o problema de transferência de imagens reais (a transferência é reduzida a dados de posicionamento). Dessa forma, o tempo de resposta do sistema é reduzido, resultando assim, em uma resposta mais rápida e fornecendo um melhor controle ao operador. Além disso, o AV empregado nesse trabalho permite ao operador simular as tarefas de operação no robô virtual, observando futuros movimentos, onde poderá salvar essas operações, e quando quiser, enviar ao robô real para execução.

Para tratar com a questão da deficiência na representação virtual de ambientes físicos, neste trabalho é apresentada uma solução baseada no uso de RPs para fazer a sincronização dos ambientes virtual e físico através de técnicas de PDI, de modo que, os mesmos se mantenham integrados garantindo atualizações do AV para operações subseqüentes.

Tendo em vista essas características, é necessário desenvolver um STI em arquitetura cliente-servidor, consistindo dos subsistemas servidor e cliente:

- Subsistema Servidor: Responsável pelo envio dos comandos de controle (enviados pelo subsistema cliente) ao Sistema de Controle do Robô (SCR) para a operação do robô, bem como, pelo fornecimento dos dados de posições do ambiente físico ao subsistema cliente para manutenção do AV;

- Subsistema Cliente: Responsável por fornecer as interfaces de usuário para o operador interagir com o robô, fornecendo dados de controle e recebendo dados do robô e do ambiente, para serem visualizados no AV.

Este sistema possui uma arquitetura modular que fornece portabilidade entre diferentes plataformas de computadores e tipos de robôs.

Nas subseções seguintes são descritas a arquitetura do sistema e a arquitetura de software, dando uma visão geral sobre o projeto do STI, a fim de preparar o leitor para os capítulos posteriores.

2.3.1 Arquitetura do Sistema

O projeto de arquitetura do STI proposto inclui três partes principais, conforme Figura 2.2: Subsistema Cliente; Subsistema Servidor; Robô.

A parte Subsistema Servidor inclui os seguintes componentes:

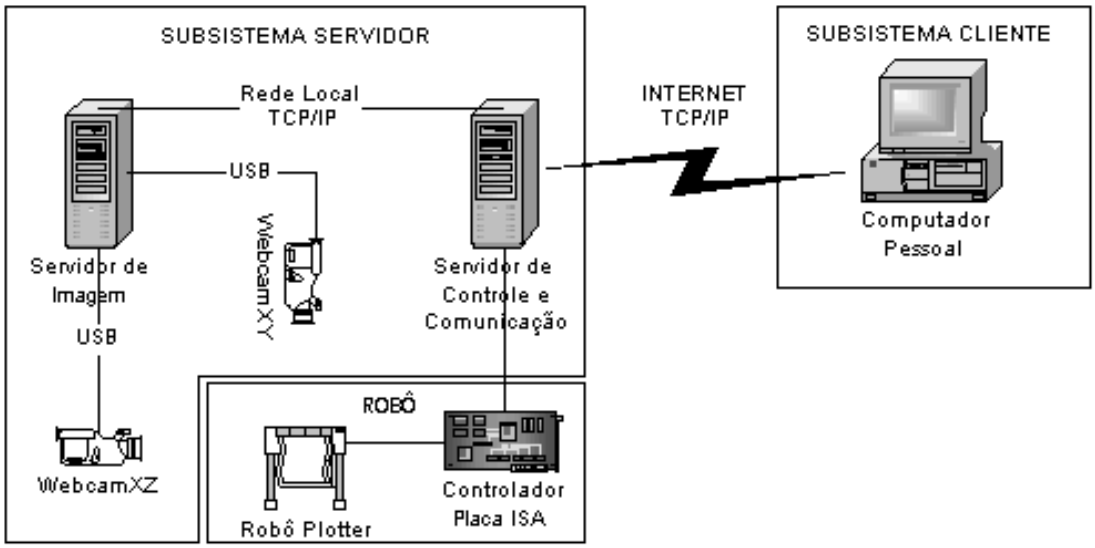

Figura 2.2: Arquitetura do Sistema

- PC Servidor de Controle e Comunicação para acessar ao controlador do Robô via barramento ISA. Este servidor também é responsável pela comunicação com o Cliente remoto via Internet e com o Servidor de Imagem via rede local;
- Placa controladora ISA para acessar o Robô;
- PC Servidor de Imagem com duas portas USB [3] para acessar as câmeras do ambiente robótico;
- Duas câmeras *webcams* com resolução de 352x280 para fazer as aquisições das imagens superior (imagem do plano x0y) e lateral (imagem do plano x0z).

O subsistema servidor é responsável pelo recebimento dos comandos de controle do robô (enviados pelo cliente), e envio dos mesmos ao controlador para operação do robô. Além disso, esse subsistema gera os parâmetros de imagens do ambiente físico (robô e área de trabalho) utilizados na manutenção do AV, no Subsistema Cliente.

A parte Subsistema Cliente inclui um PC para acessar a Internet e que possua uma configuração mínima de vídeo que suporte *opengl* versão 2.0 para operar AVs. Este subsistema é a interface gráfica de usuário usada para operação do sistema.

A parte Robô, por sua vez, inclui o robô Plotter Industrial de 3 graus de liberdade e uma placa controladora ISA [4]. Este robô é utilizado para traçar linhas retas em uma área de $40cm^2$.

[3] Tipo de conexão Plug and Play que permite a conexão de periféricos sem a necessidade de desligar o computador.

[4] Padrão de barramento desenvolvido para micros 286, que ainda é utilizado hoje em dia.

2.3.2 Arquitetura de Software

A arquitetura de Software do Sistema é ilustrada na Figura 2.3.

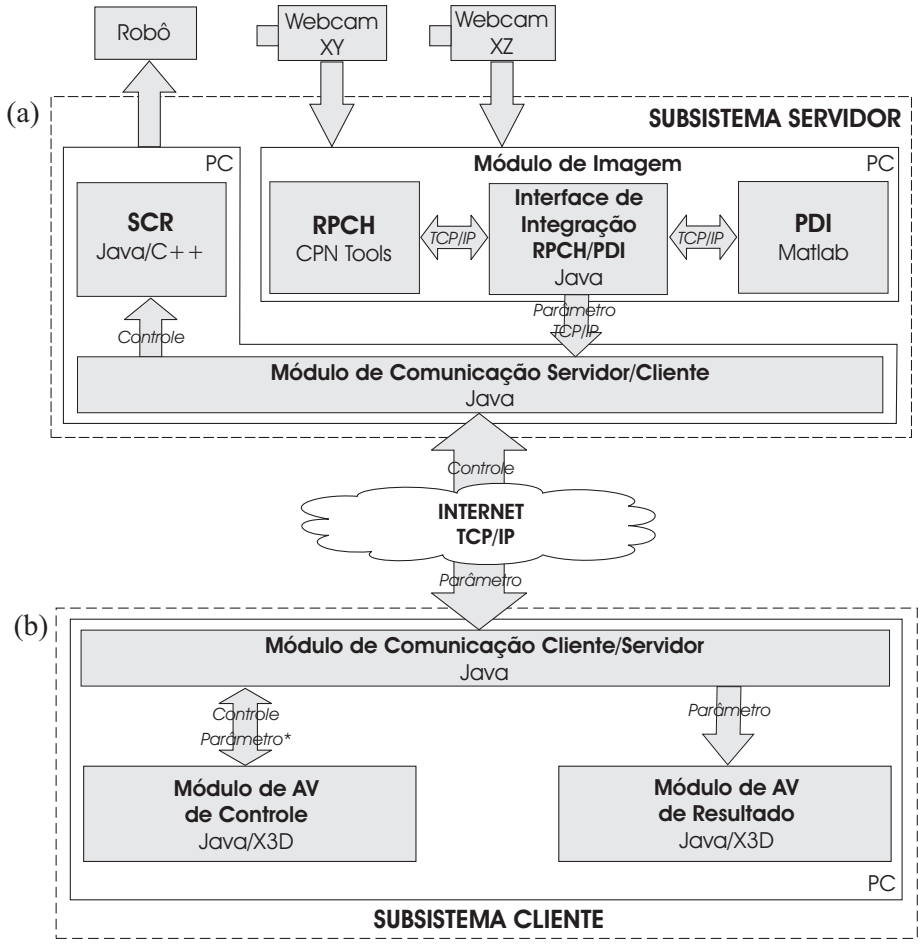

Figura 2.3: Arquitetura de Software: (a) Subsistema Servidor; (b) Subsistema Cliente

O Sistema é baseado em arquitetura cliente/servidor, consistindo dos seguintes subsistemas conectados via Internet: Subsistema Servidor; Subsistema Cliente.

O Subsistema Servidor, ilustrado na Figura 2.3(a), compreende dos seguintes componentes: SCR; Módulo de Imagem; Módulo de Comunicação Servidor/Cliente.

O SCR é utilizado neste STI para fornecer o acesso aos métodos de controle do robô. Este sistema inclui a Biblioteca de Controle e a Interface de comunicação do Robô. A Biblioteca de Controle contém os métodos em alto nível implementados em linguagem Java para controlar o robô, e a Interface de Comunicação contém o *driver* de *hardware* de baixo nível implementado na linguagem C++ para acessar o Robô.

As soluções baseadas na linguagem Java fornecem mais interatividade (o retorno ao usuário pode ser imediato), interfaces de usuário sofisticadas (interfaces gráficas de usuário fáceis de trabalhar, suporte de realidade aumentada e virtual) e arquitetura de comunicação melhorada (*Stream* ou Soquetes baseados em datagramas, *frameworks* de comunicação

avançadas para aplicações especiais, etc). Além disso, o desenvolvimento de *software* é mais eficiente porque a linguagem Java fornece ferramentas poderosas de comunicação para Internet e sincronização de programas multi-escalonados (multi-threaded).

O Módulo de Imagem implementa a solução RP-PDI. Este inclui os blocos *RPCH* e *PDI* para fazer a extração de parâmetros das imagens capturadas do ambiente físico (robô e área de trabalho). Os parâmetros são as posições do robô e dos objetos caixa e cilindro do ambiente físico e servem para atualizar o AV. Este Módulo também é responsável pela supervisão do sistema em casos de exceção, tais como, falha na identificação dos objetos e do robô na imagem, e detecção de colisão entre o robô e os objetos.

O bloco *RPCH* e o bloco *PDI* foram desenvolvidos utilizando as ferramentas CPNTools e Matlab, respectivamente. A descrição dessas ferramentas, bem como, a integração entre elas são abordadas detalhadamente no apêndice B.

O Módulo de Comunicação Servidor/Cliente, por sua vez, consiste do Servidor de Soquete de fluxo para receber conexão do Cliente (Módulo de Comunicação Cliente/Servidor). Além disso, esse módulo serve como uma ponte de dados entre os componentes SCR e Módulo de Imagem.

O Subsistema Cliente, ilustrado na Figura 2.3(b) compreende dos seguintes componentes: Módulo de AV de Controle; Módulo de AV de Resultado; Módulo de Comunicação Cliente/Servidor.

Através do AV de Controle, o usuário pode programar as tarefas de forma *off-line*, e depois enviar as mesmas ao ambiente do robô para execução. Este método é chamado de visualização antecipada onde permite ao usuário operar o robô virtual de forma instantânea, eliminando problemas de atrasos indesejáveis da Internet (SHERIDAN, 1992).

A operação do robô pode ser visualizada em tempo real pelo usuário através do AV de Resultado. O AV é usado no lugar de imagens de vídeo para minimizar o tempo de transmissão de dados através da rede. O processo de atualização do AV é executado localmente nas interfaces de Controle e de Resultado na máquina do cliente.

O Módulo de Comunicação Cliente/Servidor consiste do Cliente de Soquete de fluxo para fazer conexão com o Servidor (Módulo de Comunicação Servidor/Cliente).

O *software* da parte cliente é implementado usando as linguagens Java e X3D. A única restrição é que o computador do cliente deve ter a configuração suficiente para trabalhar com cenas virtuais com uma taxa mínima de 10 quadros por segundo (KIRNER, 2006). Além disso, a escolha desses *softwares* nesta parte do projeto, é porque eles podem ser utilizados em qualquer tipo de computador e sistema operacional como cliente.

2.4 Conclusão

Neste capítulo são apresentados os tópicos que permitiram dar um entendimento geral da solução proposta deste trabalho com respeito aos casos de uso da solução e sua estrutura. Neste capítulo também é abordado o estado da arte, fazendo-se uma revisão bibliográfica, a fim de avaliar outras soluções existentes com o trabalho proposto.

Capítulo 3

Subsistema Servidor

Neste capítulo é descrito o projeto do Subsistema Servidor do STI. Este subsistema é responsável pelo envio dos comandos de controle (enviados pelo subsistema cliente) ao SCR para a operação do robô, bem como, pelo fornecimento dos parâmetros do ambiente físico ao subsistema cliente para manutenção do AV. De acordo com a Figura 2.3(a), este subsistema inclui os componentes SCR, Módulo de Imagem, e Módulo de Comunicação Servidor/Cliente.

Na seção 3.1 é apresentado o diagrama de casos de uso para mostrar as funcionalidades do subsistema servidor. Na seção 3.2 é introduzida de forma genérica o SCR, onde são apresentados os métodos de acesso ao robô utilizados neste STI. Na seção 3.3, é apresentado o Módulo de Imagem e os blocos *RPCH* e *PDI*, utilizados na extração de parâmetros das imagens. Na seção 3.4, por sua vez, é apresentado o Módulo de Comunicação Servidor/Cliente que é utilizado para comunicação com o subsistema cliente e também para comunicação interna entre os módulos do subsistema servidor.

3.1 Casos de Uso do Subsistema Servidor

A funcionalidade do Subsistema Servidor é ilustrada no diagrama de casos de uso da Figura 3.1.

Observa-se que o subsistema cliente é um ator nesse diagrama, afim de mostrar sua interação com o subsistema servidor. Esse diagrama inclui os casos de uso referentes às funcionalidades de comunicação com o sistema robótico e de processamento de imagens do ambiente robótico para serem transmitidas ao subsistema cliente.

Em termos gerais, a funcionalidade do subsistema servidor pode ser descrita como se segue:

- O subsistema cliente submete comandos da tarefa ao subsistema servidor (caso de uso Submeter Comandos ao SCR);

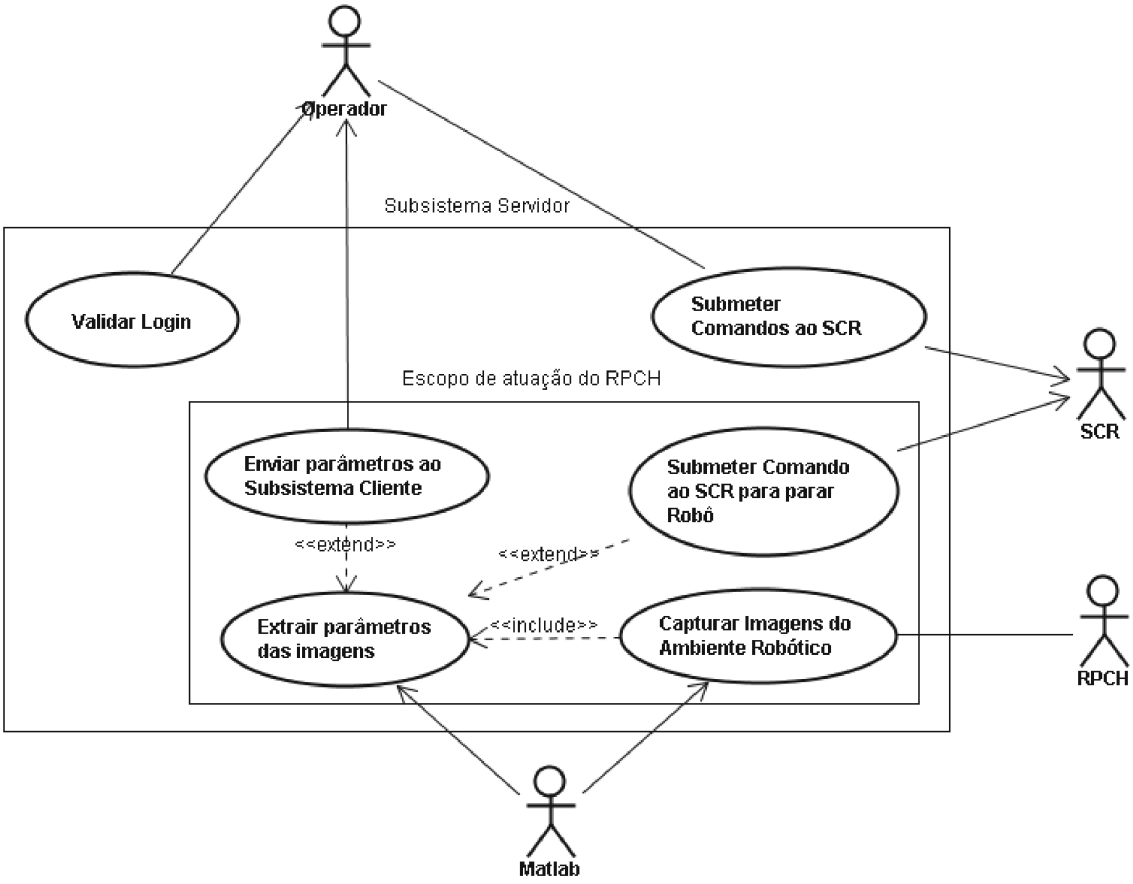

Figura 3.1: Diagrama de Casos de Uso do Subsistema Servidor

- O SCR converte os comandos em sinais de controle do robô para sua operação;
- O RPCH solicita ao Matlab a aquisição das imagens do ambiente robótico (caso de uso Capturar Imagens do Ambiente Robôtico);
- De posse das imagens, o RPCH solicita ao Matlab o processamento das imagens de forma a extrair as posições (parâmetros) do robô e dos objetos (caso de uso Extrair parâmetros das imagens);
- O RPCH verifica o resultado do processamento e executa uma das seguintes ações, abaixo:
  - Se robô e objetos são identificados nas imagens, o RPCH envia os parâmetros ao subsistema cliente (caso de uso Enviar parâmetros ao Subsistema Cliente);
  - Se robô e/ou objetos não são identificados nas imagens, o RPCH envia o comando para parar o robô ao SCR (caso de uso Submeter Comando ao SCR para parar Robô);

– Se houver iminência de colisão entre o robô e um dos objetos, o RPCH envia o comando para parar o robô ao SCR (caso de uso Submeter Comando ao SCR para parar Robô).

3.2 Sistema de Controle do Robô

O SCR foi desenvolvido no laboratório de robótica da Universidade Federal do Amazonas (UFAM) (MAISENBACHER, 2005) com a intenção de fornecer um laboratório remoto baseado na Internet. Na Figura 3.2 é ilustrado o esquema simplificado de *software* do SCR que contém os seguintes componentes: Interface de Comunicação (*driver* de *hardware* de baixo nível), implementada em C/C++, para acessar o *hardware* do robô; Biblioteca de Controle, implementada em Java, para fornecer os métodos funcionais do robô; Interface Nativa Java [1] (Java Native Interface - JNI) usada para servir como ponte entre a Biblioteca de Controle e a Interface de Comunicação.

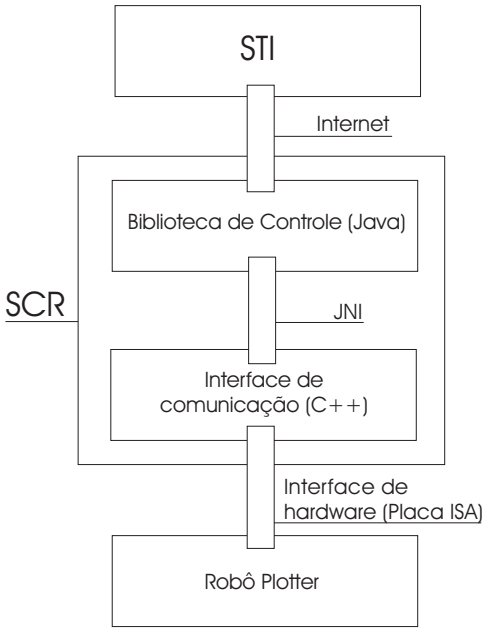

Figura 3.2: Esquema Simplificado de *Software* do SCR.

O robô utilizado neste trabalho é o robô industrial Plotter de 3 (três) graus de liberdade, conforme ilustrado na Figura 3.3. Este robô pode desenhar linhas retas em uma área de trabalho de $40cm^2$ demarcada pelo sistema de coordenadas do robô ilustrado na Figura 3.3. Uma outra aplicação desse tipo de robô é o de recorte no qual uma lâmina recorta adesivos de acordo com o que foi desenhado previamente no computador, através

[1] Padrão de programação para permitir que a máquina virtual da linguagem Java acesse bibliotecas construídas com o código nativo de um sistema.

de um programa específico. O material assim produzido é utilizado por exemplo na personalização de frotas de veículos e ambientes comerciais, como fachadas, vitrines, confecção de banners, luminosos, placas, faixas, entre outros.

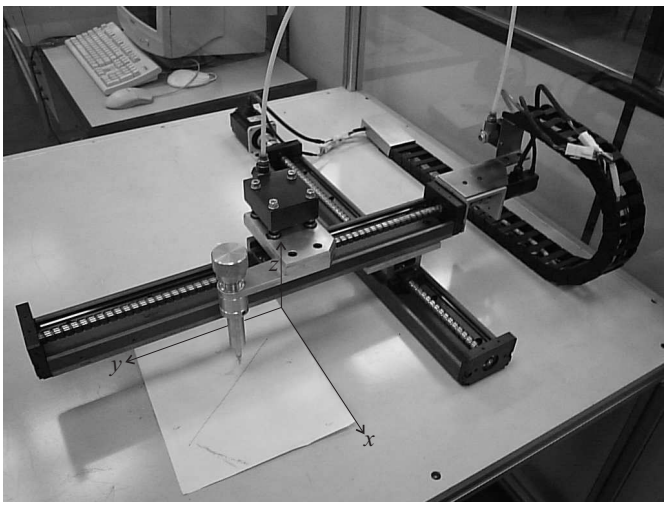

Figura 3.3: Robô Industrial Plotter utilizado no STI.

Dois servos motores são utilizados para a movimentação dos braços do robô no plano $x0y$. Para a movimentação do suporte da caneta na direção do eixo z, é utilizado um mecanismo pneumático. Cada servo motor tem um *encoder* que envia pulsos TTL [2] para o contador *up/down* de 16 Bits (conta na faixa de -65536 até +65536). Esta é uma forma de mensurar a distância máxima que o robô se moverá (65535 unidades nos eixos x e y). A cada pulso que o contador conta significa que o movimento do robô é incrementado de 0,00976mm.

O controle do robô é baseado em uma interface de *hardware* proprietária (placa ISA de controle) (MAISENBACHER, 2005) usando o barramento ISA do PC servidor de controle. O *hardware* do robô é acessado via Interface de Comunicação que pode ler e escrever nesse barramento. A Interface de Comunicação é o *driver* de *hardware* de baixo nível que serve como uma ponte entre os registradores da placa ISA e outras partes de *software*. Este *driver* não só fornece os registros, mas também uma forma legível de acesso ao *hardware*.

A Interface de Comunicação fornece um mecanismo de acesso ao *hardware*, e não uma regra de negócio. A Biblioteca de Controle, por sua vez, se caracteriza por fornecer a política de controle do sistema, e também, as funcionalidades utilizadas na interface de usuário. Ela é composta por métodos abstratos de alto nível, implementados em Java, tais como, ir a um ponto desejado, parar o robô, zerar encoders, etc. Além disso, a Biblioteca de Controle possui outros atributos, tais como, garantir a segurança do sistema, acessar a Interface de Comunicação e implementar o JNI.

[2] Os circuitos digitais TTL (Lógica Transistor-Transistor) tem como principal característica a utilização de sinais de 5 volts para níveis lógico altos.

A parte mais importante da Biblioteca de Controle é a certeza de que o ser humano não sofrerá dano pelo sistema. Ela foi desenvolvida para verificar a todo momento se a posição da garra do robô alcançou o limite de borda. Para isso, foi implementado um processo escalonado (thread) que faz a leitura das posições do robô a todo tempo, fornecendo essas informações a todos os componentes do sistema.

Os métodos abstratos descritos anteriormente, não são capazes de executar o código de baixo nível (código da Interface de Comunicação). Para resolver este problema, foi utilizado o API JNI para permitir a linguagem Java executar os métodos nativos como se fossem seus próprios métodos.

3.2.1 Métodos de Acesso ao Robô

Para se ter acesso as funcionalidades do robô é necessário utilizar os métodos da Biblioteca de Controle. O diagrama de classes, contendo os métodos de acesso ao SCR, é ilustrado na Figura 3.4.

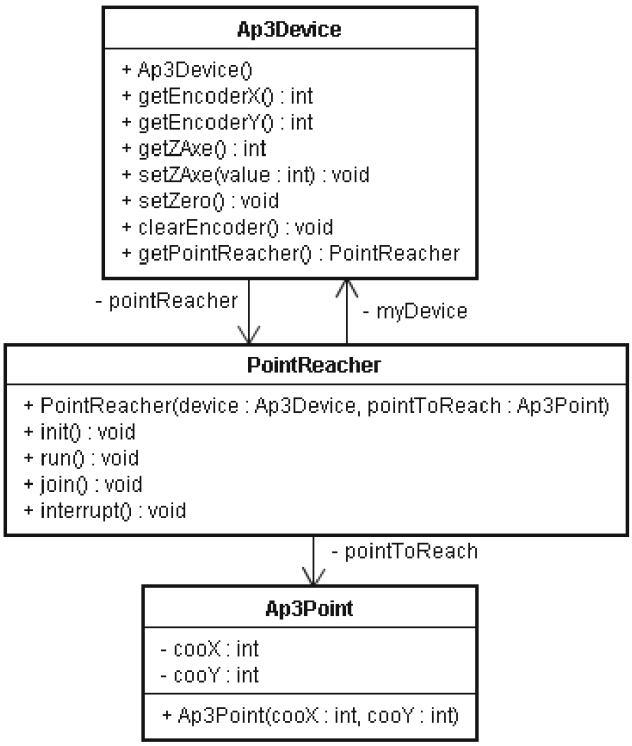

Figura 3.4: Diagrama de classes do SCR.

Os métodos principais utilizados neste trabalho são:

- $setZaxe(intz)$: Método que faz com que a caneta fique na posição baixa ($z = 1$) ou na posição alta ($z = 0$);
- $setZero()$: Método usado para fazer o robô parar;

- $AP3Point(intcooX, intcooY)$: Método usado para criar um ponto no sistema de coordenadas do ambiente físico (sistema real);
- $PointReacher(AP3DevicemyDevice, AP3Pointp)$: Método *Thread* (escalonado) que faz com que o robô se mova para o ponto especificado p;
- $clearEncoder()$: Método usado para limpar os valores dos encoders;
- $getEncoderX()$: Método usado para obter o valor da posição de x;
- $getEncoderY()$: Método usado para obter o valor da posição de y;
- $getZAxe()$: Método usado para obter o valor da posição de z.

O código completo desses métodos podem ser vistos no apêndice E.

O SCR, através do Módulo de Comunicação Servidor/Cliente, recebe os comandos enviados pelo subsistema cliente, que efetua a operação no robô. Este processo pode ser visualizado no diagrama de seqüencia Submeter Comando ao SCR da Figura 3.5. Este diagrama de seqüencia modela um cenário em que o robô desenha uma linha reta, dados os pontos inicial e final.

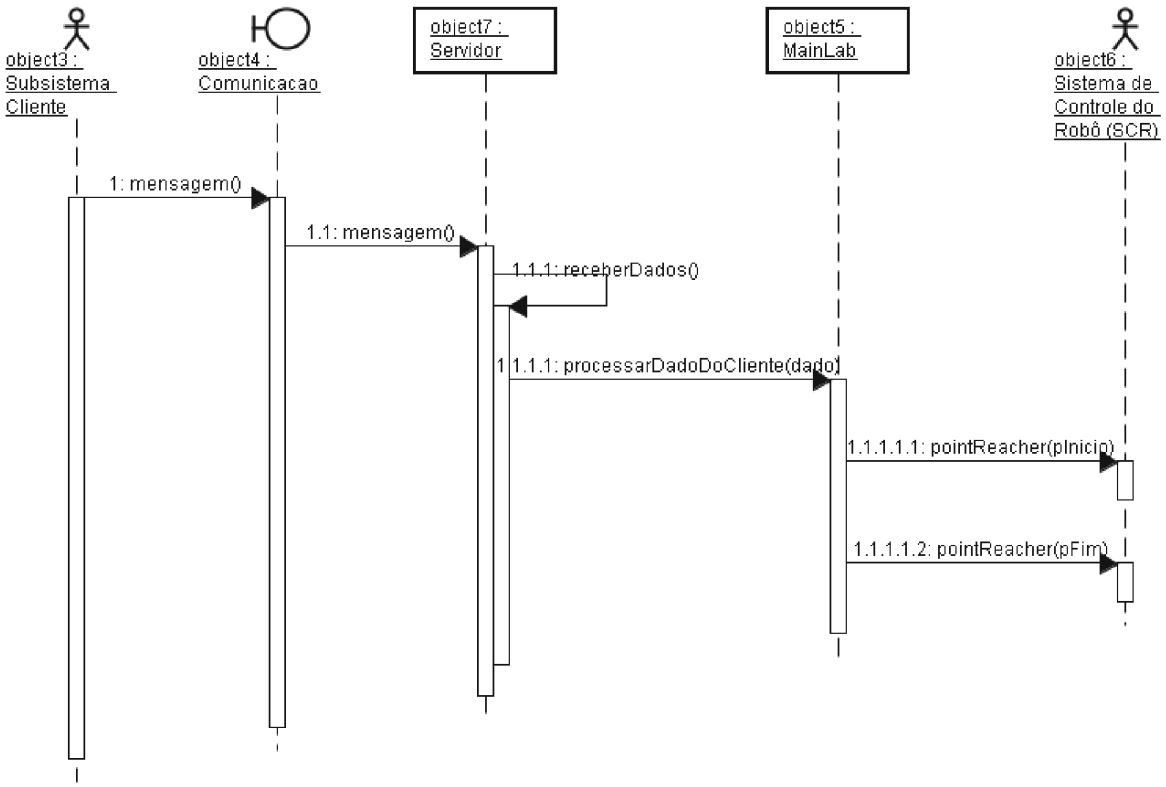

Figura 3.5: Diagrama de Seqüência do cenário Submeter Comando ao SCR.

3.3 Módulo de Imagem

O Módulo de Imagem tem por objetivo tratar a questão da sincronização entre os ambientes físico e virtual. Ele inclui os blocos *RPCH*, *PDI* e *Interface de Integração RPCH/PDI* conforme ilustrado na Figura 2.3(a). Este Módulo se baseia no uso de RPCHs e técnicas de PDI para fazer a extração de parâmetros das imagens do ambiente físico de forma a manter o AV sincronizado com o ambiente físico. Os parâmetros são as posições dos baricentros do robô e dos objetos caixa e cilindro da área de trabalho.

O processo de extração de parâmetros é ilustrado no diagrama de blocos da Figura 3.6. A RPCH controla a execução de cada etapa de PDI, partindo da etapa de aquisição das imagens até a etapa de extração de parâmetros.

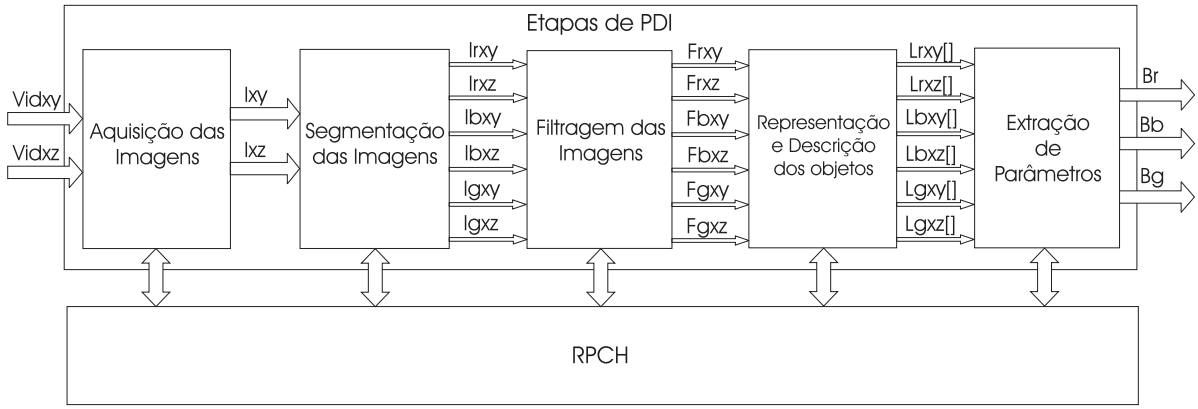

Figura 3.6: Diagrama de blocos do processo de extração de parâmetros.

A etapa de Aquisição das Imagens realiza a aquisição dos vídeos dos planos $x0y$ e $x0z$ ($Vidxy$ e $Vidxz$, respectivamente) mediante câmeras (Webcams XY e XZ) conectadas ao PC via portas USB, conforme infra-estrutura ilustrada na Figura 3.7. Estas câmeras estão dispostas de forma a fazer a aquisição das vistas superior e lateral do ambiente físico. O propósito da localização das câmeras é obter as posições (x, y, z) dos baricentros do robô e dos objetos da área de trabalho. Esta etapa produz na saída as imagens Ixy e Ixz digitalizadas do ambiente físico. Considera-se que os objetos são de forma primitiva (caixa e cilindro) e que possuem dimensões conhecidas. Cada objeto é identificado por uma cor específica que é usada na etapa de segmentação das imagens.

A etapa de Segmentação das Imagens consiste em identificar e extrair as imagens do robô e dos objetos da cena. Ela é baseada na segmentação por modelo de cor HSI. Este modelo de cor foi escolhido por ser mais adequado no processo de segmentação, pois seus componentes matiz (H), saturação (S) e intensidade (I) mostram-se menos correlacionados. A partir dos valores de H e S (valores de cromaticidade) é possível se fazer a segmentação do objeto pela cor sem a interferência de luminosidade do ambiente. Neste trabalho, o robô e os objetos são identificados com cores específicas para facilitar o pro-

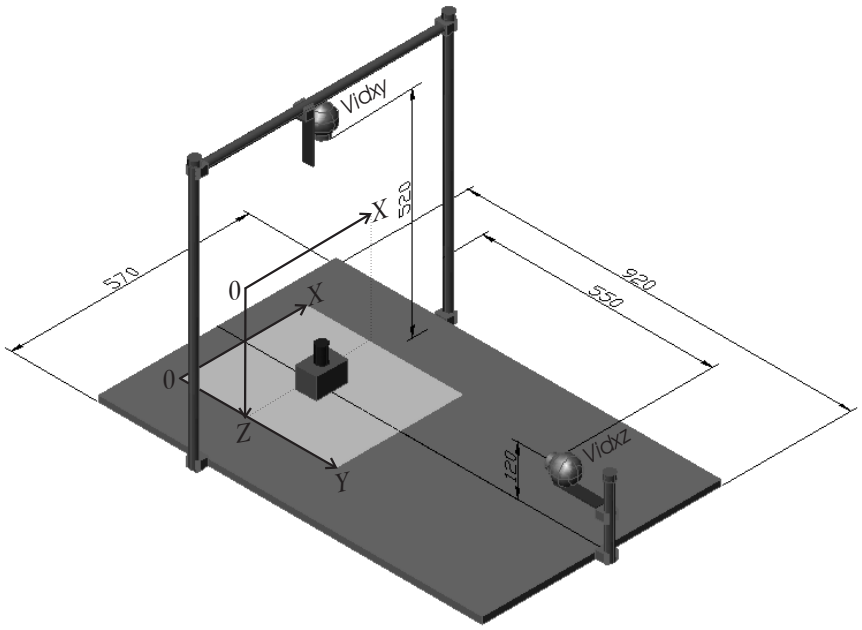

Figura 3.7: Infra-estrutura para aquisição das imagens dos planos x0y e x0z.

cesso de segmentação. Esta etapa produz na saída as imagens $Igxy$, $Igxz$, $Irxy$, $Irxz$, $Ibxy$ e $Ibxz$ representando a projeção do robô e dos objetos caixa e cilindro nos planos $x0y$ e $x0z$, respectivamente.

A etapa de Filtragem das Imagens é realizada através de filtragem morfológica que possibilita uma melhor redução de ruído nesse tipo de imagem. Este tipo de filtragem é baseada na técnica de abertura, operação utilizada na Morfologia Matemática. A Morfologia Matemática é descrita detalhadamente no apêndice A. Esta etapa produz na saída as imagens filtradas $Fgxy$, $Fgxz$, $Frxy$, $Frxz$, $Fbxy$ e $Fbxz$ representando as imagens sem ruídos do robô e dos objetos caixa e cilindro nos planos $x0y$ e $x0z$, respectivamente.

A etapa de Representação e Descrição procura extrair características das imagens resultantes da segmentação através de descritores que permitem a identificação de elementos na imagem. Os descritores são representados por uma estrutura de dados adequada ao algoritmo de reconhecimento. Neste trabalho os descritores são rótulos usados para identificar os elementos. Cada elemento rotulado é representado por um número que é usado na extração de características dos mesmos. Esta etapa produz na saída os vetores $Lgxy[]$, $Lgxz[]$, $Lrxy[]$, $Lrxz[]$, $Lbxy[]$ e $Lbxz[]$ representando os elementos rotulados das imagens do robô e dos objetos caixa e cilindro nos planos $x0y$ e $x0z$, respectivamente.

De posse da identificação dos elementos dos objetos realizados na etapa anterior, é possível se obter os seus baricentros. Para se extrair o valor do baricentro se utilizam algoritmos baseados em cálculos de geometria dos valores das massas do corpo e de suas posições no espaço. Nesta etapa é apresentado os valores dos parâmetros bg, br e bb que

correspondem aos baricentros do robô e dos objetos caixa e cilindro, de cor verde, vermelha e azul, respectivamente. Nos casos de exceção onde não se pode identificar o robô e/ou os objetos nas imagens, os parâmetros ficam com valores *flags* para identificação do tipo de exceção. Cada valor *flag* identifica o tipo de exceção a ser tratado pelo supervisor do sistema.

3.3.1 Bloco RPCH

Uma forma de reduzir o tempo de processamento na extração dos parâmetros, é efetuar as etapas de PDI das imagens Ixy e Ixz em paralelo. As RPs têm se destacado amplamente como ferramenta para modelagem de aplicações que trabalham em situações de concorrência, paralelismo e sincronização entre processos. Além disso ela apresenta um formalismo matemático com representação gráfica (MURATA, 1989).

Dentre as RPs de Alto Nível, as mais notáveis são as RPCHs. Estas redes são muito utilizadas na modelagem de aplicações complexas. Além disso, elas possibilitam a construção de um modelo através da combinação de um conjunto de redes relativamente menores denominadas páginas de forma análoga a construção de um programa a partir de um conjunto de módulos e funções.

O bloco *RPCH* inclui a RPCH para modelar as etapas de PDI que são usadas na extração de parâmetros das imagens capturadas do ambiente físico (Ixy e Ixy), conforme ilustrado na Figura 3.6. Além disso, esta rede atua como um supervisor em situações de exceção, tais como, falha na identificação dos objetos e do robô na imagem, e detecção de colisão entre o robô e os objetos. A RPCH está dividido nas seguintes páginas (subredes) descritas nas subseções seguintes: página de Controle; página de Supervisão; página de Conexão.

Página de Controle

A página de Controle, ilustrada na Figura 3.8 modela as etapas de PDI para a extração dos parâmetros *bg*, *br* e *bb*.

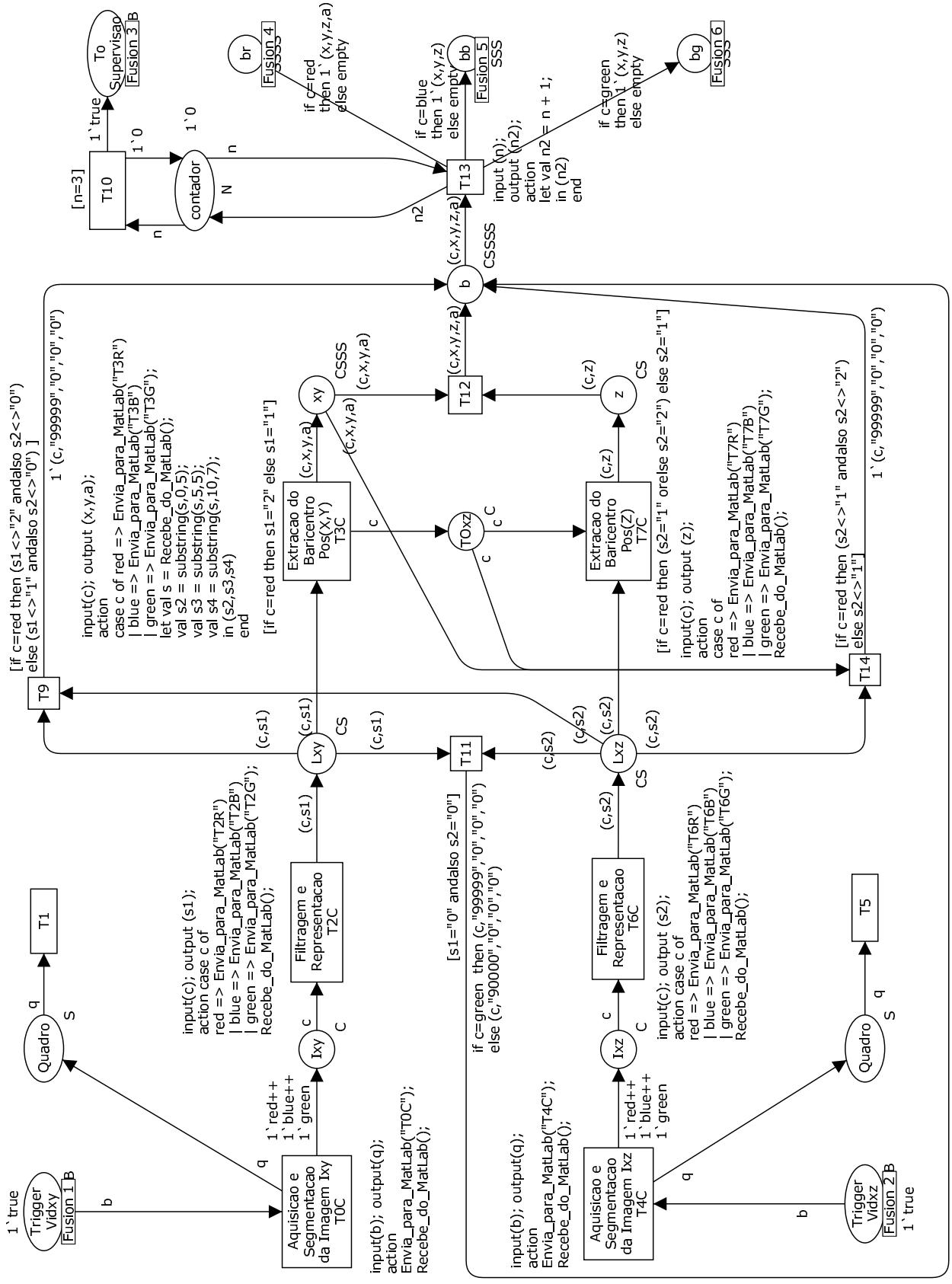

Figura 3.8: Página de Controle para execução das etapas de PDI.

Esta página contém transições que correspondem as etapas de PDI das imagens Ixy e Ixz. Na verdade, a RPCH foi desenvolvido para executar mais de uma etapa de PDI por transição. Como exemplo disso, tem-se a transição **T0C** que realiza as etapas de Aquisição e Segmentação de Imagem Ixy, e a transição **T2C** que realiza as etapas de Filtragem e Representação da Imagem Ixy. Observa-se também, que na rede existem transições idênticas para cada imagem Ixy e Ixz, tais como, as transições **T0** e **T4** (Aquisição e Segmentação da Imagem Ixy e Ixz, respectivamente), **T2C** e **T6C** (Filtragem e Representação), **T3C** e **T7C** (Extração do Baricentro Pos(x,y) e Pos(z)). A rede foi modelada dessa forma com o intuito de fornecer um processamento paralelo entre as etapas de PDI dessas imagens, e com isso, otimizar o processo de extração dos parâmetros. Na verdade, as etapas de PDI são executadas para cada objeto (caixa e cilindro) e para o robô de forma concorrente, através do disparo dessas transições. Essas transições possuem inscrições que contém as funções de envio e recebimento de dados ao/do MATLAB.

A única transição que não é executada em paralelo é a transição **T7C** (Extração do Baricentro Pos(z)) da imagem Ixz, pois esta depende do valor do componente y da imagem Ixy. Nessa condição, a extração do componente z deverá ser realizada após a extração do componente y da imagem Ixy. Esta dependência acontece porque o valor de z (valor simétrico à altura h de um dos objetos ou do robô) extraído da imagem Ixz varia à medida que o objeto se move na direção do eixo y no plano $x0y$, conforme ilustrado na Figura 3.7; quando o objeto se move no sentido negativo do eixo y (ele se distancia da câmera WebcamXZ), o valor de z aumenta (h diminui). Caso contrário, ele diminui (h aumenta) de valor.

Uma forma de manter o valor de z aproximadamente constante em qualquer ponto do plano $x0y$, é utilizar um polinômio que relacione as variáveis de z e y. Esse polinômio é gerado utilizando o método de mínimos quadrados [3] a partir de valores de z e y amostrados das imagens Ixz e Ixy.

A extração de cada parâmetros br, bb ou bg acontece durante o disparo da transição **T12**; e isso só ocorre quando existem fichas nos lugares de entrada **XY** e **Z**. Esta transição é utilizada para fazer o sincronismo dos processos de PDI concorrentes de cada imagem Ixy e Ixz. Cada parâmetro bg, br e bb é composto pelos componentes de x, y e z, exceto o parâmetro br que inclui a mais o componente angular a. O componente a fornece o ângulo de rotação do objeto caixa em torno do eixo z do plano $x0z$.

Em situações em que os objetos e/ou o robô não possam ser identificados nas imagens Ixy e/ou Ixz, o sistema atribui aos parâmetros um valor *flag* para indicar o problema a ser tratado na página de supervisão. Na verdade, pode acontecer duas situações em que

[3] Técnica de optimização matemática que procura encontrar o melhor ajustamento para um conjunto de dados tentando minimizar a soma dos quadrados das diferenças entre a curva ajustada e os dados.

os parâmetros recebem *flags*. Uma delas é quando o objeto não se encontra no ambiente robótico, e neste caso, o *flag* recebe valor igual a "90000". A outra é quando o objeto ou o robô não são identificados em pelo menos uma das imagens Ixy ou Ixz, e neste caso o *flag* assume valor "99999". Na rede, esses *flags* são armazenadas na variável x de cada parâmetro.

Página de Supervisão

A página de Supervisão, ilustrada na Figura 3.9 consiste na atuação sobre o sistema baseada nos parâmetros *br*, *bb* e *bg* obtidos da página de Controle.

Na verdade, de acordo com o valor *flag* (x) desses parâmetros, o supervisor toma uma ação específica, conforme a Tabela 3.1.

Tabela 3.1: Ação do Supervisor a partir da situação dos parâmetros

Valor *Flag* (x)	Situação do objeto	Ação no sistema
Se x=90000	*Objeto não encontrado.*	*Limpar objeto no AV.*
Se x=99999	*Objeto não identificado.*	*Erro! Parar Robô.*
Se x <> (90000 e 99999)	*Objeto identificado.*	*Atualizar localização do objeto no AV.*

Quando o valor *flag* x é igual a "90000", significa que o objeto (caixa ou cilindro) não se encontra no ambiente físico, e neste caso, o supervisor do sistema envia um comando ao operador para remover o objeto do AV. Isso acontece durante o disparo da transição **T19** ou **T22** (Limpar *br* do AV ou Limpar *bb* do AV, respectivamente).

Quando o valor *flag* x é igual a "99999", significa que o robô ou o objeto (caixa ou cilindro) não foram identificados em pelo menos uma das imagens (Ixy ou Ixz) do ambiente físico, e neste caso, o supervisor do sistema envia um comando ao SCR para parar o robô. Isso acontece durante o disparo da transição **T18** ou **T21** ou **T23** (Erro *br* Parar Robô ou Erro *bg* Parar Robô ou Erro *bb* Parar Robô, respectivamente) .

Quando o valor *flag* x é diferente de "90000"e de "99999", significa que o robô ou o objeto foi identificado, e neste caso, o supervisor do sistema envia o parâmetro ao cliente para atualização do AV. Isso acontece durante o disparo da transição **T17** ou **T20** ou **T24** (Atualizar *br* no AV ou Atualizar *bg* no AV ou Atualizar *bb* no AV, respectivamente).

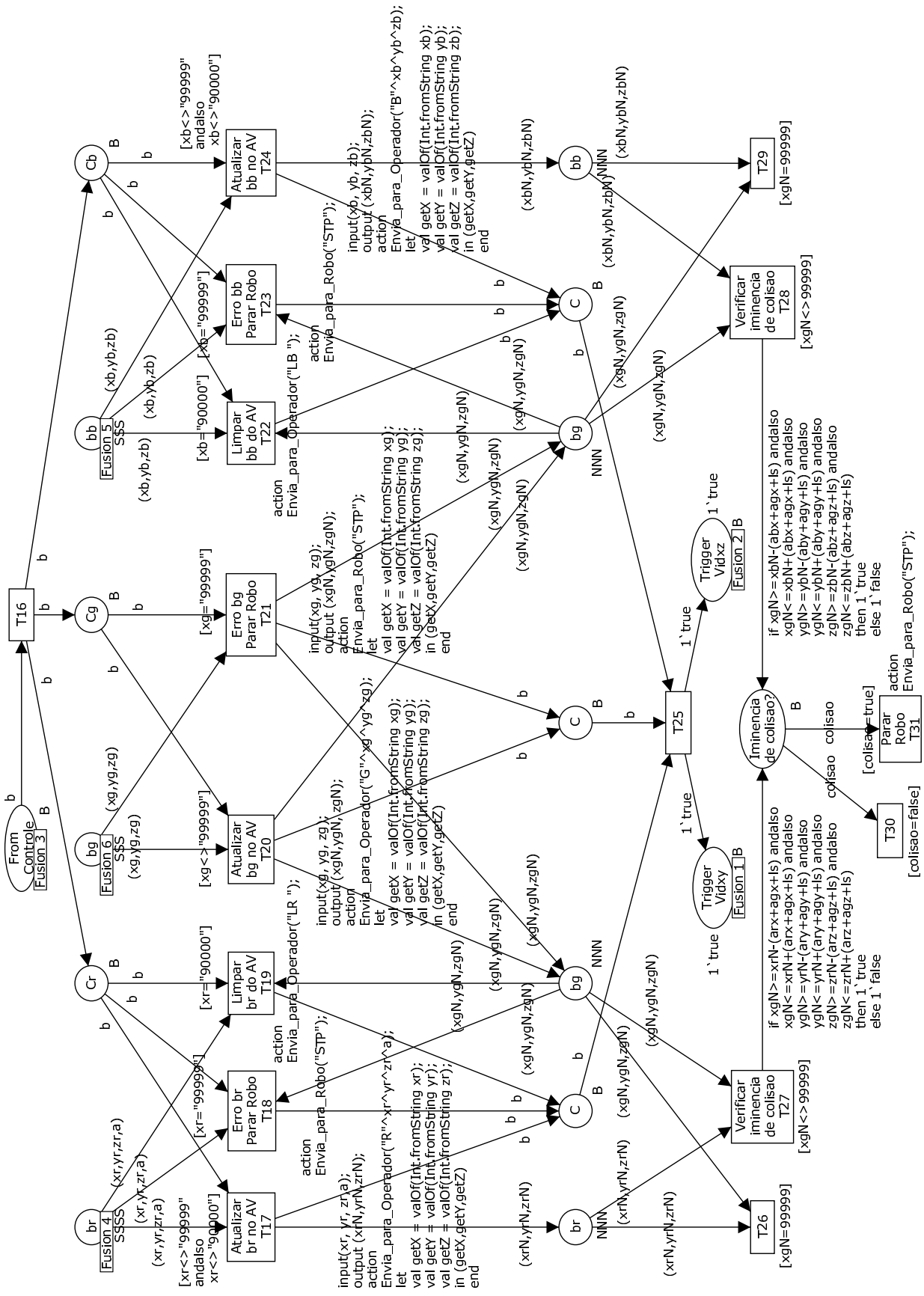

Figura 3.9: Página de Supervisão para atuar sobre o sistema.

Em caso de iminência de colisão (entre o Robô e um dos objetos), ou seja, quando uma das inscrições dos arcos de entrada do lugar **Iminência de colisão?** for satisfeita ($colisao = true$), o supervisor do sistema envia um comando ao SCR para parar o robô. Isso acontece durante o disparo da transição **T31** (Parar Robô). Essas inscrições verificam se a posição do baricentro do robô g está dentro da faixa de limite de segurança ls. Como exemplo, o caso de iminência de colisão entre o robô g e a caixa r é verificado pela condição descrita na equação 3.1. Essa condição é descrita sem as coordenadas x, y e z para simplificar o entendimento.

$$Se\ (r - (ar + ag + ls) \leq g \leq r + (ar + ag + ls))\ então\ colisao\ =\ true \tag{3.1}$$

onde ar e ag são a metade dos lados de r e g, respectivamente, e ls é o valor limite de segurança, que é aproximadamente 1,83cm. Nesta equação é verificado se o valor da posição do baricentro de g (robô) está dentro do intervalo de segurança $[(r - (ar + ag + ls), r + (ar + ag + ls)]$. Essa verificação é ilustrada no gráfico 2D da Figura 3.10, onde o ponto g está dentro da área de intersecção entre dois quadrados. Nota-se que, a distância entre os lados desses quadrados é exatamente o valor ls, escolhido de forma intuitiva, o que corresponde a uma melhor distância de segurança. O processo de verificação de colisão consiste em usar formas retangulares em todos os objetos, e com isso, simplificar os cálculos de comparação, afim de detectar colisão. Este trabalho de pesquisa não tem a intenção de desenvolver um novo mecanismo para detecção de colisão, e sim, validar se a replicação das informações reais estão sendo efetuadas no ambiente sintético. A mesma analogia serve para o teste entre o robô e o cilindro.

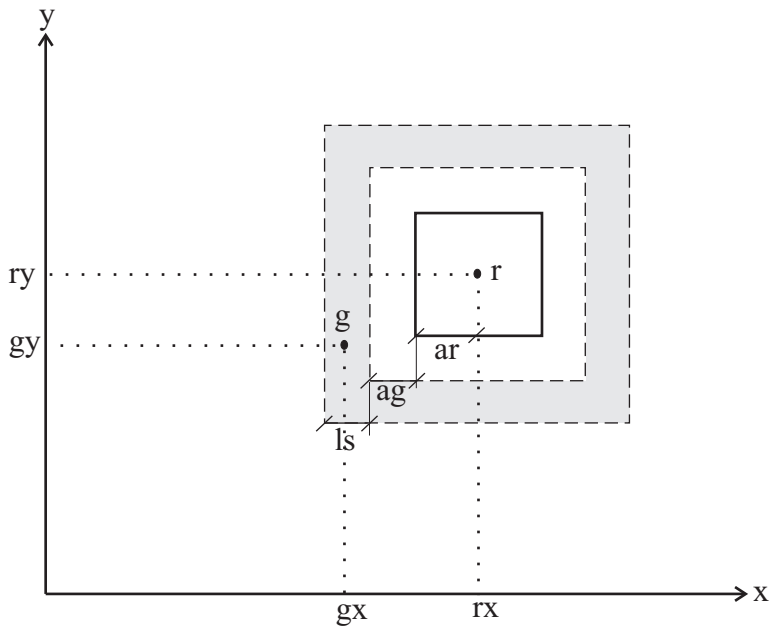

Figura 3.10: Representação gráfica de colisão entre o robô e o objeto caixa.

Página de Conexão

A página de Conexão com o Matlab, por sua vez, consiste na ação de conexão com o Matlab propriamente dita. Esta página é ilustrada na Figura 3.11.

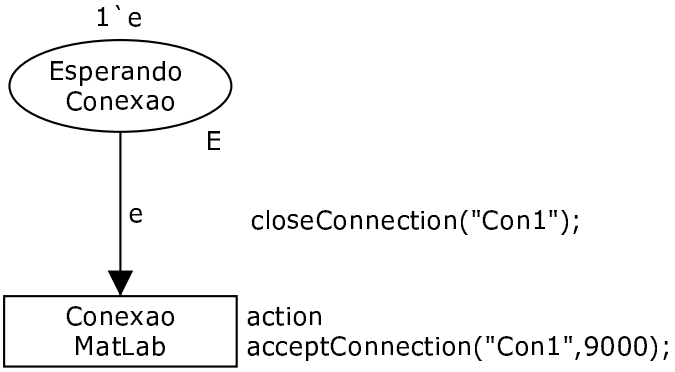

Figura 3.11: Página de Conexão com o Matlab.

3.3.2 Bloco PDI

O bloco *PDI* corresponde a implementação das etapas de PDI ilustradas na Figura 3.6. Os programas de PDI foram desenvolvidos utilizando as ferramentas de aquisição de imagem (Image Acquisition Toolbox) e processamento de imagem (processing image toolbox) do Matlab. Além disso, para fazer a integração entre os blocos *PDI* e *RPCH* foi utilizada a ferramenta de comunicação TCP/UDP/IP (TCP/UDP/IP toolbox). Estes programas incluem as funções *mainHSI.m*, *segmentarImagemHSI.m* e *segmentarImagemHSI2.m* desenvolvidas utilizando a linguagem de programação do Matlab.

Nesta subseção é apresentada estas funções de forma resumida. O código completo dessas funções podem ser vistos no apêndice E.

Função mainHSI

A função *mainHSI.m* é o programa principal que contém as seguintes rotinas:

- Rotina de Configuração dos Vídeos *vidxy* e *vidxz*;
- Rotina de Conexão entre o Matlab e o CPNTools;
- Rotinas de PDI.

Rotina de Configuração dos Vídeos vidxy e vidxz

A rotina de configuração dos vídeos *vidxy* e *vidxz* usa comandos da ferramenta de Aquisição de Imagem do Matlab. Abaixo é apresentado um trecho do código de configu-

ração do vídeo *vidxy* onde é descrito o código linha a linha. O processo é análogo para a configuração do vídeo *vidxz*:

```
vidxy = videoinput('winvideo');
triggerconfig(vidxy,'manual');
set(vidxy,'TriggerRepeat',fmax);
set(vidxy,'FramesPerTrigger',1);
vidxy.FrameGrabInterval = 1;
start(vidxy);
```

A primeira linha de comando cria o objeto de vídeo *vidxy* que a ferramenta de aquisição de imagem usa para fazer a conexão entre o Matlab e a câmera $WebcamXY$. A descrição 'winvideo', denominada adaptador de dispositivo, fornece o *driver* de conexão com a câmera de vídeo.

Uma vez estabelecida a conexão com o dispositivo externo (câmera de vídeo), a segunda linha de comando configura o modo manual de disparo para aquisição das imagens do vídeo. Esse modo permite que a captura das imagens sejam iniciadas manualmente através do comando $trigger(vidxy)$.

A terceira e quarta linhas especificam o número de disparos para aquisição das imagens e a quantidade de imagens capturadas durante a aquisição, respectivamente.

A quinta linha determina a freqüência requisitada de uma imagem durante o fluxo corrente de vídeo (video stream), que neste caso é 1 (um) quadro de imagem a ser capturado por vez.

A última linha de comando, por sua vez, é usada para iniciar o fluxo de vídeo, sem entretanto, fazer a aquisição de imagem para a memória.

Rotina de Conexão entre o Matlab e o CPNTools

A rotina de conexão para comunicação entre as ferramentas Matlab e o CPNTools usa comandos das bibliotecas *TCPIP Toolbox 1.2.4* e *TCP/UDP/IP Toolbox 2.0.5* (RYDESäTER, 1998-2002) para o MATLAB. Essas bibliotecas permitem criar as conexões TCP/IP, ou enviar/receber pacotes UDP/IP ao/do MatLab. Mais detalhes sobre essas bibliotecas podem ser vistas no apêndice B. Abaixo é apresentado a configuração dos parâmetros de comunicação do Matlab, bem como, os comandos de leitura e escrita, ou seja, recebimento e envio de dados entre o Matlab e o CPNTools via rede local.

```
con1=pnet('tcpconnect','localhost',9090);
con2=tcpip_servsocket(10000);
dadorecebido=tcpip_read(abs(conectado),3);
pnet(con1,'write',DadoEnv); %%% Envia DadoEnv
```

Existem dois parâmetros utilizados para fazer a conexão entre essas ferramentas; o *con*1 e o *con*2. O parâmetro *con*1 da primeira linha de comando cria a conexão *TCP* na porta 9090 para estabelecer a comunicação no sentido Matlab-CPNTools, enquanto que o parâmetro *con*2 da segunda linha de comando é usada no sentido oposto, isto é, da comunicação do CPNTools ao Matlab na porta 10000.

Rotina de PDI

A rotina de PDI, por sua vez, é um *loop* que contém rotinas para execução das etapas de PDI. Essas rotinas estão dentro de instruções *case* que são executadas durante os disparos das transições da página de controle. Ou seja, cada transição disparada executa uma rotina no Matlab que, por sua vez, após a execução retorna o valor de resultado ao CPNTools para prosseguimento de outra etapa de PDI, conforme a RPCH. Como exemplo, tem-se a rotina *case* T0C que é chamada durante o disparo da transição **T0C** que corresponde as etapas de Aquisição e Segmentação da Imagem Ixy. Abaixo é apresentado o código dessa rotina.

```
case {'T0C'}  %%% Aquisição e Segmentação da Imagem Ixy
                % 1. Aquisição da Imagem Ixy
                trigger(vidxy); % Faz a aquisição da imagem e armazena em memória
                Ixy = getdata(vidxy,1); % Imagem do plano x0y (Imagem vista superior)

                % 2. Segmentação dos objetos da Imagem Ixy
                % Irxy - Objeto Caixa
                % Ibxy - Objeto Cilindro
                % Igxy - Objeto Robo
                segmentarImagensHSI(Ixy);
                DadoEnv=sprintf('%s\n',num2str(vidxy.FramesAcquired));
```

O comando $trigger(vidxy)$ faz a aquisição da imagem Ixy e armazena o resultado na memória *buffer*. Para recuperar o dado da imagem na memória utiliza-se o comando *getdata*.

Uma vez que se tem o dado da imagem, a função $segmentarImagensHSI$ é chamada para extrair da imagem Ixy, o robô e os objetos caixa e cilindro que são atribuídos às variáveis $Irxy$, $Ibxy$ e $Igxy$, respectivamente. A função $segmentarImagensHSI$ é descrita com detalhes na subseção posterior *Função segmentarImagensHSI*.

Outra rotina *case* relevante na etapa de PDI é a T2C que implementa as etapas de Filtragem das imagens e Representação dos elementos das imagens $Lgxys$, $Lrxys$ e $Lbxys$, representando as imagens do robô (g), da caixa (r) e do cilindro (b), respectivamente. Esta rotina, descrita abaixo para T2R, faz a remoção de todo o ruído da imagem do objeto

caixa, e a representação (rotulação) de todos os elementos encontrados nesta imagem. O processo é análogo para as outras imagens $Igxy$ e $Ibxy$.

```
case {'T2R'}  %%% Filtragem da Imagem Irxy e Rotulação de elementos
                % FILTRAGEM DA IMAGEM
                % 1. Elimina o ruído da Imagem Ibwrxy usando filtragem morfológica
                Ibwrxy = im2bw(Irxy,0.5);
                se = strel('disk',2);
                Ibwrxy = imopen(Ibwrxy,se);
                %figure, imshow(Ibwrxy);

                % REPRESENTAÇÃO E DESCRIÇÃO (Operação Morfológica)
                % 2. Estiqueta os Elementos para posterior identificação
                [labeledrxy, numObjectsrxy] = bwlabel(Ibwrxy,4);
                % 3. Cria uma estrutura de medição para cada elemento
                % Estrutura de array com o campo de Baricentro
                datarxy = regionprops(labeledrxy,'basic');
                %imview(labeledrxy);

                DadoEnv=sprintf('%s\n',num2str(numObjectsrxy));
```

Antes da aplicação do filtro utiliza-se o comando $im2bw(Irxy, 0.5)$ para converter a imagem do formato RGB para o formato binário (1- branco e 0 - preto). Esse passo é necessário para se aplicar a filtragem morfológica por meio da operação de abertura. A operação de abertura é usada para remover os ruídos externos e do contorno do objeto e é descrita com detalhes no apêndice A.

Após isso, é definido o tipo e o tamanho do elemento estruturante se, através do comando $strel('disk', 3)$. O tamanho do elemento estruturante foi escolhido com valor maior ou igual ao aglomerado de *pixels* ruidosos, a fim de permitir uma melhor redução dos ruídos. E por fim, é utilizado o comando $imopen(Ibwrxy, se)$ para efetuar a operação de abertura.

Após a filtragem é necessário que os elementos da imagem sejam rotulados para posterior identificação. Isso é feito utilizando o comando $[labeledrxy, numObjectsrxy] = bwlabel(Ibwrxy, 4)$. Em seguida, o comando $datarxy = regionprops(labeledrxy,'basic')$ é utilizado para criar uma estrutura de vetor para cada elemento, afim de fornecer a posição de baricentro através da variável $datarxy$.

A etapa final de PDI que é a extração de parâmetro (baricentro) é efetuada pelas rotinas T3G (Gera Baricentro (X, Y) do Robô), T3R (Gera Baricentro (X, Y) do objeto Caixa), T3B (Gera Baricentro (X, Y) do objeto Cilindro), T7G (Gera Baricentro Z do Robô), T7R (Gera Baricentro Z do objeto Caixa) e T7B (Gera Baricentro Z do objeto Cilindro) que são executadas durante o disparo das transições **T3C** (Extração do Baricentro

Pos(x,y)) e **T7C** (Extração do Baricentro Pos(z)), através das funções de inscrição dessas transições. A rotina T3R é apresentada abaixo e corresponde a extração das coordenadas (x, y) do baricentro do objeto caixa. Esta rotina processa a extração do baricentro da imagem $Irxy$ que contém os elementos $c1(x1, y1)$ e $c2(x2, y2)$ utilizados para calcular o ângulo de rotação do objeto caixa em relação ao eixo z do plano x0z.

Esta rotina também trata da conversão das coordenadas do sistema de imagem para as coordenadas do sistema do robô. A conversão entre esses sistemas são definidas pelas equações 3.2 e 3.3 que foram deduzidas a partir de cada sistema de coordenadas.

$$x_R = \frac{F_R}{F_{Ix}} x_I \tag{3.2}$$

$$y_R = \frac{F_R}{F_{Iy}} y_I \tag{3.3}$$

Para a equação 3.2, x_R é o valor da abscissa no sistema Real, F_R e F_{Ix} são valores de fundo de escala dos sistemas Real e Imagem, respectivamente, e x_I é o valor da abscissa no sistema Imagem. A descrição é análoga para a equação 3.3, sendo que nesta é equalizado a ordenada do sistema Real.

O resultado da conversão é então formatado para ser enviado ao cliente (operador) pelo CPNTools durante o disparo da transição **T17** na página de Supervisão. Como o dado é um parâmetro, este é formatado conforme *layout* de formatação de parâmetro descrita no apêndice D.

```
case {'T3R'}  %%% Gera Baricentro (X,Y) do objeto Caixa de Irxy
                % 1. Pega posição do Baricentro (X,Y,A)
                c1 = datarxy(1).Centroid;
                x1 = round(c1(1));
                y1 = round(c1(2));

                c2 = datarxy(2).Centroid;
                x2 = round(c2(1));
                y2 = round(c2(2));

                xr = (x1 + x2)/2;
                yr = (y1 + y2)/2;

                % 2. Converte coordenadas do Sistema Imagem para Sistema Real
                xR = (FR/FIx)*xr;
                yR = (FR/FIy)*yr;

                % 3. Calcula angulo de rotação em radianos em torno de z no ponto (x,y)
                if (x2 ~= x1)
```

```
        a = atan((y2-y1)/(x2-x1));
    else
        a = pi/2; % a = 90; % 90 graus
    end
    a = num2str(a);
    while (length(a) < 7) % Completa tamanho à direita com zeros
        a = strcat(a,'0');
    end

    DadoEnv = strcat(formataXY(round(xR),round(yR),5), a);
    DadoEnv = sprintf('%s\n',DadoEnv);
```

O processo é análogo para as rotinas T3B e T3G, exceto que estas trabalham com as imagens $Ibxy$ e $Igxy$ e que só possuem um par de coordenadas (x, y) representando um único elemento na imagem.

A rotina T7C, por sua vez, trata da extração da componente z do baricentro, ou seja, da altura do objeto (caixa ou cilindro) ou do robô (suporte da caneta do Plotter) em relação ao plano $x0y$, conforme Figura 3.7. No código a seguir é apresentado a rotina T7R que é usada para fazer a extração do componente z da imagem $Irxz$ (imagem do objeto caixa no plano Ixz). Para as outras imagens o processo é análogo.

```
case {'T7R'}  %%% Gera Baricentro (Z) do objeto Caixa no plano Irxz
    % 1. Captura posição do Baricentro (Z)
    c = datarxz(1).Centroid;
    zi = round(c(2));
    [rows cols dim] = size(Ixz);
    zi = rows - zi;  % inverte sentido do eixo z

    % ------------------------------------------------------------
    % Faz a compensação da posição de z pela função quadrática
    % do polinomio p. Assim, o valor de z se manterá constante
    % em todo o plano xy.
    % ------------------------------------------------------------
    zp = p(1)*xr^2 + p(2)*xr + p(3);
    z = zi - zp + hr;

    % 2. Converte coordenadas do Sistema Imagem para Sistema Real
    zR = (FR/FIz)*z;

    % 3. Formata zR
    zR = num2str(round(zR));
    while (length(zR) < 5) % Completa tamanho à esquerda com zeros
        zR = strcat('0',zR);
    end
```

```
DadoEnv=sprintf('%s\n',zR);
```

O valor z do baricentro é capturado através dos comandos $c = datarxz(1).Centroid$ e $zi = round(c(2))$, e seu sentido é invertido para fornecer a altura em relação ao plano x0z. Como foi discutido na subseção *Página de Controle*, o valor z varia à medida que o objeto se move na direção do eixo y (z é função de y do plano x0y). Para resolver esse problema, se utiliza o método de mínimos quadrados para fazer a compensação de z através da função quadrática $zp = p(1) * xr^2 + p(2) * xr + p(3)$, gerada a partir de valores amostrados de y e z das imagens Ixy e Ixz.

E por fim, o valor z é convertido para o sistema Real e formatado para ser enviado ao Cliente (Operador) pelo CPNTools.

Para todas as rotinas *case*, é utilizado o comando $pnet(con1,'write', DadoEnv)$ para enviar o dado de resultado ao CPNTools para dar prosseguimento a simulação da RPCH.

Função segmentarImagensHSI

A função segmentarImagensHSI é o algoritmo de segmentação usado neste trabalho, que se baseia em cálculos estatísticos de desvio padrão e média aritmética dos componentes H e S de cada cor (R, G e B) das imagens Ixy e Ixz. O valor de desvio padrão é usado para medir a dispersão desses componentes em relação ao valor médio. Na verdade, esse algoritmo consiste em verificar se os componentes H e S de cada *pixel* da imagem estão dentro de intervalos de distribuição. Dessa forma é possível extrair objetos da mesma cor, ou seja, da mesma matiz e saturação.

Os valores de desvio padrão (s) e média aritmética (m) utilizados nesse algoritmo foram gerados pela amostragem dos componentes H e S das imagens Ixy e Ixz, escolhendo uma área da imagem de cor (r, g ou b) e fazendo o cálculo de desvio padrão e média aritmética para cada componente H e S. Tem-se então, uma lista de valores para média da matiz (mh), desvio padrão da matiz (sh), média da saturação (ms) e desvio padrão da saturação (ss) para cada cor r (vermelho), g (verde) e b (azul) de cada imagem Ixy e Ixz o que representa cada objeto no ambiente físico. Abaixo é apresentado um trecho do código que contém os valores de mh, ms, sh e ss amostrados da cor vermelha que representa o objeto caixa na imagem Ixy.

```
% Valores médios e de desvio padrão dos componentes
% H e S da cor Vermelha (Objeto Caixa)
% Valores obtidos a partir de amostragem da Imagem Ixy
mhr = 0.9499; msr = 0.7851; shr = 0.0975; ssr = 0.1166;
```

Como exemplo de utilização dos intervalos de distribuição, é apresentado abaixo um

trecho do código utilizado para extrair o objeto caixa da imagem Ixy. O processo é análogo para as imagens do robô ($Igxy$) e do objeto cilindro ($Ibxy$).

```
% Faz a segmentação das Imagens Ixy
[rows cols dim] = size(Ixy);
for i = 1:rows
    for j = 1:cols
        % Pega componentes de cromaticidade (matiz e saturação) de Ixy
        h = double(HSVxy(i,j,1));
        s = double(HSVxy(i,j,2));

        % ---------------------------------------------------------------
        % segmenta Objeto Caixa de cor vermelha na imagem Irxy (plano x0y)
        % ---------------------------------------------------------------
        if ( (h >= (mhr-2*shr)) & (h <= (mhr+2*shr)) ) &
           ( (s >= (msr-2*ssr)) & (s <= (msr+2*ssr)) )
            % Poe cor branca ao objeto identificado
            Objrxy = true;
            Irxy(i,j,1) = 255; Irxy(i,j,2) = 255; Irxy(i,j,3) = 255;
        else
            % Poe fundo preto
            Irxy(i,j,1) = 0; Irxy(i,j,2) = 0; Irxy(i,j,3) = 0;
        end
        :

    end
end
```

O algoritmo consiste em varrer toda a imagem linha a linha, fazendo uma verificação dos valores de h e s com os intervalos de distribuição. Neste exemplo, o valor dos desvios padrão (sh e ss) são multiplicados por 2 (dois) de cada lado da média, o que significa dizer, que a distribuição ocupa 95,45% dos casos.

Para a imagem Ixz, o processo de segmentação é análogo, sendo tratado na função $segmentarImagensHIS2.m$.

3.3.3 Interface de Integração

A Interface de Integração, ilustrada na Figura 2.3(a), é a ponte de ligação entre os blocos *HCPN* e *PDI*, respectivamente. Além disso, ela é usada para enviar os parâmetros ao Módulo de Comunicação Servidor/Cliente para que o mesmo possa transmiti-los ao cliente, ou ao SCR em casos de exceção (falhas de identificação de objetos e detecção de colisão). Essa interface realiza a integração entre as ferramentas Matlab e CPNTools. Para isso, é utilizado uma classe em linguagem Java chamada MatlabCPNComms.java.

Essa classe utiliza a biblioteca Comms/CPN (GALLASCH; KRISTENSEN, 2001) para implementar a comunicação com o CPN Tools, enquanto que a comunicação com o Matlab é implementada através das *toolboxes* TCP/IP Toolbox 1.2.4 e TCP/UDP/IP Toolbox 2.0.5 (RYDESäTER, 1998-2002). A Integração entre essas ferramentas é descrita com detalhes no apêndice B.

As classes utilizadas na interface de Integração são apresentadas no diagrama de classes da Figura 3.12.

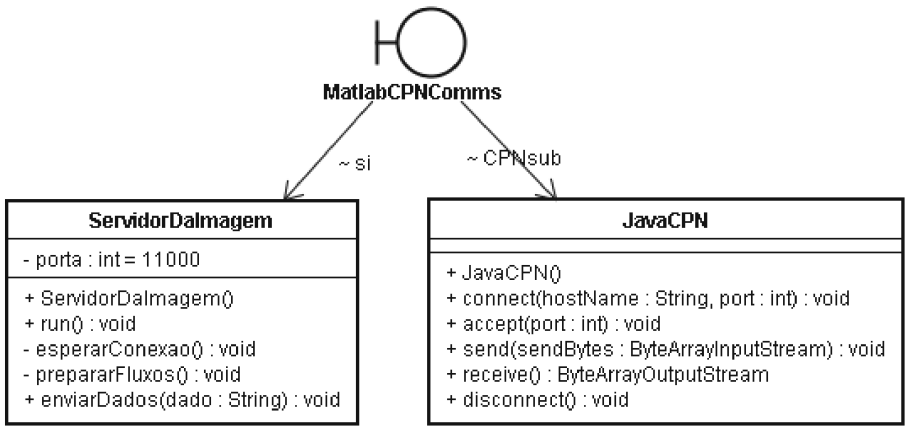

Figura 3.12: Diagrama de classes do Módulo de Imagem.

A classe ServidorDaImagem é o servidor de soquete utilizado para receber conexão do cliente de imagem (classe ClienteDaImagem) do Módulo de Comunicação Servidor/Cliente. Esta classe é responsável pelo envio dos parâmetros ao Módulo de Comunicação Servidor/cliente.

A classe JavaCPN é utilizada para realizar a comunicação com o CPNTools através da Comms/CPN, e é descrita com detalhes no apêndice B.

A classe de fronteira MatlabCPNComms, por sua vez, é a interface de conexão entre o Matlab e o CPNTools através da classe JavaCPN. Na verdade, essa classe é utilizada para instanciar as classes ServidorDaImagem e JavaCPN, e servir como fronteira para comunicação com outras classes do sistema.

3.4 Módulo de Comunicação Servidor/Cliente

O Módulo de Comunicação Servidor/Cliente é a interface de comunicação entre os subsistemas servidor e cliente via Internet, e entre o Módulo de Imagem e o SCR via rede local. De acordo com o esquema de software da Figura 2.3(a), os pacotes de dados que trafegam entre os subsistemas e entre o Módulo de Imagem e o SCR são pacotes de dados de parâmetro e de controle. Na comunicação externa entre os subsistemas os pacotes

de dados de parâmetro trafegam no sentido servidor-cliente, ao passo que, os pacotes de dados de controle trafegam no sentido oposto. Com relação a comunicação interna no subsistema servidor, os dados de parâmetro e de controle trafegam apenas no sentido Módulo de Imagem-SCR.

Na comunicação interna e externa, os dados de controle e de parâmetros são formatados usando *strings* ASCII. Cada pacote de dados é uma *string* que pode conter um comando de controle ou um parâmetro para atualização do AV. O pacote mais crítico se refere ao dado de parâmetro que contém as posições do robô ou dos objetos. Este pacote de dado pode conter um tamanho de até $23 Bytes$ que devem ser recebidos no subsistema cliente a uma taxa mínima de 5 a 6 quadros por segundo para permitir a visualização atualizada do AV com as posições correntes do robô e objetos. As formatações dos pacotes de dados de comando e de parâmetro seguem os *layouts* de formatação de comando e formatação de parâmetro descritos no apêndice D.

O Módulo de Comunicação Servidor/Cliente utiliza soquete de fluxo como mecanismo de transmissão de fluxo de dados, tanto para a comunicação via Internet, como para rede local. Este tipo de soquete fornece um serviço orientado a conexão e o protocolo utilizado é o TCP. Este protocolo fornece um serviço de fluxo *full duplex*, com tratamento automático de erro, retransmissão, reordenação de pacotes, e garantia de entrega. Em STIs, o TCP é freqüentemente usado na transmissão de comandos de controle e outros dados importantes. Para um estudo mais detalhado sobre a teoria e aplicação de protocolos TCP/IP, o leitor deve se reportar a (TANENBAUM, 2003).

As classes desse módulo são apresentadas no diagrama de classes da Figura 3.13.

A classe *MainLab* é usada para implementar os métodos *processarDadoDaImagem* e *processarDadoDoCliente* das interfaces *IOuvinteDaImagem* e *IOuvinteDoCliente*, respectivamente. Esses métodos são utilizados para processar dados vindos do cliente e do módulo de imagem. Para enviar dados (parâmetros) ao cliente, esta classe utiliza o método *enviarDados* da classe *Servidor*. Na verdade, a classe *MainLab* instancia as classes *ClienteDaImagem* e *Servidor* para ter acessos a seus métodos para enviar e receber dados ao/do cliente. Os passos utilizados para se estabelecer um servidor utilizando soquetes de fluxos são descritos com detalhes no apêndice C.

Abaixo são listados os métodos utilizados no intercâmbio de dados entre os subsistemas:

- *processarDadoDoCliente(String dado)*: Método usado para receber dado de controle do cliente;
- *servidor.enviarDados(String dado)*: Método usado para enviar dado ao cliente;

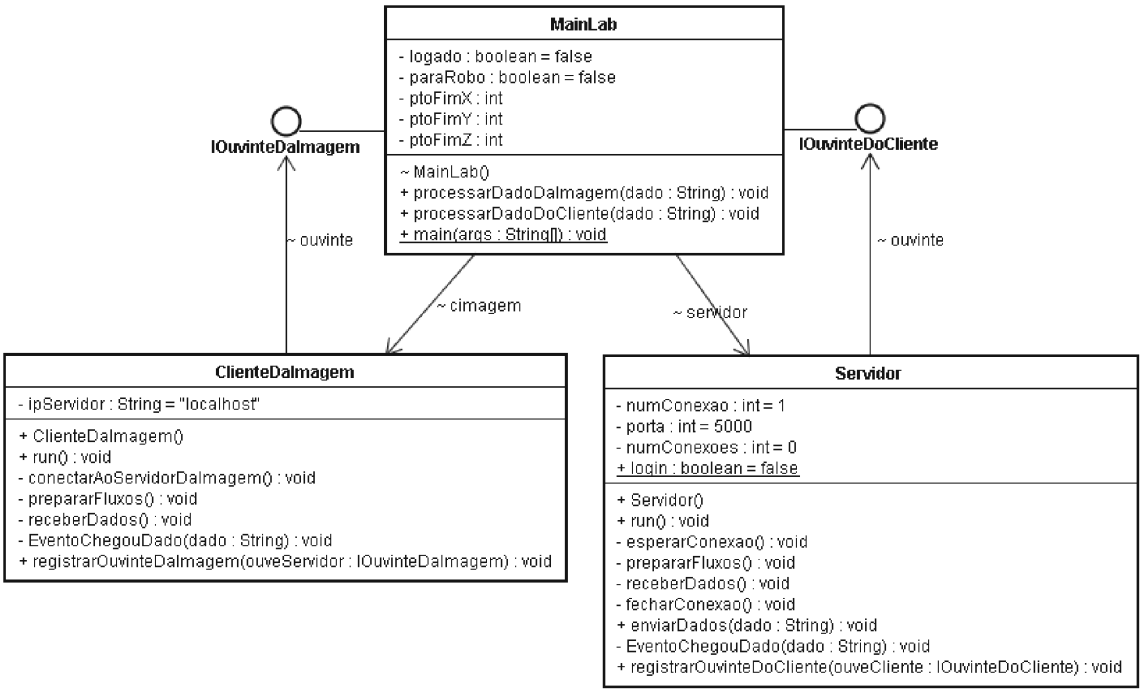

Figura 3.13: Diagrama de classes do Módulo de Comunicação Servidor/Cliente.

- *processarDadoDaImagem(String dado)*: Método usado para receber dado de parâmetro do Módulo de Imagem.

Além disso, a classe *MainLab* é usada para instanciar as classes da biblioteca de controle do SCR (classes *AP3Device* e *PointReacher* apresentadas na seção 3.2), afim de acessar os métodos de controle do robô.

O código completo das classes *MainLab*, *Servidor* e *ClienteDaImagem* pode ser visto no apêndice E.

A interação em seqüência entre esses métodos pode ser visualizada no diagrama de seqüência da Figura 3.14. Esse diagrama consiste principalmente no envio de parâmetros ao Subsistema Cliente, partindo da aquisição das imagens pelos atores RPCH e Matlab, até o envio propriamente dito ao subsistema cliente. Observa-se que, dependendo do valor do parâmetro (dado) recebido pelo método *processarDadoDaImagem*() da classe *MainLab*, este pode ser enviado ao Subsistema Cliente (quando o valor é diferente de "STP"), ou ser enviado ao SCR (quando o valor é igual a "STP"), em caso de exceção.

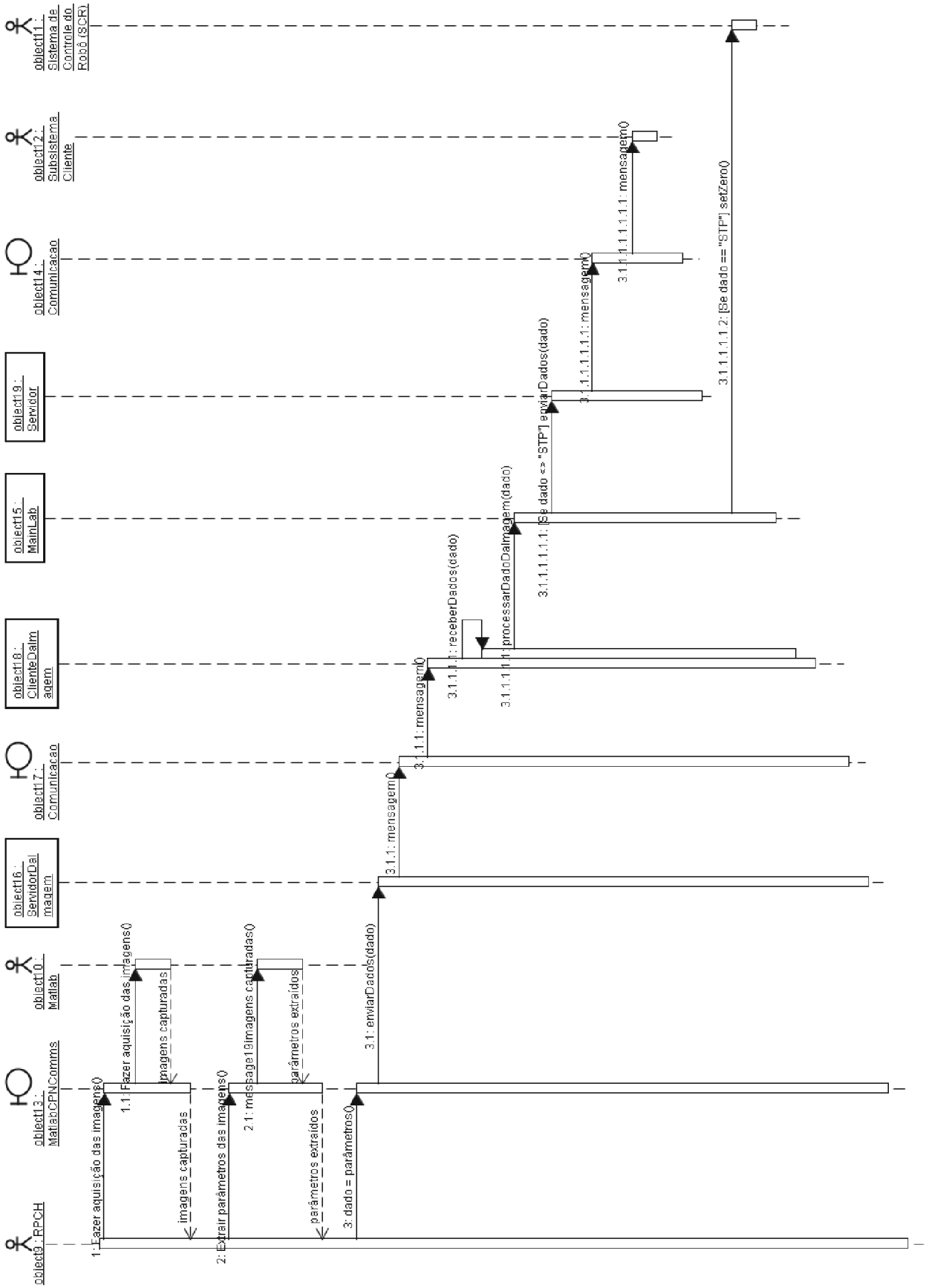

Figura 3.14: Diagrama de seqüência Enviar Parâmetro ao Subsistema Cliente.

3.5 Conclusão

Neste capítulo foi discutido o projeto do subsistema servidor, a partir da realização dos casos de uso do mesmo através de diagramas de UML (classes e seqüência) para modelagem do projeto dos módulos do subsistema. O módulo de imagem foi modelado de forma detalhada, apresentando o projeto de seus componentes, tais como, a *RPCH* e o *PDI*, e sua interface de integração. Foi também apresentado o sistema SCR que fornece os métodos de acesso ao robô Plotter, e por fim, foi discutido o módulo de comunicação servidor/cliente, apresentando o projeto de seus componentes.

Capítulo 4

Subsistema Cliente

Neste capítulo é descrito o projeto do Subsistema Cliente do STI. Este subsistema é responsável por fornecer as Interfaces de Usuário (IUs) para o operador interagir com o robô fornecendo dados de controle e recebendo dados do robô e do ambiente, para serem visualizados no AV de resultado.

De acordo com a Figura 2.3(b), este subsistema inclui os módulos de AV de controle e de resultado, e o módulo de Comunicação Cliente/Servidor.

Na seção 4.1 é apresentado a Modelagem do AV, através do uso da linguagem X3D e de métodos Java para acessar externamente o AV. Na seção 4.2 é abordado o diagrama de casos de uso para mostrar as funcionalidades do mesmo. Na seção 4.3, é apresentado o projeto do Módulo de AV de Controle por meio dos diagramas de classes e de seqüência. Na seção 4.4, é abordado o projeto do Módulo de AV de Resultado e apresentados os diagramas de classes e de seqüência. Na seção 4.5 é discutido o projeto do Módulo de Comunicação Cliente/Servidor que é utilizado para comunicação com o Subsistema Servidor.

4.1 Modelagem do Ambiente Virtual

A modelagem do AV inclui o robô virtual e a área de trabalho contendo os objetos caixa e cilindro. A modelagem do AV utilizado na IU é baseada nos conceitos de modelagem geométrica e cinemática [1]. Essas modelagens definem as características dos objetos do ambiente robótico, tais como, forma, aparência e comportamento.

O AV é criado utilizando a linguagem de modelagem virtual X3D, escolhida por empregar uma arquitetura modular e fornecer maior extensibilidade e flexibilidade (X3D, 2006). Sua principal característica para aplicações de STIs está associada a capacidade

[1] Modelagens geométrica e cinemática são conceituadas na subseção 1.2.3(Modelagem de Ambientes Virtuais) do capítulo 1 (Fundamentação Teórica) desta dissertação.

de operar em conexões com pequena largura de banda, o que é o caso da Internet.

Para a geração do robô virtual, utilizou-se o projeto de desenho auxiliado por computador (Computer Aided Design - CAD) conforme os desenhos das vistas superior e lateral com as dimensões em milímetros, ilustrados nas Figuras 4.1 e 4.2, respectivamente.

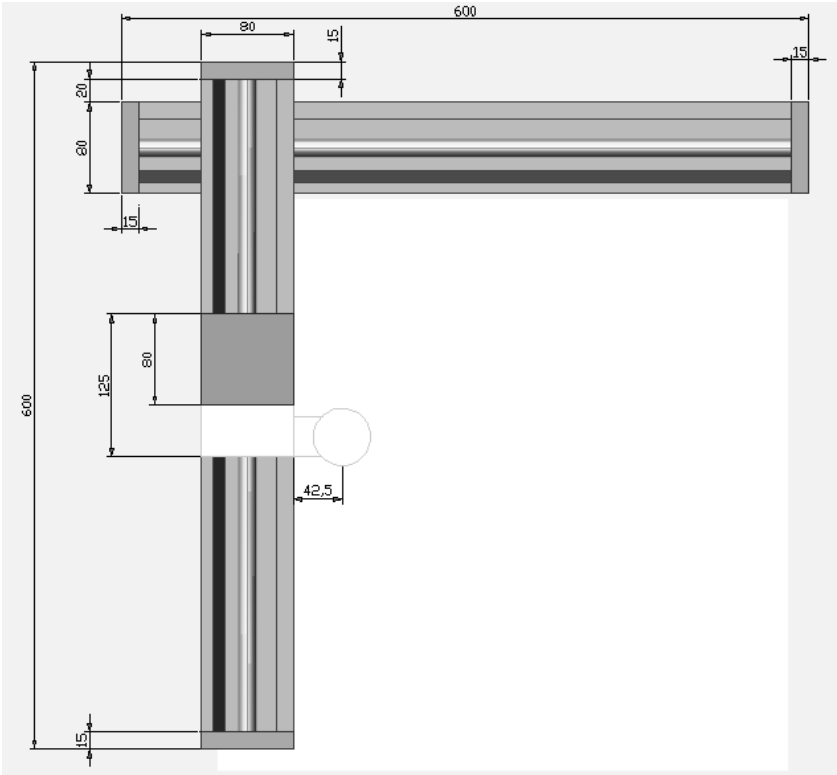

Figura 4.1: Desenho em CAD da vista superior do robô.

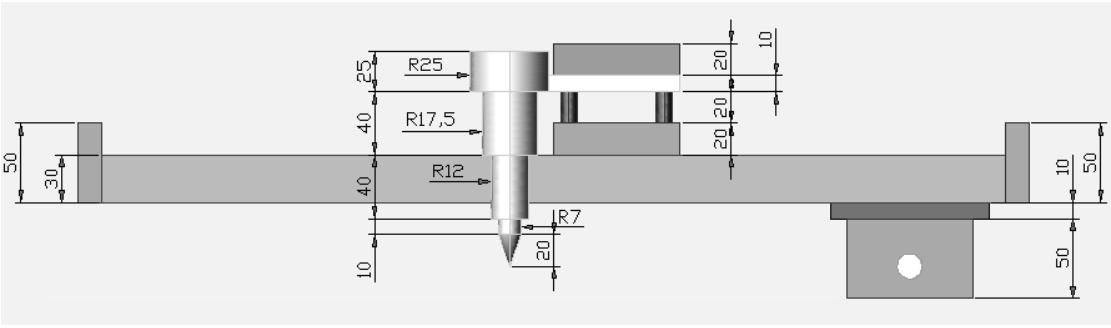

Figura 4.2: Desenho em CAD da vista lateral do robô.

De acordo com o robô real ilustrado na Figura 3.3, observa-se que o mesmo é constituído por formas primitivas simples, tais como, caixa, cilindro, e cone. Sendo assim, a modelagem do robô virtual se torna simplificada, podendo ser construída pela composição dessas formas. A linguagem X3D possui um perfil (profile) chamado *Interchange* que contempla a modelagem dessas formas sem a necessidade de se gerar formas complexas.

Para se fazer a modelagem geométrica do robô virtual, é necessário gerar a forma e a aparência do mesmo. Como exemplo, o trecho do código abaixo, na sintaxe VRML, demonstra como é gerado um braço virtual do robô:

```
DEF braco Shape {
    appearance Appearance {
         material Material {
              diffuseColor 0.3 0.3 0.3
         }
    }
    geometry Box {
         size 5.7 0.8 0.3
    }
}
```

O comando *DEF* define a variável *nó braco* que representa o braço robótico virtual a ser manipulado externamente por métodos Java. Os *nós geometry* e *appearance* são utilizados para gerar a forma e aparência, respectivamente, sem dar detalhes da aparência real do braço robótico. O comando *geometry Box* indica que a forma a ser gerada é uma caixa com dimensões de 5700mm de comprimento (5.7), 80mm de largura (0.8) e 30mm de altura (0.3).

Afim de dar realismo ao braço robótico, se utiliza a propriedade de textura que oferece um aumento do nível de detalhe e de realismo da cena, além de fornecer uma melhor visão de profundidade. Isso pode ser visto nas Figuras 4.3(a) e (b) para efeito de comparação.

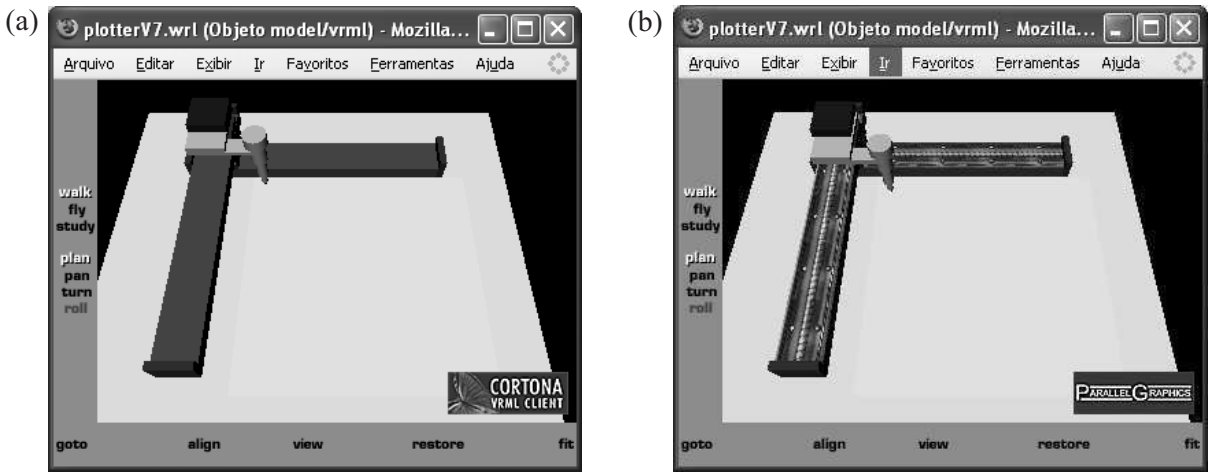

Figura 4.3: AV do robô: (a) AV sem textura; (b) AV com textura.

O código abaixo utiliza o comando *texture ImageTexture* que faz uma chamada ao arquivo da imagem de textura via *URL* [2] para ser rotulado ao braço robótico virtual.

[2] Localizador Uniformizado de Recursos, que permite identificar e acessar um serviço na rede Web.

Na verdade, o trecho de código em questão, funciona como um papel de parede que é sobreposto ao braço virtual do robô.

```
appearance Appearance {
    material Material { }
    texture ImageTexture {
        url "textura.jpg"
    }
    textureTransform TextureTransform {
        scale 5.0 1.0
    }
} geometry Box {
    size 5.7 0.8 0.01
}
```

Quanto a modelagem de outros componentes do ambiente robótico, tais como, a mesa, o papel e os objetos caixa e cilindro, o procedimento é o mesmo, com exceção de que, estes não utilizam a propriedade de textura, por não haver necessidade de detalhamento.

Com relação a modelagem cinemática do AV, ou seja, a alteração de posição do robô virtual, faz-se necessário obter as coordenadas do robô em relação ao sistema de coordenadas pai do AV [3], e o uso de matrizes de transformação para permitir a alteração de posições e mudanças de escala. O trecho do código a seguir implementa esse tipo de modelagem.

```
# Braço - Movimento no eixo X
      DEF Braco Transform {
         translation  -2.0 0.0 0.0
```

Os *nós Transform* e *translation* criam um sistema de coordenadas cartesianas posicionado em (-2,0,0) em relação ao sistema de coordenadas pai. Para se alterar a posição do braço robótico virtual, deve-se primeiro referenciar o *nó Braco* e depois fazer acesso ao campo *translation* para se alterar o valor de x. O processo é análogo para o suporte da caneta, e os objetos caixa e cilindro do AV. Na subseção a seguir são apresentados alguns métodos Java para manipular o robô e objetos no AV.

4.1.1 Obtendo Acesso Externo ao AV

Neste trabalho é utilizado o navegador Xj3D para a visualização de AVs X3D. O Xj3D é o navegador padrão aberto utilizado para navegação do ambiente virtual X3D (X3JD, 2006). Desenvolvido totalmente na linguagem Java, o Xj3D pode ser facilmente

[3] Sistema de coordenadas referencial para cada objeto construído no AV.

incorporado em aplicações Java. Essa facilidade é utilizada nesse trabalho para se ter acesso ao robô virtual externamente através de métodos Java.

Para se ter acesso externo ao AV, é necessário utilizar a API SAI (Scene Access Interface) (X3D, 2006). Essa interface permite que uma aplicação Java manipule *nós* de objetos virtuais de um AV, como ilustrado na Figura 4.4.

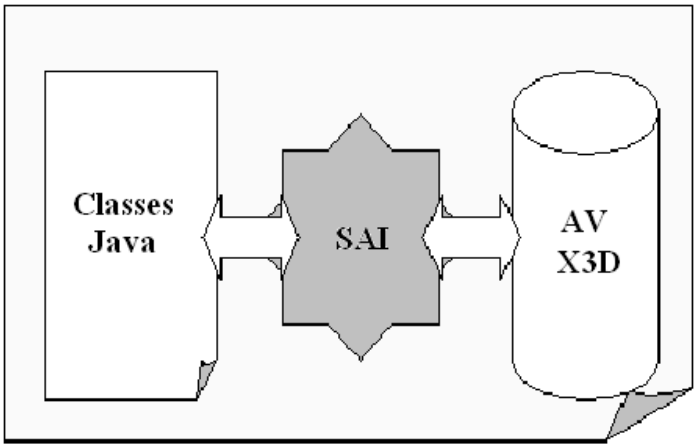

Figura 4.4: Diagrama de Acesso ao AV X3D utilizando SAI

Antes de se utilizar os métodos SAI para ter acesso ao AV, deve-se criar o componente X3D, usando a classe SAI *BrowseFactory*. Após isso, adiciona-se este componente à aplicação. O código em Java a seguir, mostra esses passos:

```
import org.web3d.x3d.sai.*;
public class GUILab extends
javax.swing.JFrame  {
    public GUILab() {
        // Cria componente X3D
        X3DComponent x3dComp = BrowserFactory.createX3DComponent(requestedParameters);

        // Adiciona o componente à Interface de Usuário
        JComponent x3dPanel = (JComponent) x3dComp.getImplementation();
        // ------------------------------------------------------------------------
        :
```

A fim de carregar o mundo virtual na janela 3D, é necessário fazer referência ao navegador X3D [4], obtendo uma instância do mesmo. Isso pode ser demonstrado no código a seguir:

```
// Obtem um navegador externo
ExternalBrowser x3dBrowser = x3dComp.getBrowser();
```

[4]Interface básica que permite que se faça tarefas como carregamento de arquivos, criação de novos *nós* e obtenção da taxa de quadros.

Uma vez que se tem a referência ao navegador, pode-se usá-la para carregar o arquivo X3D através do método *createX3DFromURL*. Este método retorna um objeto *X3Dscene* que encapsula uma cena X3D. Uma vez que se tem a referência para a cena, utiliza-se o método *replaceWorld* para carregar o AV:

```
// Cria uma cena X3D ao carregar o arquivo
mainScene = x3dBrowser.createX3DFromURL(new String[] { "plotterV7.x3dv" });
// Atualiza o mundo corrente com o novo
x3dBrowser.replaceWorld(mainScene);
```

Após a cena do AV ser carregada à janela 3D, pode-se acessar os *nós* definidos pelo construtor *DEF*. No exemplo a seguir, é definido o *nó* posCaixa que será usado externamente pelo SAI para alterar a posição do objeto Caixa.

```
# Objeto Caixa
DEF posCaixa Transform {
    translation 0.0 0.0 0.375
    rotation 1.0 0.0 0.0 1.57
    children [
        Shape {
            appearance Appearance {
                   material DEF matCaixa Material {
                      diffuseColor 1.0 0.0 0.0
                      transparency 1.0
                   }
            }
            geometry Box {
                size 1.2 0.75 0.75
            }
        }
    ]
}
```

Para acessar o *nó* posCaixa, utiliza-se o método *getNamedNode*() da classe *Browser*, onde é passado o nome do *nó*, como mostra o trecho do código a seguir:

```
// Encontrar  nó chamado posCaixa (Posição da Caixa)
X3DNode posCaixa = mainScene.getNamedNode("posCaixa");
 if (mat == null) {
       System.out.println("nó não encontrado: posCaixa");
       return;
}
```

Uma vez que a referência do *nó* é obtida, os seus campos tipo *inputOutput*[5] podem ser acessados. No exemplo citado, o valor do campo *translation* é obtido usando o método *getField*(), onde é passado o nome do *nó* (posCaixa):

```
// Obtém o valor pos do campo translation do nó posCaixa
SFVec3F pos = (SFVec3F) posCaixa.getField("translation");
```

E por fim, para modificar o valor de *pos* referenciado, utiliza-se o método *setValue*(), como se segue:

```
// Atribui o valor da nova posição
float[] novaPos = {x,y,z};
pos.setValue(novaPos);
```

Todas as classes necessárias para acessar SAI estão no pacote *org.web3d.x3d.sai.**.

4.2 Casos de Uso do Subsistema Cliente

A funcionalidade do Subsistema Cliente é ilustrada no diagrama de casos de uso da Figura 4.5. Observa-se que o subsistema servidor é um ator nesse diagrama, afim de mostrar sua interação com o subsistema cliente. Esse diagrama contém os casos de uso referentes as funcionalidades de interface gráfica de usuário, tais como, programar tarefa do robô, salvar tarefa, abrir tarefa, fazer login de conexão com o servidor, enviar comandos ao robô, etc.

Em termos gerais, a funcionalidade do subsistema cliente pode ser descrita como se segue:

- Primeiramente o operador programa o robô no AV em modo *off line* (caso de uso Programar Tarefa do Robô);
- O operador poderá, se quiser, salvar a tarefa para uso futuro (caso de uso Salvar Tarefa);
- O operador poderá abrir uma tarefa salva para ser enviada, ou ser alterada (Abrir Tarefa);
- O operador poderá enviar a tarefa desejada ao subsistema servidor do robô para execução (caso de uso Enviar Comandos ao Robô);
- Para enviar a tarefa, o operador deverá fazer *login* de conexão com o subsistema servidor do robô (caso de uso Fazer Login de conexão);

[5] Campos que podem ser modificados externamente pelo SAI.

- O resultado da operação poderá ser visualizado em tempo real no AV de resultado (caso de uso Visualizar Resultado em AV).

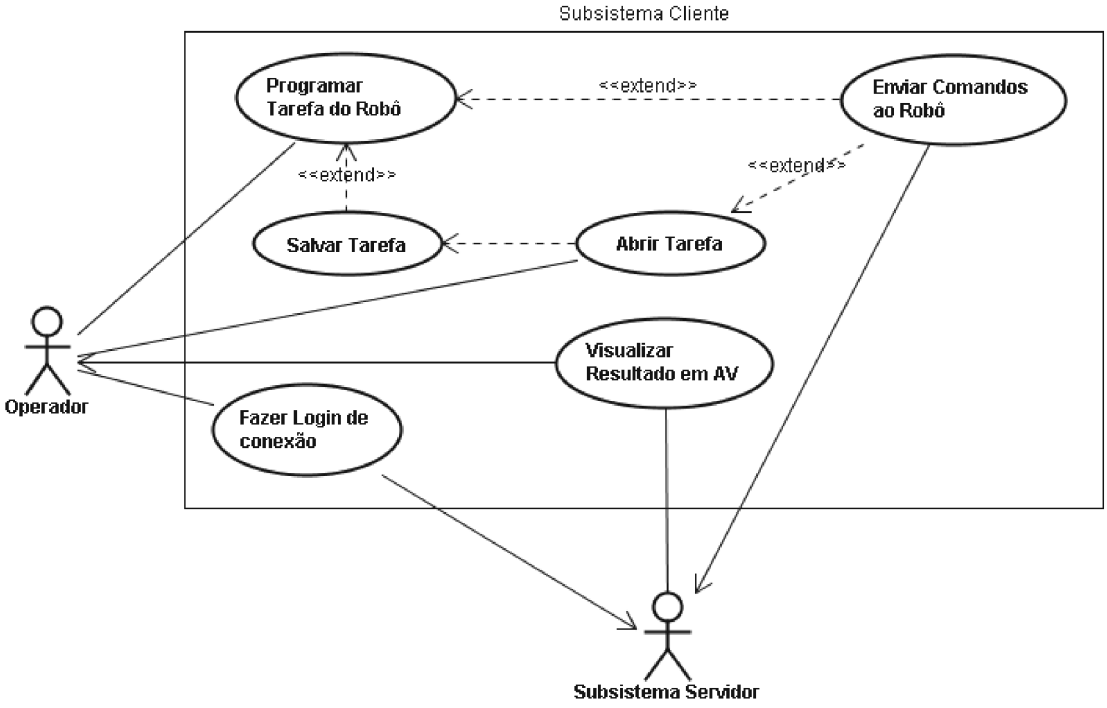

Figura 4.5: Diagrama de Casos de Uso do Subsistema Cliente

4.3 Módulo de AV de Controle

O Módulo de AV de Controle é a IU usada na programação de tarefas do robô. A programação do robô consiste na especificação *off-line* de tarefas e tem a característica de salvar a largura de banda da rede. Essa estratégia de controle é baseada na técnica de visualização antecipada (predictive display) para permitir ao operador "predizer"os movimentos do robô antes de enviar remotamente os comandos de controle ao robô real (SHERIDAN, 1992). O robô virtual é devidamente inicializado com a mesma posição do robô real, tendo a vantagem de responder instantaneamente as entradas do operador.

Nesta seção são abordados a descrição detalhada dos casos de uso Programar Tarefa do Robô e Enviar Comandos ao Robô, bem como, o projeto dos mesmos através de diagramas de seqüência e diagrama de classes do Módulo de AV de Controle.

4.3.1 Especificação do Caso de Uso Programar Tarefa do Robô

Nesta subseção é apresentado o detalhamento do caso de uso Programar Tarefa do Robô. Neste caso de uso são descritos os passos necessários para o operador programar

a tarefa do Robô. Como se trata de um robô Plotter, a tarefa consiste na elaboração *off-line* de desenhos. Na Tabela 4.1 é apresentada a descrição deste caso de uso.

Tabela 4.1: Especificação do Caso de Uso Programar Tarefa do Robô

Nome do Caso de Uso	Programar Tarefa do Robô
Ator Principal	Operador Humano
Pré-Condições	O AV deve estar carregado na IU
Condição Final com Êxito	Tarefa programada
Condição Final com Falha	Mensagem de erro
Forma de Disparo	O botão esquerdo do mouse
Ações do Ator	Ações do Sistema
1. Selecionar o tipo de figura (linha, retângulo ou círculo)	
2. Desenhar no painel de desenho usando o mouse, conforme tipo selecionado	
	3. Converter posições do desenho para o sistema de coordenadas do AV
	4. Exibir em AV para validar tarefa programada

4.3.2 Especificação do Caso de Uso Enviar Tarefa ao Robô

Nesta subseção é apresentado o detalhamento do caso de uso Enviar Comandos ao Robô. Este caso de uso descreve os passos necessários para o operador enviar comandos da tarefa ao robô, ou seja, parâmetros de posições inicial e final em que o robô deverá efetuar. Na Tabela 4.2 é apresentada a descrição deste caso de uso.

Tabela 4.2: Especificação do Caso de Uso Enviar Tarefa ao Robô

Nome do Caso de Uso	Enviar Tarefa ao Robô
Ator Principal	Operador Humano
Pré-Condições	Existir tarefa programada
Condição Final com Êxito	Tarefa enviada ao Robô
Condição Final com Falha	Mensagem de erro
Forma de Disparo	Um botão de envio
Ações do Ator	Ações do Sistema
1. Submeter a tarefa ao Robô	
	2. Converter posições do desenho para o sistema de coordenadas do ambiente real
	3. Formatar comandos de envio
	4. Enviar comandos ao Robô utilizando a Internet como meio de comunicação

4.3.3 Diagrama de Classes do Módulo de AV de Controle

Na Figura 4.6 é ilustrado o diagrama de classes do Módulo de AV de Controle. Este diagrama inclui as classes de fronteira (boundary) *GUILab* e *Login*, que são as IUs onde o operador interage com o sistema. A classe *GUILab* é a interface de controle por onde o usuário faz a programação do robô. Essa classe instancia as classes *PainelJava* (painel de desenho) e *PainelX3D* (painel de AV), por onde são realizadas a programação das tarefas do robô e a simulação da operação do robô em AV, respectivamente. A tarefa do robô consiste na elaboração de desenhos formados a partir de figuras de linhas retas, retângulos e círculos. A classe *GUILab* também instancia a classe *X3DScene* para gerar o objeto de cena X3D do AV e ter acesso aos métodos para manipulação do robô e objetos virtuais. A classe *Login*, por sua vez, é a interface formulário para que o usuário informe os dados de usuário e senha para ter acesso ao sistema robótico remotamente.

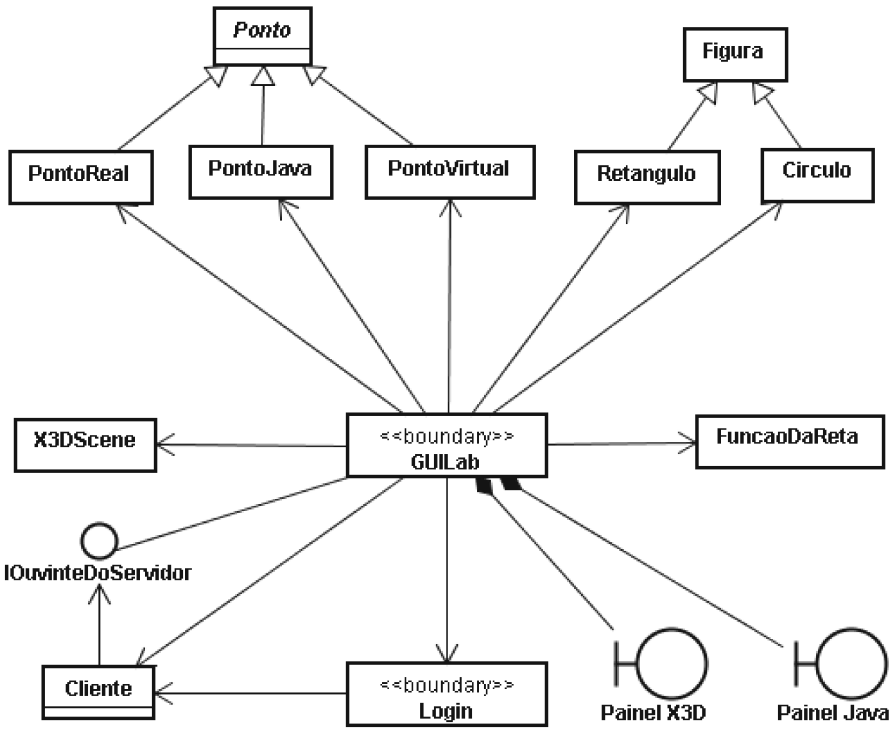

Figura 4.6: Diagrama de Classes do Módulo de AV de Controle.

Além das classes mencionadas, existem as classes auxiliares que são instanciadas pela classe *GUILab*, tais como, *PontoReal*, *PontoJava*, *PontoVirtual*, *Retangulo*, *Circulo* e *FuncaoDaReta*. Essas classes são utilizadas para dar suporte na programação do robô e simulação do robô virtual.

Os objetos *pontoReal*, *pontoJava* e *pontoVirtual* são utilizados na geração das posições do robô nos sistemas de coordenadas Real, Java e Virtual, respectivamente. A conver-

são entre esses sistemas de coordenadas é efetuado pelo próprio construtor de cada classe, utilizando o parâmetro $from$. Por exemplo, para a conversão de um ponto no sistema Java para o sistema Real, se utiliza o parâmetro $from$ com valor igual a "JAVA2REAL". Segue abaixo o código do método construtor para efetuar tal conversão.

```
// Exemplo de conversao de um Ponto Java para Ponto Real
ptoRealIni[numLinhas] = new PontoReal("JAVA2REAL", xIni, yIni, zFim,
                                      Ponto.xOffSet, Ponto.yOffSet);
:
// Método Construtor para criar Ponto Real a partir do Virtual ou do Java
public PontoReal(String from, float xCoordenada, float yCoordenada,
                 boolean zCoordenada, int xOffSet, int yOffSet) {
        float fator;

        if (from == "VIRTUAL2REAL") { // virtual2real
            fator = (float)limReal/(float)limVirt;

            this.x = (int) (fator*(xCoordenada+limVirt/2));
            this.y = (int) (fator*(-yCoordenada+limVirt/2));
            this.z = zCoordenada;

        } else { // from =="Java" (java2real)
            fator = (float)limReal/(float)limJava;

            this.x = (int) (fator*(xCoordenada-xOffSet));
            this.y = (int) (fator*(yCoordenada-yOffSet));
            this.z = zCoordenada;
        }
}
```

O processo é análogo para a conversão dos outros sistemas de coordenadas. A conversão entre esses sistemas são definidas pelas equações 4.1 e 4.2 que foram deduzidas a partir de cada sistema de coordenadas.

$$x_R = \frac{F_R}{F_J}(x_J - offset_x) \tag{4.1}$$

$$y_R = \frac{F_R}{F_J}(y_J - offset_y) \tag{4.2}$$

Para a equação 4.1, x_R é o valor da abscissa no sistema Real, F_R e F_J são valores de fundo de escala dos sistemas Real e Java, respectivamente, x_J é o valor da abscissa no sistema Java, e $offset_x$ é o valor da constante de *offset* da abscissa x do sistema Java. A descrição é análoga para a equação 4.2, sendo que nesta é equalizado a ordenada do sistema Real.

Quanto aos objetos das classes *Retangulo* e *Circulo*, estes são figuras que são criadas no painel de desenho com o uso do mouse, arrastando e soltando o mesmo para formar a figura no tamanho desejado.

A classe *FuncaoDaReta*, por sua vez, é utilizada para gerar os coeficientes linear e angular de cada reta desenhada no painel Java. Esses coeficientes servem para encontrar os pontos da reta que são utilizados na construção da mesma no AV.

E por fim, a classe *Cliente* prepara o ambiente de cliente de soquete para fazer conexão com o servidor de soquete no subsistema servidor. Para manter o cliente preparado para receber dados do subsistema servidor, é utilizada a interface *IOuvinteDoServidor*, criada para "ouvir"o servidor através da implementação do método *processarDadoDoServidor*(). É por meio dessa classe que os parâmetros de posições são recebidos do subsistema servidor, e os dados de controle são enviados ao mesmo.

Métodos Funcionais da Interface de Controle

A Interface de Controle, representada pela classe *GUILab* ilustrada na Figura 4.7, foi projetada para fornecer ao usuário uma interface interativa e intuitiva, além de ser capaz de incorporar funcionalidades adicionais. Essa interface deve permitir a programação do robô e envio ao *site*, bem como, estar preparada para atualizar em tempo real os objetos no AV de acordo com os parâmetros enviados do subsistema servidor.

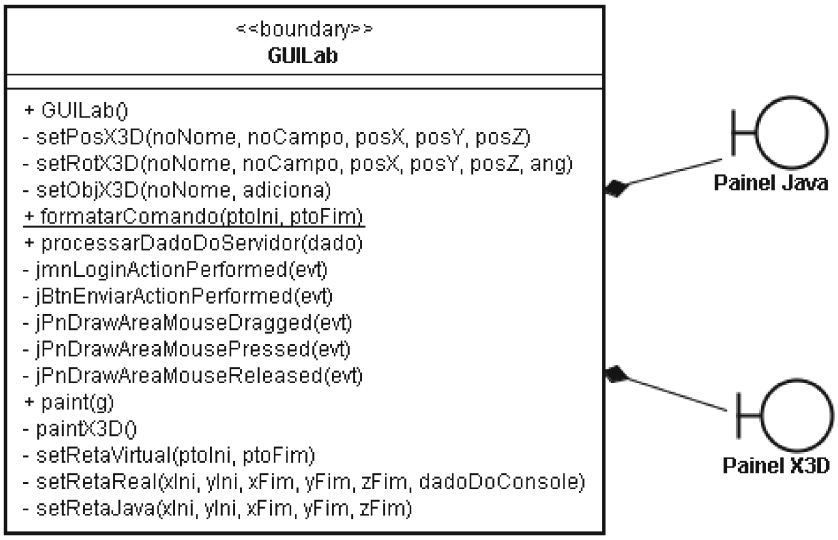

Figura 4.7: Classe GUILab com atributos e métodos.

Os métodos principais utilizados nesta classe são:

- *setPosX3D(noNome, noCampo, posX, posY, posZ)*: Método de acesso ao AV que é utilizado para alterar a posição (posX, posY, posZ) do robô ou objeto que tenha referência por nome *noNome*;

- *setRotX3D(noNome, noCampo, posX, posY, posZ, ang)*: Método de acesso ao AV que é utilizado para alterar o ângulo de rotação (posX, posY, posZ, ang) do objeto em relação ao eixo z que tenha referência por nome *noNome*;

- *setObjX3D(noNome, adiciona)*: Método de acesso ao AV que é utilizado para mudar o grau de transparência do objeto. Quando o grau de transparência for igual a 1 (um), significa dizer que o objeto está totalmente transparente, dando a sensação de que o mesmo foi removido do ambiente. Caso contrário, quando o grau de transparência for igual a 0 (zero), significa dizer que o objeto está incluso no ambiente;

- *formatarComando(ptoIni, ptoFim)*: Método que faz a formatação de comando para ser enviado ao subsistema servidor. Os comandos enviados pelo cliente são de dois tipos, a saber: comando de Login com as informações de Usuário e Senha; comando de Desenho com as informações dos pontos extremos do segmento de reta. Esses tipos de formatação seguem os *layouts* descritos no apêndice D;

- *processarDadoDoServidor(dado)*: Método usado para processar parâmetros vindos do subsistema servidor. Esse método utiliza os métodos de acesso ao AV ($setPosX3D$, $setRotX3D$, $setObjX3D$) para atualização do AV;

- *jmnLoginActionPerformed()*: Método usado para abrir o formulário de Login para fazer conexão com o servidor;

- *JBtnEnviarActionPerformed()*: Método usado para enviar os comandos de controle ao servidor do robô (subsistema servidor);

- *jPnDrawAreaMousePressed(MouseEvent)*: Método usado para obter o ponto âncora (ponto inicial) que é a posição do cursor no momento que o botão esquerdo do mouse é pressionado;

- *jPnDrawAreaMouseDragged(MouseEvent)*: Método usado para obter o ponto corrente que é a posição do cursor a cada momento de arrasto do mouse;

- *jPnDrawAreaMouseReleased(MouseEvent)*: Método usado para obter o ponto final que corresponde a posição que o botão esquerdo do mouse é solto. Esse método também é usado para desenhar a figura no painel de desenho e no painel do AV;

- *paint()*: Método que faz o desenho da figura no painel de desenho (painel Java);

- *paintX3D()*: Método que faz o desenho da figura no painel de AV;

- *setRetaVirtual()*: Método usado para gerar os pontos virtuais, atualizar a posição do robô no AV e desenhar a figura no AV;

- *setRetaJava()*: Método usado para criar os pontos inicial e final dos sistemas de coordenada Real e Virtual, a partir dos pontos inicial e final do sistema Java. Esses pontos são usados para desenhar a linha reta no painel de desenho. Esse método também faz uma chamada ao método *setRetaVirtual()* para desenhar o segmento de linha reta no AV;

- *setRetaReal()*: Método usado para criar os pontos inicial e final dos sistemas de coordenada Java e Virtual, a partir dos pontos inicial e final do sistema Real. Esses pontos são usados para desenhar a linha reta no painel de desenho. Esse método também faz uma chamada ao método *setRetaVirtual()* para desenhar o segmento de linha reta no AV.

O código completo dos métodos supracitados estão no fonte GuiLab.java referenciado no apêndice E.

Na subseção seguinte são apresentados os diagramas de seqüência dos casos de uso Programar Tarefa do Robô e Enviar Comando ao Robô que utilizam alguns desses métodos para trocar mensagens entre os objetos e, assim, contribuir para realização dos mesmos.

4.3.4 Diagrama de Seqüência Programar Tarefa do Robô

A realização do caso de uso Programar Tarefa do Robô é demonstrada pelo diagrama de seqüência Programar Tarefa do Robô, ilustrado na Figura 4.8.

Este diagrama descreve a seqüência de eventos para se realizar o processo de elaboração de tarefa do robô. O processo consiste no uso de métodos da classe $GUILab$ que trocam mensagens entre si para efetuar a tarefa do robô, que no contexto desse trabalho é a elaboração de desenhos. A seqüência de eventos se inicia quando o operador seleciona a figura de desenho e termina quando o sistema simula a mesma tarefa no AV.

Neste diagrama tem-se um ator e três objetos que interagem entre si. O ator é o operador do sistema, os três objetos são *guiLab*, *painelJava* e *painelAV*. O objeto *guiLab* é a IU de controle, e os objetos *painelJava* e *painelAV* compõem a IU para fornecer os paineis de desenho e de AV, respectivamente.

O processo de desenho se inicia quando o ator operador seleciona a figura que deseja desenhar no painel de desenho, através do método $selecionarFigura()$. A figura pode ser uma reta, retângulo ou círculo. O operador desenha a figura com a ajuda do mouse, onde se disparam os métodos $jPnDrawAreaMousePressed(MouseEvent)$ para obter o ponto inicial (ponto âncora), $jPnDrawAreaMouseReleased(MouseEvent)$ para obter o ponto final da figura, após saltar o botão esquerdo do mouse.

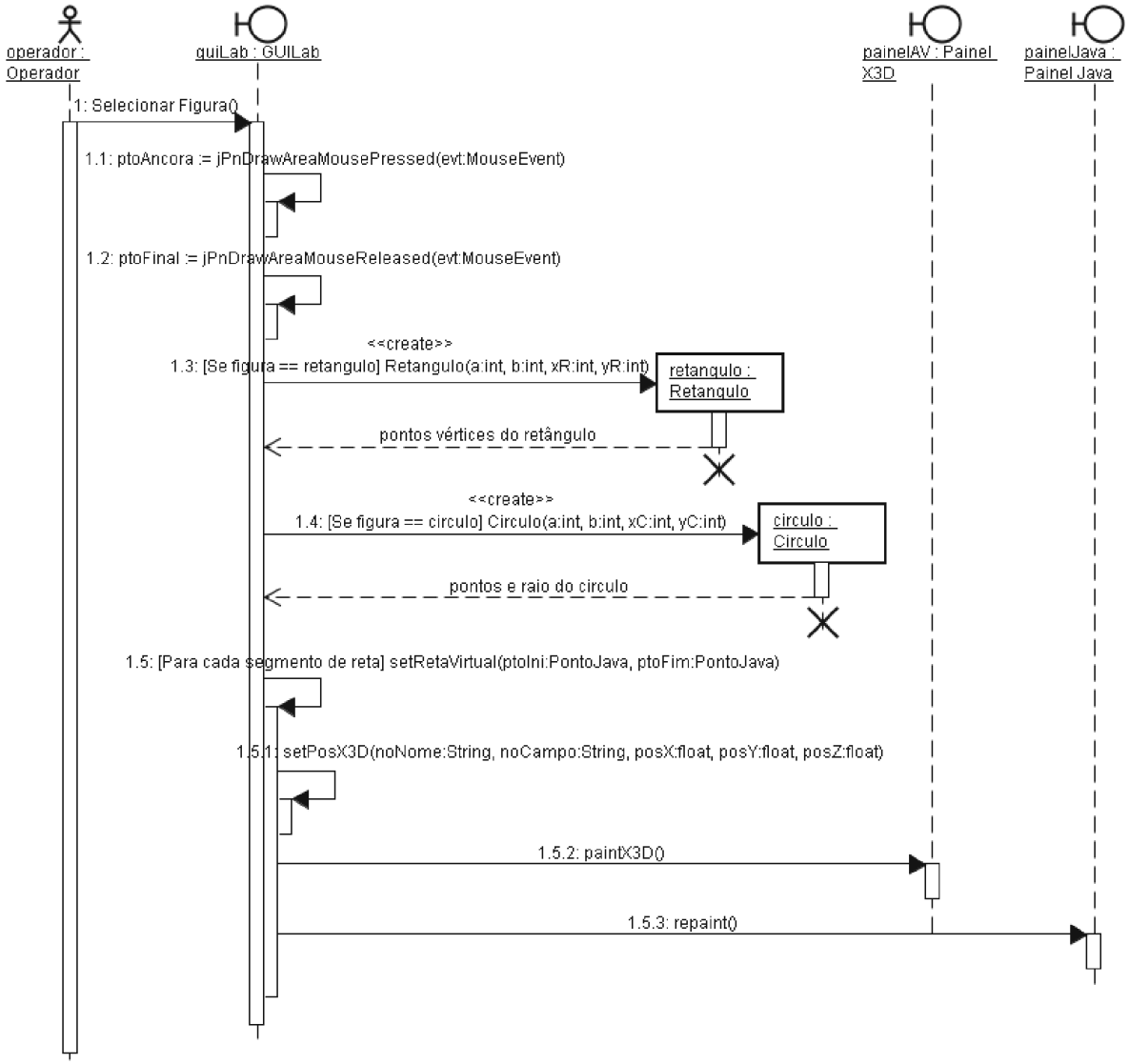

Figura 4.8: Diagrama de Seqüência Programar Tarefa do Robô.

Após isso, o sistema verifica qual figura foi selecionada, afim de obter os pontos para criar a mesma. Isso acontece utilizando o método construtor de cada figura. Como por exemplo, para se obter os pontos da figura retângulo usa-se o método construtor $retangulo(x, y)$ que é usado para fornecer os pontos de vértices da figura retângulo. Na verdade, para se gerar qualquer figura, é necessário se obter os pontos inicial e final dos segmentos de retas usadas na formação das figuras.

De posse desses pontos (inicial e final), a seqüência continua com a execução do método $setRetaVirtual()$ que é utilizado para gerar os pontos da reta no sistema virtual, e desenhar esses pontos no AV. Para que o robô virtual se movimente para formar o desenho no AV, o método $setPosX3D()$ é chamado passando como parâmetros os pontos da reta no sistema virtual. Os pontos da reta virtual são encontrados utilizando o construtor $FuncaoDaReta(ptoIni, ptoFim)$ da classe $FuncaoDaReta$ que é instanciada no método

em questão. Em seguida, o método $paintX3D()$ é executado para desenhar a figura no painel de AV.

E para finalizar, o método $repaint()$ é chamado para se desenhar a mesma figura no painel de desenho do Java.

4.3.5 Diagrama de Seqüência Enviar Comandos ao Robô

A realização do caso de uso Enviar Comandos ao Robô é demonstrada pelo diagrama de seqüência Enviar Comandos ao Robô, ilustrado na Figura 4.9.

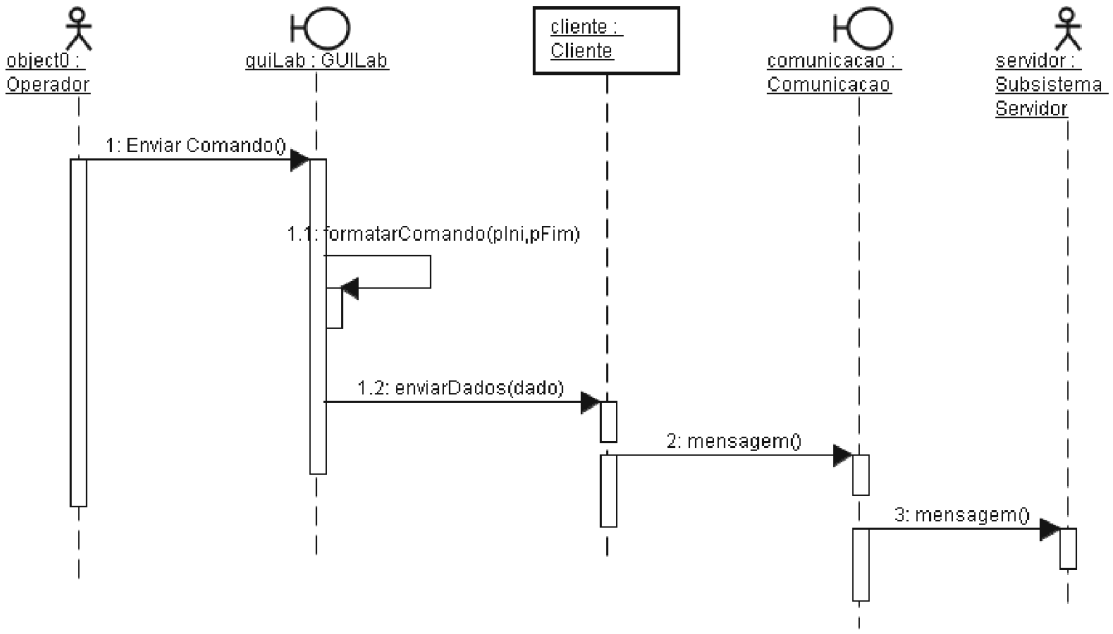

Figura 4.9: Diagrama de Seqüência Enviar Comando ao Robô.

Este diagrama descreve a seqüência dos eventos para se realizar o processo de envio do comando de controle ao subsistema servidor (robô remoto). O processo consiste no uso de métodos da classe *GUILab* que trocam mensagens entre si para efetuar o envio de comandos ao robô. A seqüência de eventos se inicia quando o operador seleciona o botão Enviar e termina quando o sistema envia o comando de controle ao subsistema servidor para a operação do robô.

Neste diagrama tem-se dois atores e três objetos que interagem entre si. Os dois atores são o operador que inicia o processo selecionando o botão Enviar e o subsistema servidor que finaliza o processo recebendo dados enviados pelo subsistema cliente. Os três objetos são o *guiLab*, o *cliente* e a *comunicacao*. O objeto *guiLab* é a IU de controle, o *cliente* é o objeto para fazer conexão com o servidor no subsistema servidor, e o objeto *comunicacao* representa a linha de comunicação da Internet.

O processo de envio se inicia quando o operador seleciona o botão Enviar que chama o método $JBtnEnviarActionPerformed()$ para formatar comandos e enviar dados para o subsistema servidor.

Os comandos são formatados com pontos do sistema real, convertidos a partir do sistema java. A formatação segue o *layout* de formatação de comando descrito no apêndice D.

Em seguida, o sistema chama o método $enviarDados(dado)$ da classe *Cliente* de soquete para enviar o dado de comando já formatado ao subsistema servidor, via mensagem de rede, representada pelo objeto *comunicacao*. Este objeto é apenas ilustrativo e serve para representar a linha de comunicação que separa os dois subsistemas.

4.3.6 Projeto da Interface de Usuário de Controle

A IU de Controle inclui duas partes. A primeira parte é composta de um painel de desenho e um painel de controle e a segunda parte é composta pelo *browser* X3D onde é carregado o AV, conforme ilustrado na Figura 4.10.

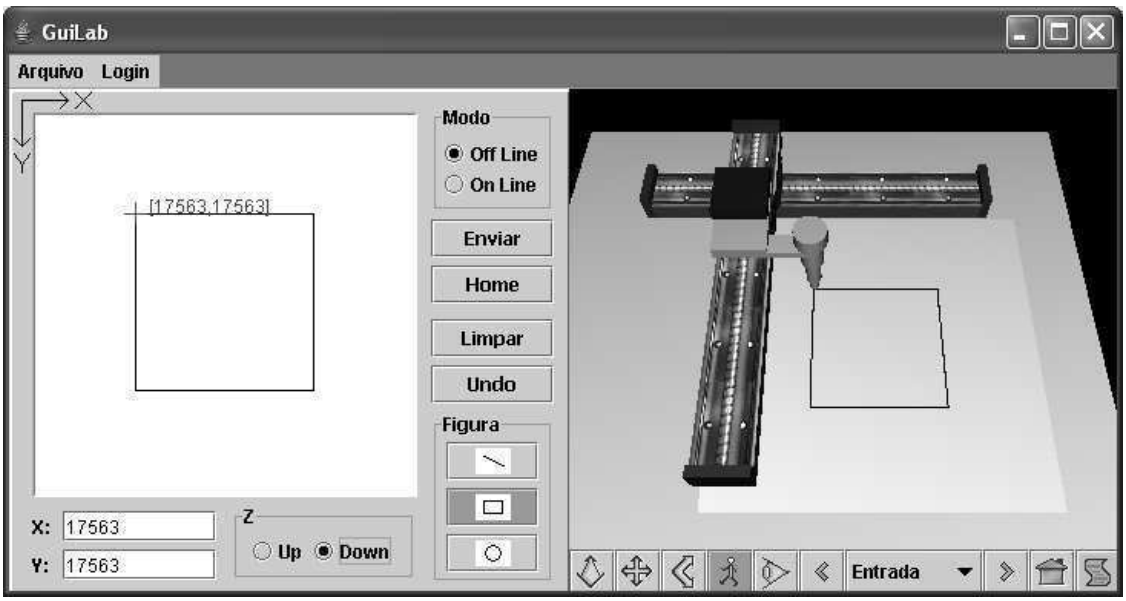

Figura 4.10: IU de Controle usada na programação do robô.

O painel de controle inclui os seguintes componentes para a interação com o operador:

- Dois *RatioButtons* para a escolha do modo de operação *off-line* e *on-line* (ao lado do painel de desenho);
- Um botão Enviar usado no modo de operação *off-line* (abaixo do modo de operação);
- Um botão Home para posicionar o robô na posição inicial (abaixo do botão Enviar);

- Um botão Limpar usado para apagar o desenho (abaixo do botão Home);
- Um botão Undo usado para desfazer a última figura (abaixo do botão Limpar);
- Três botões de figuras usados para selecionar a figura reta, retângulo e círculo (abaixo do botão Undo);
- Dois campos de edição X e Y usados para posicionamento pontual (abaixo do painel de desenho);
- Dois Ratio Buttons para a escolha do posicionamento do suporte da caneta (para Cima ou para Baixo no eixo Z).

A operação da tarefa pode ser realizada em ambiente virtual de forma *off-line* ou de forma *on-line*.

De forma *off-line*, o usuário simula a operação e depois envia ao robô (subsistema servidor). A simulação da operação não requer que usuário esteja conectado ao servidor, mas no envio da operação a conexão é necessária (efetuar *login*). Neste modo, o usuário opera o robô de forma indireta, obtendo resposta imediata do robô virtual, sem estar sujeito aos atrasos indesejáveis da Internet. Após o término da operação, o operador poderá salvar e/ou enviar ao subsistema servidor para execução do robô. Essa é uma característica da técnica de visualização antecipada utilizando a RV.

De forma *on-line*, o usuário executa a operação que estará sendo enviado ao robô. Na verdade, esse modo possibilita o envio de cada segmento de reta que compõe a figura desenhada. Isso acontece no momento em que o usuário solta o botão esquerdo do mouse. Esta opção requer conexão prévia à rede Internet (efetuar login).

Processo para Elaboração de Tarefas

A tarefa do robô consiste na elaboração de desenhos formados a partir de figuras de linhas retas, retângulos e círculos. Na verdade, a unidade básica de desenho no robô Plotter é o segmento de linha reta. Qualquer desenho é formado por um conjunto de segmentos de linhas retas, como por exemplo, o desenho de um retângulo que possui quatro segmentos de linhas retas AB, BC, CD e DA como ilustrado na Figura 4.11.

No painel de desenho, cada segmento de linha reta pode ser desenhada da seguinte forma: O usuário pressiona o botão esquerdo do mouse em qualquer área do painel de desenho (ponto inicial) e arrasta o mouse para a posição desejada (ponto final) onde deverá soltar o botão para se completar a ação. Após completar essa ação (para cada figura desenhada), o sistema faz a simulação da tarefa em AV, utilizando o robô virtual para desenhar a mesma figura no painel de AV.

Figura 4.11: Desenho de um retângulo composto de quatro linhas retas AB, BC, CD e DA.

A mesma analogia serve também para os desenhos das figuras retângulo e círculo, sendo que os pontos inicial e final são os vértices extremos para o retângulo, e os pontos de origem e o do raio para o círculo.

Na Figura 4.10 é ilustrado o desenho de um retângulo na IU de controle. Nota-se que o robô virtual efetua o mesmo desenho no AV, o que possibilita ao operador simular a operação antes da tarefa ser enviada ao robô real.

4.4 Módulo de AV de Resultado

O Módulo de AV de Resultado é a IU usada na visualização *on-line* da operação do telerobô. Na telerobótica, o retorno visual é importante, pois fornece ao operador o status corrente do robô, testificando a ocorrência da operação. A qualidade da visualização do resultado é de suma importância em um sistema telerobótico (BURDEA, 1999). A visualização deve permitir ao operador interagir com o robô de maneira clara, tendo a noção de sua localização e da localização dos objetos no ambiente físico.

Neste trabalho de pesquisa, é utilizada uma IU com AV para visualizar o resultado da operação, no lugar de imagens de vídeo. A substituição de imagens de vídeo pelo AV diminui o tempo de resposta do sistema devido a redução no volume de dados na rede. Essa diminuição do tempo de resposta pode ser constatada utilizando a equação de tempo de resposta 1.1. Esta equação afirma que, para minimizar o tempo de resposta, deve-se transmitir um volume mínimo de dados. Nesse contexto, o AV é utilizado neste trabalho com o intuito de acelerar a teleoperação, devido a transmissão de poucos dados na rede (dados de posicionamento), se comparado com o grande volume de dados de imagem de vídeo.

Além disso, o AV empregado nessa abordagem, oferece a vantagem de possibilitar a visualização do resultado em qualquer ângulo (sensação de navegação) no ambiente, oferecendo ao usuário a sensação de estar presente no local da operação (característica de imersão), o que não aconteceria utilizando apenas uma câmera de vídeo no local da

operação.

Nesta seção são abordados a descrição detalhada do caso de uso Visualizar Resultado em AV, bem como, o projeto do mesmo através de diagramas de seqüência e diagrama de classes do Módulo de AV de Resultado.

4.4.1 Especificação do Caso de Uso Visualizar Resultado em AV

Nesta subseção é apresentado o detalhamento do caso de uso Visualizar Resultado em AV. Este caso de uso descreve os passos necessários para se visualizar o resultado da operação do robô em AV. O AV é atualizado utilizando os parâmetros enviados pelo subsistema servidor. Na Tabela 4.3 é apresentada a descrição deste caso de uso.

Tabela 4.3: Especificação do Caso de Uso Visualizar Resultado em AV

Nome do Caso de Uso	Visualizar Resultado em AV
Ator Principal	Operador Humano
Pré-Condições	Usuário conectado esperando retorno dos parâmetros
Condição Final com Êxito	Visualização no AV em tempo real
Condição Final com Falha	Mensagem de erro
Ações do Ator	Ações do Sistema
	1. Receber parâmetros enviados do Subsistema Servidor
	2. Converter posições do robô e objetos para o sistema de coordenadas do AV
	3. Exibir em AV o resultado da operação em tempo real

4.4.2 Diagrama de Classes do Módulo de AV de Resultado

Na Figura 4.12 é ilustrado o diagrama de classes do Módulo de AV de Resultado. Este diagrama inclui a classe de fronteira (boundary) $GUILabR$, que é a IU onde o operador visualiza o resultado da operação do telerobô. Essa classe instancia as classes $PainelJava$ (painel de desenho) e $PainelX3D$ (painel de AV), por onde são visualizadas as tarefas do telerobô no painel de desenho e no painel de AV, respectivamente. A classe $GUILabR$ também instancia a classe $X3DScene$ para gerar o objeto de cena X3D e ter acesso aos métodos para manipulação do robô virtual.

Da mesma forma como acontece com a classe $GUILab$, as classes auxiliares $PontoJava$ e $PontoVirtual$ também são instanciadas pela classe $GUILabR$. Essas classes são instanciadas para gerar as posições do robô nos sistemas de coordenadas Java e Virtual, que por sua vez, são utilizadas no processo de desenho das figuras no painel de desenho e painel de AV.

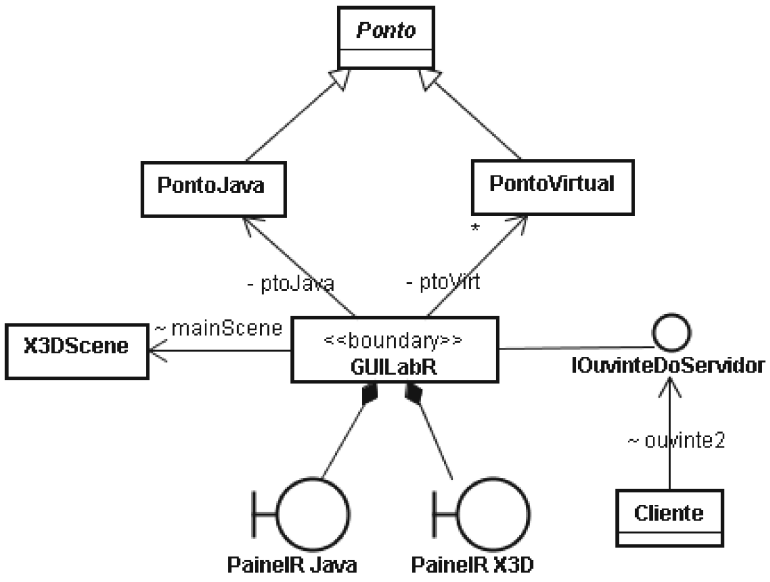

Figura 4.12: Diagrama de Classes do Módulo de AV de Resultado.

E por fim, a classe *Cliente* prepara o ambiente de cliente de soquete para fazer conexão com o servidor de soquete. É por meio dessa classe que os parâmetros de posições são recebidos do subsistema servidor, onde são usados para atualizar o AV de resultado.

Na subseção seguinte é apresentado o diagrama de seqüência do caso de uso Visualizar Resultado em AV que utiliza os seguintes métodos da classe $GUILabR$. Esses métodos já foram descritos acima, não necessitando reescrevê-los nessa subseção.

- $setPosX3D(noNome, noCampo, posX, posY, posZ)$;
- $setRotX3D(noNome, noCampo, posX, posY, posZ, ang)$;
- $setObjX3D(noNome, adiciona)$;
- $processarDadoDoServidor(dado)$;
- $paint()$;
- $paintX3D()$.

O código completo dos métodos supracitados estão no fonte GuiLabR.java referenciado no apêndice E.

4.4.3 Diagrama de Seqüência Visualizar Resultado em AV

A realização do caso de uso Visualizar Resultado em AV é demonstrada pelo diagrama de seqüência Visualizar Resultado em AV, ilustrado na Figura 4.13. Este diagrama descreve a seqüência dos eventos para se visualizar a operação da tarefa do telerobô em AV.

O processo consiste no uso de métodos da classe *GUILabR* que trocam mensagens entre si para efetuar esse processo. A seqüência de eventos se inicia quando o subsistema servidor envia os parâmetros de posições dos objetos (caixa e cilindro) e do robô ao subsistema cliente para atualizar o AV.

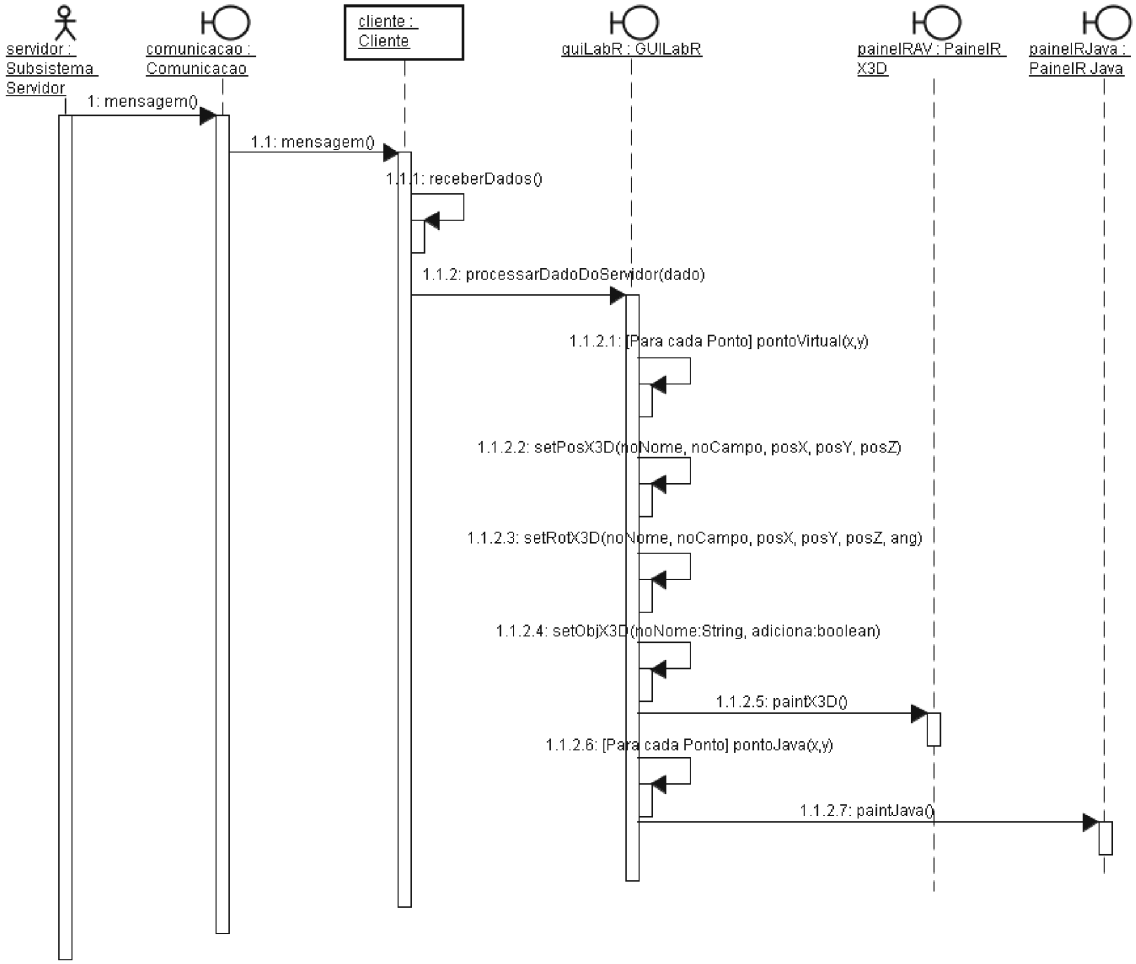

Figura 4.13: Diagrama de Seqüência Visualizar Resultado em AV.

Neste diagrama tem-se um ator e cinco objetos que interagem entre si. O ator é o subsistema servidor que inicia o processo enviando os parâmetros ao subsistema cliente. Os cinco objetos são *comunicao*, *cliente*, *guiLabR*, *painelRJava* e *painelRAV*. O objeto *comunicacao* representa a linha de comunicação que separa fisicamente os dois subsistemas, o *cliente* é o objeto usado para receber os dados do subsistema servidor, o objeto *guiLabR* é a IU de visualização em tempo real da operação do telerobô, o *painelRJava* é o objeto usado para visualizar o desenho em tempo real no painel de desenho, e o objeto *painelRAV*, por sua vez, é o painel de navegador X3D para visualizar o AV.

O processo de visualização em AV se inicia quando o subsistema servidor envia os parâmetros de posições dos objetos e do robô ao subsistema cliente. Como os subsistemas

cliente e servidor estão remotamente separados via linha de comunicação (que no caso é a Internet), utilizou-se o objeto *comunicacao* para ilustrar a passagem de mensagens de dados entre os subsistemas. Cada mensagem de dado corresponde a um parâmetro de posição. Os parâmetros são os pontos (x, y, z) do sistema real e sua formatação segue o *layout* de formatação de parâmetro descrito no apêndice D.

Assim que os parâmetros chegam ao subsistema cliente, estes são recebidos pelo objeto *cliente* através do método *receberDados*() e processados pelo objeto *guiLabR* através do método *processarDadoDoServidor*(*dado*) que é a implementação da interface Java *IOuvinteDoServidor*. No método *processarDadoDoServidor*(*dado*) são executados os métodos de acesso *setPosX3D*(), *setRotX3D*() e *setObjX3D*() para manipular o robô virtual e/os objetos virtuais, de acordo com o tipo de parâmetro.

Se existe parâmetro para movimentação do robô Plotter então o sistema chama os métodos *setPosX3D*() e *paintX3D*() para fazer o robô desenhar no AV.

Em seguida, os pontos no sistema real são convertidos para o sistema java através do construtor *PontoJava*(), onde são utilizados para desenhar no painel de desenho (objeto *painelRJava*()) através do método *repaint*().

4.4.4 Projeto da Interface de Usuário de Resultado

A IU de Resultado é idêntica a IU de Controle, com exceção, de que esta não possui painel de controle. Ou seja, ela inclui na primeira parte, um painel de desenho, e na segunda parte um painel de AV (browser X3D) onde é carregado o AV, conforme ilustrado na Figura 4.14. O painel de desenho, nesta IU, é utilizado para o operador visualizar o resultado de forma mais legível, oferecendo uma melhor resolução, se comparado ao painel de AV.

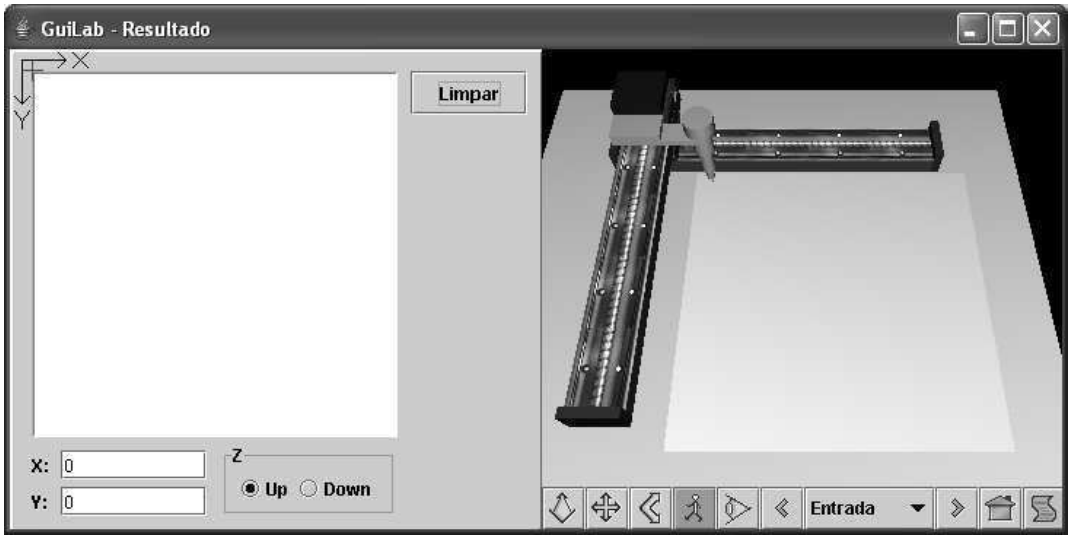

Figura 4.14: IU de Resultado usada na visualização da operação do robô.

O processo de atualização do AV (robô virtual e objetos caixa e cilindro) utiliza os parâmetros enviados pelo servidor de forma simples. A formatação dos parâmetros segue o *layout* de formatação de parâmetro descrito no apêndice D.

Na atualização do AV, os objetos com que o robô interage são inseridos ou removidos dependendo da dinâmica do sistema. O operador envia comandos de controle ao robô, e pequenos pacotes de dados contendo as posições correntes do robô e dos objetos são transmitidos ao módulo de AV de resultado. Estes são então remodelados em tempo real, fazendo com que os atrasos de tempo sejam suprimidos. Esta abordagem reduz consideravelmente o tráfego no sistema se comparado com transmissão de imagens de vídeo.

4.5 Módulo de Comunicação Cliente/Servidor

O Módulo de Comunicação Cliente/Servidor é a interface de comunicação entre os subsistemas cliente e servidor via Internet. Este módulo é responsável por fazer a conexão de soquete de fluxo com o subsistema servidor. De acordo com o esquema de software da Figura 2.3(b), os dados que trafegam entre os subsistemas são dados de parâmetro e de controle. Na comunicação entre os subsistemas os dados de controle trafegam no sentido cliente-servidor, ao passo que, os dados de parâmetros trafegam no sentido oposto.

As classes utilizadas nesse módulo são as mesmas apresentadas no diagrama de classes da Figura 4.6. A classe principal *GUILab* instancia a classe *Cliente* para ter acesso aos métodos para enviar e receber dados. Os passos utilizados para se estabelecer um cliente utilizando soquetes de fluxos são descritos com detalhes no apêndice C.

Abaixo são listados os métodos utilizados no intercâmbio de dados entre os subsistemas:

- *processarDadoDoServidor*(*Stringdado*): Método usado para receber dado de parâmetros do servidor;
- *servidor.enviarDados*(*Stringdado*): Método usado para enviar dado ao servidor.

Além disso, a classe *GUILab* é usada para instanciar a classe *Login* para apresentar o formulário de Login para o usuário ter acesso ao SCR do robô, e assim, poder controlar o mesmo.

4.6 Conclusão

Neste capítulo foram desenvolvidas as interfaces gráficas de usuário para controlar o robô remotamente e para visualizar a operação do mesmo. A estratégia de controle

adotada neste trabalho se baseia em especificar as tarefas do robô em modo *off-line* utilizando AV com o objetivo de melhorar a interação operador-sistema. Ou seja, o operador opera o robô virtual sem atrasos indesejáveis da Internet. Após a conclusão da tarefa, o operador poderá visualizar a operação do robô através da interface gráfica de usuário de resultado.

Capítulo 5

Resultados Experimentais

Após a implementação dos componentes do sistema, que são descritos nos capítulos 3 e 4, o sistema foi integrado e testado. Neste capítulo são apresentados alguns casos de teste escolhidos dentre os muitos realizados para demonstrar e validar a solução RP-PDI usada na integração do ambiente físico e AV, e a viabilidade do uso da RV em STIs.

Na seção 5.1 é apresentada a plataforma de testes utilizada para validar a solução RP-PDI. Na seção 5.2 é apresentado o caso de teste 1 - Extração de Parâmetros, utilizado para demonstrar a execução das etapas de PDI. Na seção 5.3 é abordado o caso de teste 2 - Atualização do AV, usado para verificar se o AV está sendo atualizado a partir dos parâmetros validados pela rede. Na seção 5.4, é discutido sobre o caso de teste 3 - Tratamento de Exceção, que é utilizado para demonstrar se a RP está tratando com as exceções de falha na identificação de objetos nas imagens e de detecção de colisão entre o robô e os objetos no ambiente. Na seção 5.5, é apresentado o caso de teste 4 - Teleoperação robótica, utilizado para demonstrar a operação robótica remotamente, afim de validar a viabilidade do uso da RV em STIs.

5.1 Ativação da Plataforma de Teste

Nesta seção, apresenta-se a plataforma de testes utilizada na representação do ambiente robótico, onde se pode validar a solução RP-PDI proposta neste trabalho. Esta plataforma inclui a infra-estrutura ilustrada na Figura 5.1, desenvolvida no Laboratório de Instrumentação Eletrônica da UFCG a partir do projeto CAD ilustrado da Figura 3.7.

Essa infra-estrutura contém as seguintes partes: duas *webcams* para capturar as imagens dos planos Ixy e Ixz do ambiente, um cilindro de cor verde (g) para representar o robô Plotter, um cilindro de cor azul (b) e uma caixa de cor vermelha (r) para servirem de obstáculos ao robô. O propósito da localização das câmeras é obter as posições dos baricentros no espaço (x, y, z) do robô e dos objetos da área de trabalho. Considera-se que

os objetos são de forma primitiva (caixa e cilindro) e que possuem dimensões conhecidas. Cada objeto é identificado por uma cor específica que é usada na etapa de segmentação das imagens.

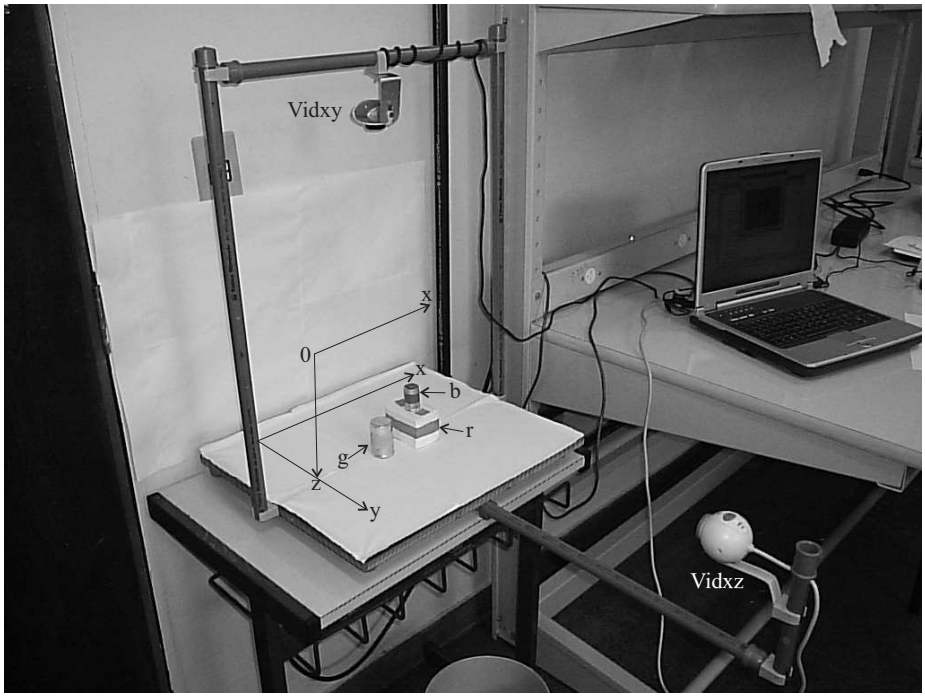

Figura 5.1: Infra-estrutura para simular o ambiente de teste.

Além disso, essa plataforma inclui os componentes de *software* necessários no processamento das imagens do ambiente e na comunicação de dados entre os subsistemas servidor/cliente, conforme ilustrado na Figura 2.3. Esses componentes incluem as ferramentas CPNTools para modelar a RPCH e o Matlab para implementar o programa de PDI, além das interfaces de comunicação, implementadas em Java, para integrar essas ferramentas com o servidor de controle e a interface do cliente através de soquetes de fluxos.

Para ativação da plataforma de testes é necessário seguir os seguintes passos:

1. Executar página de Conexão com o Matlab no CPNTools;
2. Executar *MatlabCPNComms.java* para preparar interface entre o Matlab e o CPNTools;
3. Executar *Mainlab.java* para preparar Servidor para receber conexão com o cliente;
4. Executar programa *Main.m* no Matlab;
5. Executar página de Controle no CPNTools;

6. Executar *GuiLab* (IU para fazer conexão com o cliente). Este passo é executado somente para os casos de teste 2 e 3.

Nas subseções a seguir são descritos os casos de teste usados para validar o funcionamento da RPCH, partindo do processo de extração de parâmetros, até o tratamento de exceções (falha de identificação e iminência de colisão).

5.2 Caso de Teste 1 - Extração de Parâmetros

O objetivo desse caso de teste é verificar a extração dos parâmetros *bg*, *br* e *bb* através do processamento da página de controle da RPCH. O teste deve verificar se as etapas de PDI são processadas, apresentando os resultados de cada etapa.

Conforme a página de controle, ilustrada na Figura 5.2, as transições **T0C**, **T2C** e **T3C** realizam as etapas de PDI para a imagem Ixy. As transições **T4C**, **T6C** e **T7C**, por sua vez, realizam as mesmas etapas para a imagem Ixz.

Neste caso de teste, é verificado a extração do parâmetro *br* (baricentro do objeto caixa), através do disparo de cada transição na ordem das etapas de PDI. As extrações dos outros parâmetros (*bg* e *bb*) são idênticas, não necessitando ser demonstradas.

As fichas nos lugares **Trigger Vidxy** e **Trigger Vidxz** representam a marcação inicial da RPCH. Esses lugares guardam fichas que representam a aquisição das imagens dos respectivos vídeos. Nessa marcação, as transições **T0C** e **T4C** são processos concorrentes e estão habilitadas para disparar.

Nas subseções seguintes são realizadas demonstrações dessas etapas com detalhes.

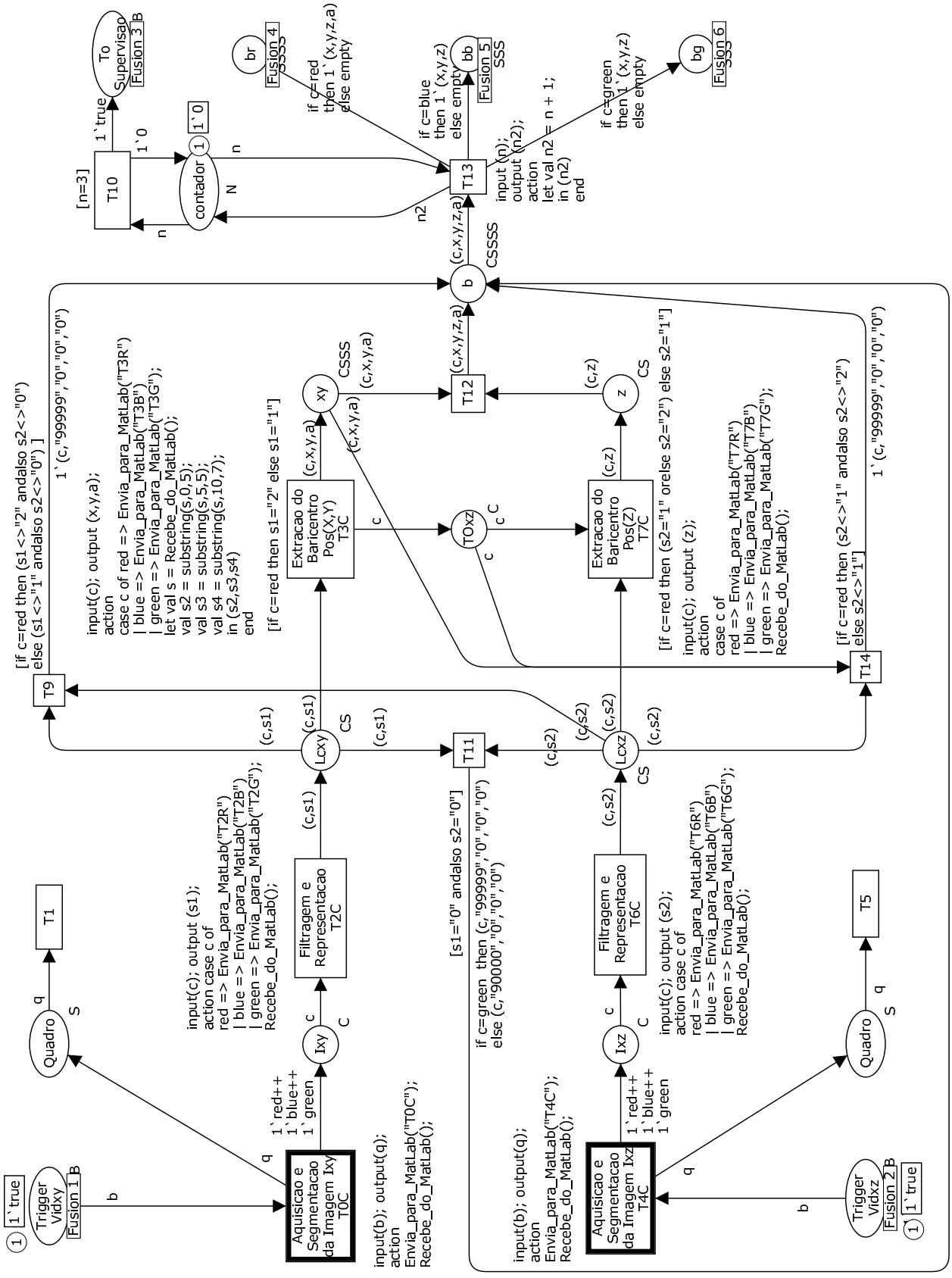

Figura 5.2: Página de Controle apresentando a marcação inicial.

5.2.1 Demonstração 1 - Etapas de Aquisição e Segmentação

As etapas de aquisição e segmentação das imagens, como o próprio nome diz, consistem na captura das imagens Ixy e Ixz do ambiente e no processo de extração de objetos das imagens conforme algoritmo de segmentação proposto na seção 3.3.

A demonstração destas etapas para o objeto r (caixa) acontece durante o disparo da transição **T0C** onde uma mensagem é enviada através da função $EnviaParaMatLab("T0C")$ ao Matlab para se fazer a aquisição e segmentação da imagem Ixy. Como resultado da execução da etapa de Aquisição é gerada a imagem Ixy ilustrada na Figura 5.3.

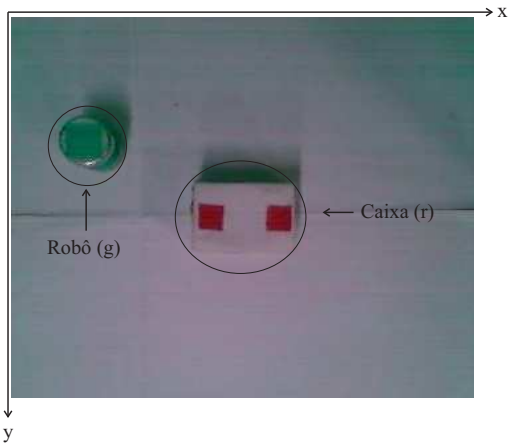

Figura 5.3: Imagem Ixy capturada no momento da aquisição.

A execução da etapa de segmentação desta imagem, gera as imagens $Irxy$, $Ibxy$ e $Igxy$ que representam as imagens segmentadas dos objetos caixa e cilindro, e do robô, respectivamente. Para esse caso de teste, a imagem $Irxy$ é ilustrada na Figura 5.4. Nota-se que, essa imagem ainda não passou pelo processo de filtragem, possuindo ruídos em torno do objeto r.

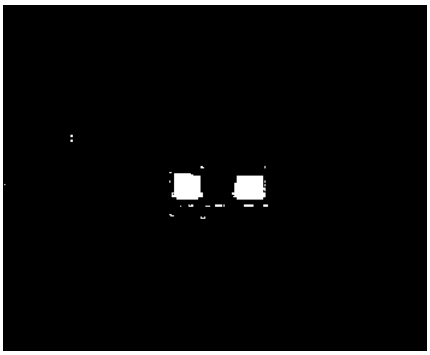

Figura 5.4: Imagem Irxy que representa o objeto caixa antes da filtragem.

Após o disparo dessa transição, são incluídas três fichas *1'red*, *1'blue* e *1'green* (que

correspondem as imagens segmentadas $Irxy$, $Ibxy$ e $Igxy$, respectivamente) no lugar **Ixy**. A Figura 5.5 ilustra a situação da rede após o disparo da transição **T0C**.

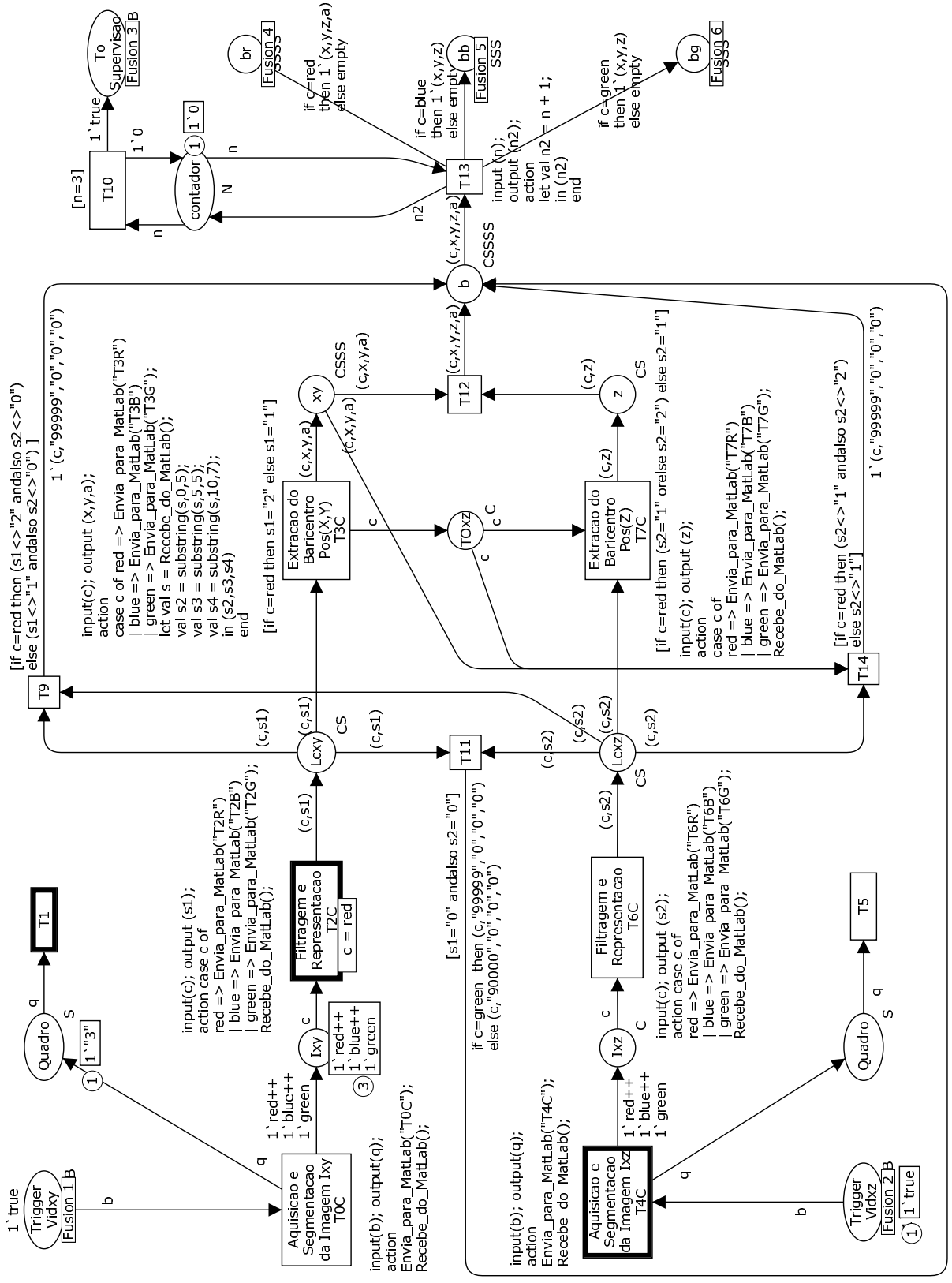

Figura 5.5: Marcação da rede após disparo de T0C.

Com essa marcação da rede, a transição **T2C** fica habilitada para disparar. Na subseção seguinte são demonstrados os resultados das etapas de Filtragem e Representação durante o disparo dessa transição.

5.2.2 Demonstração 2 - Etapas de Filtragem e Representação

As etapas de Filtragem e Representação consistem nas operações de remoção dos ruídos presentes nas imagens e na rotulagem dos elementos das imagens obtidos da etapa de Segmentação.

A demonstração destas etapas para o objeto r acontece durante o disparo da transição **T2C**, através da ficha *1'red*, onde se utiliza a função $EnviaParaMatlab("T2R")$ para solicitar ao Matlab que execute a filtragem da imagem Ixy e representação dos elementos do objeto em questão.

Na execução da etapa de Filtragem da imagem $Irxy$ é gerada a imagem $Lrxy$ ilustrada na Figura 5.6. Nota-se que a filtragem morfológica aplicada [1] removeu todos os ruídos da imagem, além de regularizar os contornos da mesma. Nesta imagem, são encontrados 2 (dois) elementos que representam o objeto caixa na imagem Ixy e que são usados para gerar o ângulo de rotação da mesma no plano x0y.

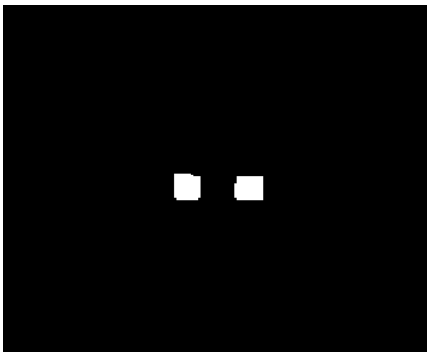

Figura 5.6: Imagem Irxy após filtragem.

Na execução da etapa de Representação são rotulados os dois elementos da imagem supracitados, a fim de identificá-los na etapa de Extração de Parâmetros. Essa etapa também cria uma estrutura de vetor para esses elementos, com o intuito de gerar os baricentros dos mesmos. Além disso, essa etapa fornece a quantidade dos elementos encontrados na imagem que é usado na RPCH para validar se a imagem foi filtrada de forma satisfatória.

[1] Filtragem detalhada no apêndice A

Como resultado do disparo dessa transição, o valor da quantidade de elementos da imagem *Lrxy* (que nesse caso é o valor "2") é enviado a rede e uma ficha *1'(red,"2")* é incluída no lugar **Lcxy**. Na rede esse valor é armazenado na variável $s1$ que é usada como parâmetro de condição para habilitar o disparo da transição **T3C**. A Figura 5.7 ilustra a situação da rede após o disparo da transição **T2C**.

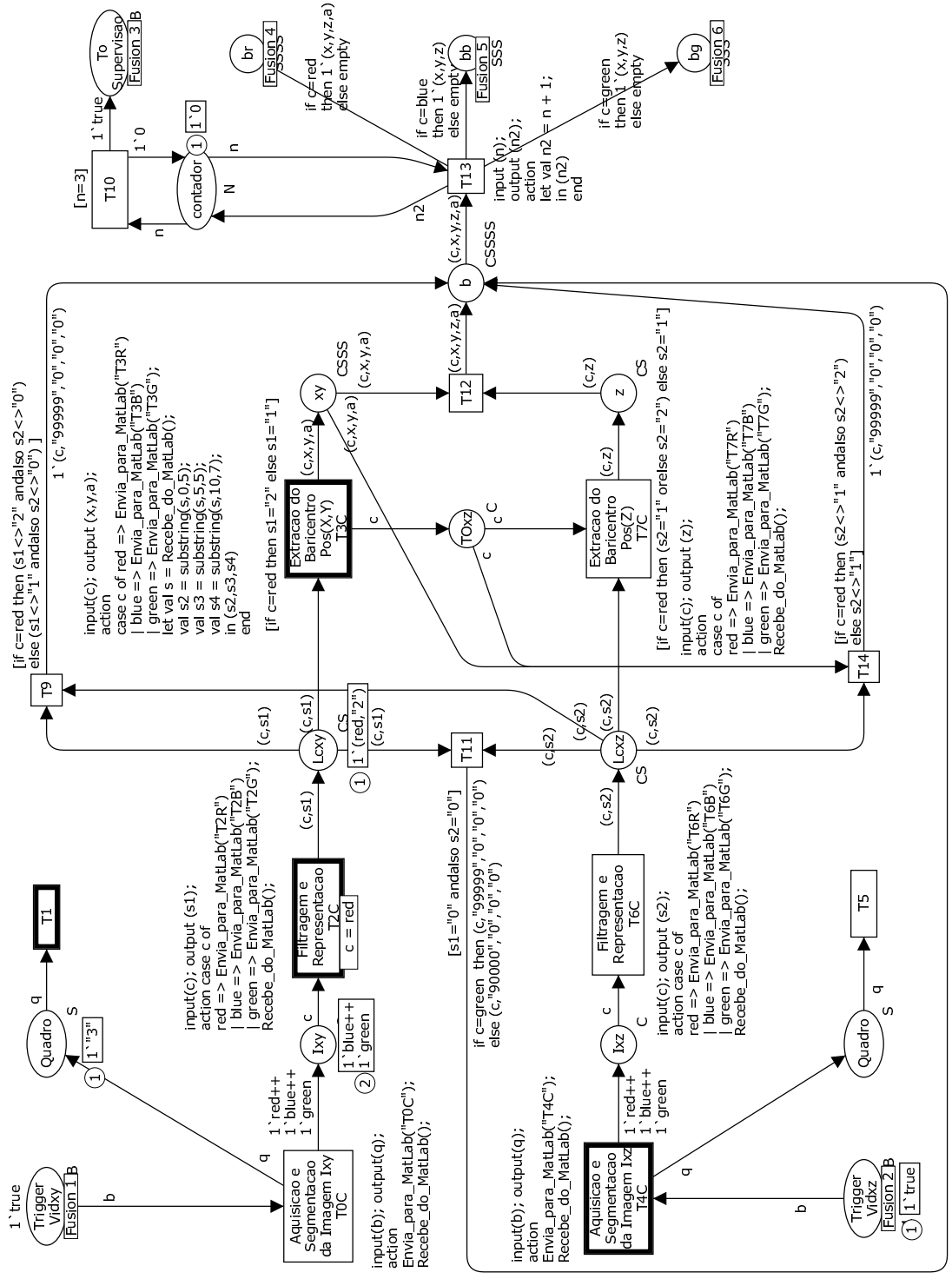

Figura 5.7: Marcação da rede após disparo de T2C.

Com essa marcação, a transição **T3C** (transição que corresponde a extração das coordenadas (x, y) do baricentro do objeto caixa) fica habilitada para disparar. Na subseção seguinte é demonstrado o resultado da etapa de Extração de Parâmetros durante o disparo dessa transição.

5.2.3 Demonstração 3 - Etapa de Extração de Parâmetros

A etapa de Extração de Parâmetros consiste em processar a extração dos baricentros dos elementos das imagens, identificados através da etapa de Representação.

A demonstração desta etapa para o objeto r acontece durante o disparo da transição **T3C** onde é enviado uma mensagem através da função $EnviaParaMatLab("T3R")$ ao Matlab para se fazer a extração do baricentro do objeto caixa a partir dos baricentros dos elementos da imagem $Lrxy$. Essa transição fica habilitada para disparar quando a variável $s1$ for igual a "2"(conforme condição descrita na inscrição de guarda dessa transição). Após o disparo dessa transição, uma ficha com os valores da cor red e coordenadas x,y e o componente angular a são adicionadas no lugar **XY**.

Para se completar este caso de teste, é necessário executar as etapas de PDI para a imagem Ixz, através do disparo das transições **T4C**, **T6C** e **T7C**. Estas transições são idênticas as transições **T0C**, **T2C** e **T3C**, não sendo necessário demonstrá-las nesse teste. Durante o disparo da transição **T7C**, que faz o Matlab executar a extração da posição z do baricentro, a ficha $1'red++Z$ é adicionada no lugar **Z**. Com uma ficha no lugar **XY** e uma ficha no lugar **Z**, a transição **T12** fica habilitada para disparar. Após o disparo dessa transição, tem-se o valor do parâmetro (baricentro) do objeto r, resultado do caso de teste em questão. As Figuras 5.8 e 5.9 ilustram as situações da rede antes e depois do disparo dessas transições, respectivamente.

De posse dos valores dos parâmetros (baricentros) obtidos neste caso de teste, é possível analisar o comportamento do sistema através da execução da página de supervisão. Os casos de testes seguintes são utilizados para demonstrar o funcionamento da página de supervisão na validação desses parâmetros.

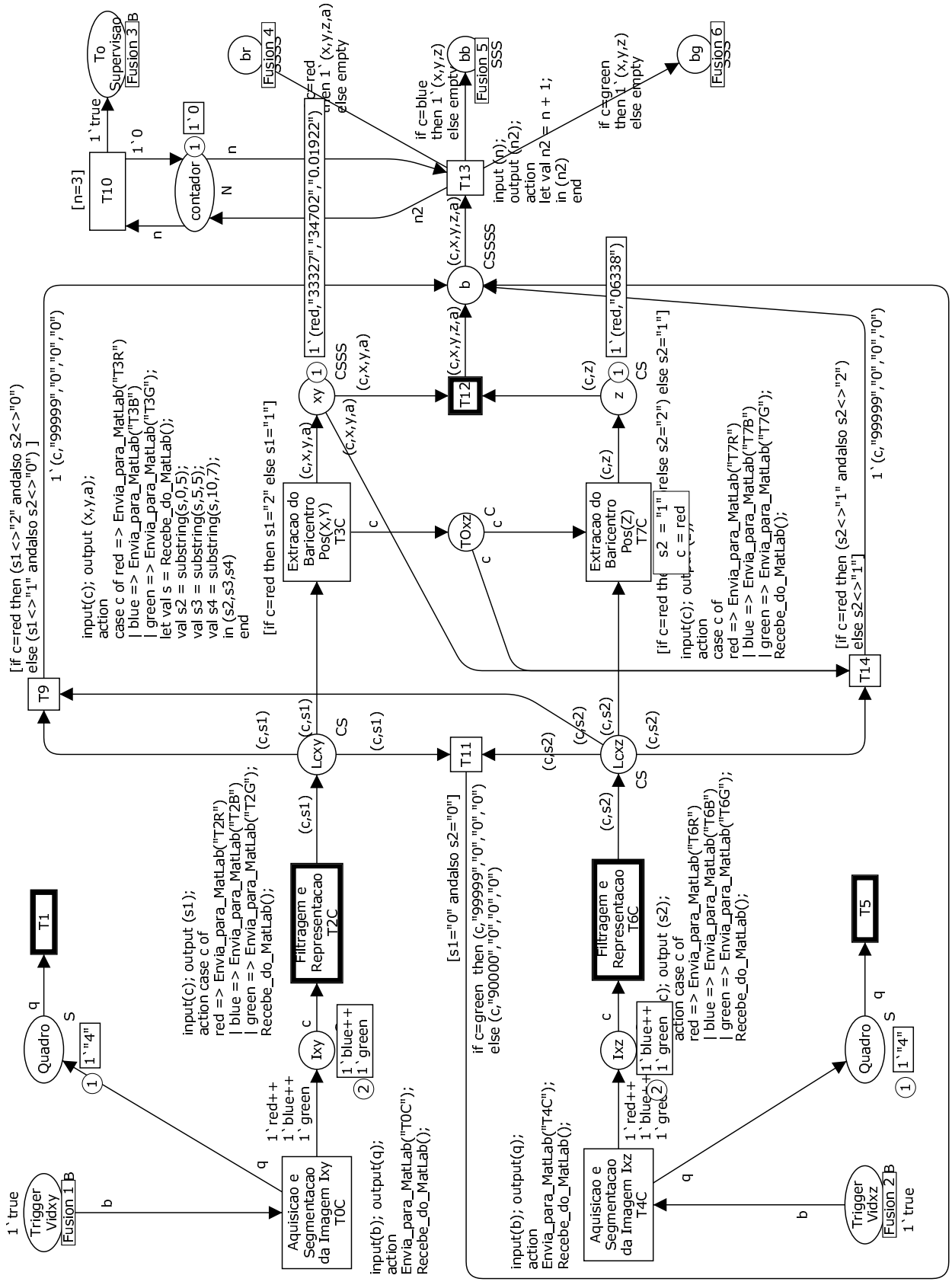

Figura 5.8: Marcação da rede antes do disparo de T12.

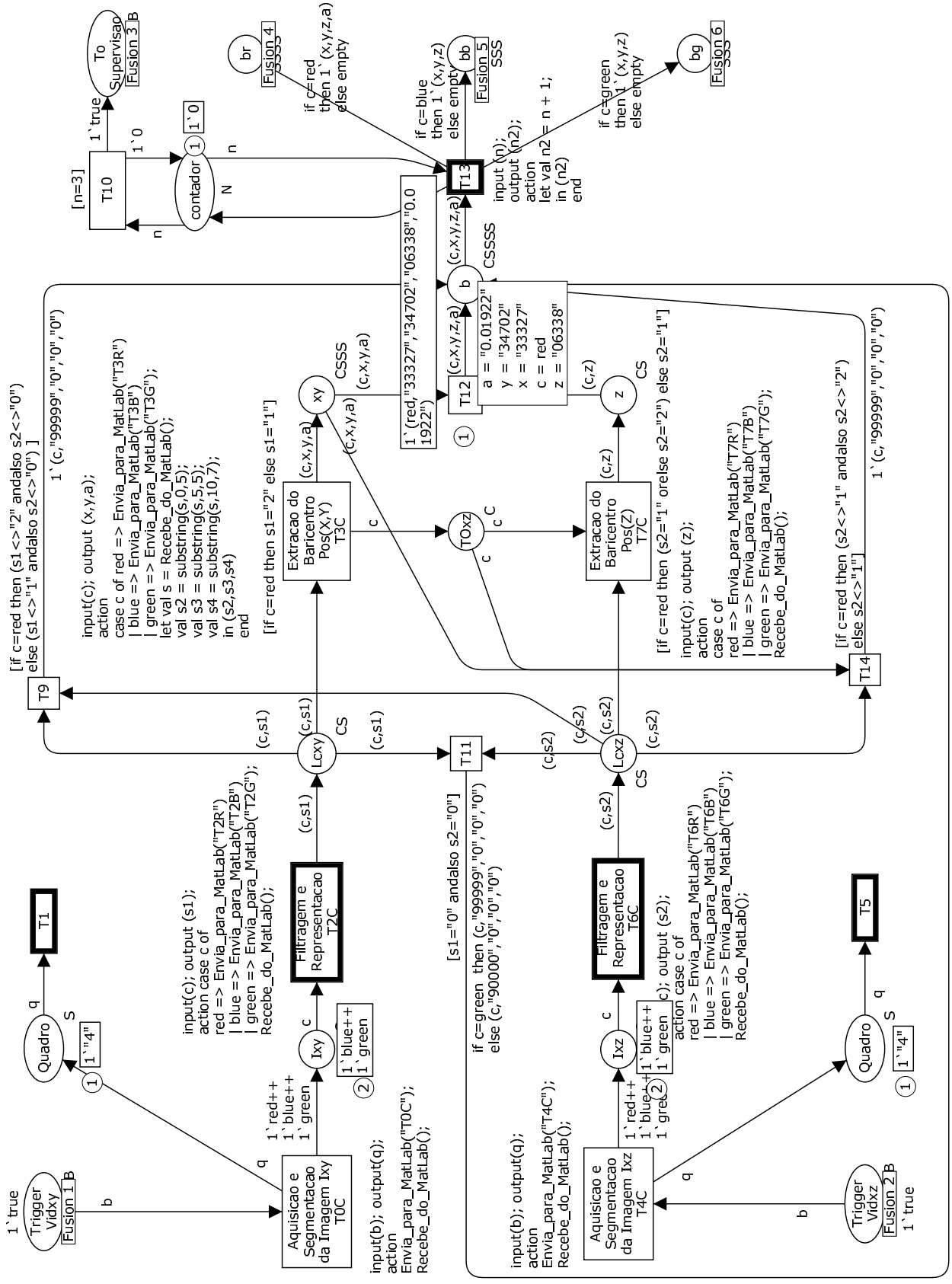

Figura 5.9: Marcação da rede após disparo de T12.

5.3 Caso de Teste 2 - Atualização do AV

O objetivo desse caso de teste é verificar se o AV está sendo atualizado toda vez que os parâmetros *br*, *bb* e *bg* (obtidos da página de controle) são processados através da página de supervisão. Na verdade, é nesta página que os parâmetros são avaliados e, dependendo de seus valores, estes são enviados ao cliente para atualização do AV. Para esse caso de teste, apresenta-se o objeto caixa (*br*) onde são efetuados os seguintes passos:

1. O objeto caixa é rotacionado no sentido do ponteiro de relógio em 90 graus;
2. O objeto caixa é removido do ambiente.

Esse caso de teste parte do princípio que o parâmetro *br* já foi extraído mediante o uso da página de controle no Caso de Teste 1 - Extração de Parâmetro, não necessitando ser demonstrado nesse caso de teste.

Na Figura 5.10 é ilustrada a imagem $Irxy$ do objeto caixa antes do teste ser realizado. Este objeto está localizado em paralelo ao eixo 0x. Ou seja, os elementos desse objeto (dois quadrados de cor vermelha) estão alinhados com o eixo 0x' que está em paralelo ao eixo 0x.

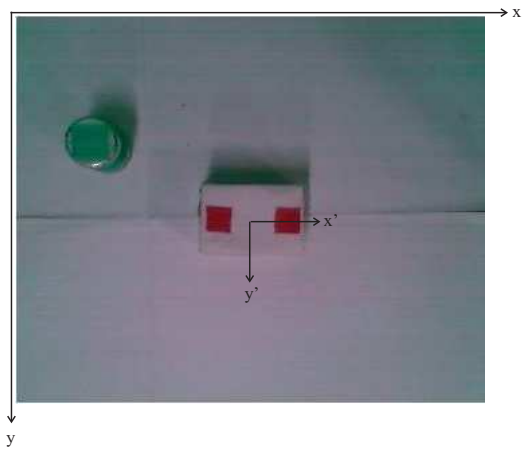

Figura 5.10: Imagem do objeto caixa (Irxy) antes do teste.

Nesse primeiro momento, para atualizar o AV com o ambiente real da Figura 5.10 é necessário se processar a página de supervisão da RP, ilustrada na Figura 5.11. Nesta página é tomada uma determinada ação após o parâmetro *br* ter sido extraído da imagem. Nesse caso de teste, a ação tomada é enviar o parâmetro *br* ao cliente para atualizar o AV, e isso acontece através do uso da função $EnviaParaOperador("R"xr^y r^z r^a)$ durante o disparo da transição **T17** (Atualizar *br* no AV). A formatação do parâmetro segue o *layout* de formatação de parâmetro descrito no apêndice D.

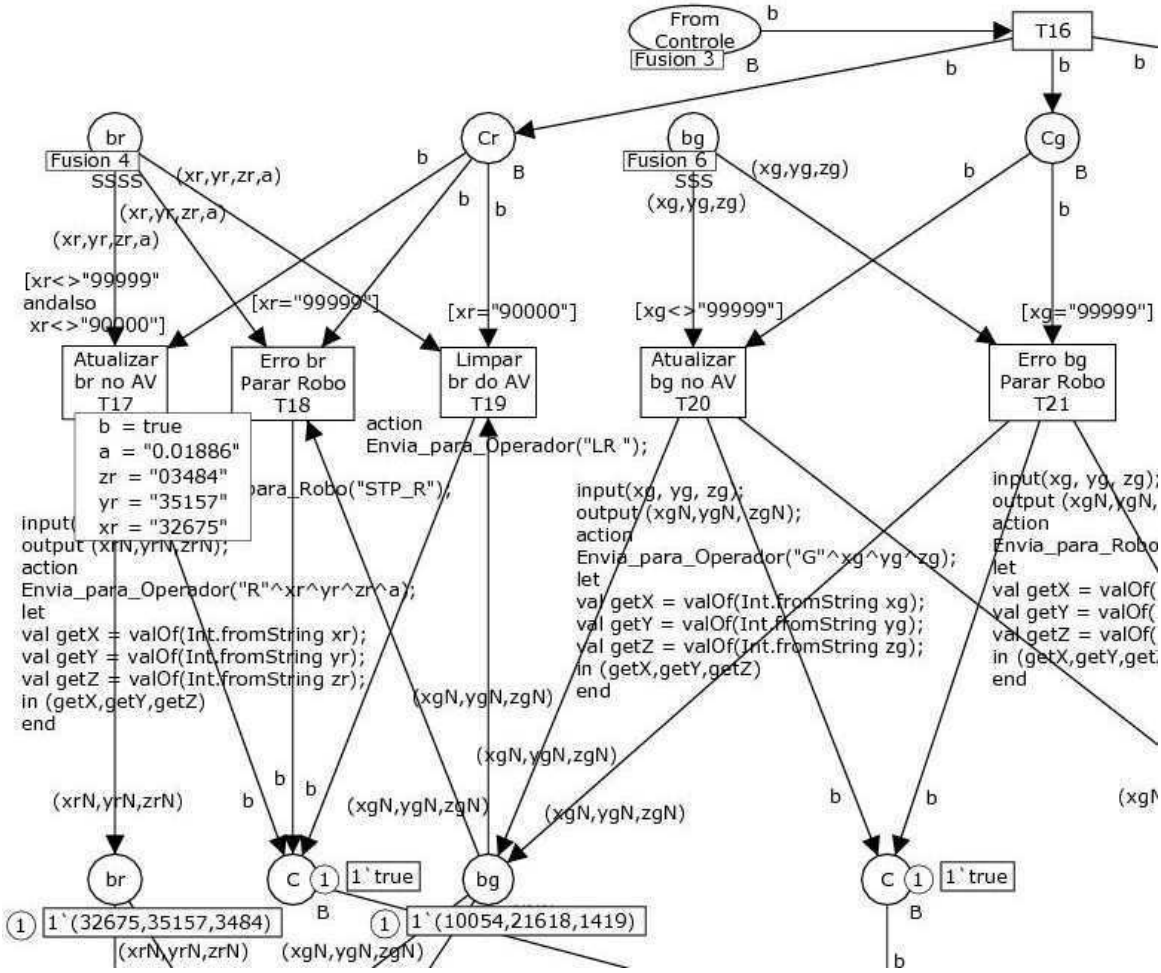

Figura 5.11: Marcação da rede durante o disparo da transição T17.

Ao final do disparo dessa transição, a título de validação, é apresentado os valores das coordenadas do parâmetro em uma caixa de diálogo logo abaixo da transição, conforme ilustrado na Figura 5.11. E por fim, o AV é atualizado no lado do cliente, conforme ilustrado na IU da Figura 5.12.

Para cada passo do teste são apresentadas as marcações da rede (na página de supervisão) durante o disparo das transições, afim de validar o parâmetro *br* para se tomar uma determinada ação. Os passos supracitados desse caso de teste são apresentados com detalhes nas subseções seguintes.

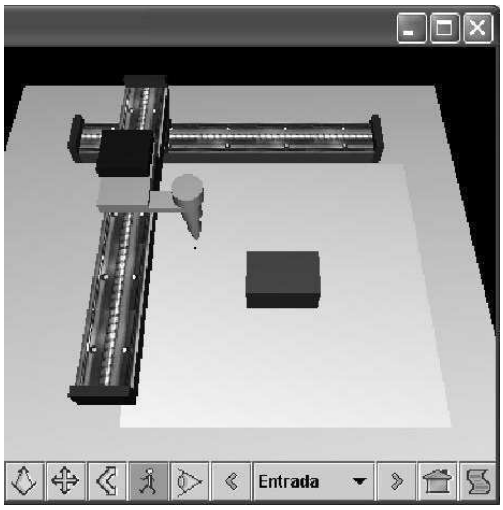

Figura 5.12: Imagem do objeto virtual caixa antes do teste.

5.3.1 Demonstração 1 - Objeto Caixa Rotacionado de 90 graus

O objetivo da primeira demonstração do caso de teste 2 - Atualização do AV, é verificar se o AV é atualizado quando o objeto caixa é rotacionado de 90 graus no sentido do ponteiro de relógio. De posse dos valores das coordenadas e do valor angular do parâmetro *br*, é possível se reconstruir o objeto virtual caixa com a mesma posição angular do seu correspondente no ambiente real.

A demonstração desse passo se inicia quando os lugares **br** e **Cr** da página de supervisão possuem fichas que habilitam o disparo da transição **T17** (Atualizar br no AV). Essa marcação da página de supervisão é ilustrada na Figura 5.13.

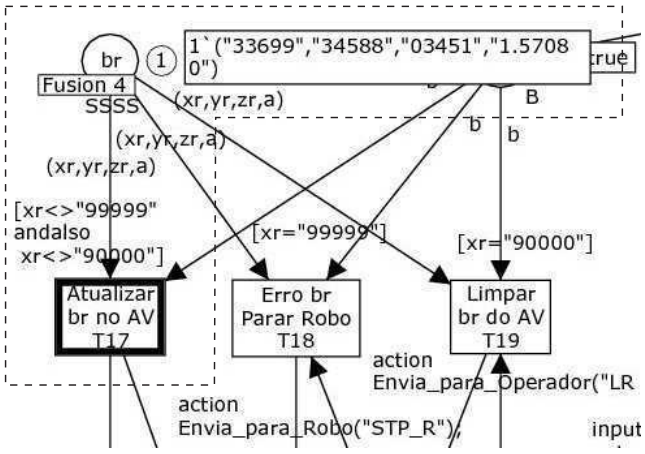

Figura 5.13: Marcação da rede antes do disparo da transição T17.

O lugar **br** contém a ficha do parâmetro *br* 1‘(”33699”,”34588”,”03451”,”1.57080”) que são os valores das coordenadas x, y, z e a do objeto caixa já rotacionado de 90 graus. Nota-se que a transição **T17** fica habilitada porque a condição da inscrição de guarda da

mesma (quando a variável xr é diferente de "99999"e "90000") é satisfeita. Na verdade, as três transições **T17**, **T18** e **T19** são mutualmente exclusivas (XOR), podendo ser habilitada apenas uma destas transições por vez.

O lugar **Cr**, por sua vez, bem como, os lugares **Cg** e **Cb** são usados para dar início ao processo de supervisão; e isso só acontece durante o disparo da transição **T16** da referida página. Na verdade, o início da supervisão só ocorre no momento em que as fichas dos parâmetros *br*, *bg* e *bb* estão nos seus respectivos lugares **br**, **bg** e **bb**, permitindo assim, habilitar o disparo da transição **T16** para incluir as fichas nos lugares **Cr**, **Cg** e **Cb**.

Durante o disparo da transição **T17** é enviado o parâmetro *br* através da função $EnviaParaOperador("R"^x r^y r^z r^a)$ ao Cliente para se fazer a atualização do AV (que significa a rotação de 90 graus do objeto virtual caixa).

Após o disparo dessa transição, é apresentado os valores das coordenadas do parâmetro em uma caixa de diálogo logo abaixo da transição, conforme ilustrado em destaque na Figura 5.14.

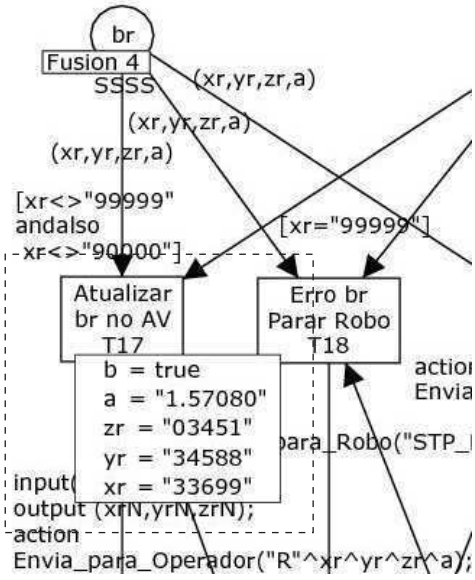

Figura 5.14: Marcação da rede após o disparo da transição T17.

Observa-se que a variável angular a [2] está com valor de 1,5708 radianos, o que vale a 90 graus de rotação do objeto. Dessa forma, conclui-se que o processo de extração desse parâmetro foi realizado de forma satisfatória.

Na Figura 5.15 é ilustrado o AV após a atualização da posição do objeto virtual caixa (rotacionado de 90 graus). Este objeto está agora localizado de forma perpendicular ao eixo 0x, em conformidade com o objeto real, ilustrado na Figura 5.16.

[2] Valor angular gerado pela média aritmética dos valores de baricentro dos elementos $c1(x1, y1)$ e $c2(x2, y2)$ encontrados durante o disparo da transição T3C (Extração do Baricentro Pos(X,Y)), conforme descrito na seção 3.3.

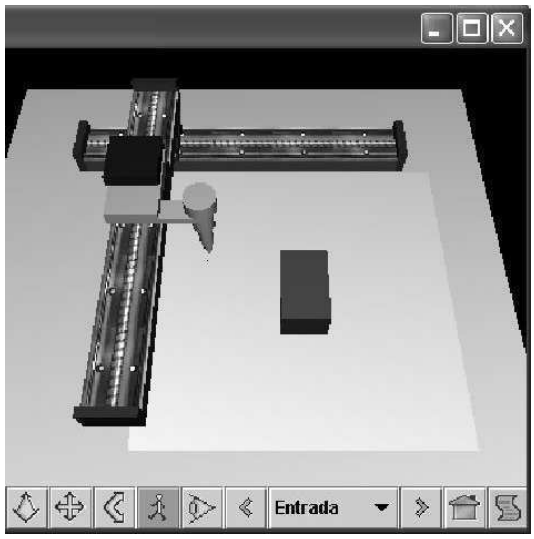

Figura 5.15: Imagem virtual do objeto caixa após demonstração do passo 1.

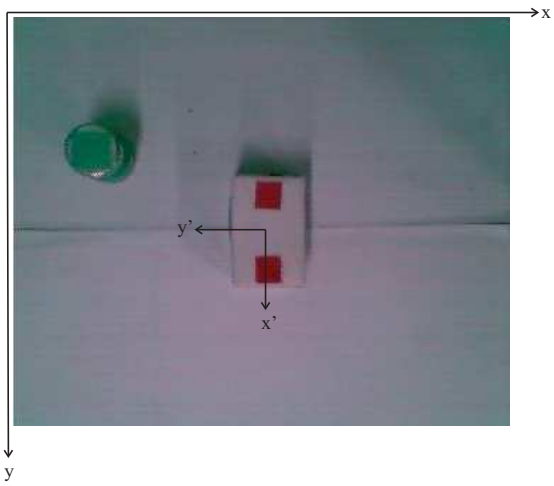

Figura 5.16: Imagem real do objeto caixa após demonstração do passo 1.

Para concluir esse caso de teste, na subseção seguinte é demonstrado a situação em que o objeto é removido do ambiente. Neste caso, o parâmetro *br* não possui valor de posição, e sim, valor *flag* para indicar a ausência do objeto no ambiente.

5.3.2 Demonstração 2 - Objeto Caixa Removido do Ambiente

O objetivo da segunda demonstração do caso de teste 2, é verificar se o AV é atualizado (sem o objeto virtual caixa) quando o objeto caixa é removido do ambiente real. Na verdade, de posse do valor *flag* "90000"atribuído a variável *br*, é possível se reconstruir o AV sem o objeto virtual caixa.

A demonstração desse passo se inicia quando os lugares **br** e **Cr** da página de supervisão possuem fichas que habilitam o disparo da transição **T19** (Limpar br do AV). Essa

marcação da página de supervisão é ilustrada na Figura 5.17.

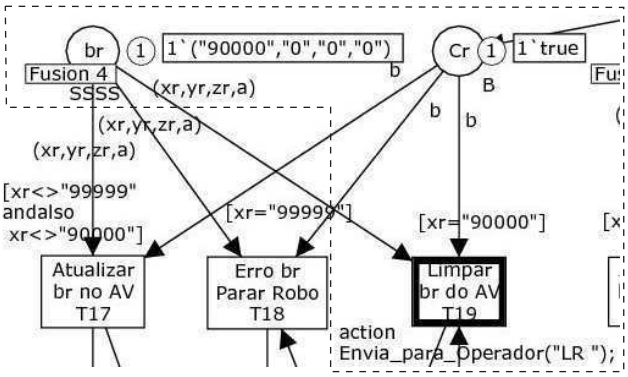

Figura 5.17: Marcação da rede antes do disparo da transição T19.

O lugar **br** contém a ficha do parâmetro *br* 1‘(”90000”,”0”,”0”,”0”) que é o valor *flag* usado para indicar que o objeto caixa não se encontra no ambiente real. Nota-se que a transição **T19** fica habilitada porque a condição da inscrição de guarda da mesma (quando a variável *xr* é igual a "90000") é satisfeita.

Durante o disparo da transição **T19** é enviado o parâmetro "LR"através da função *EnviaParaOperador*(”*LR*”) ao Cliente para remover o objeto virtual caixa do AV. A regra para formatação do parâmetro segue o *layout* de formatação de parâmetro descrito no apêndice D.

Na Figura 5.18 é ilustrado o AV após a remoção do objeto virtual caixa. Este AV está agora em conformidade com o ambiente real, ilustrado na Figura 5.19.

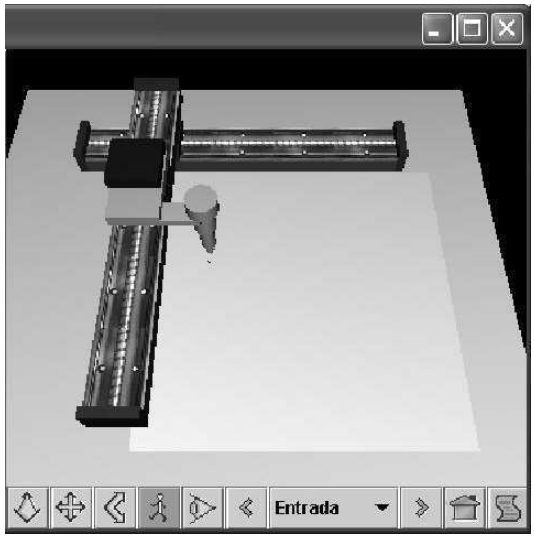

Figura 5.18: AV após demonstração do passo 2.

Figura 5.19: Imagem real após demonstração do passo 2.

5.4 Caso de Teste 3 - Tratamento de Exceção

O objetivo desse caso de teste é verificar se a página de supervisão está tratando os casos de exceção do sistema. Segundo a modelagem da RPCH, podem acontecer as seguintes exceções:

1. Falha de identificação de objeto nas imagens;

2. Eminência de colisão entre o robô e um dos objetos.

O tratamento dessas exceções consiste na atuação sobre o sistema baseado nos parâmetros *br*, *bb* e *bg* obtidos da página de controle. Na verdade, de acordo com o valor *flag* (x) desses parâmetros, o supervisor toma uma ação específica, conforme a Tabela 3.1 da seção 3.3 do capítulo 3.

O primeiro tipo de exceção acontece quando o valor *flag* x é igual a 99999, o que significa dizer que o robô ou o objeto (caixa ou cilindro) não foram identificados em pelo menos uma das imagens (Ixy ou Ixz) do ambiente operacional, e neste caso, o supervisor do sistema envia um comando ao SCR para parar o robô.

O segundo tipo de exceção, por sua vez, acontece quando um dos pares de parâmetros *bg* e *br* e/ou *bg* e *bb*, representando as posições dos baricentros do robô e caixa, e robô e cilindro, respectivamente, se coincidem, conforme a condição descrita no capítulo 3 deste trabalho de pesquisa. Quando isso ocorre, a transição **T31** (Parar Robô) da página de supervisão dispara, solicitando a parada imediata do robô.

Nas subseções seguintes são descritas as demonstrações dessas exceções com detalhes, com o intuito de validar o tratamento das mesmas pelo sistema.

5.4.1 Demonstração 1 - Falha de identificação de objetos

Para demonstrar como o sistema atua em casos de falha de identificação de objetos nas imagens, é utilizado o parâmetro *br* do objeto caixa. Conforme modelado na RPCH, o valor da variável *xr* desse parâmetro é usado para verificar se o referido objeto foi identificado nas imagens Ixy e Ixz. Quanto aos outros parâmetros *bg* e *bb*, as demonstrações são idênticas, não necessitando ser demonstradas.

A demonstração dessa exceção se inicia quando os lugares **Lcxy** e **Lcxz** da página de controle possuem fichas que habilitam o disparo da transição **T9**, conforme ilustrado na Figura 5.20. A transição **T9** torna-se habilitada pois a condição de guarda ($s1$ <> "2" e $s2$ <> "0") da mesma é satisfeita porque as variáveis $s1$ e $s2$ possuem valores "0"e "1", respectivamente. Esses valores informam a quantidade de elementos encontrados nas imagens $Lrxy$ e $Lrxz$ e são atribuídos durante o disparo das transições **T2C** (Filtragem e Representação) e **T6C** (Filtragem e Representação).

O lugar **Lcxy** contém a ficha 1‘(red,"0") que indica a não existência de elementos na imagem $Lrxy$ após a etapa de filtragem e representação; o que não é verdade. Ao passo que, o lugar **Lcxz**, por sua vez, contém a ficha 1‘(red,"1") que indica a existência de 1 (um) elemento na imagem $Lrxz$. Em condições normais, o lugar **Lcxy** contém a ficha 1‘(red,"2") indicando que foi encontrado dois elementos na imagem $Lrxy$, o que representa a identificação do objeto caixa.

Durante o disparo da transição **T9**, uma ficha com valor 1‘(c,"99999","0","0","0") é inserida no lugar **b**, informando que houve falha na identificação do parâmetro *br*. Essa marcação é ilustrada na Figura 5.21. Em seguida, durante o disparo da transição **T13**, a mesma ficha é incluída no lugar **br**, onde será utilizada na página de supervisão da rede.

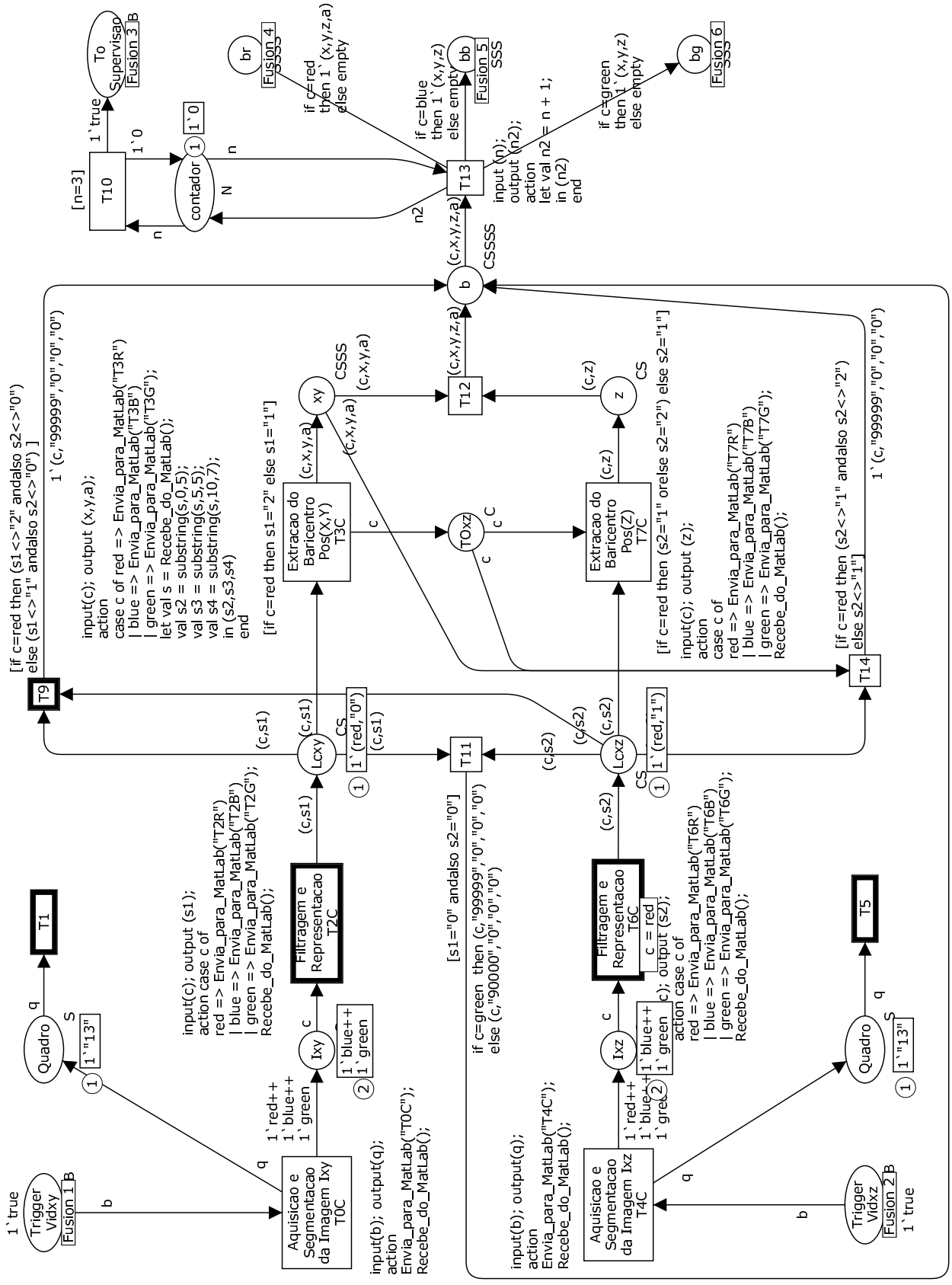

Figura 5.20: Marcação da rede antes do disparo da transição T9.

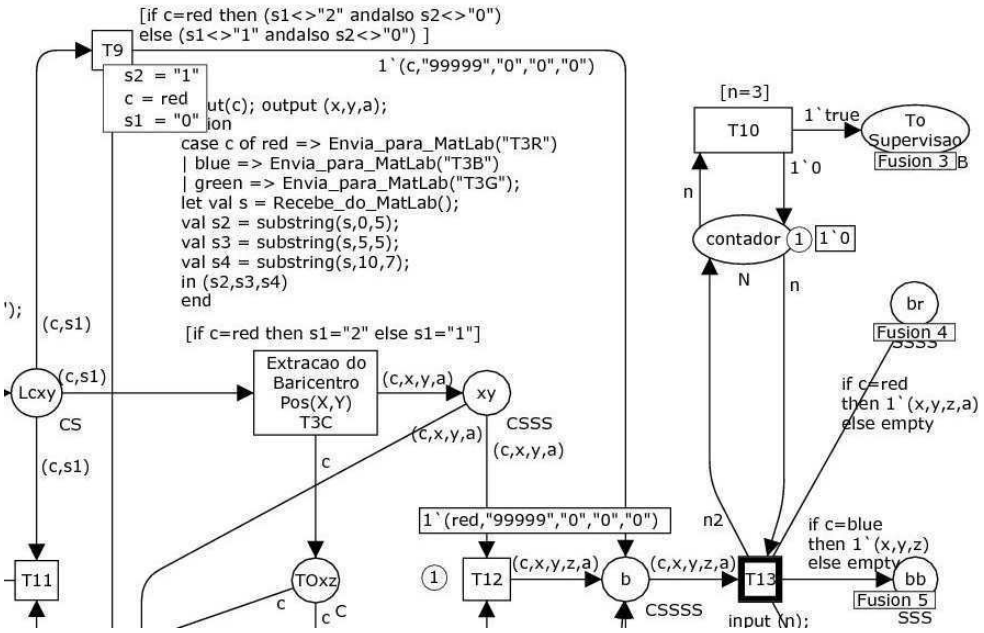

Figura 5.21: Marcação da rede após o disparo da transição T9.

Uma vez que os lugares **br** e **Cr** possuem fichas, estes habilitam o disparo da transição **T18** (Erro br Parar Robo). Essa marcação da página de supervisão é ilustrada na Figura 5.22.

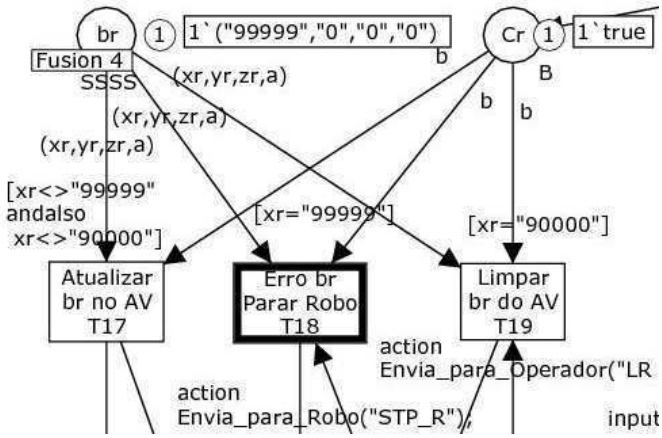

Figura 5.22: Marcação da rede antes do disparo da transição T18.

O lugar **br** contém a ficha 1‘(”99999”,”0”,”0”,”0”) que é o valor *flag* usado para indicar que houve falha na identificação do objeto caixa em pelo menos uma das imagens (Ixy ou Ixz) do ambiente operacional. Nota-se que a transição **T18** fica habilitada porque a condição da inscrição de guarda da mesma (quando a variável xr é igual a "99999") é satisfeita.

Durante o disparo da transição **T18** é enviado o parâmetro "STP"através da função *EnviaparaRobo*(”STP”) ao SCR para solicitar parada imediata do robô.

Para concluir esse caso de teste, na subseção seguinte é demonstrado a última exceção que trata da situação em que o robô está na iminência de se colidir com o objeto caixa.

5.4.2 Demonstração 2 - Eminência de colisão entre o robô e um dos objetos

Para demonstrar o segundo tipo de exceção que faz o tratamento de detecção de colisão, são utilizados os parâmetros de baricentro *bg* e *br*, para verificar a iminência de colisão entre o robô e o objeto caixa. O processo de verificação de colisão deste trabalho é simplório e se baseia na comparação desses baricentros, conforme descrito no capítulo 3 deste trabalho de pesquisa. A mesma analogia serve para o teste entre o robô e o cilindro.

Esse tipo exceção acontece quando o robô, ao movimentar-se em direção ao objeto caixa, entra na faixa limite de segurança *ls* (aproximadamente igual a 1,83cm), conforme ilustrado no gráfico da Figura 3.10. Essa situação pode ser vista nas Figuras das imagens de vista superior 5.23(a) e lateral 5.23(b) do robô (g) e do objeto caixa (r).

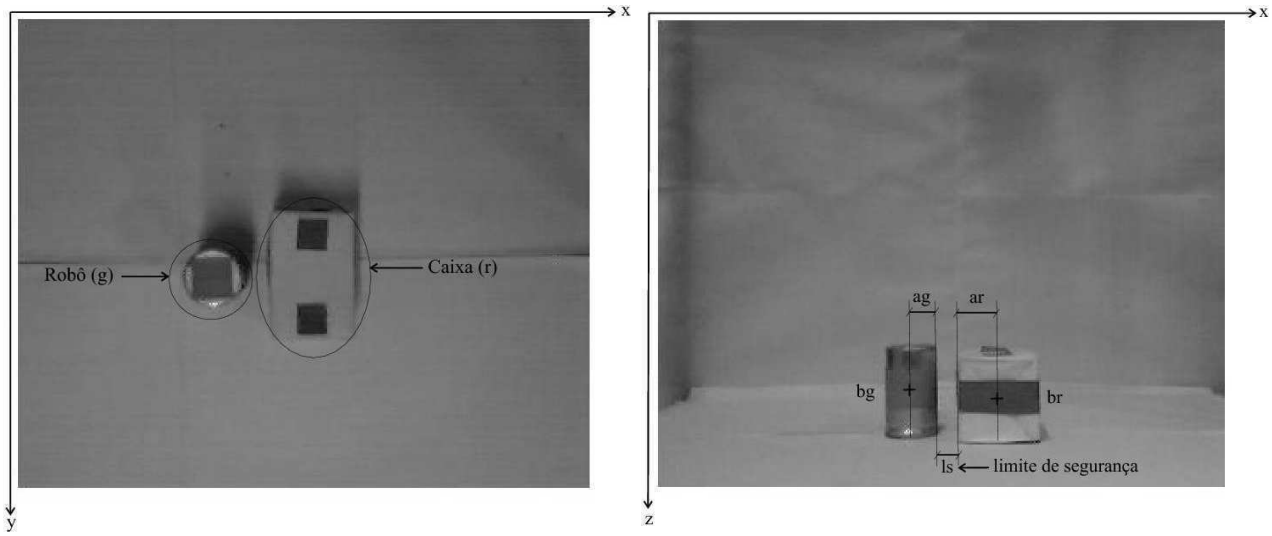

Figura 5.23: Imagens apresentando iminência de colisão. (a) Vista superior; (b) Vista lateral.

A demonstração dessa exceção se inicia quando os lugares **br**, **bg**, **Cr** e **Cg** da página de supervisão possuem fichas que habilitam os disparos das transições **T17** (Atualizar br no AV) e **T20** (Atualizar bg no AV), conforme ilustradas na Figura 5.24.

Nota-se que as transições **T17** e **T20** ficam habilitadas porque as condições das suas respectivas inscrições de guarda (quando as variáveis *xr* e *xg* são diferentes de "99999"e "90000") são satisfeitas.

Durante o disparo da transição **T17**, uma ficha contendo os valores das coordenadas do baricentro *br* 1‘(31185, 39708, 02298) é inserida no lugar **br**, para ser utilizada no processo

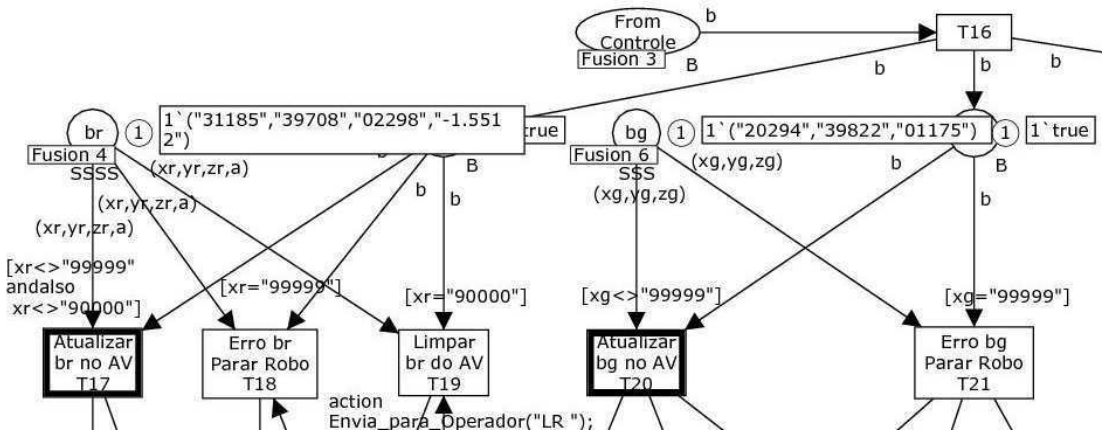

Figura 5.24: Marcação da rede antes dos disparos das transições T17 e T20.

de verificação de colisão. Do mesmo modo, acontece durante o disparo da transição **T20**, que inclui uma ficha $1`(20294, 39822, 01175)$ no lugar **bg**. Isso pode ser visualizado na marcação da rede na Figura 5.25.

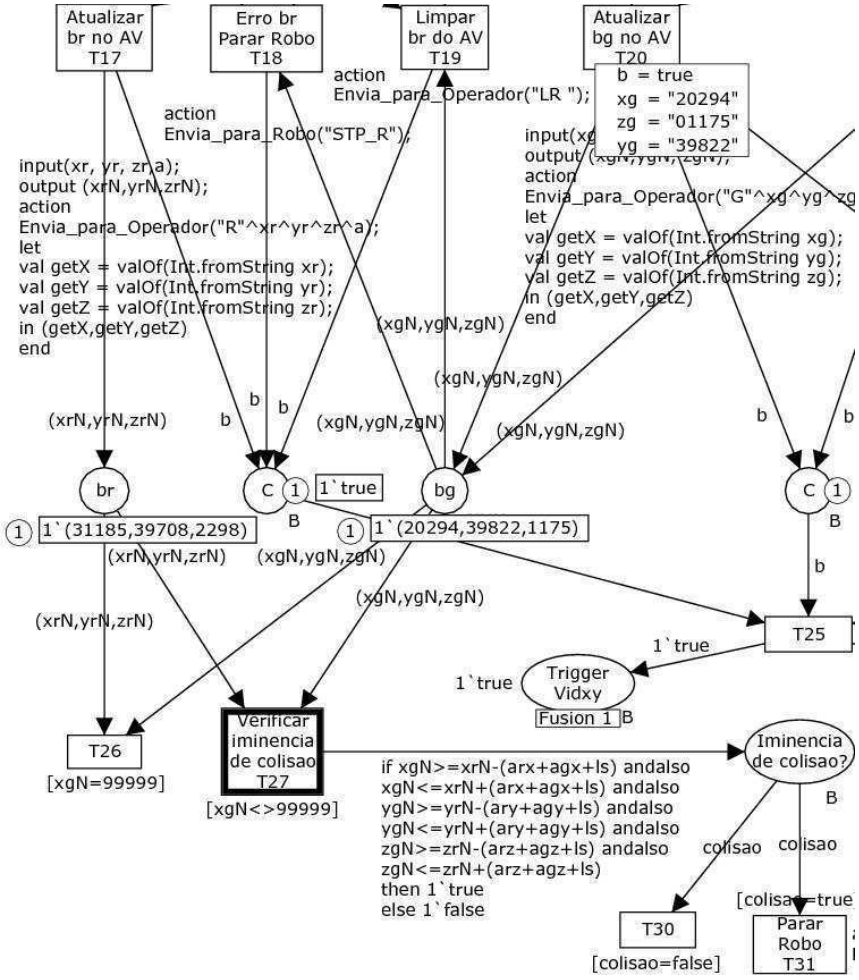

Figura 5.25: Marcação da rede após os disparos das transições T17 e T20.

Observa-se que, os valores dessas coordenadas são convertidos do tipo *string* para o

tipo *numérico* utilizando a função CPN ML [3] $valOf(Int.fromStringx)$ localizada nas inscrições de segmento de código de cada transição. Essa conversão é necessário, pois na fase posterior, esses valores são comparados afim de verificar a iminência de colisão entre o robô e o objeto caixa.

Uma vez que os lugares **br** e **bg** possuem fichas, estes habilitam o disparo da transição **T27** (Verificar iminência de colisão). Nota-se que a transição **T27** fica habilitada porque a condição da sua inscrição de guarda (quando a variável xgN é diferente de "99999") é satisfeita.

Durante o disparo da transição **T27** é verificado a condição descrita na inscrição do arco de entrada do lugar **Iminência de colisão?**. Essa condição tem o propósito de verificar se cada coordenada do robô está dentro do intervalo de segurança $[(r-(ar+ag+ls), r+(ar+ag+ls)]$. De acordo com os valores das coordenadas do parâmetro *br* $(xrN, yrN, zrN) = (31185, 39708, 02298)$ e do parâmetro *bg* $(xgN, ygN, zgN) = (20294, 39822, 01175)$, constata-se que a condição descrita na inscrição do arco de entrada do lugar **Iminência de colisão?** é satisfeita, gerando uma ficha 1‘*true*, que é incluída no referido lugar. Isso pode ser visualizado na marcação da rede na Figura 5.26.

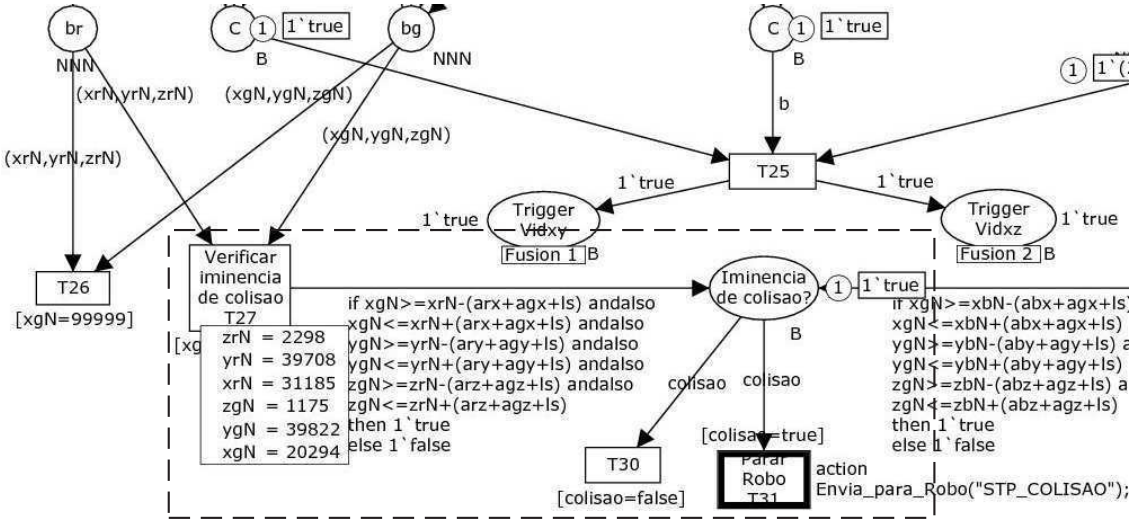

Figura 5.26: Marcação da rede após o disparo da transição T27.

Na marcação da rede, o lugar **Iminência de colisão?** contém a ficha 1‘*true* que é o valor lógico usado para habilitar a transição **T31** (Parar Robô). Essa transição fica habilitada porque a condição da inscrição de guarda da mesma (quando a variável *colisao* é igual a true) é satisfeita.

Durante o disparo da transição **T31** é enviado o parâmetro "STP"através da função $EnviaparaRobo("STP")$ ao SCR para solicitar parada imediata do robô.

Caso a condição descrita no arco não for satisfeita, é incluída uma ficha 1‘$false$ ao

[3] Linguagem para declarações e inscrições da rede

lugar **Iminência de colisão?** que, por sua vez, habilita o disparo da transição **T30**, que é usada para consumir a ficha sem efetuar tarefa alguma.

5.5 Caso de Teste 4 - Teleoperação Robótica

O objetivo desse caso de teste é demonstrar o funcionamento do STI para verificar se o uso da RV em STIs propicia a redução do tempo de resposta, apesar da taxa de transferência da Internet ser imprevisível.

Na Figura 5.27 é ilustrado a interação entre o subsistema cliente (interface de operação) e o subsistema servidor (robô Plotter) que são conectados via Internet. No subsistema cliente, o operador efetua a tarefa do robô em modo *off-line* usando o AV e envia a tarefa ao robô remoto para sua operação. A operação do robô é capturada em imagens de vídeo, que são processadas pela RP para obter as posições do robô e dos objetos no ambiente físico. Após isto, o subsistema servidor retorna os dados de posições ao subsistema cliente para serem atualizados no AV da interface de resultado.

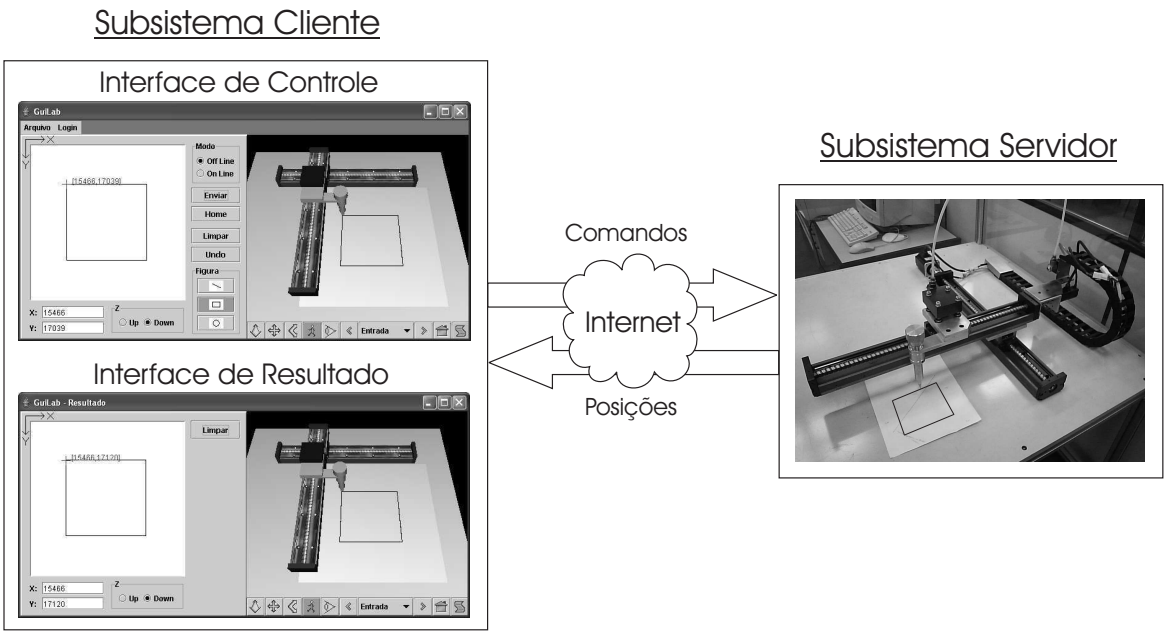

Figura 5.27: Diagrama de interação entre os subsistemas cliente e servidor.

Aqui é demonstrado a teleoperação robótica do STI desenvolvido neste trabalho, entre as cidades de Campina Grande (PB) e Manaus (AM) (distância de 2.705Km (GOOGLE, 2007)). O robô utilizado nessa demonstração é um Plotter de 3 graus de liberdade, conforme ilustrado na Figura 3.3. Este robô Plotter é controlado mediante o sistema de controle (SCR) desenvolvido no departamento de automação da Universidade Federal do Amazonas (UFAM) por Maisenbacher (2005). Este sistema foi projetado com a intenção de fornecer um laboratório remoto baseado na Internet.

A demonstração do presente caso de teste inicia quando o operador entra com as informações de usuário e senha no formulário de Login, ilustrado na Figura 5.28. Antes de submeter o *login*, os dados de usuário e senha são concatenados segundo o *layout* de formatação de comando descrito no apêndice D. Quando o operador clica no botão *Login*, o comando de *login* é enviado ao servidor para se efetuar a conexão com o mesmo. Após ter efetuado a conexão, o servidor manda o robô ir a posição de origem (posição HOME) para iniciar operação, e envia uma mensagem ao subsistema cliente (operador) para indicar que o robô está pronto para ser operado.

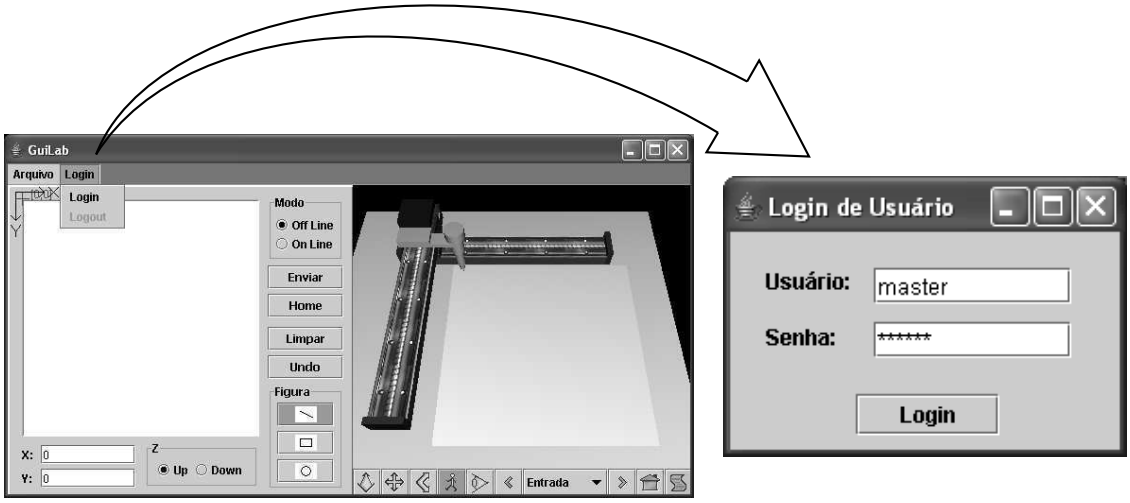

Figura 5.28: Interface de usuário para realização de Login.

Como se trata de um robô Plotter, a tarefa consiste na elaboração de desenhos na forma de linhas. A tarefa a ser demonstrada consiste em elaborar o desenho de um retângulo, conforme ilustrado na Figura 5.29. Os passos necessários para efetuar essa tarefa são descritos com detalhes na especificação do caso de uso *Programar Tarefa do Robô* descrita na Tabela 4.1. O AV nesta interface serve para mostrar os estados futuros do robô real, antes da tarefa ser enviada ao subsistema servidor.

Após concluir a tarefa, o operador envia a mesma ao servidor clicando no botão Enviar. Os passos necessários para se enviar essa tarefa são descritos com detalhes na especificação do caso de uso *Enviar Tarefa ao Robô* descrita na Tabela 4.2. Antes do envio dos comandos ao robô, o sistema prepara a formatação do comando segundo *layout* de formatação de comando descrito no apêndice D. O envio desses comandos podem ser visualizados no console de usuário ilustrado na Figura 5.30.

Cada comando enviado ao robô significa um segmento de reta que contém as informações de identificador do comando, que no caso é a letra "R", o ponto inicial (X1,Y1), o ponto final (X2,Y2) e o valor de Z para informar a posição da caneta do Plotter. Como se trata de um retângulo, a tarefa consiste de quatro comandos contendo as informações dos pontos de vértices do retângulo.

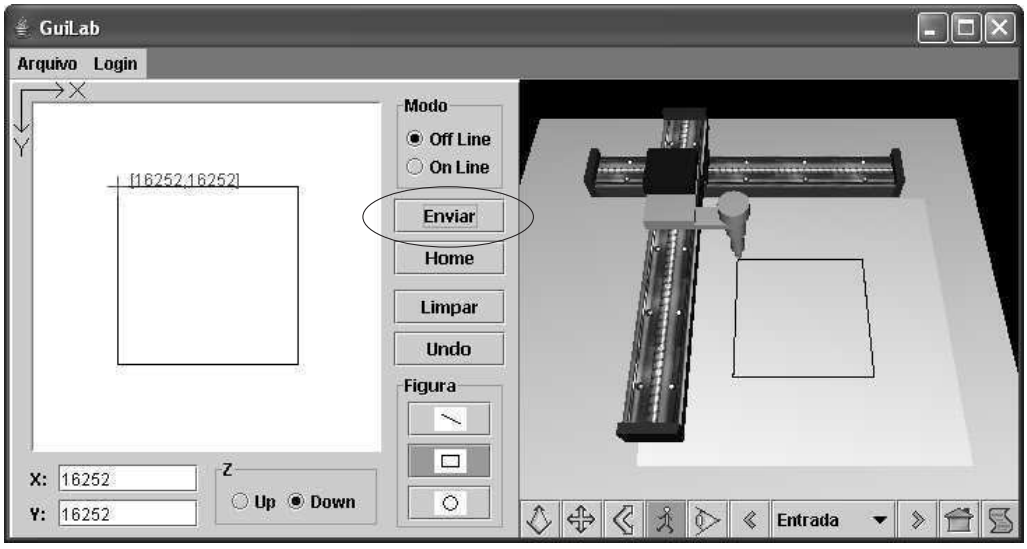

Figura 5.29: Interface de usuário para operação de tarefas.

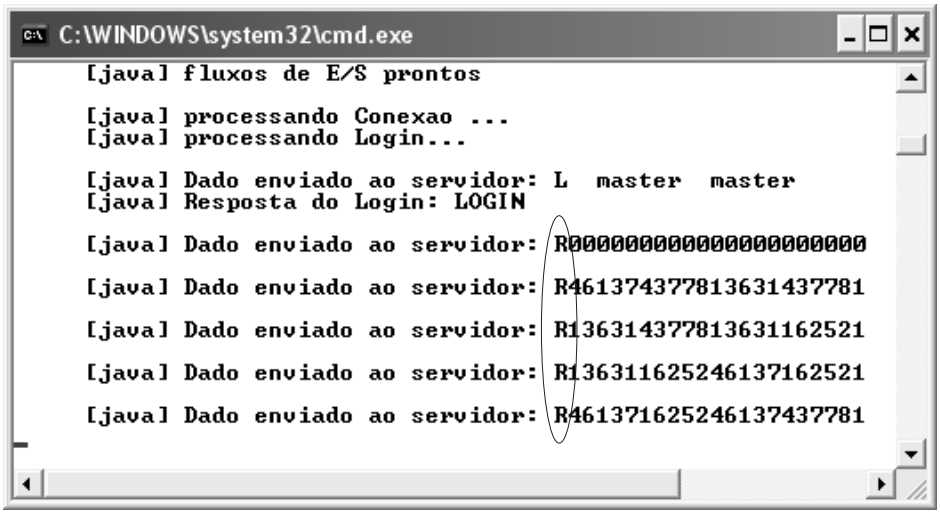

Figura 5.30: Console de usuário mostrando os dados enviados ao servidor.

No lado do servidor, esses comandos são recebidos no SCR, onde são submetidos ao controle do robô para operação.

Durante a operação, imagens do ambiente físico (vista superior e lateral) são capturadas por meio de câmeras (webcams) e processadas para se extrair as posições de baricentro do robô e dos objetos da área de trabalho. Essas posições, são então enviadas ao subsistema cliente para atualização do AV que é usado pelo operador para visualizar o resultado da operação.

O processamento das imagens do ambiente físico é descrito com detalhes na seção 3.3. Antes do envio das posições ao subsistema cliente (operador), o sistema prepara a formatação da posição com o *layout* de formatação de parâmetro descrito no apêndice D. O recebimento dessas posições podem ser visualizados no console de usuário ilustrado na

Figura 5.31.

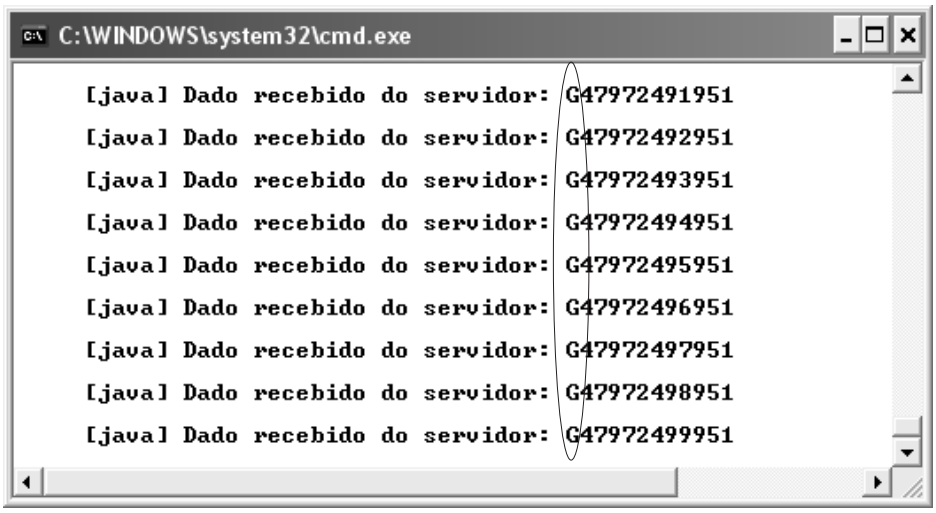

Figura 5.31: Console de usuário mostrando os dados recebidos do servidor.

Cada posição enviada ao subsistema cliente é um parâmetro que contém as informações de identificador de posição (a letra "G"para identificar o robô, a letra "R"para identificar o objeto caixa ou a letra "B"para identificar o objeto cilindro) e o ponto de posição (X,Y,Z).

No lado do cliente, esses parâmetros são recebidos e atualizados no AV da interface de usuário para visualização da operação, conforme pode ser visto na interface de usuário, ilustrado na Figura 5.32

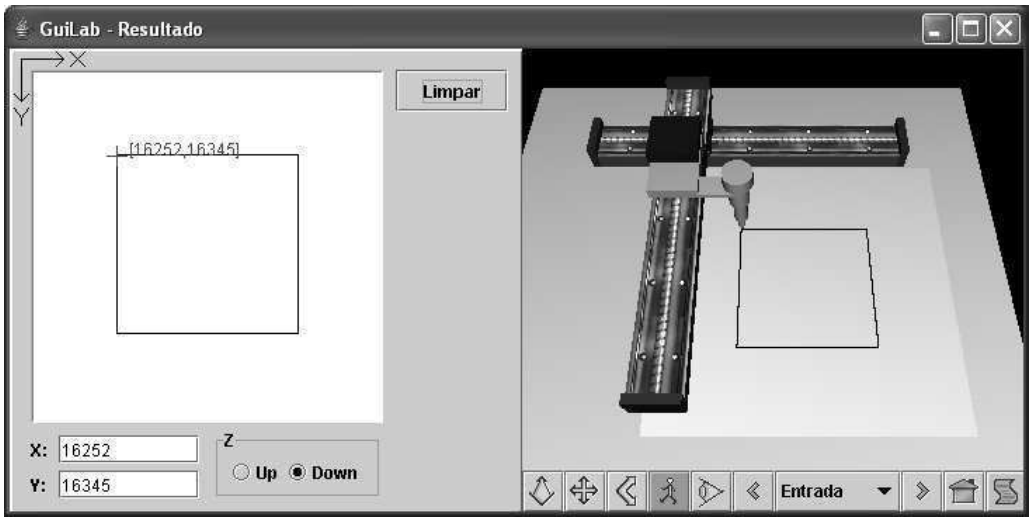

Figura 5.32: Interface de usuário para visualização do resultado da operação de tarefas.

5.5.1 Resultados da Teleoperação

O caso de teste 4 permite validar o uso da RV neste STI, onde se pôde obter respostas rápidas de teleoperação em grandes distâncias, em torno de 4,5 segundos após o envio de tarefas ao servidor. Essa redução no tempo de resposta é devido a transmissão de poucos dados na Internet.

A substituição de imagens de vídeo pelo AV diminui o tempo de resposta do sistema devido a redução no volume de dados na rede. Essa diminuição do tempo de resposta pode ser constatada utilizando a equação de tempo de resposta 1.1. Esta equação afirma que, para minimizar o tempo de resposta, deve-se transmitir um volume mínimo de dados. Como exemplo disto, considera-se uma imagem de vídeo em preto e branco com resolução de 256 x 256 pixels, com 256 níveis de cinza. Com essas características, são necessários transmitir $65 KBytes$ de dados da referida imagem na rede a uma taxa mínima de 5 a 6 quadros por segundo para se manter a imagem em movimento sem trazer desconforto ao usuário (TAN; CLAPWORTHY, 2003). Em contraste, para atualizar o AV, são necessários poucos *Bytes* de dados, cerca de $23 Bytes$ contendo apenas as informações de posições do robô ou dos objetos.

Os pacotes de dados de comandos e de posições deste STI são formatados para conter uma quantidade de poucos *Bytes*, em torno de 20 a 23 *Bytes*. O pacote de dados mais crítico usado durante a teleoperação é o de posição do robô. Esse pacote pode conter um tamanho de até 23 *Bytes* que devem ser recebidos no subsistema cliente a uma taxa mínima de 5 a 6 quadros por segundo para permitir a visualização atualizada do AV com as posições correntes do robô e objetos.

Para reduzir o tempo de resposta, pode-se diminuir a quantidade de dados transmitidos, uma vez que, estes parâmetros são relacionados pela equação 1.1. Mas deve-se ter uma relação de compromisso, para que, mesmo com a redução da quantidade de dados, o objetivo de teleoperação seja atendido.

5.6 Conclusão

Neste capítulo foram testados os casos de testes propostos para validar a RP usada na sincronização do ambiente físico com o AV, e verificar a viabilidade do uso da RV em STIs. Os casos de testes permitiram verificar os seguintes pontos:

1. A extração de parâmetros (posições do robô e objetos) foi realizada adequadamente durante os disparos das transições que representam as etapas de PDI;

2. O AV foi atualizado adequadamente através da página de supervisão após esta ter avaliado os parâmetros de forma satisfatória.

3. As exceções do sistema (falhas de identificação e eminência de colisão) foram tratadas adequadamente após a página de supervisão avaliar os valores *flags* dos parâmetros do sistema;

4. O uso da RV no STI permitiu a redução do tempo de resposta (em torno de 4,5 segundos) para grandes distâncias (Campina Grande-Manaus), apesar da taxa de transferência da Internet ser imprevisível.

Capítulo 6

Conclusões e Trabalhos Futuros

Um sistema telerobótico baseado na Internet em arquitetura cliente/servidor foi desenvolvido neste trabalho de pesquisa. O sistema consiste de uma interface gráfica de usuário com ambiente virtual do robô e da área de trabalho, e um painel para programação *off-line* de tarefas do robô.

O ambiente virtual foi modelado utilizando a linguagem X3D (sucessora do VRML) e conceitos de modelagens geométrica e cinemática que são usadas para definir as características dos objetos do ambiente físico (ambiente robótico), tais como, forma, aparência e comportamento. O ambiente virtual foi usado neste trabalho de pesquisa para reduzir o volume de dados que são transmitidos entre o local de controle (operador) e a área de trabalho remota (robô), e permitir que as respostas no controle desses sistemas se tornem mais rápidas.

Uma infra-estrutura utilizando câmeras de baixa qualidade (*webcams*) foi desenvolvida para fazer a aquisição das imagens de vistas superior e lateral do ambiente físico com o intuito de se extrair parâmetros (posições do robô e objetos) que são utilizados na atualização do ambiente virtual. Os ambientes virtual e físico devem se manter sincronizados, de modo que, o ambiente virtual possa mapear de forma mais realista o ambiente físico, principalmente se este for dinâmico. Para tanto, uma solução baseada em redes de Petri e processamento de imagens (solução RP-PDI) foi desenvolvida para fazer a integração desses ambientes.

Uma rede de Petri colorida hierárquica foi criada, utilizando a ferramenta CPNTools (sucessora do DesignCPN), para modelar as etapas de processamento de imagem (aquisição, segmentação, filtragem, representação e extração) que são usadas para extrair os parâmetros na atualização do ambiente virtual. Essas etapas são executadas por rotinas do Matlab durante o disparo de uma seqüência de transições na rede. Além disso, esta rede atua como um supervisor em situações de exceção, tais como, falha na identificação dos objetos nas imagens, e detecção de colisão entre o robô e os objetos.

Testes de teleoperação realizados a grandes distâncias (Campina Grande-Manaus) utilizando um robô Plotter relataram tempos de respostas em torno de 4,5 segundos em conexões usando a RNP [1] entre a UFCG e a UFAM. Esses testes puderam mostrar a viabilidade do uso da realidade virtual como um fator redutor do tempo de resposta em sistemas telerobóticos baseados na Internet, apesar da taxa de transferência da Internet ser imprevisível. Além disso, constatou-se que, o uso da realidade virtual também facilita a interação usuário-sistema, pois permite respostas imediatas às ações do operador, servindo como um simulador de tarefas antes de serem submetidas ao robô remoto para operação.

Para validar a integração do ambiente físico com o ambiente virtual, foram efetuados testes em situações possíveis na rede de Petri, tais como, extração de parâmetros, atualização do ambiente virtual e o tratamento de exceções (falhas de identificação de objetos em imagens e detecção de colisão), que apresentaram os seguintes resultados, respectivamente:

1. Os parâmetros (posições do robô e objetos) foram extraídos das imagens durante os disparos das transições que executam as etapas de processamento de imagens;

2. O ambiente virtual foi atualizado quando os parâmetros não apresentaram falhas de identificação;

3. As exceções do sistema (falhas de identificação e eminência de colisão) foram tratadas adequadamente após a rede avaliar os valores dos parâmetros do sistema.

Esses resultados mostraram que, o uso de redes de Petri e processamento de imagem permitiram a sincronização entre os ambientes físico e virtual, garantindo atualizações do ambiente virtual para operações subseqüentes.

6.1 Trabalhos Futuros

Nesta seção são sugeridas algumas futuras investigações que dariam prosseguimento a este trabalho de pesquisa. Os principais trabalhos identificados a serem desenvolvidos são:

1. Implementar dispositivos não convencionais (Luvas de Dados, Vídeo-Capacete) ao ambiente virtual para fornecer mais interatividade ao operador;

2. Adaptar o sistema para trabalhar com ambiente de serviço multi-usuário;

[1] Rede Nacional de Ensino e Pesquisa.

3. Implementar lógica *Fuzzy* no tratamento de incertezas do ambiente robótico, a fim de melhorar a integração dos ambientes virtual e físico;

4. Implementar múltiplos canais de comunicação entre o cliente e o servidor afim de permitir transferência de informações em paralelo. Cada *bit* de informação poderia ser enviado imediatamente quando este estivesse disponível;

5. Adaptar ao sistema o uso de uma câmera *webcam* com o intuito de verificar se o sistema está operando normalmente. Entretanto, esta não apresentaria as imagens em tempo real;

6. Adaptar o sistema para trabalhar com robôs de maior complexidade (robôs manipuladores com maior graus de liberdade) com o intuito de validar a arquitetura genérica;

7. Adaptar o sistema para aplicação *web*, e com isso, poder utilizar seus recursos de segurança, tais como, VPN (Virtual Private Network) que fornece confidencialidade, autenticação e integridade das comunicações requeridas e SSL (Security Socket Layer) que provê a privacidade e a integridade de dados entre duas aplicações que estejam se comunicando pela Internet.

Apêndice A

Morfologia Matemática

A palavra morfologia vem do grego (morphê=forma e logos=ciência); portanto é uma ciência que trata das formas que a matéria pode tomar. Como exemplo, a morfologia vegetal refere-se ao estudo da estrutura dos organismos vegetais. Seguindo esse exemplo, a morfologia matemática leva em consideração modelos matemáticos, estudando assim estruturas matemáticas.

Em termos de imagens, a morfologia matemática, que representa um ramo do processamento não linear, permite processar imagens com objetivos de realce, segmentação, detecção de bordas, esqueletização, afinamento, análise de formas, compressão, entre outras.

O princípio básico da morfologia matemática consiste em extrair informações relativas à geometria e à topologia de um conjunto desconhecido de uma imagem, utilizando uma entidade chamada elemento estruturante, que consiste num conjunto, completamente definido e conhecido (forma, tamanho), que é comparado, a partir de uma transformação, ao conjunto desconhecido cujo resultado permite avaliar este último. O formato e o tamanho do elemento estruturante possibilitam testar e quantificar de que maneira este elemento está ou não contido na imagem.

A.1 Operações de Dilatação e Erosão

Os pilares a morfologia são duas operações básicas, dilatação e erosão a partir das quais, por composição é possível realizar as operações de filtragem por meio do operador de Abertura.

Inicialmente,algumas definições básicas em relação a teoria de conjuntos são necessárias com o intuito de definir estas operações.

Sejam A e B conjuntos em Z^2, cujos componentes são $a = (a_1, a_2)$ e $b = (b_1, b_2)$, respectivamente. A translação de A por $x = (x_1, x_2)$, denotada $(A)_x$, é definida como:

$$(A)_x = \{c|c = a + x, a \in A\} \tag{A.1}$$

A reflexão de B, denotada por $\hat{B}$ é definida como:

$$\hat{B} = \{x|x = -b, b \in B\} \tag{A.2}$$

O complemento do conjunto A é:

$$A^c = \{x|x \notin A\} \tag{A.3}$$

Finalmente, a diferença entre dois conjuntos A e B, denotada por $A - B$, é definida na equação A.4 :

$$A - B = \{x|x \notin A, x \notin B\} = A \cap B \tag{A.4}$$

A.1.1 Dilatação

Sejam A e B conjuntos no espaço Z^2 e seja $\varnothing$ o conjunto vazio. A dilatação de A por B, denotada $A \oplus B$, é definida na equação A.5:

$$A \oplus B = \{x|(\hat{B})_x \cap A \neq \varnothing \tag{A.5}$$

Portanto, o processo de dilatação consiste em obter a reflexão de B sobre sua origem e depois deslocar esta reflexão de x. A dilatação de A por B é, então, o conjunto de todos os x deslocamentos para os quais a intersecção de $(\hat{B})_x$ e A inclui pelo menos um elemento diferente de zero. O conjunto B é normalmente denominado elemento estruturante.

A Figura A.1 mostra os efeitos da dilatação de um conjunto A usando o elemento estruturante B. Isso acontece porque o ponto central do conjunto B passa sobre todo o contorno do conjunto A.

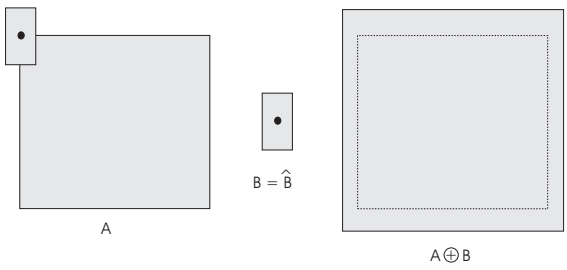

Figura A.1: Dilatação.

A.1.2 Erosão

Sejam A e B conjuntos no espaço Z^2. A erosão de A por B, denotada $A \ominus B$, é definida na equação A.6 :

$$A \ominus B = \{x|(B)_x \subseteq A\} \tag{A.6}$$

o que em outras palavras significa dizer que a erosão de A por B resulta no conjunto de pontos x tais que B, transladado de x, está contido em A.

A Figura A.2 mostra os efeitos da erosão de um conjunto A usando o elemento estruturante B.

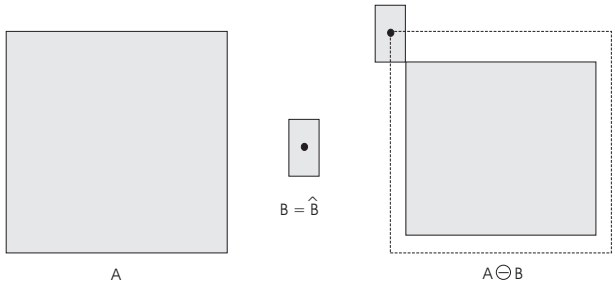

Figura A.2: Erosão.

A.2 Abertura

A aplicação de uma erosão seguida imediatamente por uma dilatação usando o mesmo elemento estruturante é chamada de uma operação de abertura. Este nome descreve a observação de que esta operação tende a "abrir"pequenas lacunas ou espaços entre objetos próximos em uma imagem. Este operador pode ser matematicamente definido pela equação A.7:

$$A \circ B = (A \ominus B) \oplus B \tag{A.7}$$

Um outro uso bastante comum da abertura é a filtragem morfológica que consiste na remoção de ruídos presentes em imagens binárias; isto decorre do fato de que a etapa de erosão tende a remover *pixels* isolados bem como o contorno dos objetos, enquanto que a etapa de dilatação restaura a maior parte dos *pixels* pertencentes ao contorno dos objetos sem restaurar o ruído.

Para um exemplo de filtragem morfológica, na Figura A.3 apresenta-se uma imagem de um objeto retangular com ruído a qual se aplica o filtro $A \circ B$. Após a operação de abertura, os pontos ruidosos externos ao objeto são removidos. Convém mencionar que o

sucesso dessa técnica depende do elemento estruturante ser maior que o maior aglomerado de *pixels* ruidosos conectados presente na imagem original.

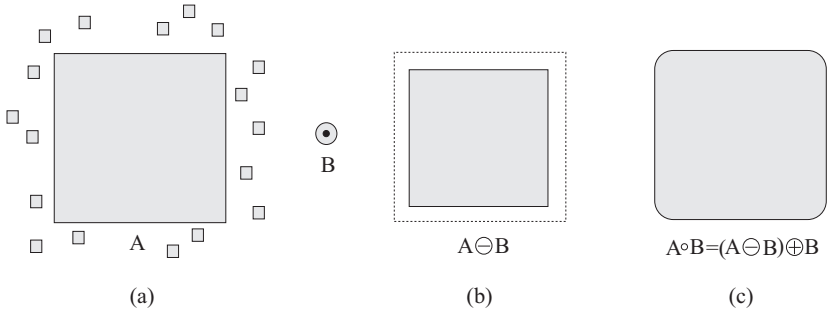

Figura A.3: Filtro Morfológico: (a) imagem ruidosa; (b) resultado da erosão; (c) resultado da abertura.

Apêndice B

Integração Matlab/CPN Tools

Neste Apêndice é apresentada a integração entre a ferramenta matemática computacional MATLAB© e a ferramenta de modelagem de sistemas a eventos discretos CPN Tools© (SOUSA, 2005). O objetivo é utilizar de forma integrada o potencial de cada ferramenta, e assim poder modelar o processo de PDI na extração de parâmetros.

A integração apresentada neste capítulo é realizada através da utilização de protocolos TCP/IP e UDP/IP. Assim, as ferramentas podem se comunicar em uma mesma máquina (modo local) ou em máquinas diferentes (modo remoto) dentro de uma rede. Na verdade, este processo pode ser aplicado na integração de outras ferramentas computacionais.

Para um estudo mais detalhado sobre a teoria e aplicação de protocolos TCP/IP, o leitor deve se reportar a (**??**).

B.1 Biblioteca TCP/IP para o MATLAB©

As bibliotecas *TCPIP Toolbox 1.2.4* e *TCP/UDP/IP Toolbox 2.0.5* (RYDESäTER, 1998-2002) permitem a criação de conexões TCP/IP ou enviar/receber pacotes UDP/IP no MatLab. Elas possibilitam a transmissão de dados através de Intranet/Internet entre aplicações ou processos do MatLab e outros aplicativos.

Portanto, o MatLab pode atuar como *servidor* e/ou *cliente* para transmitir cadeias de texto, matrizes e vetores de qualquer tipo de dado, arquivos ou variáveis. Para cada biblioteca, o leitor encontrará a lista de funções disponíveis com as respectivas descrições de utilização.

B.2 Ferramenta CPN Tools©

A ferramenta CPN Tools foi concebida e ainda está em fase de desenvolvimento com o objetivo de substituir a famosa ferramenta Design/CPN. O CPN Tools é uma ferramenta

para editar, simular e analisar Redes de Petri Coloridas (JENSEN, 1992) temporizadas, não temporizadas e hierárquicas.

Enquanto o Design/CPN foi desenvolvido para os sistemas operacionais Linux, Solaris, HP-UX, Irix e Mac OS, o CPN Tools deve ser utilizado nos sistemas operacionais Windows 2000© e Windows XP©.

O CPN Tools pode ser obtido em *http://wiki.daimi.au.dk/cpntools/cpntools.wiki*. Para obter mais detalhes sobre a ferramenta CPN Tools, o leitor deve se reportar a (BEAUDOUIN-LAFON et al., 2001; RATZER et al., 2003).

B.3 Biblioteca Comms/CPN

A biblioteca Comms/CPN foi desenvolvida para permitir comunicação entre o CPN Tools (originalmente para o Design/CPN) e processos externos via TCP/IP. Foi desenvolvida por Guy Gallasch e Lars M. Kristensen (GALLASCH; KRISTENSEN, 2001), do Computer Systems Engineering Centre, University of South Australia.

A motivação para o desenvolvimento desta infraestrutura veio a partir do desejo de visualizar, fora do CPN Tools (Design/CPN), a simulação de seus modelos. Aplicações externas podem ser desenvolvidas com maior capacidade gráfica do que o CPN Tools (Design/CPN). Existe também o potencial para executar uma aplicação externa numa máquina remota. Generalizando, a Comms/CPN torna possível integrar o CPN Tools (Design/CPN) a uma ou várias aplicações externas via TCP/IP.

A biblioteca Comms/CPN (commscpn1_3.tar.Z) para o DesignCPN está disponível na página http://www.daimi.au.dk/designCPN/libs/commscpn/. Enquanto que o CPN Tools possui a biblioteca Comms/CPN dentro do seu kernel desde a versão 1.0.1. A arquitetura da biblioteca Comms/CPN é mostrada na Figura B.1 e é comentada a seguir.

- Communication Layer: Contém a interface de suporte para o protocolo de transporte, neste caso o TCP/IP. Contém todas as funções primitivas relacionadas com o TCP/IP e socket: *connect*, *accept*, *send*, *receive* e *disconnect*.

- Messaging Layer: É responsável pela transformação do serviço (reliable byte stream) fornecido pela (transport layer) em um serviço apropriado para enviar e receber mensagens entre o Design/CPN (CPN Tools) e aplicações externas. As suas funções são: *send* e *receive*.

- Connection Management Layer: Permite aos usuários abrir, fechar, enviar para, e receber de múltiplas conexões. É com esta camada que o modelo CPN nor-

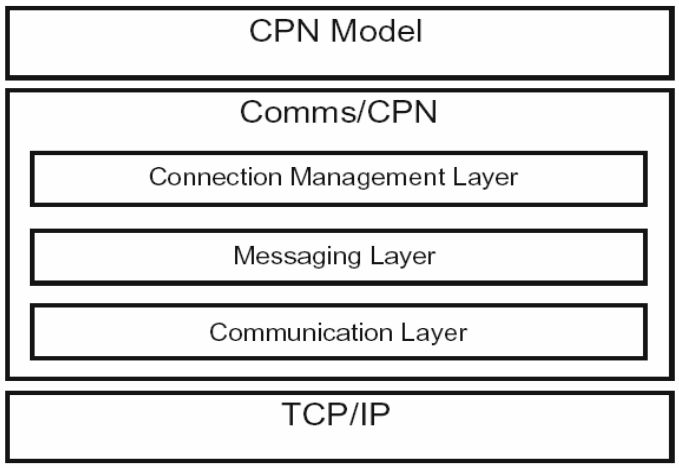

Figura B.1: Arquitetura da biblioteca Comms/CPN.

malmente se conectará. As suas funções são: *openConnection*(), *acceptConnection*("id_con",porta), *send*(), *receive*() e *closeConnection*().

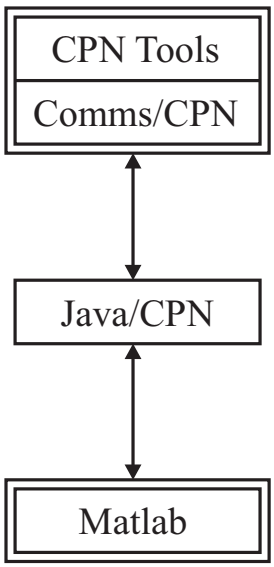

Figura B.2: Estrutura de integração entre o CPN Tools e o MATLAB.

Na Figura B.2 é mostrada a estrutura de integração entre o CPN Tools e o MATLAB. O Java/CPN realiza a comunicação com o CPN Tools através da Comms/CPN. O Java/CPN é composto de três arquivos: JavaCPNInterface.java, JavaCPN.java e EncodeDecode.java. O Java/CPN acompanha o CPN Tools, e para o Design/CPN faz parte do arquivo *commscpn1_3.tar.Z*.

Para completar a integração é necessário implementar a comunicação entre o MATLAB e o Java/CPN. Para isso, foi criado a classe de fronteira chamada de *MatlabCPNComms.java*. Assim, o Java/CPN realiza a interface de comunicação entre as duas ferramentas. Para a integração em questão, o código java dessa classe pode ser visto no

apêndice E. Neste código, tanto o CPN Tools como o MatLab estão conectados como *clientes* e *servidores* com relação ao Java/CPN.

Apêndice C

Comunicação via Soquete de Fluxo

A comunicação via soquete de fluxo é um mecanismo de transmissão de fluxo de dados, tanto para a comunicação via Internet, como para rede local. Este tipo de soquete fornece um serviço orientado a conexão e o protocolo utilizado é o Protocolo de Controle de Transmissão (Trasmission Control Protocol - TCP). Para se estabelecer uma comunicação de soquete de fluxo é necessário se ter um servidor de soquete para receber conexão de um cliente. Uma vez que a conexão é estabelecida, dados podem trafegar na rede como fluxos de *Bytes*. Nas seções subseqüentes são descritas os passos necessários para se estabelecer o servidor e o cliente utilizando soquetes de fluxos utilizando a linguagem Java.

C.1 Estabelecendo um servidor utilizando soquetes de fluxos

Estabelecer um servidor em Java requer cinco passos. O passo 1 é criar um objeto *ServerSocket*. Uma chamada ao construtor *ServerSocket* como

```
ServerSocket s = new ServerSocket( porta, tamFila );
```

registra um número de *porta* disponível e especifica um número máximo de clientes que podem solicitar conexões ao servidor. Se a fila estiver cheia, as conexões de clientes são automaticamente recusadas. A instrução precedente estabelece a porta em que o servidor espera conexões de clientes. Cada cliente solicitará conectar-se ao servidor nessa porta.

Cada conexão de cliente é gerenciada com um objeto *Socket*. Uma vez que o *ServerSocket* é estabelecido (passo 2), o servidor ouve indefinidamente (ou bloqueia) uma tentativa de se conectar por parte de um cliente. Isso é realizado com uma chamada ao método *accept* de *ServerSocket*, como em

```
Socket connection = s.accept();
```

que retorna um objeto *Socket* quando uma conexão é estabelecida.

O passo 3 é obter os objetos *ObjectOutputStream* e *ObjectInputStream* que permitem que o servidor se comunique com o cliente. O servidor envia as informações para o cliente via um objeto *ObjectOutputStream*. O servidor recebe as informações do cliente via um objeto *ObjectInputStream*. Para obter os fluxos, o servidor invoca o método *getOutputStream* para o *Socket* para obter uma referência ao *ObjectOutputStream* associado com o *Socket* e invoca o método *getInputStream* para o *Socket* para obter uma referência ao *ObjectInputStream* associado com o *Socket*. Por exemplo,

```
ObjectInputStream input = new
    ObjectInputStream(connection.getInputStream());

ObjectOutputStream input = new
    ObjectOutputStream(connection.getOutputStream());
```

O passo 4 é a fase de processamento em que o servidor e o cliente se comunicam via os objetos *InputStream* e *OutputStream*. No passo 5, quando a transmissão estiver completa, o servidor fecha a conexão invocando o método close para o *Socket*.

C.2 Estabelecendo um cliente utilizando soquetes de fluxos

Estabelecer um cliente em Java exige quatro passos. No passo 1, cria-se um *Socket* para conectar-se ao servidor. A conexão ao servidor é estabelecida utilizando uma chamada para o construtor de *Socket* com dois argumentos - o endereço Internet do servidor e o número de porta - como em

```
Socket connection = new Socket( serverAddress, port );
```

Se a tentativa de conexão for bem-sucedida, essa instrução retorna um *Socket*. Uma tentativa de conexão que falha dispara uma instância de uma subclasse de *IOException*, de modo que muitos programas simplesmente capturam *IOException*.

No passo 2, os métodos *getInputStream* e *getOutputStream* de Socket são utilizados para obter referências ao *ObjectInputStream* e ao *ObjectOutputStream* associados ao Socket, respecivamente.

O passo 3 é a fase de processamento em que o cliente e o servidor se comunicam via os objetos *ObjectInputStream* e *ObjectOutputStream*.

No passo 4, quando a transmissão estiver completa, o cliente fecha a transmissão invocando o método *close* para o *Socket*. Ao processar as informações enviadas por um

servidor, o cliente deve determinar quando o servidor terminou de enviar informações de modo que o cliente possa chamar *close* para fechar a conexão de *Socket*.

Apêndice D

Formatação de Dados

A formatação de dados oferece um meio de padronizar os pacotes de dados que trafegam entre os subsistemas, e facilitar a manutenção do sistema. Neste trabalho são apresentadas duas formatações, tais como, Formatação de Comandos de Controle e Formatação de Parâmetros. A Formatação de Comandos de Controle é usada para formatar os dados de comandos a serem enviados ao servidor do telerobô. A Formatação de Parâmetro, por sua vez, é utilizada para formatar os parâmetros de posições dos objetos e do robô, utilizados na atualização do AV. Nas seções a seguir são apresentadas essas formatações com o *layout* de formatação.

D.1 Formatação de Comando

Os comandos enviados pelo cliente são de dois tipos, a saber: comando de *Login* com as informações de Usuário e Senha; comando de Desenho com as informações dos pontos extremos do segmento de reta. O comando é formatado como se segue:

- Comando de Login: $LUUUUUSSSSSSSS$ onde:
  - L - Identificador do Comando. Indica que o comando é do tipo Login;
  - $UUUUU$ - Indica uma string de 5 caracteres contendo a informação do nome de usuário;
  - $SSSSSSSS$ - Indica uma string de 8 caracteres contendo a informação de senha.
- Comando de Desenho: $R1111122222333334444 4Z$ (Unidade Básica de Desenho) onde:
  - R - Identificador do Comando. Indica que o comando é do tipo Desenho (de Linha Reta);

- 11111 - Indica uma string de 5 caracteres contendo a informação da posição X do ponto inicial;
- 22222 - Indica uma string de 5 caracteres contendo a informação da posição Y do ponto inicial;
- 33333 - Indica uma string de 5 caracteres contendo a informação da posição X do ponto final;
- 44444 - Indica uma string de 5 caracteres contendo a informação da posição Y do ponto final;
- Z - Indica uma string de 1 caracter contendo a informação da posição Z (0 ou 1).

Observações:

- As informações do comando de Login que contiverem tamanho menor que o especificado são preenchidas com caracter branco à direita.
- As informações do comando de Desenho que contiverem tamanho menor que o especificado são preenchidas com caracter "0"à esquerda.

D.2 Formatação de Parâmetros

Os parâmetros enviados pelo subsistema servidor são de três tipos, a saber: parâmetro de baricentro para atualização da posição do robô ou objeto no AV; parâmetro flag de remoção para limpar objeto no AV; parâmetro flag de erro para mandar parar robô. Os parâmetros são formatados como se segue:

- Parâmetro de baricentro: $CXXXXXYYYYYZZZZZAAAAAAA$ onde:
  - C - Identificador do robô ou objeto, ou seja, C pode assumir os valores G, R e B que corresponde ao robô, caixa e cilindro respectivamente;
  - $XXXXX$ - Indica uma string de 5 caracteres contendo o valor de X do baricentro;
  - $YYYYY$ - Indica uma string de 5 caracteres contendo o valor de Y do baricentro;
  - $ZZZZZ$ - Indica uma string de 5 caracteres contendo o valor de Z do baricentro;
  - $AAAAAAA$ - Indica uma string de 7 caracteres contendo o valor angular do objeto caixa.

- Parâmetro flag de remoção de objeto: LC onde:
  - L Identificador do parâmetro de remoção do objeto no AV;
  - C Identificador do objeto que será removido do AV. O valor de C pode assumir os valores r e b que corresponde a caixa e cilindro respectivamente.
- Parâmetro *flag* de erro: STP Parâmetro enviado ao SCR para mandar parar o robô. Este tipo de parâmetro é gerado para o caso em que o robô e/ou o objeto não forem identificados nas imagens do ambiente operacional (Ixy e Ixz).

Observações:

- O valor de $AAAAAAA$ só é preenchido para o identificador C = r. Caso contrário, ele não é preenchido;
- As informações do parâmetro de baricentro que contiverem tamanho menor que o especificado são preenchidas com caracter "0"à esquerda.

Apêndice E

Arquivos Eletrônicos

Os arquivos eletrônicos que acompanham o CD em anexo, incluem os códigos fontes dos subsistemas servidor e cliente (em linguagem Java), do modelo do AV (em linguagem X3D), dos programas de PDI (em linguagem do Matlab), do modelo de RP (do CPNTools), e arquivos do projeto SCR (em linguagem C++, Java, doc, etc.) que estão, respectivamente, nas seguintes pastas:

- Subsistema servidor;
- Subsistema cliente;
- Modelo do AV;
- Programas de PDI;
- Modelo de RP;
- Outros;

Referências Bibliográficas

AMES, A. L.; NADEAU, D. R.; MORELAND, J. L. *VRML 2.0 Sourcebook*. 2nd. ed. 605 Third Avenue, New York, N.Y. 10158-0012: John Wiley & Sons, Inc., 1997.

APOSTOLOPOULOS, J.; TAN, W.; WEE, S. Video streaming: Concepts, algorithms, and systems. Http://www.hpl.hp.com/techreports/2002/HPL-2002-260.pdf. Setembro 2006.

BACKES, P.; TSO, K. Internet-based operations for the mars polar lander mission. *Proceedings of the IEEE International Conference on Robotics and Automation*, p. 2025–2032, April 2004.

BEAUDOUIN-LAFON, M. et al. CPN/Tools: A Post-WIMP Interface for Editing and Simulating Coloured Petri Nets. *ICATPN 2001*, p. 71–80, 2001.

BELOUSOV, I. R.; CHELLALI, R.; CLAPWORTHY, G. J. Virtual reality tools for internet robotics. Proceedings of the IEEE International Conference on Robotics and Automation. May 2001.

BRITTO, A. S. et al. Técnica interativa de segmentação por cor: aplicação à quantificação da mucina (muco intestinal). Http://www.ppgia.pucpr.br/ facon/Artigos/JaSemi97.PDF. 2007.

BURDEA, G. Invited review: The synergy between virtual reality and robotics. *Proceedings of the IEEE International Conference on Robotics and Automation*, v. 15, n. 3, p. 400–410, june 1999.

CASHER, O. et al. Virtual reality modelling language (vrml) in chemistry. Http://www.ch.ic.ac.uk/rzepa/vrml/. june 2007.

FILHO, O. M.; NETO, H. V. *Processamento Digital de Imagens*. 1st. ed. São Paulo: Brasport, 1999.

FRIZ, H. *Design of an Augmented Reality User Interface for an Internet based Telerobot using Multiple Monoscopic Views*. Tese — Institute for Process and Production Control Techniques, Clausthal-Zellerfeld, Germany, September 1998.

GALLASCH, G.; KRISTENSEN, L. M. Comms/cpn: A communication infrastructure for external communication with design/cpn. *Workshop on Modelling of Objects, Components, and Agents - MOCA'01, Joint Workshops at University of Aarhus, Denmark*, August 2001.

GOLDBERG, K.; MASCHNA, M.; GENTNER, S. Desktop teleoperation via the www. *Proceedings of the IEEE International Conference on Robotics and Automation*, p. 654–659, 1995.

GOLDBERG, K.; SANTARROMANA, J. The telegarden. Http://www.usc.edu/dept/garden/. Setembro 2006.

GONZALEZ, R.; WOODS, R. *Processamento de Imagens Digitais*. 3rd. ed. Rua Pedroso Alvarenga, 1245 - cj 2204531-012, S. Paulo- SP: Addison-Wesley, 1992.

GOOGLE. Google earth. Http://earth.google.com/. Maio 2007.

GORGÔNIO, K. C. *Adaptação de Modelos em Redes de Petri Coloridas*. Dissertação — Universidade Federal de Campina Grande, Campina Grande, Março 2001.

HARALICK, R.; SHAPIRO, L. G. *Computer and Robot Vision*. 3rd. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1992. ISBN 0201569434.

JENSEN, K. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. 2nd. ed. Ny Munkegade, Bldg. 540 DK-8000 Aahus C, Debnark: Springer-Verlag, 1992.

KIRNER, C. Sistemas de realidade virtual. Http://www.dc.ufscar.br/ grv/tutrv/tutrv.htm. 2006.

LUO, R.; LEE, W. Tele-control of rapid prototyping machine via internet for automated tele manufacturing. *Proceedings of the IEEE International Conference on Robotics and Automation*, p. 1999–2005, may 1999.

MAISENBACHER, A. Development of a control system for a 3-axis plotter. Undergraduate Project Conclusion. march 2005.

MORIE, J. Inspiring the future: merging mass communication, art, entertainment and virtual environment. *Computer Graphics*, v. 28(2), p. 135–138, may 1994.

MURATA, T. Petri nets: Propreties, analysis and applications. *Proceedings of the IEEE*, v. 77, n. 4, p. 541–580, April 1989.

NETTO, A. V.; MACHADO, L. S.; OLIVEIRA, M. C. F. Realidade virtual - definições, dispositivos e aplicações. Http://www.di.ufpb.br/liliane/publicacoes/reic2002.pdf. 2007.

OHTA, Y.; KANADE, T.; SAKAI, T. Color information for region segmentation. *Computer Graphics and Image Processing*, v. 13, p. 222–241, 1980.

OLIVEIRA, J. J. *Sistema de Processamento Digital de Imagens para Fins Didático/Científico: Estudo, Seleção e Implementação de Algoritmos de Filtragem Espacial*. Julho 2007.

OPENGL. Opengl, http://www.opengl.org//. Http://www.opengl.org/. 2007.

RATZER, A. V. et al. CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. *ICATPN 2003*, p. 450–462, 2003.

REDD, J. et al. Remote inspection system for hazardous sites. Http://www.uraweb.org/reports/anrc9918.pdf. April 2006.

ROVETTA, A.; BEJCZY, A.; SALA, R. Telerobotic surgery: applications on human patients and training with virtual reality. *PubMed*, n. 39, p. 508–517, 1997.

RYDESäTER, P. TCP/UDP/IP Toolbox 2.0.5, TCPIP Toolbox 1.2.4 - network comunication with other applications & remote matlab control. Http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=345. 1998–2002.

SAFARIC, R.; SORGO, B. Remote controlled robot arm. *Proceedings of the IEEE Transactions on Robotics and Automation*, p. 1202–1207, 2003.

SHERIDAN, T. Telerobotics, automation and human supervisory control. Cambridge, MA: MIT Press. 1992.

SILVA, L. D.; PERKUSICH, A. Uso de realidade virtual para validação de modelos de sistemas flexíveis de manufatura. *VI Simpósio Brasileiro de Automação Inteligente*, Setembro 2003.

SOUSA, J. R. B. Verificação de Propriedades de Sistemas Híbridos Modelados com Redes de Petri. Relatório Técnico. Universidade Federal de Campina Grande. Agosto 2005.

SUN. Sun microsystems, http://java.sun.com/. Http://java.sun.com/. 2006.

TAN, J.; CLAPWORTHY, G. J. Virtual environments for internet-based robots - i: Modeling a dynamic environment. *Proceedings of the IEEE - Invited Paper*, v. 91, n. 3, p. 383–388, march 2003.

TANENBAUM, A. S. *Computer Networks*. 4rd. ed. Upper Saddle River, New Jersey 07458: Prentice Hall, 2003.

TAYLOR, K.; TREVELYAN, J. A telerobot on the world wide web. *National Conference of the Australian Robot Association*, 1995. Http://telerobot.mech.uwa.edu.au/robot/telerobo.htm.

VRML. The virtual reality modeling language. Http://www.demotride.net/vrml97-spec-html/index.html. 2006.

X3D, C. X3d consortium. Http://www.web3d.org/. 2006.

X3JD. X3jd. Http://www.web3d.org/. 2006.

YANG, X.; CHEN, Q. Virtual reality tools for internet-based robotic teleoperation. *Proceedings of the Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications*, 2004.

YANG, X. et al. A web-based 3d virtual robot remote control system. *Proceedings of the IEEE International Conference on Robotics and Automation*, p. 0955–0958, maio 2004.

ZAKARIA, M.; AMIN, S.; MAMAT, R. Design and development of control system for internet-based telerobotics. *Proceedings of the IEEE International Conference on Robotics and Automation*, p. 338–342, 2000.

ZHANG, P. et al. A teleoperation system for underwater manipulator with virtual aid. *Proceedings of the IEEE International Conference on Robotics and Automation*, p. 882–886, 2003.