CRIAÇÃO E VISUALIZAÇÃO DE INTERFACES DO USUÁRIO CIENTES DE CONTEXTO PARA REALIDADE AUMENTADA

AUTOR(A):

ÁLLAN CÉSAR MOREIRA DE OLIVEIRA

Orientador(a):

Regina Borges de Araujo
São Carlos, abril de 2011.

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO

ÁLLAN CÉSAR MOREIRA DE OLIVEIRA

CRIAÇÃO E VISUALIZAÇÃO DE INTERFACES DO USUÁRIO CIENTES DE CONTEXTO PARA REALIDADE AUMENTADA

Dissertação apresentado ao Programa de Pós-Graduação em ciência da computação, para obtenção do título de mestre em ciência da computação

Orientação:Prof.Dr. Regina Borges de Araujo

SÃO CARLOS - SP
2011

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

O48cv

Oliveira, Állan César Moreira de.
Criação e visualização de interfaces do usuário cientes de contexto para realidade aumentada / Állan César Moreira de Oliveira. -- São Carlos : UFSCar, 2012.
125 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2011.

1. Interfaces de usuário. 2. Realidade aumentada. 3. Computação ciente de contexto. I. Título.

CDD: 004.6 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Criação e Visualização de Interfaces do Usuário Cientes de Contexto para Realidade Aumentada"

ÁLLAN CÉSAR MOREIRA DE OLIVEIRA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Profa. Dra. Regina Borges de Araujo
(Orientadora - DC/UFSCar)

Prof. Dr. Jander Moreira
(Co-orientador - DC/UFSCar)

Prof. Dr. Ednaldo Brigante Pizzolato
(DC/UFSCar)

Prof. Dr. Edgard Afonso Lamounier Júnior
(UFU – Faculdade de Engenharia Elétrica)

São Carlos
Abril/2011

RESUMO

A Realidade Aumentada (RA) é uma área de pesquisa que visa enriquecer a experiência de interação do usuário com o mundo que o cerca, entrelaçando elementos sintéticos, gerados pelo computador, com o mundo real, tornando cada vez menos perceptível a distinção entre o real e o virtual. A crescente integração de sensores em entidades como dispositivos móveis, edifícios e infraestruturas inteligentes, veículos, roupas e até o corpo-humano, proporciona que informações sobre essas entidades (denominadas contextos), em especial as dinâmicas, possam ser utilizadas como parte da interface e da experiência de interação do usuário, o que torna o projeto de interfaces de RA um desafio.

Segundo pesquisadores da área de RA, o aspecto de interfaces de usuário na Realidade Aumenta é ainda uma área sub explorada, com vários desafios de projeto e implementação. Essas interfaces têm o potencial de usar elementos, por exemplo, 2D e 3D adaptando-os a mudanças de contextos em tempo-real. Não há muitas ferramentas disponíveis para o desenvolvimento e visualização de interfaces de RA, muito menos que sejam cientes de contexto. Sendo assim, esta dissertação apresenta duas ferramentas que podem ser usadas de forma integrada para a criação e visualização de interfaces de RA cientes de contexto: um framework de visualização de interfaces de RA (VISAR) orientado a componentes de interface (padrões), que adapta, em tempo de execução, interfaces de RA em resposta a mudanças de contexto do ambiente; e um editor de interfaces de RA (VISAR-IE) que utiliza padrões para criar interfaces de RA cientes de contexto, permitindo o reúso e compartilhamento de componentes da interface. O uso de padrões acelera o tempo de projeto e implementação de interfaces de RA. Três estudos de caso são também apresentados para validar o uso das ferramentas desenvolvidas: RA em mesa tangível multitoque, “arraste-e-jogue-lá” (*drag and drop*) na RA e um sistema de RA para bombeiros.

ABSTRACT

Augmented reality is a research area that aims to enrich the user´s interaction experience with his/her surrounding environment by weaving together computer generated synthetic elements with the real world, making the distinction between real and virtual world less perceptible. The increasing integration of sensors in entities like mobile devices, intelligent buildings and infrastructures, vehicles, clothes, and even the human body, allows information about these entities (denominated contexts) to be used as part of both the interface and the user interaction experience, what makes AR interface project a challenge.

According to researchers of AR area, Augmented Reality (AR) User Interface (UI) is an underexplored area, which faces many design and implementation challenges. These interfaces have the potential to use elements such as 3D and 2D and adapt them according to contexts in real-time. There are not many AR tools available for the building and visualization of AR UIs, especially considering context awareness. Therefore, this work presents two different tools that together can assist in the creation and visualization of context aware augmented reality interfaces: a pattern-driven AR interface visualization framework (VISAR), which allows AR interfaces to be developed by using pre-programmed interface components (patterns); and an AR Interface Editor (VISAR-IE) that supports interface design through modeling their patterns and adaptation rules. These patterns help to speed up implementation and design time, enabling the design of adaptive interfaces, and providing sharing and reuse of interfaces components. Three study cases are also presented to validate the developed tools: AR in a tangible multi-touch tabletop, AR drag and drop, and an AR system for firefighters in emergency management applications.

Dissertação 438
01/01

ATA DO EXAME DE DISSERTAÇÃO DE MESTRADO DO CANDIDATO:

ÁLLAN CÉSAR MOREIRA DE OLIVEIRA

Aos vinte nove dias do mês de abril de dois mil e onze, às 9 h, na Universidade Federal de São Carlos, no Auditório "Prof. Mauro Biajiz" do DC, reuniu-se a Comissão Examinadora nas formas e termos do Artigo 37 do Regimento Interno do Programa de Pós-Graduação em Ciência da Computação composta pelos membros: Profa. Dra. Regina Borges de Araujo (Orientadora - DC/UFSCar), Prof. Dr. Jander Moreira (Co-orientador - DC/UFSCar), Prof. Dr. Ednaldo Brigante Pizzolato (DC/UFSCar) e Prof. Dr. Edgard Afonso Lamounier Júnior (UFU - Faculdade de Engenharia Elétrica) para o Exame de Dissertação de Mestrado sob o título **"Criação e Visualização de Interfaces do Usuário Cientes de Contexto para Realidade Aumentada"**. A sessão foi aberta pela presidente Profa. Dra. Regina Borges de Araujo. Após a explanação do candidato, a Presidente passou a palavra aos componentes da Comissão Examinadora. Terminada a argüição, a Comissão Examinadora reuniu-se em sessão secreta e elaborou o seguinte parecer: A comissão examinadora considerou que o trabalho satisfaz plenamente as exigências do programa de mestrado. Na apresentação o candidato mostrou bom conhecimento do assunto, tendo respondido com segurança as questões formuladas pelos membros da comissão. A comissão solicita, no entanto, que o candidato revise o texto de sua dissertação conforme correções sugeridas pela banca. De acordo com o artigo 31 do Regimento Geral dos Programas de Pós-Graduação da UFSCar o candidato foi APROVADO. A Presidente declarou por encerrado o exame de dissertação. Nada mais havendo a tratar, eu, Maria Cristina Carreira Trevelin, lavrei a presente ata, que assino juntamente com os membros da Comissão Examinadora.

Profa. Dra. Regina Borges de Araujo
Presidente

Prof. Dr. Jander Moreira
Membro Interno

Prof. Dr. Ednaldo Brigante Pizzolato
Membro Interno

Prof. Dr. Edgard Afonso Lamounier Júnior
Membro Externo

Maria Cristina Carreira Trevelin
Assistente Administrativo do PPG-CC

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3DOF	3 Degree of Freedom
6DOF	6 Degree of Freedom
AOA	Angle of Arrival
AP	Access Point
API	Application Programming Interface
CAD	Computer-Aided Design
CAVEs	Cave Automatic Virtual Environment
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DFN	Data Flow Network
GPS	Differential Global Positioning System
GPS	Global Positioning System
GPS RTK	GPS Real Time Kinematic
GSM	Global System for Mobile
VISAR	*Framework* Ciente de Contexto para Visualização de Interfaces de Realidade Aumentada
HMD	Head-Mounted Display
HUD	Head-Up Display
IHC	Interface humano-computador
ISMAR	International Symposium on Mixed and Augmented Reality

LED	Light-emitting Diode
LOD	Level of Detail
LOE	Level of Error
MAV	Micro Air Vehicleou Miniature Unmanned Air Vehicles
MVC	Model–View–Controller
PDA	Personal Digital Assistant
P2P	Peer-to-peer
RA	Realidade Aumentada
RFID	Radio-frequency Identification
RSSI	Received Signal Strength Indication
RV	Realidade Virtual
SCAAT	Single Constraint At A Time
SLAM	Simultaneous Localization and Mapping
SRD	Retinal Scan Display
SRG	Spatial Relationship Graph
TDOA	Time Difference of Arrival
UMTS	Universal Mobile Telecommunications System
UWB	Ultra-Wideband
VRD	Virtual Retinal Display
WIM	World in Miniature

SUMÁRIO

1 INTRODUÇÃO

A Realidade Aumentada (RA) tem o potencial de enriquecer a experiência de interação do usuário com o mundo que o cerca, entrelaçando elementos sintéticos, gerados pelo computador com o mundo real, tornando cada vez menos perceptível a distinção entre o real e o virtual. A crescente integração de sensores (que coletam informações como temperatura, posição geográfica, pressão de impacto, aceleração, dentre várias outras) nos vários objetos (e pessoas) que nos cercam, proporciona que informações sobre essas entidades (denominadas contextos), em especial as dinâmicas, possam ser utilizadas como parte da interface e da experiência de interação do usuário, o que torna o projeto de interfaces de RA um desafio.

Com a crescente evolução tecnológica de dispositivos de visualização, miniaturização, processamento e mobilidade, a Realidade Aumentada tem sido cada vez mais aplicada e difundida nas mais diversas áreas, sendo as mais comuns medicina, indústria, propaganda, robótica e entretenimento. Exemplos de aplicações vão desde o provimento de informações adicionais no celular ou PDA para localização de restaurantes, bancos [Bradesco09], pontos turísticos [Wikitude08], até manutenção de aviões [De Crescenzio11] e cirurgias complexas [Fuchs98].

A RA possibilita a adição de informações virtuais ao mundo real, facilitando a execução de tarefas e transpondo a tela do monitor para uma visualização de dados mais próxima das pessoas e menos estática. Tal maneira de apresentar dados não possui/retira as restrições de espaço e mobilidade, podendo acompanhar o usuário.

Além da simples exibição de dados em *displays* móveis, como óculos estereoscópicos e em *displays* de visualização retinal (figura 8), a RA suporta a projeção de dados no ambiente real, o que, aliás, é o seu intuito. Através de uma projeção fotorealística, é possível iludir o usuário, dando-lhe a sensação de um ambiente unicamente real e criando a possibilidade de

interação com os objetos virtuais.

Apesar do alto potencial de uso de RA em diferentes áreas de aplicação, vários projetos de RA são fortemente acoplados ao *hardware* dos dispositivos, podendo ser estes específicos ou personalizados. Os mecanismos que correlacionam as informações de contexto ao mundo real são normalmente complexos, o que dificulta a criação mais eficiente de interfaces de aplicações de RA, tornando-as de alto custo e de difícil manutenção [Educause05]. Outros desafios da RA incluem o desenvolvimento de dispositivos de visualização, interação do usuário com os objetos virtuais, calibração de equipamentos em tempo de execução, rastreamento do usuário e de objetos de interesse e registro (alinhamento) de objetos virtuais no ambiente real, dentre outros.

A complexidade dos desafios aumenta quando as interfaces de RA são cientes de contexto. Segundo Dey [Dey00], contexto pode ser definido como qualquer informação que possa ser utilizada para caracterizar a situação de entidades (pessoa, lugar ou objeto) que sejam consideradas relevantes para interação entre um usuário e uma aplicação (incluindo o usuário e a aplicação). Interfaces cientes de contexto são consideradas neste trabalho como sendo interfaces que se adaptam como resultado de alterações de contexto no ambiente da aplicação.

Não há muitas ferramentas disponíveis para o desenvolvimento e visualização de interfaces de RA, muito menos que sejam cientes de contexto. Como essas interfaces de RA possuem características diferentes das interfaces mais tradicionais, tais como *desktop*, *web* ou sistemas móveis, ferramentas e métodos para projetos de interfaces existentes devem ser criados, complementados ou adaptados para suporte à criação e visualização de interfaces de RA cientes de contexto. Pouco trabalho foi realizado nesta área específica [Swan05].

Atualmente os desenvolvedores de sistemas de RA se preocupam mais com questões de hardware, como rastrear com precisão o usuário ou ter um dispositivo de visualização ótimo. Quando as tecnologias para implementar RA estiverem mais avançadas e consolidadas (por exemplo para tratar o rastreamento) e os sensores se tornarem comuns e mais amplamente utilizados, os desafios da RA serão mais voltados à interface do usuário.

Tendo em vista as limitações e desafios apresentados na criação e visualização de interfaces de RA, este trabalho visa contribuir apresentando duas ferramentas que podem ser usadas de forma integrada para a criação e visualização de interfaces de RA cientes de contexto:

- *Framework* de visualização de interfaces de RA (VISAR) orientado a componentes de

interface (padrões), que adapta, em tempo de execução, interfaces de RA em resposta a mudanças de contexto do ambiente; VISAR é um *framework* de desenvolvimento de RA que auxilia na criação de interfaces do usuário, através da utilização de padrões de interface (componentes gráficos implementados para solucionar problemas conhecidos) desenvolvidos, por exemplo, para prédios inteligentes (ambientes com o equipamento necessário para rastreamento do usuário e para sensoriamento do ambiente). VISAR faz uma conexão entre elementos da interface do usuário e contextos considerados pela aplicação, permitindo a adaptação da interface em tempo de execução de acordo com a situação das entidades de interesse da aplicação;

- Editor de interfaces de RA (VISAR-IE), que utiliza padrões para criar interfaces de RA cientes de contexto, permitindo o reúso e compartilhamento de componentes da interface. VISAR-IE é um editor que permite aos desenvolvedores modelar interfaces de RA baseado nos padrões de interface e em regras de adaptação a contextos (ou cenários). Ele pode ser utilizado por um especialista na área da aplicação (comandante dos bombeiros, gestor de segurança da indústria, gerentes comerciais, etc.) que não tenha conhecimento de programação, mas exige um entendimento de ferramentas de modelagens (modelagem de fluxograma, modelagem de mapa mental) e interfaces de RA;

Enquanto o VISAR-IE suporta o projeto de interfaces de usuário de RA cientes de contexto, o *framework* VISAR implementa e executa (em tempo de execução) a interface, sendo controlado pela aplicação. Similar ao padrão MVC (*model-view-control*), o *framework* seria o modelo e a visualização e a aplicação seria o controle, que é responsável por atualizar os dados do modelo, que serão refletidos na visualização.

Essas ferramentas foram criadas para reduzir a complexidade do desenvolvimento de interfaces de RA. Suas características principais são:

- Permitem a elaboração de interfaces 3D projetadas no ambiente real e de interfaces 2D projetadas na tela do dispositivo de visualização (que pode ser, por exemplo, a viseira de um capacete do tipo *Head Mounted Display* - HMD);
- Possibilitam a modelagem de interfaces cientes de contexto;
- Promovem a implementação e elaboração mais ágil de interfaces, por meio do uso de componentes (padrões).
- Facilitam o reúso e compartilhamento de padrões entre desenvolvedores.

O VISAR funciona basicamente como uma camada de apresentação/visualização em aplicações de RA. Exemplos de aplicações de RA que o VISAR poderia ajuda a construir incluem:

- Escritório: uma aplicação para auxiliar empregados em escritórios. Por exemplo, quando um trabalhador quer localizar seu chefe, a RA pode guiar o usuário através de um caminho virtual, ou quando o usuário manda imprimir um documento ela pode iluminar a impressora utilizada.

- Casa: uma aplicação para auxiliar os moradores a fazer tarefas diárias. Por exemplo, quando o usuário quer cozinhar uma receita, a RA pode ajudar a achar os ingredientes mostrando rótulos virtuais nas portas de armários da cozinha onde cada ingrediente está, ou ajudar a achar o controle remoto da televisão apontando uma seta virtual na direção dele.

- Emergência (bombeiros): uma aplicação para auxiliar bombeiros em ambientes de emergência. Por exemplo, a RA pode ajudar mostrando um modelo 3D de cada pessoa no ambiente, para que os bombeiros saibam onde cada um de seus colegas de time está e onde estão as vitimas que já foram localizadas. Outro exemplo seria um mapa 2D mostrando o que está acontecendo em cada quarto do local.

Pode-se perceber que cada elemento da interface do usuário usado em uma dessas aplicações poderia ser utilizado nas outras também, sendo que estes elementos são os padrões de interface presentes no VISAR.

1.1 **Motivação**

Considerando-se os trabalhos atuais de RA de maior importância, nota-se pouco investimento e pesquisa no desenvolvimento de ferramentas para modelagem e desenvolvimento de interfaces do usuário [Zhou08]. A maioria foca em áreas que trabalham no funcionamento básico da tecnologia, como rastreamento, calibração e *display*.

Poucos trabalhos conseguem unir as técnicas de criação de interface mais utilizadas, *frameworks*, ferramentas de autoria e linguagens de descrição, para gerar um ferramental mais poderoso capaz de dar suporte ao desenvolvedor tanto na modelagem como na codificação da aplicação.

Mesmo entre os trabalhos que conseguem explorar melhor as técnicas de criação de interface, até onde é de conhecimento do autor desta dissertação, nenhum ajuda o desenvolvedor a

considerar contexto tanto na modelagem como no processo de visualização.

O VISAR foi idealizado com o objetivo de superar essas limitações.Com o VISAR, o desenvolvedor poderá projetar e implementar uma interface de RA em três partes: 1) modelagem: durante a modelagem da interface o desenvolvedor escolhe os componentes de interface pré-programados que irão compor a interface da aplicação, bem como seus cenários e contextos, além de possíveis respostas às interações do usuário e como a interface responderá a eles; 2) geração de arquivo XML: o resultado da modelagem é armazenado em um arquivo XML, contendo a estrutura dessa interface, que é carregado no *framework*; 3) execução da aplicação: a interface pode ser controlada por meio de comandos enviados à API do *framework*.

1.2 **Objetivos Gerais e Específicos**

O objetivo geral deste trabalho é criar ferramentas de suporte à modelagem e visualização de interfaces de Realidade Aumentada cientes de contexto, que facilitem o trabalho dos desenvolvedores desse tipo de interface. Para isso, os seguintes objetivos específicos caracterizam as etapas deste trabalho:

- Criar um framework de visualização (VISAR) que permita implementar interfaces de RA sem a necessidade de codificá-las, utilizando padrões de interface.
- Utilizar componentes como forma de desenvolver aplicações de RA, utilizando, por exemplo, componentes para: rastreamento, captura e interpretação de dados do ambiente, recepção de entrada de dados dos usuários, além de controle e renderização da interface (VISAR).
- Criar um ambiente para modelagem de interfaces (VISAR-IE) que permita projetar interfaces de RA com interfaces tradicionais.
- Implementar três casos de estudo para validação das ferramentas geradas.

1.3 **Organização**

O capítulo 2 apresenta os fundamentos de Realidade Aumentada, suas principais características e desafios, bem como algumas aplicações.

O capítulo 3 introduz os conceitos e usos de contextos. Além da apresentação dos problemas de localização e rastreamento.

O capítulo 4 apresenta a proposta de projeto, com a descrição do VISAR, seu funcionamento e suas estruturas internas, além da introdução do VISAR-IE, seguidos de apresentação de três estudos de casos construídos com o ferramental desenvolvido.

Finalmente, no capítulo 5, são descritas as conclusões, seguidas de Referências Bibliográficas e Anexos.

2 REALIDADE AUMENTADA

A Realidade Aumentada (RA) é uma área de pesquisa em que objetos sintéticos gerados pelo computador suplementam e/ou compõem ambientes reais, fornecendo uma visualização de informações mais rica e presente no dia-a-dia, mesmo em situações de mobilidade do usuário e sem restrições de espaço.

A RA surgiu como um conceito que se completa à Realidade Virtual (RV), uma vez que considera o real como o ambiente que envolve o usuário, ao contrário da realidade virtual, em que o usuário é envolto por ambiente sintético e simulado. Como na RA não se tem o controle total do ambiente, ela é considerada mais complexa do que a RV, na qual o ambiente é mantido sob total controle [Bimber05].Considera-se que essas duas áreas de estudo são correlatas e que descobertas e pesquisas feitas em uma também afetam a outra.

A figura 1 mostra um diagrama da virtualidade contínua que vai de um ambiente puramente real até um ambiente puramente virtual, onde podem ser observadas as variações da realidade mista (Realidade Aumentada e virtualidade aumentada) [Milgram94].

Neste caso, na Realidade Aumentada o usuário está em um ambiente real, mas este ambiente é sobreposto por objetos virtuais criados por meio de computação gráfica. Já na virtualidade aumentada o usuário está em um ambiente virtual e é capaz de alterar este ambiente a partir de objetos reais (em geral usando o próprio corpo como agente de atuação). Um exemplo de virtualidade aumentada é a nova geração de consoles de games, em que o usuário controla seu avatar através dos movimentos do seu corpo (figuras 18,19 e 20).

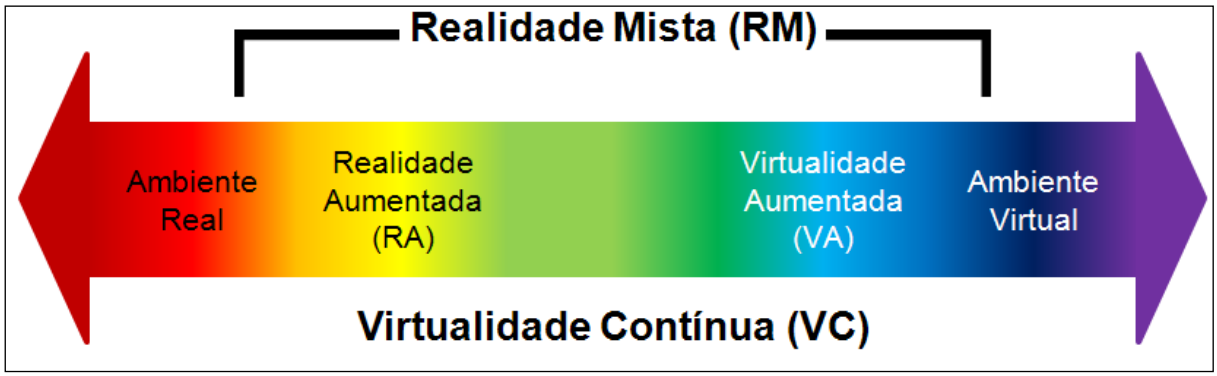

Figura 1: Diagrama da virtualidade contínua em relação às realidades mistas [Milgram94].

Através da RA, a computação é empregada para aumentar a percepção e interação de uma pessoa com o mundo real [Azuma97]. O seu uso mais comum é através da computação gráfica para amplificar o sentido da visão, por meio da inserção de objetos 3D no mundo real ou da rotulação de elementos reais. Além de influenciar o sentido da visão, essa tecnologia permite a inserção de sons virtuais e até mesmo aumentar o tato.[1]

A RA vem sendo cada vez mais aplicada em diferentes áreas, como por exemplo, a indústria automobilística [Schreiber08], jogos [GhostWire11], propaganda [Sprite10], aplicações para auxiliar a localização [Wikitude08] e outros. Uma das características que torna a RA cada vez mais incorporada ao dia-a-dia das pessoas é o fato de ela ser menos intrusiva do que a RV [Schmalstieg02].

Consoante a essa afirmação, Ronald Azuma define a RA deforma independente de tecnologias visuais, como um sistema com as seguintes características [Azuma97]:

1. combina objetos reais com virtuais em um ambiente real;
2. oferece interatividade em tempo de execução;
3. alinha (registra) elementos do mundo real com objetos virtuais;

Embora a RA tenha varias áreas de estudos repletos de desafios, como as áreas de *displa*ys, autoria, interação e usabilidade, o problema mais discutido e trabalhado pelos pesquisadores é o do registro (*registration*) [Ahad04].Essa tendência pode ser observada através da tabela 1, que mostra a quantidade de artigos para cada área da RA, em que foram computados

[1] Projetos ligados ao tato estão sendo desenvolvidos através de sensores de retorno de força (*force feedback*), os quais são ativados quando o usuário toca um objeto virtual. Com isso, há a sensação de toque de um objeto que não existe no mundo real.

todos os artigos que já apresentados no ISMAR (IEEE *International Symposium on Mixed and Augmented Reality*), o maior evento científico de Realidade Aumentada.

Tabela 1: Relação dos artigos e citações de acordo com a área em que se inserem, adaptado de [Zhou08].

Área / Tópicos	Artigos %	Citações %
Rastreamento / Localização	20.1	32.1
Interação	14.7	12.5
Calibração	14.1	12.5
Aplicações de RA	14.4	12.5
Display	11.8	5.4
Teste	5.8	1.8
RA móvel	6.1	7.1
Autoria	3.8	8.9
Visualização	4.8	5.4
Multimodal	2.6	0.0
Renderização	1.9	1.8

Os principais desafios de RA bem como os trabalhos de maior influência de cada área serão apresentados a seguir.

2.1 Displays

A RA é um exemplo da amplificação da inteligência [Brooks96], ou seja, o uso da tecnologia para facilitar o cumprimento de tarefas. Sua principal característica é a sobreposição do mundo real através de elementos virtuais. Essa sobreposição pode inclusive ocorrer de duas formas: acrescentando-se algo virtual ao ambiente real ou removendo-se um objeto real (escondendo o objeto com computação gráfica) [Azuma01].

Conforme explicado anteriormente neste trabalho, pode ocorrer com qualquer um dos sentidos, desde que haja o equipamento certo. Em consonância com a abordagem empregada na maior parte dos trabalhos e projetos de RA, neste o foco será voltado para o sentido da visão. Assim, nos próximos tópicos, são apresentados os três tipos mais utilizados de *displays* de visualização para a RA.

2.1.1 Head Mounted Display

O *head-mounted display* (HMD) é um *display* para se usar na cabeça como um capacete,

ou, em casos mais simples, um par de óculos. A vantagem dele sobre os outros tipos é a portabilidade e conforto, já que nenhum dos outros tipos permite o livre movimento do usuário. Há dois tipos de HMD: o ótico e o de vídeo.

No *display* ótico, os óculos permitem ao usuário ver o mundo real combinado com a projeção de elementos da Realidade Aumentada, os quais são projetados nas lentes do aparelho.Várias tecnologias podem ser usadas para a lente, como um espelho de duas faces (*half-silvered mirror*) ou um LCD transparente. A figura a seguir mostra o esquema definido por Azuma de um visor ótico:

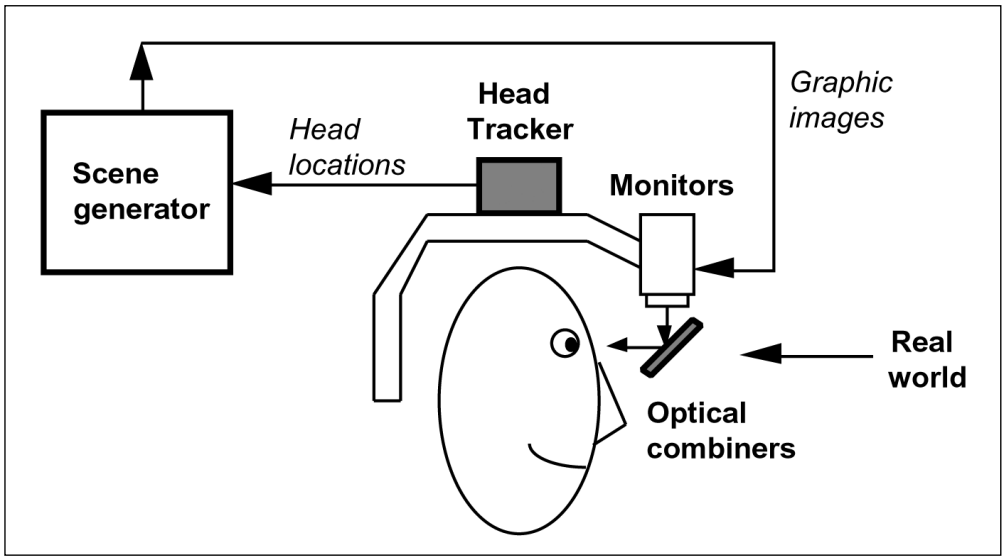

Figura 2: Diagrama conceitual do *display* de visão ótica [Azuma97].

Algumas vantagens do *display* ótico sobre o de vídeo são:

- sua produção é mais fácil e o custo envolvido é menor, pois não é necessária sincronização do vídeo com os elementos 3D;
- não há limitação da resolução do que o usuário vê, pois a visão do mundo é feita diretamente pelos olhos do usuário;
- há maior segurança, uma vez que, se acabar a energia, o usuário vai continuar vendo o mundo real normalmente e não ficar sem visão;
- não há um desvio de visão, que acontece no de vídeo por que a câmera não está posicionada exatamente onde estão os olhos.

Já no *display* de vídeo, o usuário vê o que é filmado pelas câmeras o tempo todo. É como se ficasse totalmente imerso, pois não tem uma visão direta do mundo real. A imagem da câ-

mera é combinada com a imagem dos objetos 3D para formar a Realidade Aumentada. A figura a seguir, é um esquema para definir o HMD de vídeo:

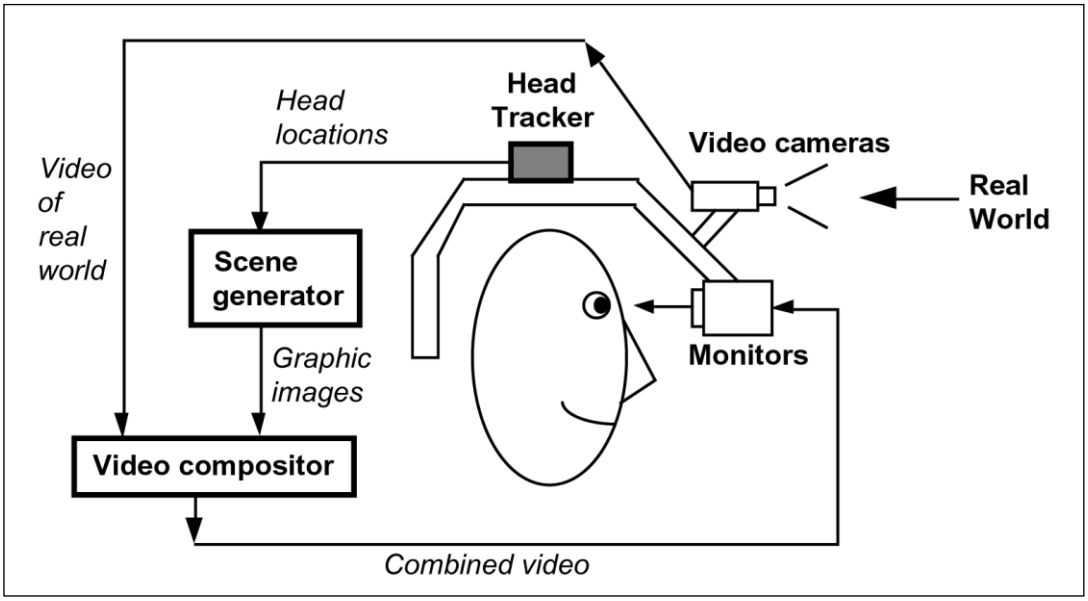

Figura 3: Diagrama conceitual do *display* de visão por vídeo [Azuma97].

As vantagens do de vídeo sobre o ótico são:

- facilidade de colocar os objetos virtuais na cena real, pois, como tudo é baseado no vídeo, a edição dos objetos virtuais se torna uma edição de vídeo em tempo de execução;
- maior facilidade de controle de problemas de atraso entre o real e o virtual, porque tudo é digitalizado;
- maior facilidade de equalizar o brilho e contraste do virtual e do real.

Os *displays* de vídeo acabaram se tornando o *display* tradicional de pesquisadores de RA, por serem mais fáceis de gerar RA registrada.

2.1.2 Monitor

A RA pode ser visualizada através de um monitor fixo ou um portátil (celular, *handheld*, PDA). Neste tipo de *display*, uma câmera, que pode ser uma *webcam* ligada a um computador ou a câmera do celular, filma o mundo e usando a estratégia do HMD de vídeo combina as imagens filmadas do mundo real com a Realidade Aumentada. O esquema a seguir foi retira-

do de [Azuma97]:

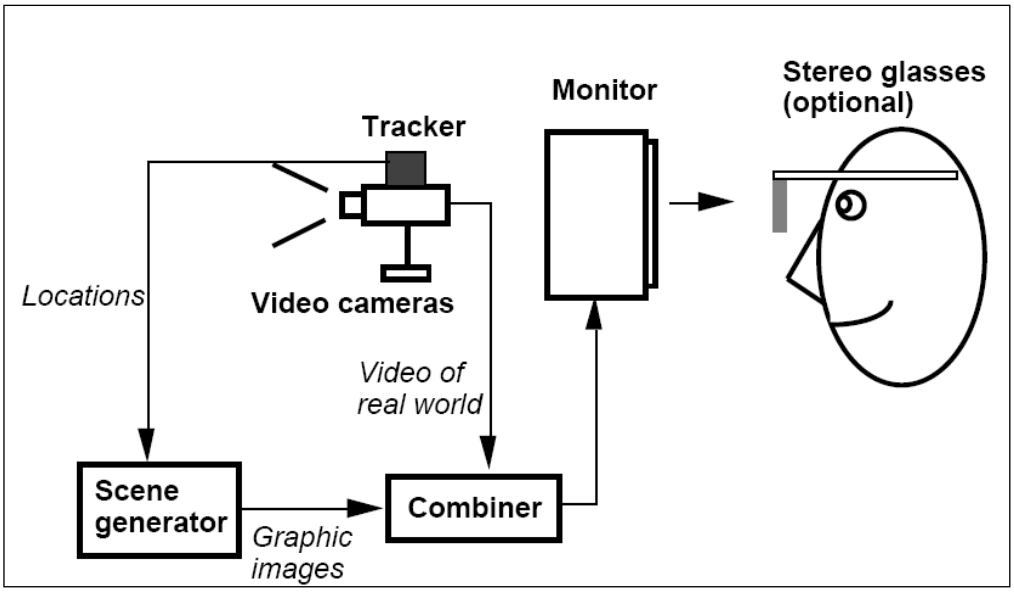

Figura 4: Diagrama conceitual do *display* baseado em monitor ou portátil [Azuma97].

2.1.3 Projetor

Este tipo de *display* consiste em vários projetores exibirem no ambiente real os objetos aumentados. A vantagem dessa tecnologia é ser menos intrusiva, já que os usuários não precisam carregar ou utilizar nenhum equipamento no corpo. Um exemplo são as CAVEs (*Cave automatic virtual environment*).

Contudo, o projetor apresenta uma grande desvantagem em relação aos outros tipos de *display*,uma vez que exige que o ambiente seja previamente preparado para a sua utilização. Em vista disso, o usuário, que não pode sair do ambiente preparado, tem sua mobilidade severamente limitada. É uma tecnologia mais comum em laboratórios ou indústrias e dificilmente vai chegar ao consumidor.

2.1.4 Evolução dos *Displays*

A evolução dos *display*s de RA foi grande nos últimos anos, podendo ser percebida claramente nas figuras a seguir:

Figura 5: MARS - *Mobile Augmented Reality Systems*, Universidade de Columbia, 1997.

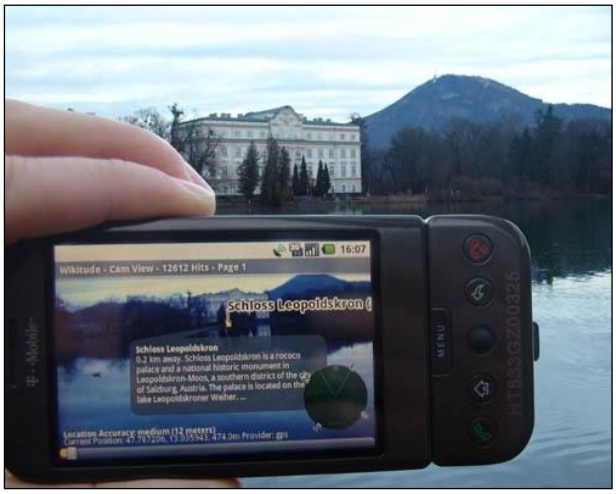

Figura 6: *Wikitude AR Travel Guide* no celular G1, do Google, 2008 [Wikitude08].

Figura 7: *Display* de visão ótica da empresa Microvision, ainda é um projeto inacabado.

A figura 5 mostra o sistema MARS, de 1997, da universidade de Columbia. A figura 6 mostra o *Wikitude AR Travel Guide,*uma aplicação de RA que executa no sistema operacional Android, usado no celular HTC G1. A figura 7 mostra o dispositivo de visualização da empresa Microvision. Comparando as figuras, pode ser visto que os *displays* de visualização já chegaram a um ponto em que seu uso não incomoda mais o usuário e embora óculos como o da empresa Microvision ainda não sejam comercializados, não é mais necessário um sistema complexo para conseguir gerar a RA, um celular com uma câmera pode fazer isso muito bem.

O próximo passo da evolução dos *displays* já está sendo dado por pesquisadores da Universidade de Washington (UW), que desenvolveram um protótipo de lentes de contato com luzes e circuitos eletrônicos. O nome da tecnologia é *virtual retinal display* (VRD), ou *retinalscan display* (RSD) e já existe desde 1991 nos laboratórios da UW. A figura 8 mostra o protótipo em duas situações: em um, está sendo usado por um coelho; no outro, está na mão de uma pessoa.

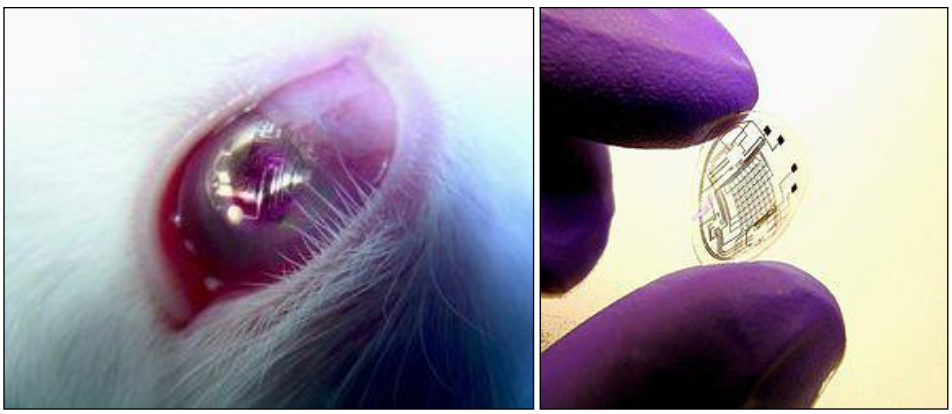

Figura 8: Protótipo da tecnologia VRD da Universidade de Washington.

2.2 **Registro**

De acordo com a tabela 1, a porcentagem de artigos das áreas de rastreamento e calibração somadas chega a quase 35%. Como essas áreas existem para resolver o problema do registro, é correto afirmar que é este o escopo de 35% dos artigos do maior evento de RA.

O problema de registro está ligado ao alinhamento. Na RA o ambiente real se mistura com o ambiente virtual. Em outras palavras, o ambiente virtual deve se encaixar no ambiente real perfeitamente, de forma que o usuário se sinta dentro de um ambiente misto. Para esse encaixe ocorrer de forma perfeita, o alinhamento entre os objetos 3D e o mundo real deve ser exato. Pequenos erros de registro são perceptíveis devido à alta capacidade visual humana (resolução da fóvea), que permite identificar erros de poucos píxeis.

Imagine um cenário com duas portas lado a lado, sendo que um objeto virtual deve ficar na frente de uma das portas para impedir o usuário de entrar. Um alto erro de registro poderia acabar posicionando o objeto virtual na frente da porta errada.

Este tipo de erro é causado por várias fontes de duas diferentes naturezas, estáticas e dinâmicas. Os problemas de registro estáticos são aqueles que acontecem mesmo quando o usuário e seu ponto de vista se mantêm imóveis, e, portanto, deveriam ser resolvidos antes mesmo da execução do sistema.

Os problemas de registro dinâmico se apresentam na forma de atraso na comunicação do sistema (*delay*), ou seja o atraso entre o rastreamento da posição do usuário e objetos de interesse na cena à renderização dos objetos virtuais no *display* do usuário. Esse tipo de erro somente acontece enquanto o usuário está se movimentando.

As principais fontes do problema de registro estático são:

- Distorção óptica: a distorção óptica pode ocorrer tanto nas câmeras quanto no *display* do dispositivo de visualização. Esta distorção é uma função da distância radial pela distância do eixo óptico. Isso causa distorções em imagens próximas à borda do display de HMDs com amplo campo de visão. É um problema que pode ser mapeado e compensado, com a inclusão de lentes óticas, ou digitalmente, embora Holloway tenha determinado que o atraso adicional gerado para compensar a distorção óptica digitalmente cause mais erro de registro do que remove, para movimentos da cabeça do usuário [Holloway95].

- Erros no sistema de rastreamento: erros nos sistemas de rastreamento são as maiores causas de problemas de registro estático e somente podem ser eliminados utilizando um sistema de rastreamento que oferece maior acurácia.
- Desalinhamento mecânicos: são problemas causados pela especificação errônea dos atributos de equipamentos, oferecidos pelos seus fornecedores. Podem ser corrigidos com calibração e testes para adaptar o sistema aos atributos corretos dos equipamentos.
- Parâmetros incorretos: este erro acontece quando os parâmetros de visão são passados incorretamente para o sistema. Alguns exemplos de parâmetros são o centro da projeção, à distância (translação e orientação) entre o rastreador inserido no capacete e os olhos do usuário e o campo de visão. Pode-se modelar estes parâmetros manualmente, usando sensores, ou utilizando visão computacional.

Quando a aplicação usa projetores como *display* de visualização da Realidade Aumentada, ainda existe um tipo de problema de registro estático que não existe em outros *displays*. Nesse caso também é preciso preocupar com o acerto geométrico entre as imagens sobrepostas projetadas pelos projetores [Bimber05].

Os problemas de registro dinâmico causados por atrasos são a principal causa de erros de registro em um sistema [Holloway95]. Algumas formas de reduzir esse problema em sistemas de RA são:

- Reduzir o atraso do sistema: é o método mais eficiente, pois se o atraso no sistema for nulo o problema do registro dinâmico é eliminado. Para isso pode-se usar técnicas que melhoram a latência sacrificando vazão (*throughput*), ou utilizar equipamentos melhores que reduzem o atraso.
- Reduzir o atraso aparente: é uma técnica que utiliza a medição mais recente da orientação da cabeça do usuário no ultimo estágio da renderização do objeto virtual. Para isso renderiza-se o objeto ligeiramente maior que seu tamanho real e espera-se receber a orientação da cabeça do usuário para saber a fração do *frame buffer*[2] para se enviar ao display, já que pequenas mudanças na orientação equivalem a pequenos deslocamentos horizontais e verticais no *frame buffer*.

[2] Memória especializada capaz de armazenar e transferir para a tela os dados de um quadro de imagem completo.

- Combinar fluxos de tempo: em sistemas com *displays* do tipo vídeo, a digitalização da imagem impõe atrasos de tempo na visualização do mundo real pelo usuário.Este atraso pode ser usado para retirar problemas de registro dinâmicos, controlando dinamicamente o atraso no vídeo para equalizar com o atraso do sistema. Dessa forma o vídeo do ambiente real e objetos virtuais estariam na mesma faixa de tempo. O problema dessa técnica é quando o atraso é muito grande, o que deixaria o usuário confuso e desorientado.

- Predizer: este método consiste em medições em tempo de execução da futura posição onde o usuário estará. Dessa forma quando o objeto virtual for renderizado estará na posição correta apesar do atraso do sistema.

Uma visão mais profunda sobre o erro de registro e sua quantificação pode ser vista no trabalho de Holloway [Holloway95].

Alguns outros trabalhos ajudam a enxergar o erro de registro de outra forma, não como um problema a ser totalmente eliminado, mas como uma variável presente em sistemas de RA que pode ser estimada e assim um sistema incerto pode ser modelado, capaz de adaptação em resposta ao problema.

O trabalho de MacIntyre e Julier propõe a estimativa estatística do erro de registro utilizando a expansão ou contração da envoltória convexa[3] (*convex hull*) dos objetos da cena [MacIntyre02]. A partir desta técnica é possível determinar o erro de registro quantitativamente.

Coelho et al. utilizam a técnica citada para desenvolver um *framework* com transformações dinâmicas no grafo de cena, que utiliza a incerteza no rastreamento como razão para se adaptar [Coelho04]. O *framework*, chamado de OSGAR (*Open Scene Graph for Augmented Reality*), possui três componentes principais: um mecanismo de propagação de erro que consegue calcular o erro de registro em qualquer ponto do grafo de cena, um conjunto de componentes para adaptar a cena as condições de incerteza e funções comuns para sistemas de RA como suporte a rastreadores, rastreamento de marcadores, câmeras e *displays*. Dessa forma é possível modelar uma aplicação que se adapte ao erro de registro em tempo de execução.

Uma das técnicas que o *framework* OSGAR pode utilizar na adaptação da aplicação a incerteza é o LOE (L*evel of Error*) [MacIntyre00]. Esta técnica é semelhante à conhecida LOD

[3]O termo designa o limiar do conjunto convexo mínimo contendo um dado conjunto finito e não vazio de pontos no plano. Uma analogia de fácil entendimento é imaginar a envoltória convexa como um elástico posto em volta do objeto, cobrindo e assumindo a forma de sua superfície.

(*Level Of Detail*) da computação gráfica, em que um objeto gráfico é alterado dinamicamente para aumentar ou diminuir seu nível de detalhamento em função da distância do usuário ao objeto, normalmente utilizada para diminuir processamento das placas gráficas. Na LOE o objeto virtual é alterado dinamicamente dependendo da estimativa do erro de registro deste objeto, ao invés da distância entre ele e o usuário. Desta forma, quanto maior o erro de registro que o objeto carrega, mais ou menos detalhes podem ser mostrados para o usuário, na tentativa de melhorar sua compreensão dos objetos aumentados na cena.

Robertson e MacIntyre também utilizam a quantificação do registro, mas usando a técnica de intenção comunicativa, que é o conjunto de objetivos que a RA pretende alcançar, para adicionar informação contextual na cena [Robertson02]. A idéia é utilizar conhecimento semântico da cena para alterar os objetos aumentados através de estratégias clássicas e consagradas da IHC (interface humano-computador) e assim amenizar o erro de registro, ou então tornar clara a intenção dos objetos aumentados para o usuário. Dessa forma o usuário não se sentiria confuso em situações que o erro de registro causa ambiguidade na função dos objetos virtuais.

No mais recente trabalho de Robertson et al., é feito uma avaliação com usuários de sua técnica, em uma tarefa simples em que a RA auxilia o usuário no cumprimento do objetivo [Robertson09]. A avaliação é feita em casos sem nenhum erro de registro, com erro de registro fixo e com erro de registro randômico, em casos com e sem a adição de informação contextual na cena. A conclusão é que a adição de informação contextual para amenizar erros de registro melhorou o trabalho dos usuários causando maior porcentagem de acerto na tarefa proposta.

Por fim, nesta área, existem alguns trabalhos que pretendem avaliar a eficiência do usuário realizando um trabalho com erro de registro constante. Livingston e Ai avaliam três tipos de erro de registro na eficiência do usuário, o ruído, a latência e erros na orientação ou posição do usuário [Livingston08]. A conclusão é que o usuário é resistente a tais erros. Em erros de latência a eficiência diminui, mas o usuário consegue tolerar, enquanto em erros de ruído, os usuários reagiram negativamente, apesar de seu desempenho ter sido o mesma que com erros de latência.

2.3 **Interface do Usuário e Autoria**

Na área de interface do usuário (IU), os desafios de IHC (interação humano-computador)

para a RA são em sua base os mesmo de sistemas tradicionais da computação. Esta área tem trabalhos com diferentes objetivos específicos, mas todos têm como meta geral a melhoria da experiência do usuário do sistema de RA. Alguns projetos tentam facilitar o *design* de interfaces, outros procuram aprimorar a interface e sua usabilidade e outros ainda oferecem uma plataforma para desenvolvimento de aplicativos e suas interfaces.

Como pode ser visto na tabela 1, de acordo com [Zhou08], este campo tem potencial para ser mais intensamente explorado (somente 8,6% dos artigos sobre RA são dedicados a essa área, obtido somando autoria e visualização), mas atualmente o número de pesquisas tem aumentado.

Em relação à criação de interfaces do usuário e aplicações, existem três abordagens principais para auxiliar desenvolvedores: *frameworks*, ferramentas de autoria e linguagens de descrição.

Frameworks cobrem o processo completo de implementação de aplicações. Eles requerem que os desenvolvedores entendam e implementem em um baixo nível de programação, requerendo mais tempo, mas permitindo que a IU seja projetada e controlada em detalhes. Eles normalmente oferecem funções para rastreamento e para gerar interfaces de RA básicas.

O *framework* mais famoso e utilizado é o ARToolkit [Kato99a]. Ele oferece funções prontas para rastreamento de marcadores retangulares e também oferece funções para renderizar objetos 3D (usando o formato X3D) sobre estes marcadores. Embora seja gratuito ele não é atualizado desde 2007, mas recebeu algumas extensões como o ARTag (com melhoria no sistema de rastreamento) ou o ARToolkitplus (voltado para dispositivos móveis). Atualmente a empresa ARToolworks produz uma versão comercial do *framework*.

Liarokapis et al. apresentam um *toolkit* de alto nível voltado para criação de aplicações de RA [Liarokapis04]. O *toolkit* se baseia em uma arquitetura em camadas em que a aplicação se comunica com o *toolkit* e suporta áudio, vídeo e interação do usuário. O *toolkit* se destaca por prover controle da RA na forma audiovisual ao mesmo tempo em que permite interação do usuário.

Já o Studierstube é um *framework* que possibilita construir aplicações distribuídas de RA e também para dispositivos móveis [Schmalstieg02]. Ele possui o OpenTracker, uma camada de abstração para dispositivos de rastreamento que processa entradas multimodais, ou seja, captura a entrada de dados de diferentes dispositivos simultaneamente [Reitmayr01]. O O-

penTracker funciona como um gerenciador de rastreamento, que são explicados na seção 3.

As ferramentas de autoria têm como objetivo desenvolver um *software* ou ferramenta que permita criação e edição de conteúdo de RA através de programação visual.Elas permitem a elaboração de interfaces para o usuário em nível mais alto do que *frameworks*. Tipicamente, elas proporcionam elementos padrões de interfaces implementados que podem ser usados para montar e adaptar uma interface. Para facilitar seu uso o ideal seria que tais *softwares* pudessem ser empregados também por pessoas que não são especialistas em computação. Alguns desses programas serão apresentados neste tópico.

O *software* BuildAR é uma ferramenta para criar cenas de RA [BuildAR08]. Ele é de fácil uso, sendo necessário apenas carregar o modelo do marcador e associar um objeto virtual a este marcador e a cena está pronta. Este *software* também permite alterar a posição, a escala e a orientação do objeto virtual dentro do marcador. Pode ser usado por qualquer um, pois não exige nenhum conhecimento em reconhecimento de padrões e programação.

Outro trabalho nessa área é o ComposAR, voltado para o suporte de interatividade na RA [Seichter08]. Ele suporta tanto programação de conteúdo de RA através da sua interface, quanto através de *scripts* interpretativos em *Python*. Seu principal foco é a possibilidade de associar objetos virtuais com reais e definir interações para esses objetos.

Outra solução é o *software* DART (*toolkit* para *designer* de RA) [MacIntyre04]. Construído sobre o *Macromedia Director*, permite criar aplicações de RA com certo grau de complexidade. Provê suporte de baixo nível para rastreadores, sensores e câmeras e também permite criar aplicações de realidade virtual. O DART chega a oferecer um serviço de gerenciador de localização, mas voltado para realidade virtual.

Linguagens de descrição (ou linguagens de marcação) utilizam linguagens de alto nível de descrição para apresentar a IU em arquivos de texto. Essa abordagem é a mais rápida no *design* de IU, mas também é a que mais limita a manipulação da interface e a extensão de comportamentos e padrões criados.

Um exemplo de linguagem de descrição é a APRIL [Ledermann05], para criação de apresentações de RA e parte do sistema Studierstube. Ela permite que desenvolvedores criem apresentações, da interface até o comportamento (incluindo interações), enquanto oculta a complexidade da implementação e depois executa as apresentações no sistema Studierstube.

Um trabalho mais completo é o de Broll et al., que abordam a criação de uma infraestru-

tura para integração de técnicas de interação e interfaces do usuário para aplicações de RA [Broll05,Broll08]. O foco do trabalho é a programação visual de aplicações e a facilidade em estender interações já existentes.

Como primeiro passo, é apresentado um *framework* de AR/VR em que o foco é o suporte a múltiplos usuários distribuídos compartilhando diversos dispositivos de entrada (para interação com os objetos virtuais) e saída (para visualização). A abordagem usada é a de uma abstração universal de dispositivos integrados. Uma grande vantagem do *framework* é ter seu próprio motor de renderização (*rendering engine*).

O passo seguinte é uma abordagem avançada na prototipação da interação. Esta abordagem prove flexibilidade em tempo de execução e troca a programação (baixo nível) por modelagem (alto nível). Por fim, é definido uma linguagem baseada em XML para configurar técnicas de interação e definir a interface do usuário. Nesta linguagem você define a interface do usuário através de um vocabulário que consiste em:

- elementos de interface comum como *labels*, botões, *sliders* e entrada de texto;
- componentes de container como *frame*, que é usado para agrupar elementos de interface, *sequence*, que pode ser usado para mostrar elementos por um certo período de tempo, *SelectionSet*, que é um conjunto de botões e menu;
- sensores de entrada (*input listeners*) que conseguem captar entrada do usuário. Eles são: sensor de posição que recebe a posição e orientação do usuário, sensor de foco que é ativo quando o objeto está sobre o foco do usuário, sensor de escolha que é ativo quando o objeto é selecionado e sensor de texto que recebe entrada de texto;

Já no campo do *design* de IU, Hollerer et al. propõem três técnicas de *design* de interface do usuário para facilitar a construção de interfaces e melhorar a usabilidade [Hollerer01a]. A primeira técnica é a filtragem de informação, que realiza uma análise de todas as informações disponíveis para a aplicação e exibe apenas as mais relevantes para o usuário naquele momento. Um exemplo seria mostrar somente dados da tarefa do usuário no momento, alterando-os quando a atividade for modificada ou terminada [Julier00].

A segunda técnica é a de *design* de componentes de interface do usuário, possibilitando alterar a interface tendo como base os recursos de visualização da aplicação e a acurácia do rastreamento oferecido pelos sensores. Quando algum desses recursos muda, a aplicação pode

modificar sua interface em resposta. Se, por exemplo, a precisão da localização diminuir e a aplicação não conseguir mais registrar os objetos virtuais corretamente, a interface poderá ser alterada para mostrar somente objetos virtuais afixados ao HUD, os quais não dependem de localização para auxiliar o usuário.

A última técnica é chamada de gerenciamento da visualização (*view management*), que é a tentativa de arranjar os objetos virtuais para serem mostrados no *display* de modo que eles não fiquem próximos demais para serem confundidos. O trabalho de Azuma e Furmanski propõe e avalia vários algoritmos de gerenciamento da visualização com uma análise estatística e um estudo empírico com o usuário [Azuma03].

Kulas et al. dizem que os processos de desenvolvimento de interface formais usados na computação não são possíveis de serem usados na RA e portanto a RA deve ter seus próprios processos de desenvolvimento de interface, assim como suas próprias ferramentas para tal [Kulas04]. Eles propõem que os participantes principais no desenvolvimento da interface do usuário devem ser os seguintes, como pode ser visto na figura9 abaixo:

- programador: implementa a interface do usuário em baixo nível, ou seja, com uma linguagem de programação;
- *designer*3D: modela os objetos 3D (virtuais) que serão inseridos no ambiente aumentado;
- *designer* de Tela: cuida do *layout* da tela, ou seja, dá a posição de cada objeto 2D que aparecerá na tela na melhor disposição para não atrapalhar o usuário;
- *designer* de Interação:determina as interações entre a RA e o usuário, quais interações são possíveis com os objetos aumentados e qual a entrada necessária para as interações;
- engenheiro de Usabilidade: verifica a qualidade da usabilidade da interface. Para isso ele prepara testes para avaliação da interface, conduz tal estudo com o usuário e avalia os resultados;
- usuário: navega pela interface para testá-la em uma situação real de uso;

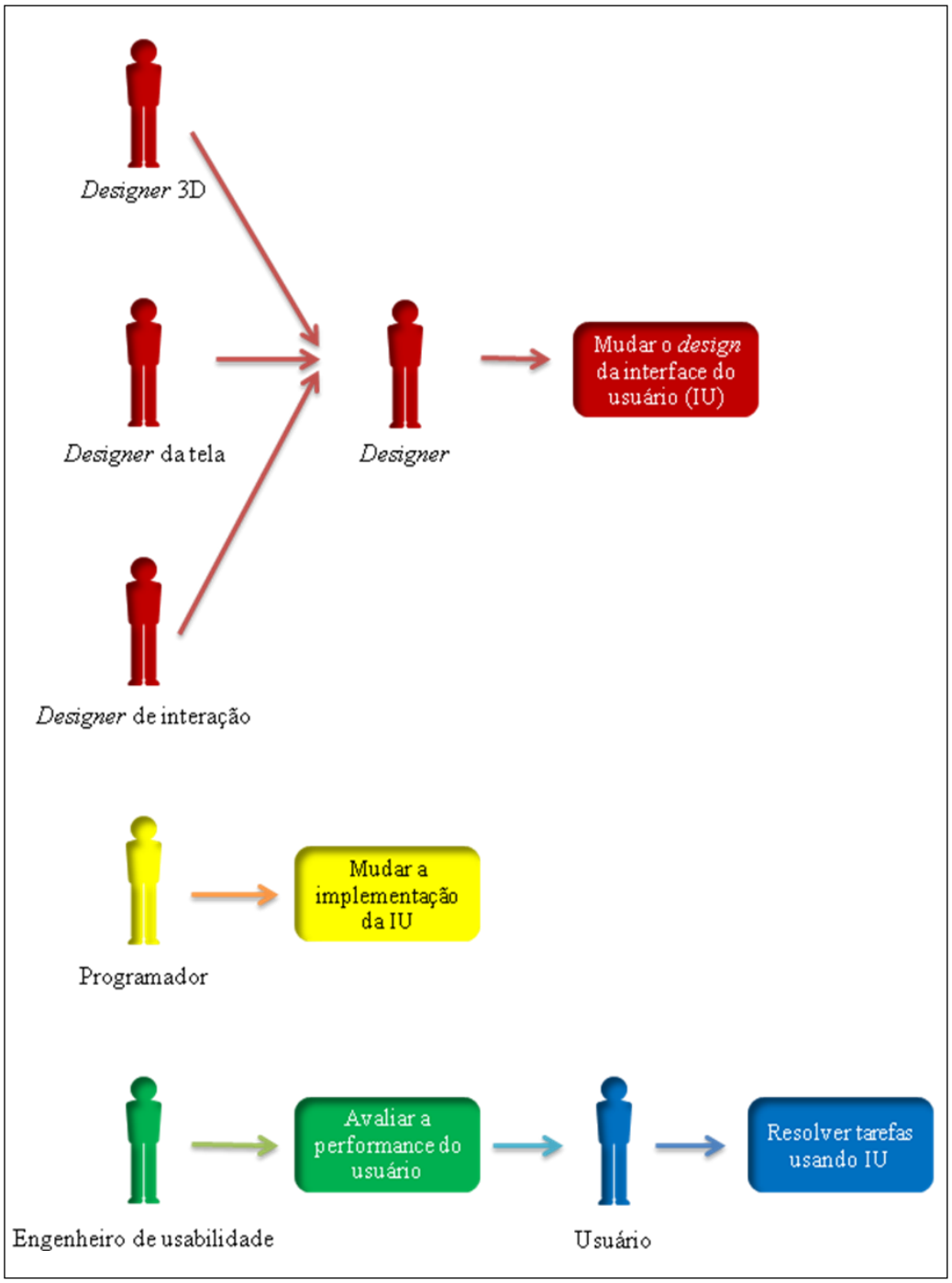

Figura 9: Representação dos envolvidos no processo de construção da interface do usuário (IU) e suas funções, adaptado de [Kulas04].

Vitzthum e Hussmann desenvolveram uma linguagem de modelagem visual para especificação abstrata de aplicações de RA e sua IU, chamada SSIML/AR [Vitzthum06]. Utilizando a linguagem de modelagem SSIML (*Scene Structure and Integration Modelling Language*), que é focada no suporte a integração de conteúdo 3D em aplicações, eles desenvolveram uma linguagem que permite modelar as tarefas do usuário como é feito através do UML na enge-

nharia de software. Essa linguagem também permite modelar a IU de RA e sua conexão com outros componentes (como rastreadores e renderizadores). O resultado passa por um gerador de código automático, gerando um código esqueleto que pode ser usado por programadores diretamente no baixo nível. Dessa forma eles tentam resolver a dificuldade na RA de se criar aplicações complexas com ferramentas de baixo nível, o que demanda tempo, oferecendo uma ferramenta de modelagem estilo UML da aplicação e da IU em conjunto.

Nesta área também existem vários trabalhos que tentam mostrar qual a melhor interface para determinadas tarefas. O trabalho de Petersen et al. é a criação de uma metodologia para uma interface intuitiva chamada interface do usuário natural contínua [Petersen09], uma interface baseada na utilização de gestos para manipulação do conteúdo virtual. Outro trabalho é o de Buchmann et al. que analisam a melhor interface para orientar o usuário na navegação do ambiente, concluindo que a melhor forma de representação é a de uma bússola circular [Buchmann08]. Por fim um último trabalho interessante é o de Liuet al., que avalia a melhor interface de orientação para usuários com problemas cognitivos [Liu06].

2.4 **Arquitetura de Sistemas de RA**

Existem poucas referências às arquiteturas de sistemas de RA. Quando um desenvolvedor de sistemas de RA planeja seu sistema, ele pensa em uma arquitetura sem nenhuma base como referência, somente conhecimento a priori. Reicher et al. com seus dois trabalhos foram os primeiros a oferecer modelos de arquitetura para desenvolvedores de RA [Reicher03a] [Reicher03b].

Nestes trabalhos foram classificadas as qualidades relevantes para arquiteturas de RA com base no nível de prioridade. São qualidades de alta prioridade: latência de rastreamento e renderização, funcionamento com redes sem fio e sem o uso de redes, uso de múltiplos dispositivos de rastreamento, reúso e adição de componentes e capacidade de integração com componentes existentes.

Por outro lado, fazem parte do grupo de baixa prioridade as seguintes qualidades: limitação de carga na CPU, tolerância a falha, segurança, capacidade de reconfiguração em tempo de execução, suporte para entrada de diferentes dispositivos simultaneamente, suporte a múltiplos usuários, capacidade de adaptação às preferências dos usuários e suporte a diferentes plataformas de hardware.

Com as qualidades relevantes listadas, eles desenvolveram uma arquitetura abstrata genérica baseada no modelo MVC (*model-view-controller*) para comparação com outras arquiteturas. O modelo MVC funciona utilizando padrões de arquitetura para auxiliar na construção de um sistema. Ele separa uma arquitetura em subsistemas, sendo eles: códigos de dados e controle e entrada e saída de dados dos usuários. Para esse modelo ser usado na RA, ele foi estendido para adicionar subsistemas de rastreamento, contexto e modelagem do mundo. Com isso os subsistemas classificados para RA são: aplicação (modelo MVC), renderização (visualização MVC), interação (controle MVC), rastreamento, contexto e modelo de mundo.

Nesta arquitetura genérica de subsistemas, padrões de *design* de arquiteturas são utilizados para compor cada subsistema e facilitar a implementação de cada. Estes padrões são estruturas utilizadas com sucesso para resolver problemas conhecidos e que podem ser reutilizados para facilitar o trabalho dos desenvolvedores. Há pelo menos dois tipos de padrões: os que dependem de outros para funcionar e os que são mutuamente exclusivos. Os padrões são mostrados na Figura 10. As próximas subseções descrevem os módulos da referida arquitetura.

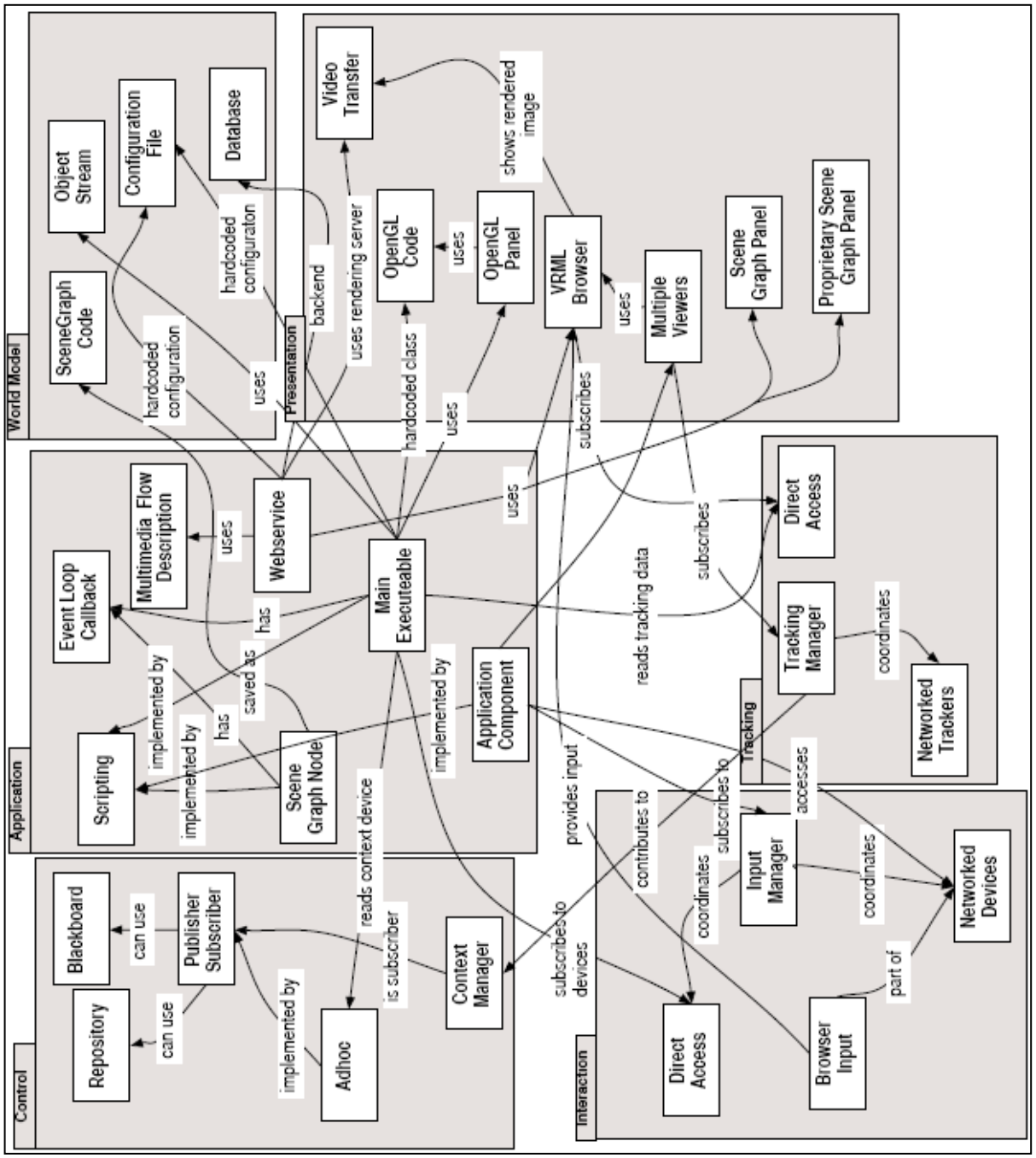

Figura 10: Relação entre padrões de *design* para implementação dos subsistemas de arquitetura de RA [Reicher0b].

2.4.1 Aplicação (*Application*)

Este subsistema é onde os desenvolvedores podem inserir a lógica relacionada à aplicação. Os padrões são:

- executável (*Main Executable*): escreve a aplicação em uma linguagem de programação de alto nível, como C++;

- *scripting*: utiliza um *script wrapper* em torno de todos os componentes que têm restrições de performance;
- nós do grafo de cena (*Scene Graph Node*): modela o mundo em forma de uma árvore de nós, um grafo de cena;
- laço de chamada de eventos (*Event Loop Callback*): provê um gancho que pode ser chamado dentro de um laço de atualização da biblioteca gráfica e que reage a mudanças na cena;
- *webService*: controla o fluxo de dados de um servidor *web* (*Web Server*), publica conteúdo de RA para clientes;
- descrição do fluxo de dados multimídia (*Multimedia Flow Description*): usa uma linguagem de marcação de alto nível para conteúdos específicos, como informação do fluxo de trabalho e para conteúdo de RA;
- componente de aplicação (*Application Component*): encapsula toda a lógica da aplicação em um componente que se comunica com os outros;

2.4.2 Rastreamento (*Tracking*)

Este subsistema é essencial para o funcionamento da RA por garantir o registro. Os padrões,relacionados à obtenção de dados de rastreamento são:

- gerenciador de localização (*Tracking Manager*): transfere todo o processamento de dados de localização para um servidor e somente envia para o cliente o resultado do rastreamento;
- rastreadores via rede: (*Networked Trackers*):provê um *middleware* de acesso para cada dispositivo de rastreamento, deixando a comunicação com eles transparente;
- acesso direto (*Direct Access*): os dispositivos de rastreamento são acessados diretamente pelos seus *drivers*;

2.4.3 Interação (*Interaction*)

A interação com objetos virtuais na RA ainda é um desafio pouco explorado por pesquisadores, embora já existam alguns trabalhos, como o de Petersen et al.[Petersen09]. Os padrões aqui, relativos às formas de entradas de dados e seu gerenciamento são:

- acesso direto (*Direct Access*): adiciona ao código da aplicação código para lidar com dispositivos de entrada, com referência a cada tipo de dispositivo;
- funções de entrada do browser (*Browser Input*): utilizada o envio de eventos do *browser* VRML para capturar cliques em objetos da cena;
- dispositivos de entrada pela rede (*Networked Devices*): provê uma camada de abstração para os dispositivos de entrada e uma descrição de como as entradas do usuário podem ser combinadas. Interpreta esta descrição utilizando um componente de controle;
- gerenciador de entradas (*Input Manager*): coordena os diversos dispositivos de entrada de baixo nível para criar uma entrada de alto nível;

2.4.4 Renderização (*Presentation*)

Este subsistema lida com a apresentação dos objetos virtuais, portanto a maioria dos padrões, listados abaixo, é voltada para a renderização:

- VRML *Browser*: utiliza um *browser* VRML, normalmente na forma de *plugin* de um *web browser*, para visualizar informação 3D;
- OpenGL:emprega o OpenGL para renderização 3D;
- grafo de cena (*Scene Graph Panel*):usa bibliotecas de grafo de cena, como o *Open Scene Graph*;
- grafo de cena proprietário (*Proprietary Scene Graph Panel*):aplica o seu próprio grafo de cena para renderização gráfica em cima do OpenGL;
- transferência de vídeo (*Video Transfer*): um servidor gera vídeo de RA e envia para o cliente visualizar;
- múltiplas classes de visualização (*Multiple Viewers*): provê uma camada de abstração para diferentes tipos de visualizadores (RA, fala, texto, etc.), que podem lidar com diferentes tipos de documentos;

2.4.5 Contexto (*Control*)

Os contextos devem ser capturados, interpretados e entregues aos usuários. Os padrões são:

- Quadro Negro *(Blackboard)*: produtores de informação escrevem no quadro negro, enquanto consumidores lêem as informações, processam e podem escrever novas informações;
- Repositório (*Repository*): componentes que produzem informação contextual escrevem no repositório e componentes interessados em informação contextual lêem os dados lá contidos;
- *Publisher/Subscriber*: provedores de contexto se conectam como *publishers* enquanto consumidores se conectam como *subscribers*;
- *Ad Hoc*:um componente interessado em uma informação consulta diretamente o produtor da informação, ou se registra como *subscriber*;

2.4.6 Modelo do Mundo (*World Model*)

O modelo do mundo é usado para descrever o ambiente em que o sistema está inserido, como a posição de objetos virtuais, objetos de interesse para a aplicação e marcadores. Os padrões são:

- Código OpenGL: o desenvolvedor cria código OpenGL e o chama para renderizar;
- Códigodo grafo de cena (*Scene Graph Code*): através de uma ferramenta de autoria o desenvolvedor cria o modelo da cena virtual;
- *Stream* de objetos (*Stream Object*): o ambiente de execução em tempo de execução (*runtime environment*) permite serializar e desserializar objetos para o disco e do disco;
- Arquivo de configuração (*Configuration File*): carrega um arquivo ao iniciar ou em tempo de execução, que pode conter, por exemplo, a posição dos marcadores;
- Banco de dados (*Database*): ao invés de carregar uma cena particular de um arquivo, o sistema tem acesso a um banco de dados com informações sobre o ambiente;

2.5 Aplicações

É crescente o número de áreas de aplicação de RA, visto que o aprimoramento desta tec-

nologia, bem como a crescente evolução tecnológica de dispositivos de visualização como miniaturização, processamento e mobilidade, contribuíram para a sua popularização na indústria. É comum encontrar aplicações de RA em campos como medicina, indústria, propaganda, robótica e entretenimento. Provavelmente diferentes formas de emprego da RA em novas áreas serão criadas.

Algumas aplicações brasileiras de destaque são o *Presença do Bradesco*[Bradesco09], que funciona no IPhone e indica para o usuário a localização do banco *Bradesco* mais próximo do usuário, ou a aplicação da Tok&Stok [TokStok09], vista na figura 11,em que se pode visualizar como os móveis a serem adquiridos ficariam na casa do comprador.

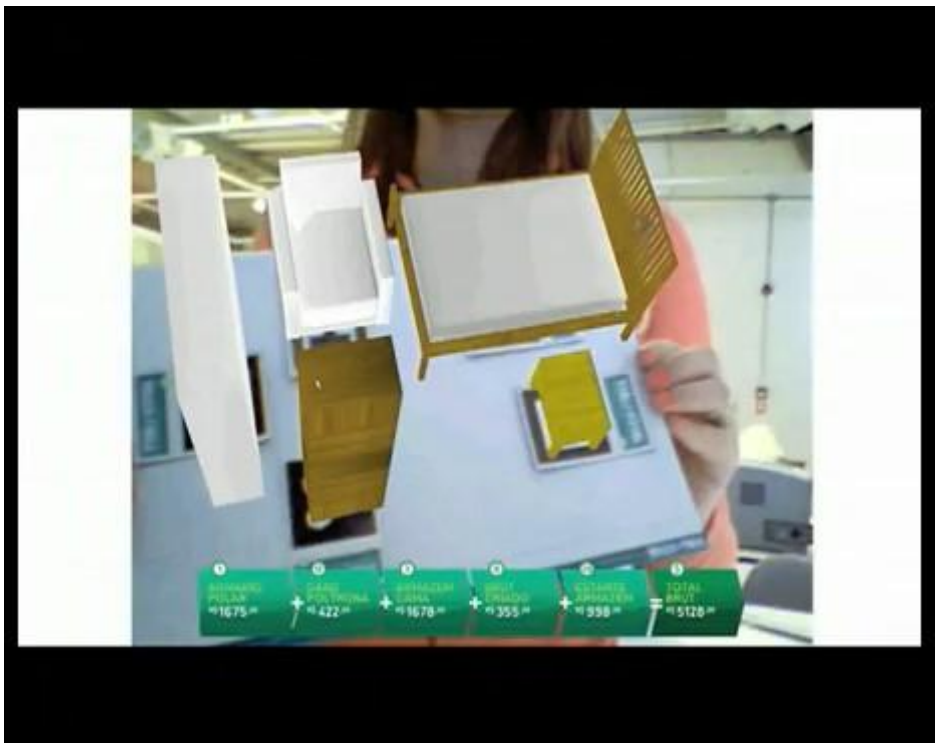

Figura 11: Aplicação da Tok&Stok de RA que ajuda o cliente a visualizar os móveis antes de serem comprados [TokStok09].

Atualmente as áreas de celulares e *handhelds* são as mais providas de aplicações comerciais. Como os atuais aparelhos já têm boas capacidades gráficas e de processamento, câmeras embutidas, GPS, acelerômetro e giroscópio, eles se tornaram a plataforma ideal para desenvolver aplicações de RA de baixo custo e sem a necessidade adicional de investir em equipamentos. Dentre as aplicações existentes, destacam-se: MARA (mo*bile augmented reality applications)* da *Nokia*, *Virtual Santa* da *Metaio* para o IPhone, aplicações da *ARToolworks*, também para o IPhone.

Outra aplicação famosa é o *Wikitude ARTravel Guide*, um guia móvel de abrangência mundial feito para o sistema operacional *Android*, usado no celular *HTC G1* e outros celulares (Figura 6). Usando o GPS do celular para saber a localização do usuário, ele mostra informações sobre pontos de interesse (como prédios, lojas e monumentos) quando estes são filmados pela câmera. São 350 mil pontos de interesse no total espalhados ao redor do mundo e obtidos da Wikipédia e Qype.

Jogos e entretenimento estão em crescimento nesta área, principalmente considerando a realidade mista no geral, onde existem grandes quantidades de aplicações ou *hardwares* vendidos, como o Wii da Nintendo que tem 84.8 milhões de unidades vendidas, ou o Kinect da Microsoft que vendeu 8 milhões de unidades em 60 dias desde seu lançamento. Bons exemplos de jogos de RA é o Ghost Wire, para o Nintendo DSI, ou o quadricóptero Parrot AR Drone, onde é possível batalhar com outros quadricópteros controlando-os através do iPhone e tendo a RA como visualizador das batalhas. Ambos podem ser vistos nas figuras abaixo:

Figura12: Ghost Wire, jogo de RA para o Nintendo DSI [GhostWire11].

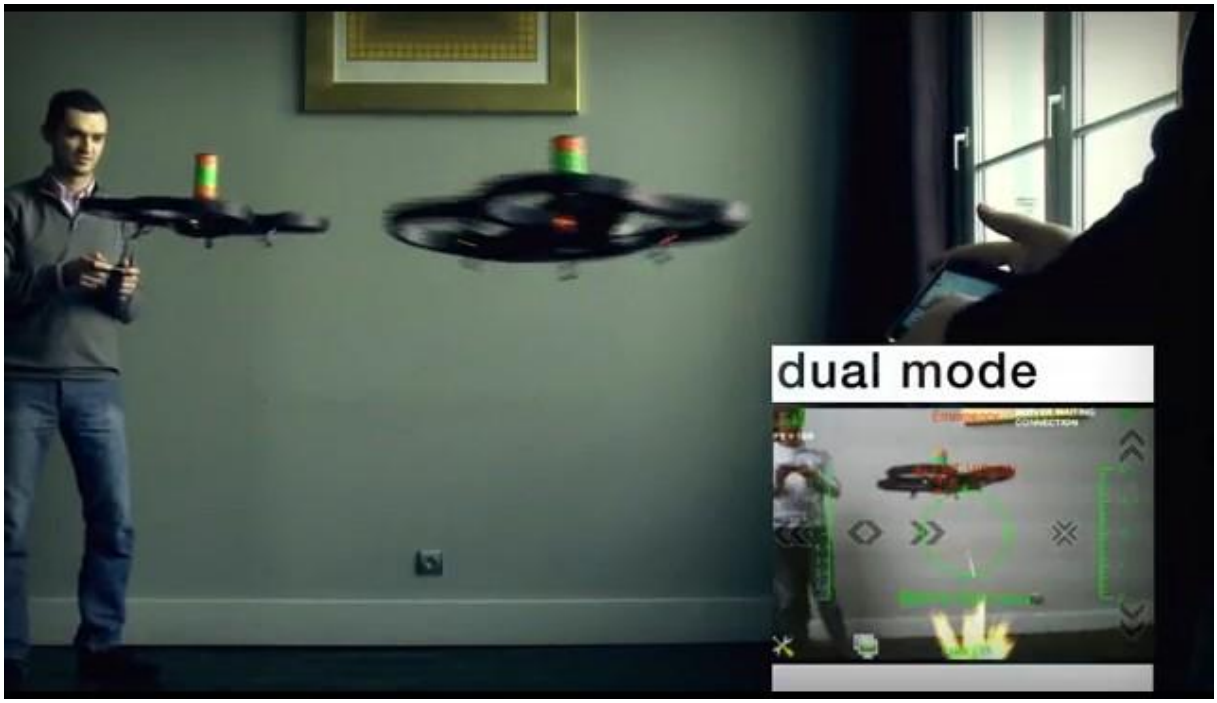

Figura13: Parrot AR Drone, quadricóptero que é controlado pelo iPhone e oferece disputas em Realidade Aumentada [Parrot10].

Algumas aplicações mais complexas e avançadas estão em construção, como o iARM (*Intelligent Augmented Reality Model*) [iARM11], que é um sistema para melhorar o conhecimento do seu redor (*situational awareness*) de um soldado, um projeto financiado pela DARPA (*Defense Advanced Research Projects Agency*) e realizado pela empresa Tanagram.

O sistema iARM suporta geolocalização e triangulação, juntamente com serviços que integram voz, vídeo e imagem para reconhecimento de faces, objetos e padrões. É suportado por um equipamento que combina um processador, redes sem fio criptografadas, câmera e um dispositivo de visualização.

A empresa garante ter resolvido o problema do registro, mas como a patente de sua solução ainda está pendente ela não mostra a solução, mas fez apresentações em eventos, como o ARE2010 (*Augmented Reality Event*).

As figuras abaixo mostram o iARM em funcionamento. O usuário pode marcar o ambiente através de gestos, como por exemplo marcar uma janela como sendo o local onde um atirador inimigo está. Ele vê seus companheiros com uma borda azul e civis com uma borda amarela, ele pode visualizar o mapa 2D da cidade, assim como o mapa 3D do quarteirão e ainda por cima uma equipe de apoio assiste tudo o que está acontecendo em tempo real e pode

coordenar as estratégias das equipes em campo.

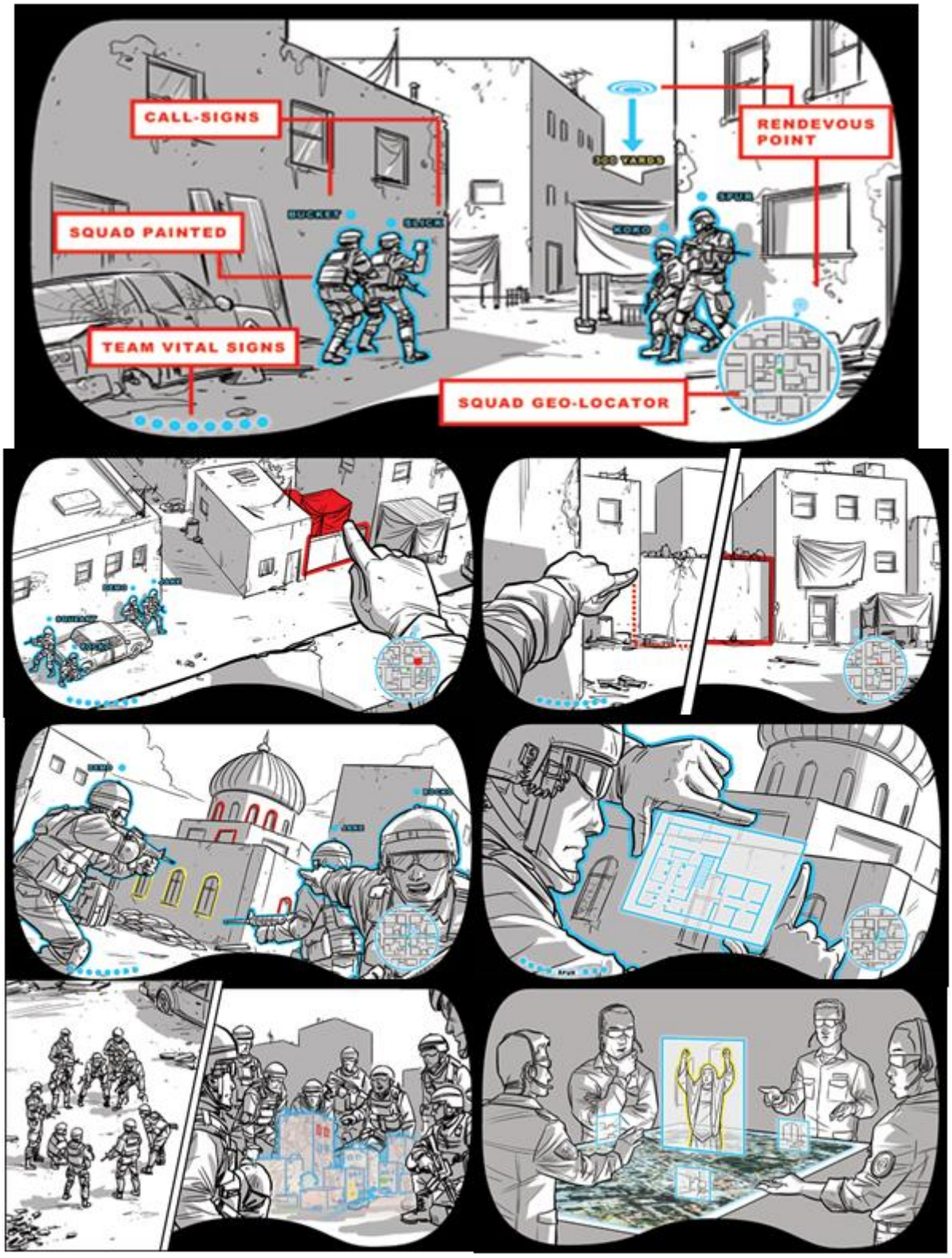

Figure 14: iARM, um sistema para militares aumentarem sua eficiência em campo, capaz de mostrar localização de inimigos e receber comandos dos usuários atraves de gestos e voz [iARM11].

Outra aplicação, mais voltada para o ambiente urbano, é o AR Walker, da empresa NTT Docomo, que mostra para o usuário o caminho para um prédio através de anotações virtuais e ícones registrados com o ambiente real. Seu par de óculos, desenvolvido pela Olympus, pode ser visto na figura abaixo:

Figure 15: AR Walker, para guiar o usuário pela cidade, da empresa NTT Docomo [ARWalker11].

Embora as aplicações mencionadas acima ainda estejam em fase de pesquisa, o aumento de investimentos nesse campo, proporcionado pelo crescente interesse da indústria, pode permitir que esses projetos logo sejam incorporados no cotidiano das pessoas.

2.6 **Discussões dos trabalhos relatados na literatura**

O *framework* OSGAR [Coelho04], que verifica erros de registro estatisticamente em tempo de execução e consegue alterar o grafo de cena dinamicamente para amenizar o erro para o usuário, tem uma eficiente visão de adaptação ao registro em tempo de execução.

Semelhante é o caso do trabalho de Robertson [Robertson02], ao propor a utilização do contexto, empregando somente o erro de registro para, juntamente com estratégias clássicas da IHC de visualização, alterar os objetos aumentados. Dessa forma, o usuário compreende o objetivo da RA nos casos em que o erro de registro pode torná-la confusa e ambígua.

O problema de ambos os projetos pesquisados é que utilizam apenas o erro de registro para a contextualização da aplicação.

Na área de interface do usuário, em relação aos *frameworks*, embora permitam construir

aplicações complexas, não oferecem padrões prontos de interfaces, forçando o desenvolvedor a utilizar linguagens de baixo nível para desenvolver sua interface, tarefa que demanda tempo.

Já os *softwares* de autoria oferecem uma visão ligeiramente diferente de criação de interfaces, mas todos estão limitados à atrelagem de um marcador a um objeto virtual. Eles oferecem elementos de IU de RA muito básicos, faltando uma amplitude maior de padrões de interface. Além disso, ferramentas de autoria, assim como as linguagens de descrição de interface, têm limitações em controlar a aplicação em tempo de execução (normalmente sendo necessário a construção de *scripts* para isso) e embora elas possam ser usadas para criar rapidamente aplicações, não ajudam na criação de aplicações complexas.

Por sua vez, o trabalho de Broll et al. [Broll05] permite a criação de uma interface simples através de um arquivo XML, como o *framework* VISAR. Entretanto, a ausência de uma estrutura mais complexa, como os padrões de interface, não permite ao desenvolvedor uma flexibilidade na criação da interface.

Nenhum dos trabalhos citados nesta seção utiliza, de forma mais profunda, o conceito de padrões de interface e quando são utilizados eles são poucos e simples, não incluem componentes 2D e 3D e não permitem que desenvolvedores construam aplicações além de apresentações (como o APRIL). Já as ferramentas propostas como parte deste trabalho de mestrado (descritas no Capítulo 4), tem como foco a interface do usuário, oferecendo uma solução para *design* e implementação da IU, enquanto permite que desenvolvedores mantenham controle total de suas aplicações.

2.7 **Considerações Finais**

A Realidade Aumentada tem marcado presença cada vez mais constante nos lançamentos de aplicativos no mercado. Esse crescimento encontra paralelo no aumento do número de publicações científicas na área.

Conforme visto neste capítulo, contextos, rastreamento e localização são aspectos chave para a RA. Este capítulo apresentou uma visão geral sobre a área de RA, suas características, desafios, arquiteturas de suporte e aplicações. A próxima seção introduz os conceitos de contexto, ciência de contexto, localização e rastreamento e sua importância para a RA, discutindo questões como RA ciente de contexto ou adaptativa, mapeamento de contextos, localização externa e interna e rastreamento adaptativo.

3 CONTEXTO, CIÊNCIA DE CONTEXTO E LOCALIZAÇÃO

Na RA, é importante saber as informações do ambiente que podem ser úteis para auxiliar o usuário em seus objetivos. Interfaces de RA cientes de contexto ou adaptativas auxiliam usuários a ter uma percepção mais precisa da sua condição atual (ou de outros). Por exemplo, a interface pode se adaptar a tarefa que o usuário está realizando no momento, ou ela pode se adaptar ao nível de erro na localização do usuário para deixar mais claro seu propósito, ou ainda pode adaptar para mostrar alguma situação inesperada que esteja ocorrendo.Estas informações são os contextos do ambiente.

Várias definições de contexto foram criadas, com cada pesquisador da área tendo a sua.Para Schilit e Theimer, é a localização, identificação de pessoas e objetos próximos e mudanças nesses objetos [Schilit94]. Já Brown et al. definem contexto como localização e identidade de pessoas próximas do usuário, adicionando as informações de hora, período e temperatura do ambiente [Brown97]. Dey e Abowd mencionam que o contexto deve ser definido para cada aplicação em particular, considerando tudo o que é importante para a aplicação [Dey00].

Um exemplo de interpretação de contexto é a ocorrência de um incêndio, onde as variáveis envolvidas são temperatura do ambiente, densidade da fumaça do ambiente e taxa de oxigênio no ambiente. Juntas elas podem ser interpretados de forma a possibilitar a descoberta de um incêndio e seu estado (fase inicial, queima livre e queima lenta). Portanto, se capturados e interpretados corretamente os contextos do ambiente em que o usuário está inserido, a qualidade da informação mostrada ao usuário poderá ser de grande valia com a redução de perdas de vidas e de patrimônio.

Para possibilitar este tipo de interpretação de contexto, é necessário um ambiente com infraestrutura, como os prédios inteligentes. Dentro deles, pode haver uma rede de sensores sem fio, capaz de mesurar variados fenômenos físicos. Aliado a isso pode haver equipamentos de

rastreamento no prédio e como o seu modelo 3D (CAD)está disponível [Christiansson00], esses ambientes são naturalmente apropriados para utilização da Realidade Aumentada.

Na RA, o contexto vem sendo usado por algumas aplicações mais recentes [Hollerer01a] [Robertson02] [Coelho04] [Lee08] [Oh09], mas ainda é pouco explorado. Como parte deste trabalho de mestrado foram catalogados os contextos mais importantes para a RA, a saber: localização, iluminação e usuário.

O primeiro se refere ao rastreamento do usuário e de objetos de interesse para a aplicação. Este contexto é necessário para a realização do registro de objetos virtuais no ambiente real. Algumas vão mais longe e utilizam este contexto para modelar uma aplicação que possa se adaptar em relação a sua precisão.

O segundo, iluminação, indica a irradiação da luz no local, como, por exemplo, claridade, luminosidade, brilho. Sua influência principal ocorre em dois pontos: na tela de visualização de dispositivos, ou quando a visão computacional é usada para rastreamento.

O usuário é o terceiro dos contextos de adaptação e se refere a quem está usando o sistema. A identificação do usuário pode gerar RA personalizada, mostrando, por exemplo, somente informações relacionadas a tarefa que o usuário está realizando no momento (e com suas preferências de visualização).

3.1 **Mapeamento de Contextos**

O mapeamento aqui mostrado representa os contextos de uma maneira abstrata, ainda não convertidos para uma linguagem que possa ser interpretada pela máquina, ou seja, ainda não é a representação do contexto.

Poucos mapeamentos são vistos na literatura, sendo que o interesse maior da área gira em torno de [Baldauf07]:

- modelagem dos contextos: forma como os contextos são representados em uma forma processável, sendo que os principais modelos são [Strang04]: chave e valor, linguagens de marcação, gráfico (UML), orientado a objeto, baseado em lógica e baseado em ontologias;
- arquiteturas de implementação de sistemas cientes de contexto: existem várias abordagens. Para adquirir informação contextual pode-se acessar diretamente o sensor, usar uma infraestrutura de *middleware*, ou um servidor de contexto

[Chen04]. Os modelos de gerenciamento de contexto podem ser *widgets*, serviços de rede ou modelo quadro-negro (*blackboard*) [Winograd01]. Os sensores podem ser classificados, de acordo com a maneira como capturam dados, em físicos, virtuais ou lógicos [Indulska03].

Um mapeamento é demonstrado por Korpipääet al. no trabalho sobre um *framework* para gerenciar contextos [Korpipää03]. O mapeamento, que pode ser visto na figura 16, do contexto de alto nível *Outdoors* (ambiente externo), é feito em níveis, em que cada átomo é a probabilidade de acontecer seu valor chave. Portanto, quando se toma como exemplo a intensidade da luz (Environment->Light->Intensity), tem-se que a probabilidade de ela ser escura é de zero (dark->0), a probabilidade de ela ser normal é de 40% (Normal->0,4) e a probabilidade dela ser bem clara é de 60% (Bright->0,6).

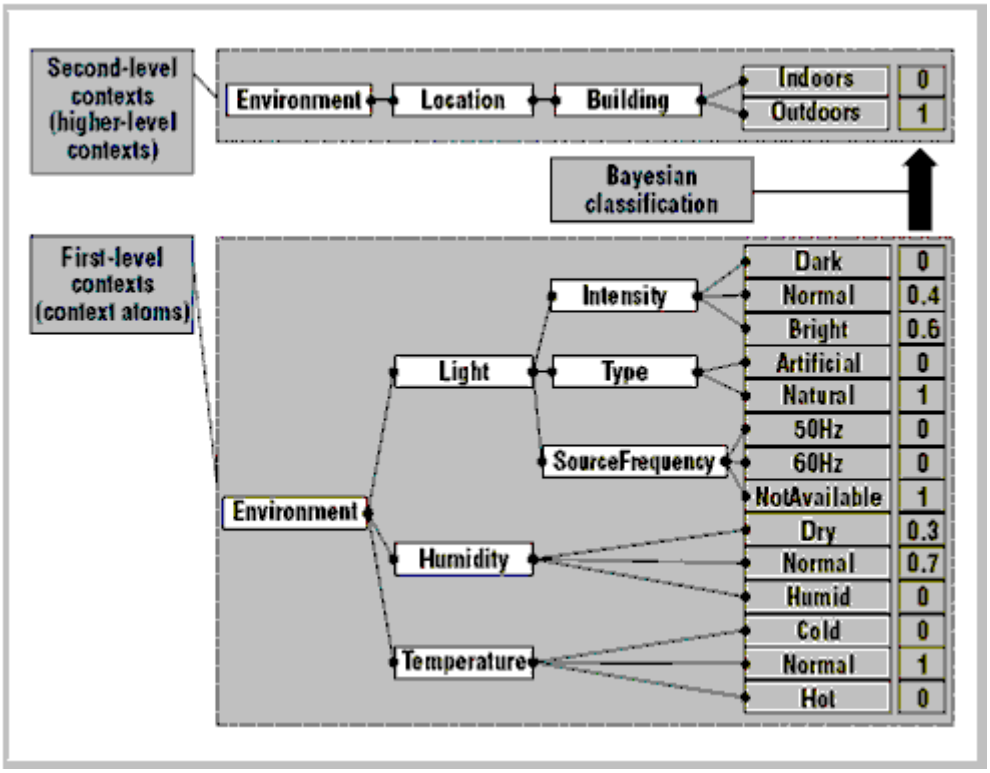

Figura 16: Exemplo de mapeamento de contexto, para o contexto de alto nível *Outdoors* [Korpipää03].

Já na figura17 pode ser visto a divisão utilizada no WINDIS dos contextos (somente no campo do monitoramento a emergência) e as variáveis, ou valores, relacionados a cada contexto. O mapeamento dos contextos para o Sistema de RA para Ambientes de Emergência é mostrado na tabela 3no anexo II.

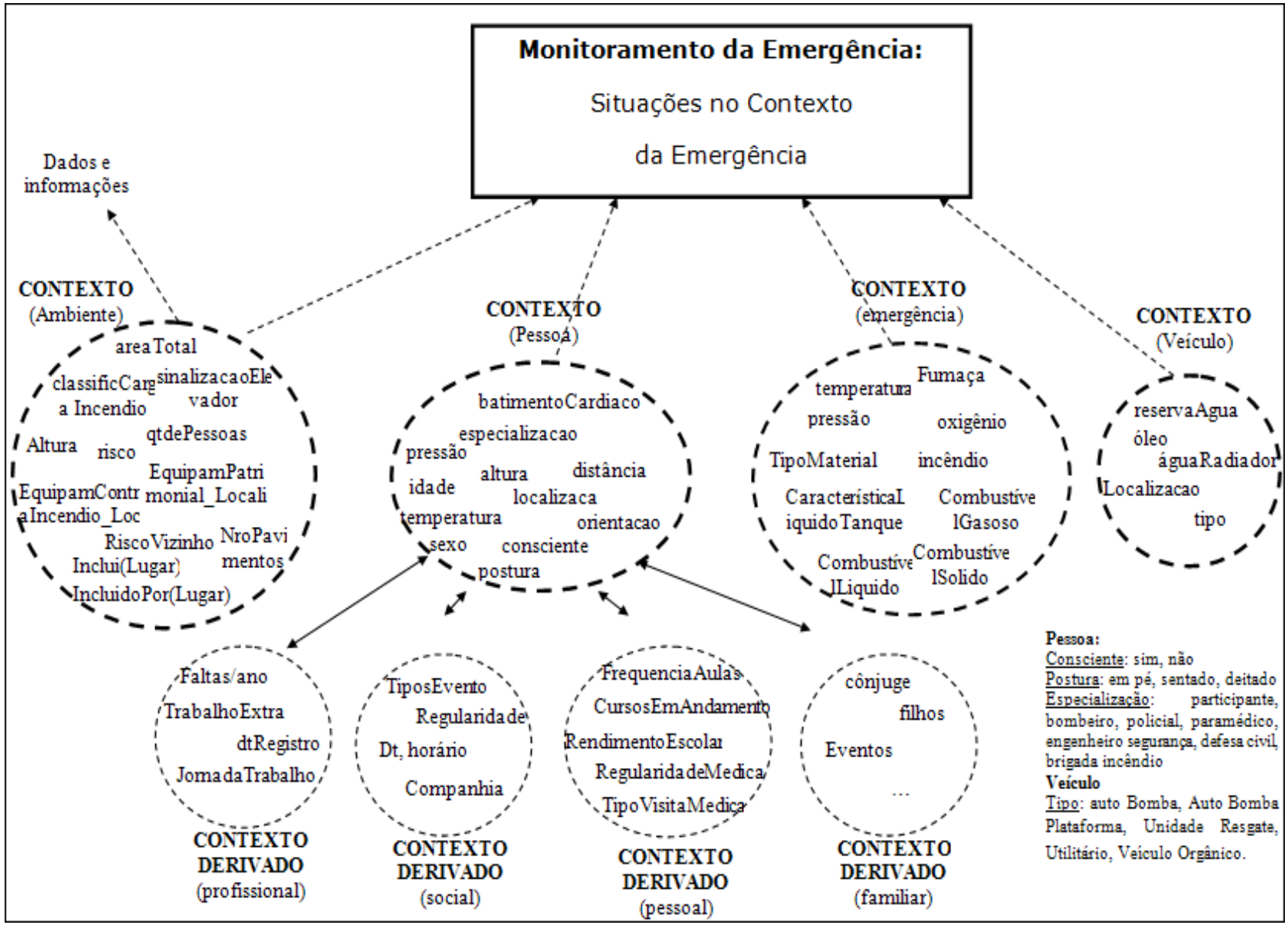

Figura 17: Divisão dos contextos na classe de aplicação Emergência.

3.2 **RA Adaptativa**

RA adaptativa é a área de trabalhos na qual a aplicação se adapta a contextos. É uma área recente e ainda pouco explorada da RA, em que a maior parte dos trabalhos foca em métodos ou arquiteturas para adaptar a interface do usuário.

Dentre os projetos dedicados a esta área, um dos destaques é o de Hollerer et al. [Hollerer01a], sobre interfaces do usuário, no qual são desenvolvidas três técnicas eficientes de adaptação: a filtragem de informação, a adaptação da interface sobre sistemas de rastreamento e o gerenciamento da visão.

A técnica de filtragem da informação, primeiramente mostrada por Julier et al.[Julier00], foi um dos passos iniciais dados na área de RA adaptativa. Como já explicado, a metodologia consiste em reunir as informações disponíveis para a aplicação e mostrar para o usuário somente o necessário para a execução da tarefa atual. Desta forma, é necessário analisar nesta ordem: quem é o usuário, qual tarefa está sendo realizada no momento, quais as melhores informações para auxiliar o usuário na tarefa e como mostrar estas informações na interface.

Outro procedimento é a adaptação da interface sobre diferentes sistemas de rastreamento. Para isso, analisam-se os atuais sistemas de rastreamento disponíveis e sua capacidade de gerenciar informação precisa e, a partir disso, adapta-se a interface à precisão da localização. Desta forma, com diferentes sistemas de rastreamento, é possível ter diferentes formas de apresentação de informação. O exemplo pode ser visto na figura 23, onde duas interfaces da mesma aplicação estão sendo mostradas, mas cada qual é usada quando determinado sistema de rastreamento está atuando.

Essa técnica foi ampliada por Coelho et al. [Coelho04] e Robertson e MacIntyre [Robertson02]. Os primeiros criaram o *framework* OSGAR, que permite analisar o erro de registro em tempo de execução e adaptar a interface ao erro através de alterações no grafo de cena. Isso tornou desnecessário ter conhecimento prévio sobre os sistemas de rastreamento utilizados.

Já Robertson e MacIntyre se concentraram em quais adaptações são necessárias no grafo de cena para uma compreensão melhor da interface pelo usuário. Cada interface de RA tem o objetivo de auxiliar o usuário ou disponibilizar uma informação, então eles usaram conhecimento semântico da cena de modo a mostrar ao usuário a intenção da RA, alterando a interface com a variação do erro de registro. Eles propõem a utilização do erro de registro, junta-

mente com estratégias de visualização clássicas da IHC, para alterar objetos aumentados, de um modo que o usuário compreenda o objetivo da RA em casos onde o erro de registro a torna confusa ou ambígua. O artigo mais recente de Robertson et al. [Robertson09], faz uma extensa avaliação dos métodos com usuários e conclui que o método ajuda o usuário a entender cenas que estão ambíguas devido ao erro de registro.

A última técnica, de gerenciamento da visualização, visa organizar os objetos virtuais de modo que eles se mantenham próximos do objeto real que referenciam, mas distante de outros objetos virtuais, prevenindo que objetos virtuais se sobreponham ou confundam o usuário [Bell01]. Vários algoritmos para gerenciamento de visualização já foram apresentados e analisados[Azuma03].

Por sua vez Mendez et al. [Mendez06] utilizam a técnica Lentes Mágicas [Matthew96] para alterar dinamicamente as cenas a partir de informação contextual, mostrando diversas visualizações possíveis para um objeto ou diversos contextos, como eles chamam.Um exemplo disso é a filtragem de informações a cada lente utilizada.

Em outro trabalho, Mendez et al. propõem um *framework* que faz uso de estilos de visualização e marcadores de contexto, no qual um usuário pode selecionar um marcador de contexto durante a execução da aplicação, ativando um estilo de visualização que altera todos os objetos do contexto selecionado [Mendez07].

Isto é feito adicionando informação contextual ao grafo de cena, o que permite à aplicação utilizar esta técnica sem necessidade de conhecimento prévio dos objetos de cena visualizados. Um exemplo seria, em um carro aumentado, escolher o contexto “objetos que representam perigo” e a aplicação mostraria o motor do carro com o estilo de visualização atribuído a tal contexto, pois o motor tem temperatura e pressão alta e, portanto foi classificado como pertencente a tal contexto.

Lee com Woo [Lee08] e Oh com Woo [Oh09] apresentam focos diferente. Lee e Woo desenvolveram um *framework* ciente de contexto, o qual usa o contexto em dois níveis de processamento, alto e baixo, com o intuito de ajudar as aplicações de RA. No nível baixo o exemplo citado é um detector de iluminação do ambiente para compensar mudanças na iluminação para a aplicação, que assim consegue usar o reconhecimento de marcadores sobre condições que outras aplicações não conseguiriam. No alto nível ele detecta o usuário corrente da aplicação e usa preferências dele para gerar RA personalizada.

Oh e Woo utilizam o mesmo trabalho, mas voltado a RA móvel, criando a técnica CA-MAR (RA móvel ciente de contexto) [Oh09].

Os projetos da área são basicamente voltados aos contextos de localização, iluminação e usuário, pois estes são os contextos mais utilizados na RA.

3.3 **Rastreamento e Localização**

Rastreamento e localização são a base da RA. Somente através de um bom sistema de rastreamento é possível gerar uma RA registrada, sendo que a RA se torna inconveniente sem o alinhamento entre os objetos virtuais com o ambiente real, podendo ser arriscado seu uso em sistemas em que a vida do usuário depende da sua eficácia.

A localização é um termo que se refere a posição de algo em um espaço físico. Já o rastreamento consiste em se obter a posição do usuário no ambiente em que está inserido e a posição dos objetos de importância para o sistema. É essencial para geração da Realidade Aumentada uma forma de rastreamento precisa e que possa mapear o ambiente para o sistema. Se o rastreamento contiver erros e inexatidões, os objetos virtuais aumentados serão posicionados incorretamente.

As técnicas de localização e rastreamento de usuários são importantes para viabilização de várias tecnologias, como a computação ubíqua ou Realidade Aumentada. Em ambientes internos o rastreamento é mais complexo, pois poucas das técnicas citadas a seguir são eficazes e de baixo custo, visto que é sempre necessário que previamente haja uma infraestrutura no ambiente. Já em ambientes externos o GPS é o grande destaque e a tecnologia mais utilizada. Afinal, apesar de ele possuir certa margem de erro (5 – 15 metros), consegue rastrear a posição do usuário em qualquer lugar do globo terrestre.

Na área de localização e rastreamento, as maiores dificuldades são lidar com ambientes não preparados previamente ou sem estrutura e com condições de operação imprevisíveis (como a possibilidade do sol de ofuscar a câmera usada para reconhecimento de padrões) [Azuma99].

A base das duas possibilidades de rastreamento utilizados na RA são os sensores e a visão. No caso dos primeiros, empregam-se sensores de onda (*Wi-Fi* ou GPS), magnéticos (utilizam o campo magnético), acústicos (ultra-som), óticos (*laser*), inerciais (giroscópio e acelerômetro) e mecânicos.

Por outro lado, a técnica baseada em visão computacional usa câmeras para filmar o ambiente e algoritmos de reconhecimentos de padrões e processamento de imagens para analisar a imagem e, assim, capturar a posição dos objetos de interesse.

A maioria dos sistemas de RA utiliza uma combinação das estratégias, para juntar as vantagens de cada área e ter robusteza e resistência a falhas. A visão computacional, por exemplo, tem uma ótima precisão, consegue detectar com erros baixíssimos a posição do usuário, mas perde sua capacidade quando trabalhando com movimentos ligeiros. Tal fraqueza pode ser compensada com uma tecnologia inercial para prever a futura posição do usuário em rápidas movimentações.

A tabela 2 mostra uma comparação das tecnologias de localização. Nesta tabela os nomes das tecnologias utilizadas são listados na primeira coluna. Na segunda coluna tem-se a distância que pode ser abrangida por cada metodologia. O tempo que se leva para montar o equipamento necessário para o funcionamento é registrado na terceira coluna. Na quarta coluna é arrolado o alcance de cada técnica, ou sua granularidade, o que influencia na precisão obtida. A quinta coluna trabalha o tempo de funcionamento da tecnologia, após o qual as medições têm um acúmulo de erro muito grande e por isso não podem mais ser confiadas. Finalmente,na sexta coluna está o ambiente em que a tecnologia funciona, podendo ser interno ou externo (*indoor/outdoor*).

Tabela 2: tabela com o estudo sobre as tecnologias de localização, adaptada [DiVerdi07].

Tecnologia	Distância (m)	Montagem (horas)	Resolução (mm)	Tempo (s) até acúmulo de erro	Ambiente
Magnética	1	1	1	∞	Interno Externo
Ultra-Som	10	1	10	∞	Interno
Inercial	1	0	1	10	Interno Externo
Pedômetro	1000	0	100	1000	Interno Externo
UWB	100	10	500	∞	Interno

Óptico	Usuário é o marcador	10	10	10	∞	Interno
	Com marcadores	10	0	10	∞	Interno Externo
	Sem marcadores	50	0-1	10	∞	Interno Externo
Híbrido		10	10	1	∞	Interno
GPS		∞	0	1000	∞	Externo
Wi-Fi		100	10	1000	∞	Interno Externo

3.3.1 Rastreamento Externo

O rastreamento externo é normalmente mais complexo de ser executado e planejado do que o interno, pois é difícil ter uma infraestrutura instalada, o que poderia limitar a capacidade de rastreamento.

Tendo em mente a dificuldade de utilizar uma infraestrutura, nota-se que tecnologias para rastreamento externo devem ser as mais abrangentes possíveis, cobrindo grandes áreas. Por isso, as metodologias que se destacaram são as baseadas em satélites (GPS) ou telefonia móvel (GSM, UMTS). Atualmente a tecnologia mais utilizada é o GPS, devido ao fato de sua principal função ser a de rastreamento. Porém essa ferramenta, por si só, não comporta a precisão necessária para grande parte das aplicações de RA. Para corrigir essa deficiência, vários trabalhos surgiram nesta última década com a temática de formas de melhoramento da precisão do GPS, ou junção dele com outras tecnologias.

Azuma et al. descrevem quatro técnicas para lidar com o rastreamento em ambientes externos despreparados (sem infraestrutura) [Azuma99]:

1. Duas delas são para corrigir os efeitos de erros na orientação do usuário. De acordo com Azuma, os maiores erros na orientação são causados pela latência ou pela distorção dos sensores. Em vista disso, a primeira técnica proposta é baseada no uso de uma bússola (permitindo a calibração) e de um giroscópio de três eixos (estabilizando os objetos aumentados em situação de rápidos movimentos do usuário);
2. A segunda é um aditamento da primeira, utilizando os recursos anteriores somados à visão computacional. O uso da bússola e giroscópio fornece a orientação para o sis-

tema, o que reduz o processamento exigido pelo algoritmo de processamento de imagem.Além disso, quando o usuário para de se movimentar, o algoritmo reconhece locais do ambiente e corrige o erro acumulado pelos sensores inerciais (bússola e giroscópio). Isto aumenta a precisão da RA, como pode ser visto na figura 18, onde os objetos azuis foram gerados pela primeira técnica e os objetos vermelhos foram gerados pela segunda;

3. A terceira técnica tem o objetivo de estimar a direção da movimentação do usuário por meio do emprego da projeção panorâmica para reduzir o erro causado pela projeção planar no cálculo da translação por algoritmos de processamento de imagens. Uma simulação, mostrada na figura 18, demonstra que a projeção panorâmica tem precisão superior.

4. A quarta técnica consiste na conjunção de dois métodos de cálculo da posição do usuário através de processamento de imagens, o que melhora significativamente a performance da calibração dinâmica, possibilitando rastreamento em seis graus de liberdade (6DOF). Um método consiste na média robusta da solução de 3 pontos e o outro é baseado em um filtro de Kalman estendido iterativo e um filtro de uma restrição por vez (*single constraint at a time* - SCAAT).Para mais detalhes sobre os métodos conferir Park [Park99b].

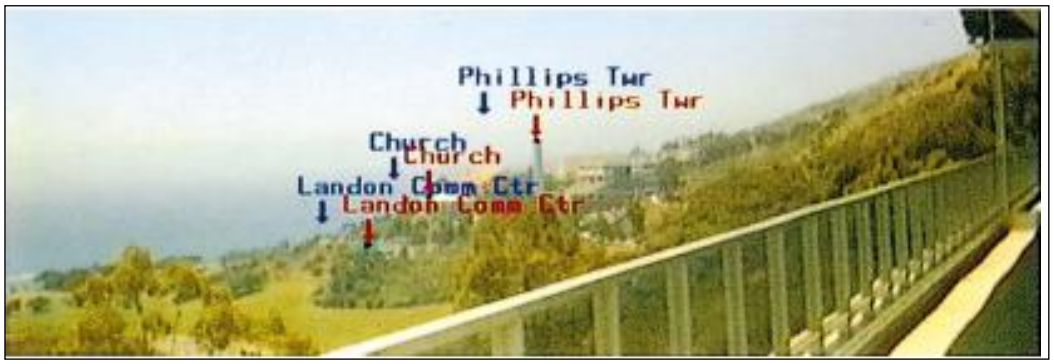

Figura 18: Diferença entre a RA gerada com o uso da visão computacional como auxiliar (vermelha) e sem ela (azul) [Azuma99].

Azuma et al. recentemente desenvolveram uma nova técnica que usa faróis (*beacons*) infravermelhos para auxiliar na localização em situações mais restritas [Azuma06], para uso em ambientes em que não há como garantir o uso de algoritmos de processamento de imagens, devido, por exemplo, a pouca luminosidade.

A condição para uso da técnica é que, a 100 metros de distância, os objetos aumentados

apareçam a 1 metro de sua verdadeira localização. É um erro normalmente considerado alto, mas aceitável para a aplicação desejada (militar). A técnica consiste em utilizar o mesmo sistema descrito anteriormente, com GPS, bússola e giroscópio, mais uma câmera infravermelha para rastrear os faróis infravermelhos, que são móveis, posicionados em veículos aéreos de miniatura não tripulados (*miniature unmanned air vehicles* -MAVs).Idéias parecidas já foram utilizadas anteriormente, mas esta conta com um diferencial: o uso de faróis em pequena quantidade (menos de dez) e móveis, o que amplia o raio de abrangência dessa prática.

Um trabalho mais atual da área é o de Schall et al., que promove um sistema para rastreamento global utilizando fusão de vários sensores [Schall09]. Um filtro de Kalman é usado para fusão de GPS Diferencial (DGPS) ou GPS cinemático em tempo de execução (*real-time kinematic* - RTK) com alturas barométricas e sensores inerciais (no caso giroscópio e acelerômetro) e um magnetômetro.

Como os sensores inerciais estão sujeitos a erro acumulado e o magnetômetro a campos eletromagnéticos, um algoritmo de processamento de imagens livre de acúmulo de erro também é aplicado.O algoritmo para rastreamento visual realiza aprendizado durante a execução, das características naturais do local, o que lhe permite produzir um mapeamento do ambiente, e, com isso, detectar e corrigir desvios no sistema de orientação.Desse modo o sistema melhora a precisão e a robustez do rastreamento, possibilitando que a RA tenha baixo erro de registro ao ser utilizada em um ambiente despreparado e externo.

3.3.2 Rastreamento Interno

O rastreamento interno difere do externo por causa da impossibilidade de utilizar o GPS e da necessidade de precisão muito maior para garantir o registro dos objetos virtuais. Para isso, normalmente é imperioso que o sistema tenha o modelo 3D do ambiente em que o usuário se encontra (podendo ser também uma planta baixa do local) ou que a utilização da RA fique restrita a um pequeno espaço. Com essas necessidades diferentes, ainda não foi encontrado um método eficaz, capaz de elidir a obrigação de haver previamente no local uma infraestrutura que garanta a localização. Os métodos de rastreamento interno são:

- câmera com marcadores (fiduciais): é o procedimento mais utilizado por ser o mais robusto e exigir menor capacidade de processamento. Suas desvantagens são: requerer manutenção, ter alcance limitado e só funcionar quando o marcador estiver no campo de visão da câmera. Park et al. criaram um artifício de detecção

de características naturais do ambiente, construindo um modelo do ambiente a partir das características detectadas, como pontos, linhas, bordas e texturas. Isso permite que seja possível continuar a oferecer o rastreamento baseado no modelo desenvolvido ainda quando o marcador não está mais visível [Park99a]. Um dos primeiros trabalhos a usar um HMD com marcadores foi o de Kato e Billinghurst[Kato99b];

- câmera baseada em modelos: este método se baseia no uso de modelos das características dos objetos rastreados (modelo CAD ou 2D) para identificá-los. Geralmente os modelos são construídos baseados em linhas ou bordas, sendo que as últimas são mais usadas por exigirem menos processamento e serem mais robustas a mudanças na iluminação. A abordagem mais popular é utilizar gradientes fortes da imagem para uma estimação inicial da posição do objeto [Fua07]. O rastreamento de múltiplos objetos 3D não havia sido endereçado até recentemente, quando Park et al. resolveram o problema da escalabilidade, considerando uma taxa de *frames* razoável,ao criar um processo para rastrear simultaneamente múltiplos objetos 3D em tempo de execução [Park08]. Entretanto esta abordagem ainda não é totalmente escalável e é limitada a uma base de dados de pouco mais de 30 objetos;

- SLAM (*Simultaneous localization and mapping*): é uma metodologia para rastreamento que não exige nenhum conhecimento *a priori* ou mapeamento do local. Ela utiliza bordas de modelos de imagens sequenciais para construir um mapa do local. Klein e Murray apresentam um procedimento voltado para RA em *handhelds* em um pequeno espaço [Klein07,Klein09]. Apresentando o sistema chamado de PTAM (*Parallel Tracking and Mapping*), eles propõem separar a execução em dois processos executando em paralelos, um voltado para o rastreamento robusto dos movimentos do usuário e outro, para o mapeamento 3D de pontos chave usando *frames* de vídeo já observados. O resultado, que pode ser visto na figura 19, produz um mapa detalhado com centenas de marcações que podem ser rastreadas em tempo de execução, com acurácia e robustez de algoritmos baseados em modelos;

- câmera baseada em LEDs infravermelhos: este rastreamento utiliza os mesmos algoritmos dos marcadores, com a diferença que coloca-se um filtro infraverme-

lho na câmera e assim toda luz ambiente fica invisível, enquanto somente os LEDs são vistos. Dessa forma a iluminação não causa nenhuma interferência no rastreamento. Naimark e Foxlin introduzem uma técnica para codificar os LEDs de modo que não seja necessário sincronização entre eles e a câmera [Naimark05]. Através do uso de código da modulação da amplitude em vez de código de piscagem binário, o LED fica sempre ligado e portanto pode ser rastreado a qualquer momento;

- RFID: o trabalho de Beckeret al. mostra que é possível utilizar RFIDs para gerar RA com baixo erro de registro [Becker08]. Sua infraestrutura emprega um procedimento de localização do usuário baseado nas coordenadas do comunicador RFID, detectada pelo leitor de RFID e tem a orientação fundamentada em um giroscópio. O erro médio é de menos de meio metro;

- UWB: a utilização do UWB pode ser feita através do sistema de localização comercial Ubisense [Steggles07], aplicando emissores e receptores do sinal UWB para captar a posição do usuário, através de uma combinação das técnicas de tempo de diferença da chegada do sinal (TDOA) e ângulo da chegada do sinal (AOA). Somente dois leitores são necessários para calcular a posição 3D do usuário, com uma precisão de 15cm;

- magnetismo: os rastreadores magnéticos são sujeitos a erros e *jitter*. Um sistema não calibrado, na presença de distorções do campo magnético, pode apresentar erro de 10cm ou mais. Apesar disso são possantes e não atrapalham a movimentação do usuário. Por isso normalmente são usados em conjunto com outras tecnologias. State et al. apresenta um sistema que utiliza o magnetismo como base, por ser mais robusto e o reconhecimento de marcadores para garantir maior precisão [State96];

- *Wi-Fi*: esta tecnologia de transmissão de dados pode ser utilizada através da medição da força do sinal (RSSI) e triangulação da posição do usuário. O trabalho de Peternier et al. utilizam o *Wi-Fi* para identificar aproximadamente a posição do usuário (erros de 3 a 5 metros) utilizando um PDA [Peternier06];

- reconhecimento de rostos e gestos: uma forma de rastreamento bem comum na RA, principalmente para capturar entrada de dados de usuário (gestos) ou sua i-

dentidade (rostos).

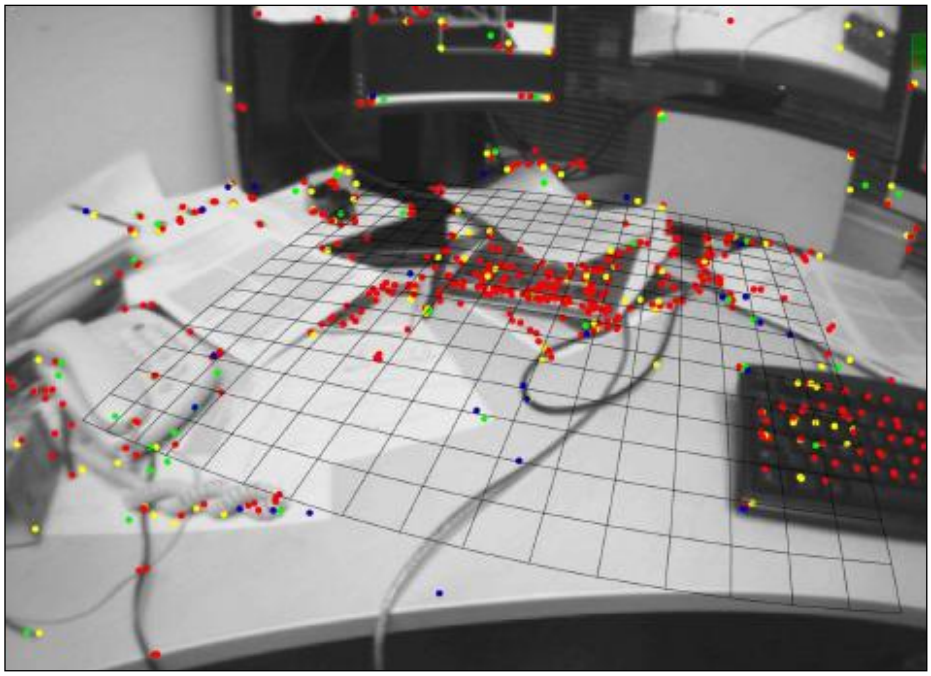

Figura 19: Exemplo de uso do SLAM em que o sistema rastreou 660 pontos chave em 18ms [Klein07].

A maioria dos sistemas hoje utiliza uma combinação de várias tecnologias para garantir o registro da RA. Um exemplo é o trabalho de Newman et al., que utiliza o UWB com o Ubisense, uma câmera para rastrear marcadores e um rastreador inercial para estimativas da orientação do usuário [Newman06].

Além disso o rastreamento interno se popularizou graças aos fabricantes de vídeo games e sua recente entrada nesta área. Os vídeo games Wii, Playstation 3 e Xbox 360 conseguiram transformar as tecnologias de rastreamento em produtos de alta vendagem, com suas respectivas tecnologias WiiMotion, PSmove e Kinect, vistos nas figuras 20, 21 e 22. As duas primeiras se baseiam no rastreamento por câmeras baseadas em LEDs infravermelhos, capazes de capturar os movimentos realizados pelo usuário através do controle, enquanto o Kinect realiza reconhecimento de faces, gestos e voz como sua interface de entrada.

Figura20: Kinect, para o Xbox 360, da Microsoft [Kinect10].

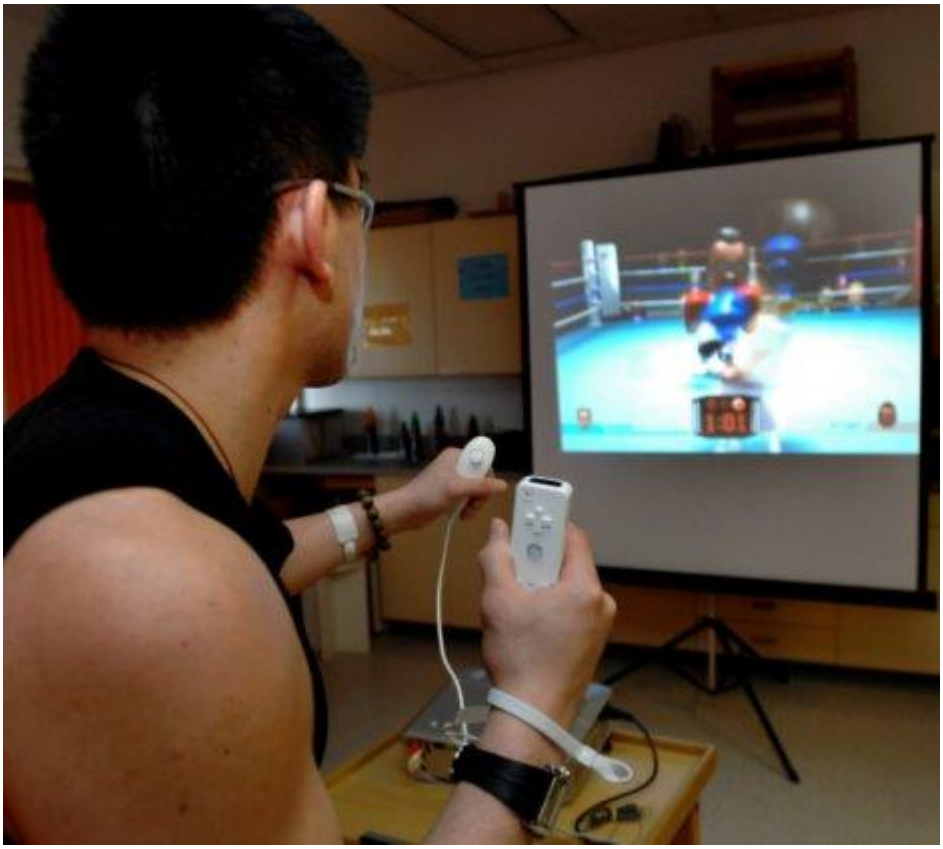

Figura 21: Wii MotionPlus, para oWii, da Nintendo [Wii06].

Figura 22: PS Move, para o Playstation 3, da Sony [PSMove10].

3.3.3 Rastreamento Adaptativo

O rastreamento adaptativo consiste em utilizar diferentes tecnologias com variadas vantagens e desvantagens para fornecer o melhor sistema de localização. Para ser possível utilizar diversas tecnologias, é preciso levar em conta que cada uma tem capacidades distintas de rastreamento e que portanto o sistema tem que estar preparado para se adaptar a diferentes níveis de erro de registro durante sua execução.

Um dos primeiros trabalhos a se preocupar com isso foi o de Höllerer et al., que descreve um sistema que adapta sua interface para diferentes tecnologias de rastreamento [Hollerer01b]. Este projeto é equivalente a segunda técnica do já comentado estudo na área de interfaces do usuário na seção de RA [Hollerer01a]. A figura abaixo mostra a diferença na interface entre duas tecnologias utilizadas para rastreamento:

(a) (b)

Figura 23: Duas interfaces de um mesmo sistema de RA. Em (a), a precisão da localização é maior; na interface (b), menor[Hollerer01a].

Na interface (a) a tecnologia de localização utilizada tem excelente precisão, fazendo com que a RA gerada possa ser exatamente registrada (alinhada) no ambiente real. Dessa forma, a interface (a) mostra objetos virtuais mais complexos e que precisam de um perfeito sistema de localização para serem utilizados.

Na figura (b) a tecnologia de localização utilizada não garante precisão na localização do usuário, mas afiança exatidão na orientação, e, portanto, a interface se torna mais simples, somente utilizando um modelo de mundo (WIM) para mostrar ao usuário.

Outro trabalho voltado para o rastreamento adaptativo na RA é a abstração de alto-nível para rastreadores [Coelho03]. Como a maioria dos sistemas precisa unir várias tecnologias para conseguir registrar a RA, é necessário a criação de uma abstração aos sistemas de localização e rastreamento na RA, de modo que os desenvolvedores possam ser capazes de criar sistemas que lidem e se adaptem com a incerteza. O resultado deste trabalho gera o *framework* OSGAR, explicado na parte sobre registro, seção 2.2.

A seguir é apresentado o Ubitrack, uma solução como gerenciador de rastreadores de alto-nível, que foi utilizado para auxiliar na construção do *framework* VISAR.

3.4 *Framework* de Rastreamento Ubitrack

O Ubitrack, abreviação de *Ubiquitous Tracking*, é um *framework* desenvolvido por vários estudantes através da parceria entre dois laboratórios de computação, das universidades técnicas de Viena (Áustria) e Munique (Alemanha). O *framework* Ubitrack foi produzido para atender a uma demanda específica: enquanto cada sistema de RA precisava se preocupar em fazer toda a parte de rastreamento e localização, seria possível criar um *framework* que abstraísse essa função para as aplicações.

Tendo tal necessidade em vista, o Ubitrack foi construído como um *framework* capaz de descobrir dinamicamente sensores heterogêneos no ambiente e fundir a informação de localização de tais sensores, de forma a obter a estimativa ótima da relação espacial obtida por eles.

Além da relação espacial, é um objetivo dessa tecnologia saber a precisão das informações obtidas em relação ao momento e entregá-las para a aplicação no menor tempo e com a maior precisão possível, mediante requisições (*query*) [Newman04].

Portanto, com o Ubitrack, será possível criar uma aplicação de Realidade Aumentada sem precisar se preocupar com o desenvolvimento e funcionamento de um sistema de rastreamen-

to e localização. A figura 24 mostra as dificuldades de rastreamento que o framework pretende resolver:

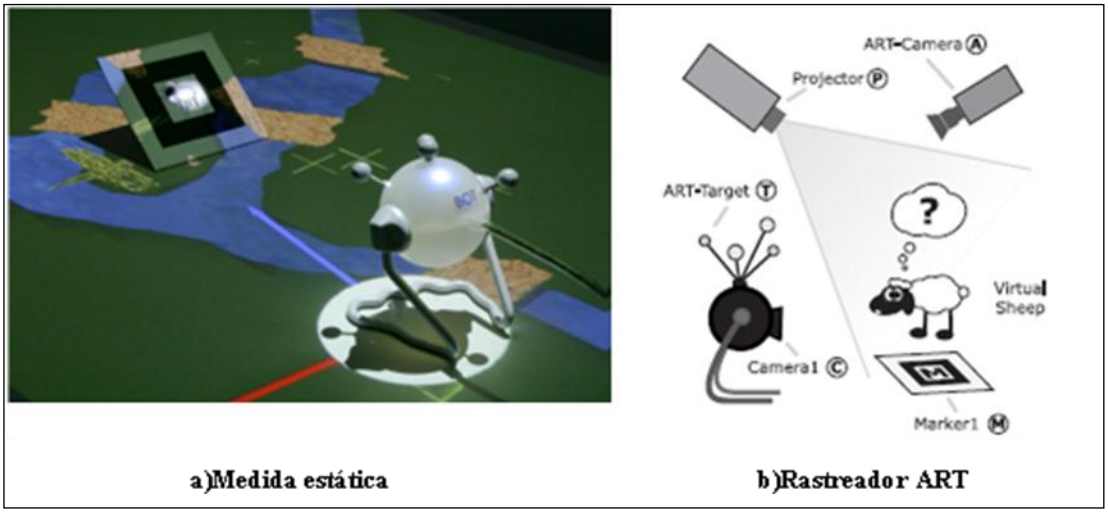

Figura 24: Exemplo de aplicação[Newman04].

Na figura 24 (b), a câmera A, que está fixa, visa rastrear a posição do marcador M e enviar a informação para o projetor P, fazendo com que ele possa delinear a imagem de uma ovelha no local exato de M. O problema é que M está fora do alcance de A. Para isso é introduzido um quarto objeto: uma câmera C com um marcador ART T. Assim, C consegue rastrear M e passar essa informação para A, que, através de T, rastreia a posição de C e converte a informação enviada por C para as coordenadas de A.

Por fim A repassa a informação para P, que converte para suas coordenadas e finalmente consegue obter a posição de M, para criar a realidade aumenta em cima dele, como mostrado na figura 24(a).Para realizar todas essas operações, o Ubitrack gera um grafo de relação espacial (SRG). Este tema é explicado mais detalhadamente no item 3.4.1.

O Ubitrack também possui um gerenciador e modelador para facilitar o seu uso e permitir seu gerenciamento em tempo de execução, o Trackman. Ele se conecta diretamente ao Ubitrack.

3.4.1 Grafo de Relação Espacial (SRG)

Os grafos de relação espacial(SRG[4]) foram definidos pela primeira vez em [Newman04]. Neles, cada nó representa um dos objetos de interesse, cuja posição interessa para a aplicação

[4] Spatial relationship graph.

e as arestas formadas representam a relação espacial entre eles, ou seja, são uma forma de se medir a matriz de transformação ou rotação entre os nós. As arestas entre os nós não representam a medição, mas apontam para um componente de software que é capaz de fazer esse cálculo em tempo de execução [Putska06a].

Outras informações podem ser passadas através da aresta, como a latência, o ruído, o *timestamp* da medição, o desvio padrão em metros. O Ubitrack também assume que os sensores são capazes de fornecer estatística dos erros de medição através de uma probabilidade de densidade, como, por exemplo, uma matriz de covariância.

Além disso, as arestas têm alguns classificadores para mostrar o tipo de informação que é medida e a forma como os dados são enviados entre os nós que compõem a aresta.Os tipos de medições nos dizem o que é calculado geometricamente, ou seja, os graus de liberdade, as coordenadas de rotação e translação e o tipo da aresta (se estático ou dinâmico).

Na aresta do tipo estático, a relação espacial não se altera. No outro caso, a relação espacial não é fixa e, portanto, deve ser medida durante toda a execução (A forma de envio entre os nós é explicada melhor na seção 3.5.3). Um exemplo de SRG e sua utilização é apresentado na figura abaixo:

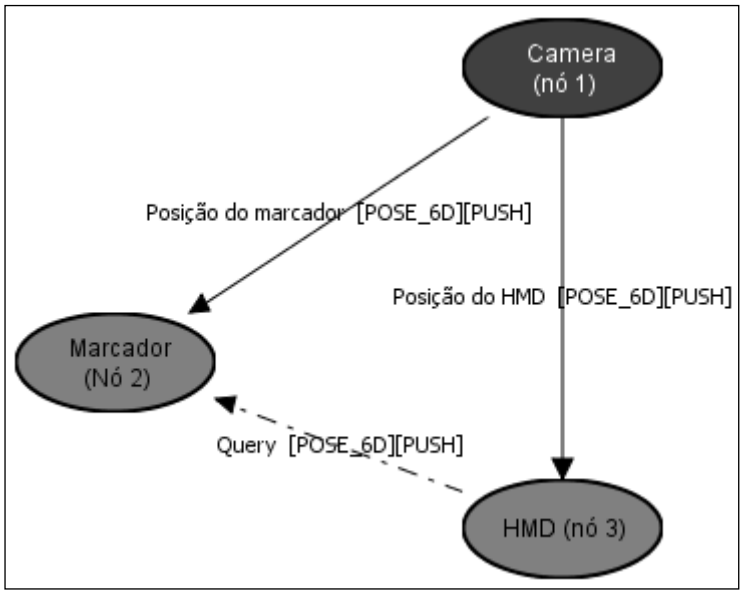

Figura 25: Grafo de relação espacial.

Na figura 25, a câmera no nó 1 consegue rastrear dois objetos, um HMD e um marcador. Deseja-se obter a posição do HMD em relação ao marcador. Ambas as arestas que saem da

câmera servem para rastrear a posição nos seis graus de liberdade (6DOF - *Degrees Of Freedom*), e, portanto, o resultado pretendido, que é a aresta Query, também é 6D.

Esse grafo consegue mostrar as relações geométricas entre todos os objetos desse ambiente, mas, para obter o resultado definitivo, é preciso fazer algumas inferências com base nas medições que nos são dadas pela câmera. Para isso, são necessários padrões, a serem explicados na próxima seção.

3.4.2 Padrões (Patterns)

Os padrões são formas de resolver problemas no grafo de relações espaciais que necessitam de um algoritmo. Eles são inseridos no SRG em partes onde se tem um algoritmo conhecido para resultar em uma informação que ainda não se tem. A partir disso o algoritmo correspondente faz a sua execução e provê o resultado na forma de uma aresta[Pustka06b].

O SRG da figura 25, se acrescido dos padrões necessários para o HMD saber a posição do marcador, geraria o SRG da figura 26.

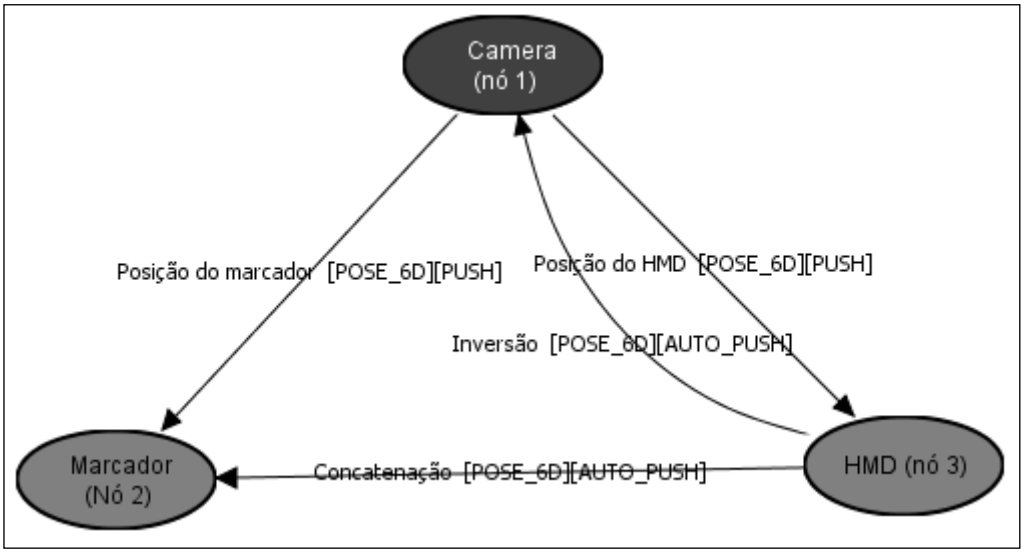

Figura 26: SRG da figura 24 acrescido de padrões.

Portanto para a aplicação obter a posição do HMD em relação ao marcador, primeiramente é necessário um padrão de inversão que aja sobre a posição do HMD obtida pela câmera e depois a concatenação das arestas de inversão com a de posição do marcador. Dessa forma, a aresta concatenação é a que a aplicação procura e ela pode ser obtida pela aplicação através de uma requisição, que é representada pelo padrão "*Application Push Sink*". Mais detalhes sobre os padrões e seus tipos pode ser visto em [Pustka06b].

3.4.3 Problemas de Sincronização

Um dos problemas em se construir um sistema de localização tão complexo como o Ubitrack, que suporta sensores heterogêneos e, ao ser requisitado por aplicações, oferece rastreamento em tempo de execução, é a sincronização das informações. Para as medidas de rastreamento de dois equipamentos serem concatenadas, devem acontecer simultaneamente, mas a sincronização por hardware para equipamentos diferentes é complicada e demanda tempo para configuração e calibração do sistema.

Por isso, quando a aplicação requer uma medição (que envolve dois equipamentos não sincronizados) é preciso utilizar alguns padrões. Alguns destes padrões, que são do tipo completos, usam os algoritmos de interpolação constante, interpolação linear e filtro de Kalman, que conseguem sincronizar para aplicação as medições dos sensores feitas em tempos diferentes.

3.4.4 Redes de Fluxo de Dados (DFN[5])

O grafo de relação espacial é uma forma abstrata de definir as relações geométricas entre objetos de interesse de um ambiente. Mas essa forma abstrata que é o SRG deve ser convertida para uma estrutura de mais baixo nível que possa ser utilizada pelo Ubitrack para calcular as medições exigida pela aplicação.

Esta estrutura, que é criada a partir do SRG, é a rede de fluxo de dados (DFN), utilizada em tempo de execução e construída de maneira dependente da implementação do Ubitrack, pois reflete os algoritmos e *drivers* utilizados por ele em execução, diferentemente do SRG que é uma estrutura abstrata.

O SRG do exemplo mostrado na figura 26 gera o DFN da figura 27.

[5] Dataflow Networks

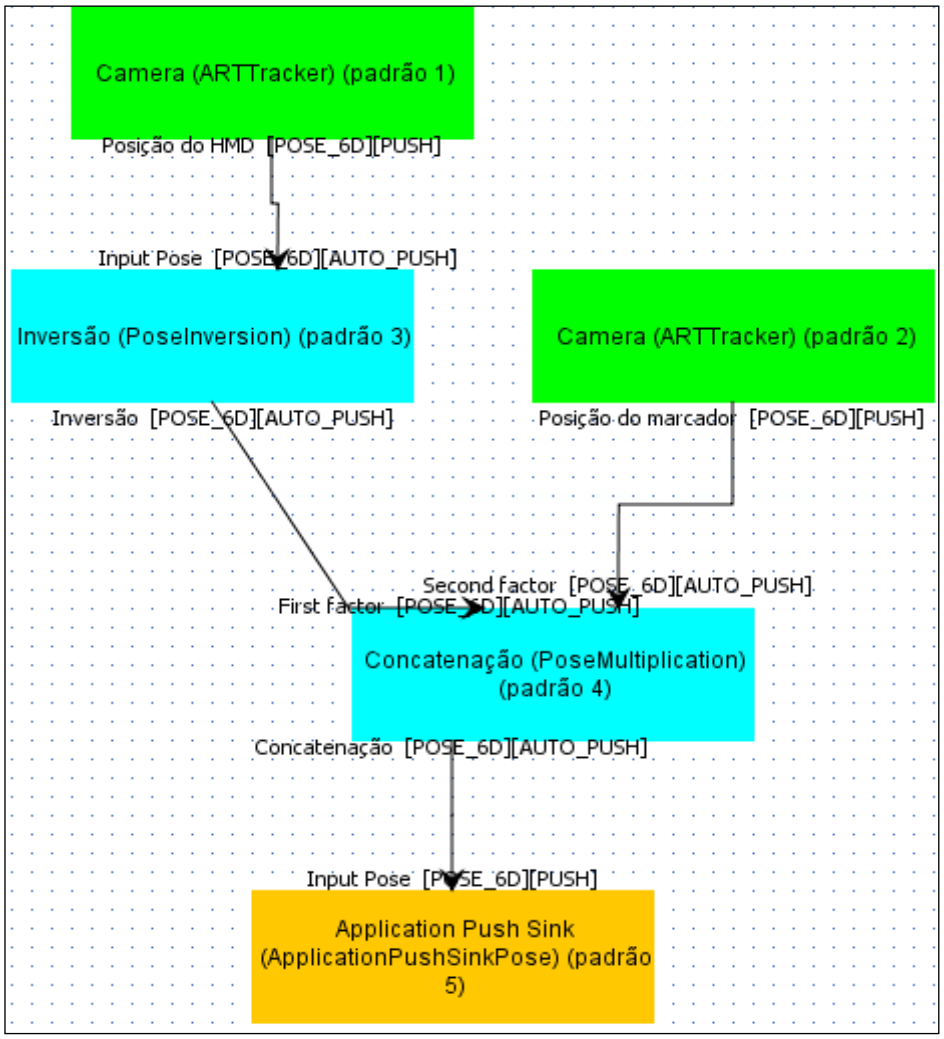

Figura 27: Tela extraída da ferramenta TRACKMAN com a DFN da figura 26.

A medição das posições dos marcadores através da câmera estão nos padrões 1 e 2, que portanto são os padrões A.R.T. Tracker. Como é preciso inverter o resultado do padrão 1, o padrão 3 é de inversão. Concatenando o padrão 3 e 2 através do padrão 4 obtém-se o padrão 5, que é responsável por receber a comunicação da aplicação para informar o resultado.

A abordagem utilizada pelo Ubitrack, com a DFN, já foi utilizada por outros sistemas com o mesmo objetivo, como o Opentracker [Reitmayr01].

3.4.5 Arquitetura do Ubitrack

Para que o Ubitrack funcionasse de modo eficiente e com capacidade de suportar as aplicações de Realidade Aumentada em tempo de execução, quatro requisitos básicos precisariam ser cumpridos[Huber07]:

- capacidade de reconfigurar-se dinamicamente durante a execução, de modo que

fosse possível adicionar novos sensores ou retirá-los sem a necessidade de reiniciar o sistema;

- comunicação eficiente de dados de rastreamento em tempo de execução, pois seria necessária uma latência baixa entre os sensores e a aplicação, de modo que o erro de registro não se tornasse alto por conta do atraso do sistema;
- capacidade de aceitar requisições de aplicações quanto à posição de objetos e transformações entre eles, pois o objetivo seria criar um sistema com o qual a aplicação pudesse se conectar e obter as informações de rastreamento e localização dos objetos de interesse, de modo que a aplicação ficasse independente do sistema de rastreamento;
- suportar capacidades de processamento variadas de clientes, de modo que seria possível um sistema executar em um cliente com pouca capacidade de processamento, como um PDA, sendo o processamento nesse caso realizado por outro cliente com capacidade maior de processamento, como um sensor preparado para tal e que os dados para os clientes com menores capacidades fossem entregues já processados e prontos para serem visualizados;

Para atender a tais requisitos, foi desenvolvida uma arquitetura de *peer-to-peer* com um coordenador central, para se aproveitar do melhor de ambas as arquiteturas, cliente-servidor e *peer-to-peer*. Com a arquitetura cliente-servidor a coordenação e manutenção do ambiente de rastreamento é facilitada e com a arquitetura *peer-to-peer* a troca de dados, principalmente de rastreamento, entre os clientes é acelerada, de modo a se evitar latência.

O cliente representa uma entidade que está diretamente envolvida com a DFN. Ele utiliza a API da biblioteca do Ubitrack para realizar todas as funções relacionadas ao rastreamento. Primeiramente ele cria o seu SRG mostrando quais as relações espaciais entre os seus sensores e outros sensores ou objetos que estejam no ambiente e depois acrescenta as suas requisições, para os objetos de interesse que ele tenha. O servidor coordena as requisições de todos os clientes e instancia um DFN para cada cliente.

Os clientes têm dois tipos básicos, mas que podem formar um híbrido. São eles: sensor, que provê medições de rastreamento para o sistema e aplicação, que consome as medições.

Além dos tipos básicos, o cliente é formado pelos seguintes componentes:

- aplicação: apóia-se na biblioteca do Ubitrack para qualquer operação de rastreamento. Uma aplicação típica é um HMD que precisa saber a relação espacial entre ele e outro objeto virtual, para poder renderizar tal objeto e para isso utiliza uma requisição para ser informado da relação espacial;
- biblioteca do Ubitrack: a biblioteca do Ubitrack encapsula todas as comunicações entre o cliente e o servidor. Com ela os sensores podem registrar quais informações fornecerão e as aplicações quais informações consumirão;
- DFN dinâmica: os componentes da DFN são instanciados assim que o servidor Ubitrack inicia sua execução e podem ser alterados a qualquer momento, caso ocorra alteração na DFN;

O servidor Ubitrack consegue coordenar os clientes para sincronizar os rastreamentos [Putska08]. Ao iniciar sua execução, ele tem um SRG vazio, o qual vai sendo preenchido com as entradas (*input*) de cada cliente, quando estes enviam seus SRGs e seus padrões para o servidor, responsável por construir um SRG global e aplicar os padrões para inferir novas arestas do SRG.

Quando uma requisição chega ao servidor, ele gera uma descrição do fluxo de dados com as arestas dependentes e envia para o cliente que fez a requisição. No caso de ser mais de um cliente, essa descrição é distribuída a eles, que podem se comunicar em tempo de execução, seguindo o modelo *peer-to-peer*, reduzindo a latência do servidor. A figura abaixo mostra a arquitetura completa do Ubitrack:

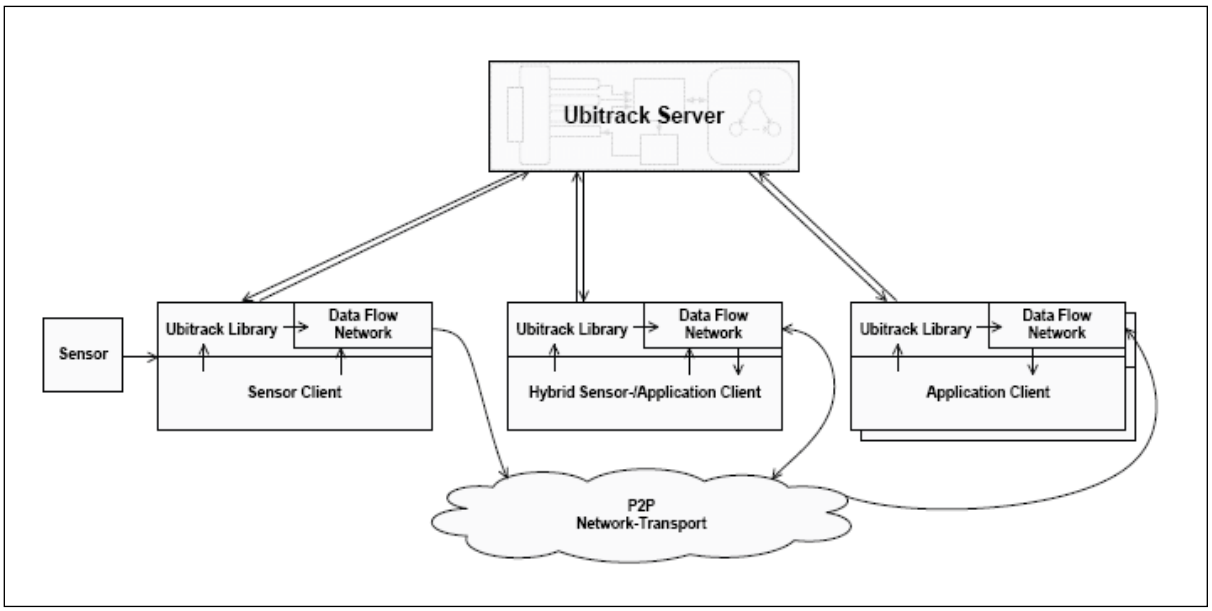

Figura 28: Arquitetura do Ubitrack [Huber07].

Um gerenciador e modelador, o Trackman, completa essa arquitetura, permitindo modelar os SRGs rapidamente e gerenciar o servidor e clientes em tempo de execução através de uma interface gráfica.

3.5 Discussão dos Trabalhos Relatados na Literatura

Alguns trabalhos da área de RA adaptativa já tentaram utilizar contexto em aplicações de RA [Hollerer01a] [Robertson02] [Coelho04] [Lee08] [Oh09], mas, embora bem sucedidos, o foco é sempre somente um ou dois contextos específicos.

As técnicas de filtragem da informação, adaptação da interface relativa ao registro e gerenciamento de visualização, mostradas por [Hollerer01a], [Coelho04] e [Robertson02], oferecem uma boa possibilidade de *design* de interfaces adaptativas. Ambas as técnicas de Mendez et al., o uso de lentes mágicas para alterar dinamicamente as cenas a partir de informação contextual [Mendez06], ou um *framework* que faz uso de estilos de visualização e marcadores de contexto [Mendez07], também têm desvantagens. No primeiro deles, há a obrigação do uso das Lentes Mágicas para ditar o contexto e alterar a visualização. No segundo, o problema é a complexidade do modelo que não foi totalmente formalizado.

Os trabalhos de Lee e Woo [Lee08] e Oh e Woo [Oh09] também permitem adaptação, mas considerando somente a iluminação ou o usuário. Em geral, os trabalhos de área têm a mesma desvantagem – embora ofereçam certo grau de ciência de contexto, eles concentram em somente um contexto, como localização (registro) ou iluminação.

Estas limitações foram consideradas no projeto do framework VISAR proposto como parte deste trabalho de mestrado. O VISAR foi construído de modo a permitir que qualquer contexto possa ser facilmente modelado para influenciar a interface do usuário na forma de cenários. Já que sua IU é baseada no modelo de padrões de interface, isso é feito através de funções que permitem que a aplicação altere e atualize a interface somente ajustando e trocando padrões de interface em tempo de execução. Ao transferir a responsabilidade da captura de contextos para a aplicação, o VISAR pode focar somente na interface do usuário, assim sendo flexível para se adaptar a contextos, desde que a IU tenha sido corretamente modelada no editor de interfaces (VISAR-IE).

3.6 **Considerações Finais**

Este capítulo apresentou os conceitos de contextos, localização e rastreamento. Os contextos são úteis para a RA, pois podem melhorar a qualidade das informações sobre o ambiente mostradas para o usuário na interface. Uma forma eficiente de modelá-los, considerando que sua função é capturar as informações que serão mostradas, pode facilitar e agilizar a construção de aplicações de RA.

Assim, foi mostrada uma forma de modelagem de contextos planejada e também classificados os principais contextos da RA, divididos em contextos de localização, respectivo ao rastreamento, iluminação, respectivo a claridade e luz no local e usuário, respectivo ao indivíduo utilizando o sistema.

Neste Capítulo foram também apresentadas as tecnologias de rastreamento atuais e o gerenciador de localização Ubitrack, utilizado na construção da ferramenta VISAR proposta como parte deste trabalho de mestrado e apresentada no próximo Capítulo. Serão apresentados também o editor de interfaces VISAR-IE e três estudos de casos utilizando ambas as ferramentas.

4 VISAR E VISAR-IE: FERRAMENTAS PARA MODELAGEM E VISUALIZAÇÃO DE INTERFACES DE RA

Conforme discutido no Capítulo 1, apesar do alto potencial de uso de RA em diferentes áreas de aplicação, são vários os desafios que limitam a ampla utilização de RA, dentre eles: forte acoplamento dos projetos de RA ao *hardware* dos dispositivos; complexidade dos mecanismos que correlacionam as informações de contexto ao mundo real, dificultando a criação mais eficiente de interfaces de RA, tornando-as de alto custo e de difícil manutenção; desenvolvimento de dispositivos adequados de visualização e de interação do usuário com os objetos virtuais; calibração de equipamentos em tempo de execução; rastreamento do usuário e de objetos de interesse; registro de objetos virtuais no ambiente real, dentre outros.

Este projeto visa criar ferramentas de suporte à modelagem e visualização de interfaces de Realidade Aumentada cientes de contexto, facilitando assim o trabalho dos desenvolvedores nesta área, tornando menos complexa a tarefa de correlacionar informações de contexto ao mundo real. Duas ferramentas foram desenvolvidas como parte deste trabalho: um *framework* de visualização (VISAR) que permite implementar interfaces de RA sem a necessidade de codificá-las, utilizando componentes de interface; e um ambiente para modelagem de interfaces (VISAR-IE) que permite modelar interfaces de RA reusando e compartilhando componentes. Três estudos de caso foram também desenvolvidos para validação das ferramentas: RA em mesa tangível multitoque; "arraste-e-jogue-lá" (RA *Drag and Drop*); e um sistema de RA para apoio aos bombeiros em aplicações de gerenciamento de emergência.

O *framework* VISAR e o editor VISAR-IE fazem parte de um projeto maior em desenvolvimento no laboratório WINDIS *(Wireless Networking and Distributed Interactive Systems),* um dos grupos participantes do Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos (INCT-SEC). Neste projeto, tecnologias de redes de sensores sem

fio, robôs e veículos aéreos não tripulados são integrados para potencializar soluções atuais de segurança e proteção de infraestruturas críticas. VISAR e VISAR-IE fazem parte das soluções de interfaces avançadas sendo desenvolvidas para suporte à visualização em diferentes aplicações do projeto INCT-SEC.

A figura 29 mostra a visão geral da arquitetura do *middleware* de suporte as aplicações sendo desenvolvido no projeto do INCT-SEC. O circulo verde mostra a localização do *framework* VISAR, responsável pela visualização das interfaces de RA cientes de contexto para as diferentes aplicações sendo criadas. Uma rede de sensores sem fio coleta dados do ambiente. Estes dados são submetidos a um sistema de interpretação de contexto que entrega à aplicação (integrada ao *framework* VISAR) o estado de uma entidade. Esta informação é utilizada pelo VISAR para adaptar a interface de RA ao estado das entidades que interessam à aplicação.

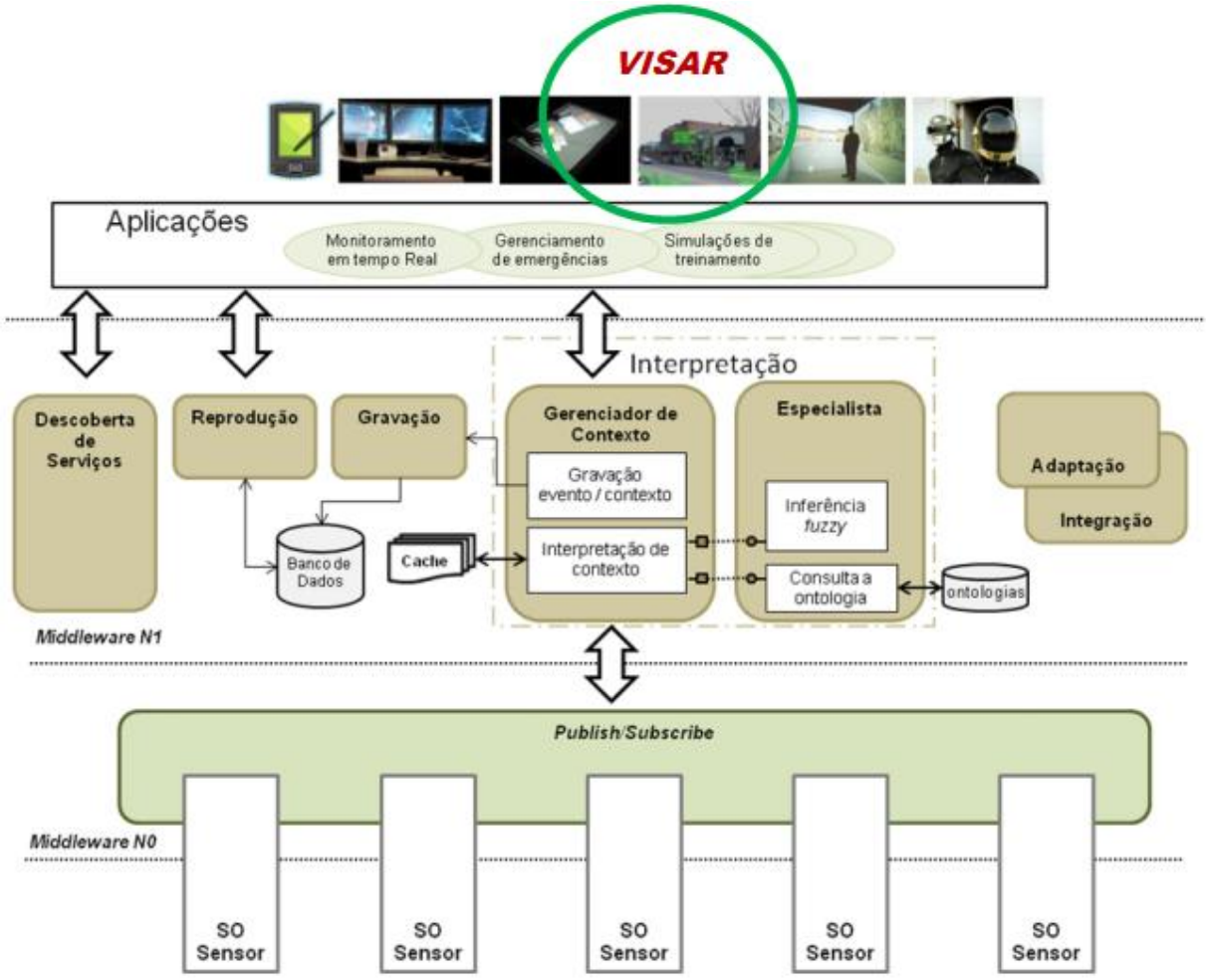

Figura29: Visão geral da arquitetura do *middleware* MIDSENSORNET (Projeto INCT-SEC) [Beder11].

4.1 **VISAR Interface Editor**

O editor VISAR-IE *(Interface Editor)* é uma ferramenta de autoria de RA, que funciona de forma integrada com o *framework* VISAR. O editor de interfaces VISAR-IE, permite aos desenvolvedores de interfaces de RA focar na modelagem da interface e em suas regras. O editor é um *software* de autoria que suporta o projeto de interfaces de RA, por meio da escolha dos padrões que devem ser exibidos em cada cenário da aplicação (isto é, situações em que o usuário ou outra entidade possa estar durante a execução da aplicação). Definição e exemplos de padrões são descritos na seção 4.2. O VISAR-IE armazena o resultado do projeto da interface em um arquivo XML que descreve a interface. Este arquivo é lido pelo *framework* VISAR (descrito na próxima seção), para carregar a estrutura da interface. O VISAR-IE foi construído usando o *plugin* Graphical Editor *Framework* do Eclipse (Eclipse GEF).

O projeto da interface no VISAR-IE pode ser feito em conjunto com um especialista na área da aplicação (ou por ele/ela próprio/a), pois a ferramenta não requer conhecimento em programação, mas exige um entendimento básico de ferramentas de modelagem, como fluxo de dados ou mapa mental, além de conhecimento sobre os padrões de interface de RA (não é necessário entender como eles funcionam, mas sim o que mostrarão para o usuário) e planejamento preciso da interface (uma boa análise de requisitos do sistema).

A figura 30 mostra a aparência do editor e para explicá-la ela foi dividida em quatro partes:

1. Paleta de padrões - onde ficam os padrões de interface disponíveis para serem usados;
2. Estrutura da interface sendo construída, com a exibição dos contextos e padrões que compõem a interface;
3. Tela de visualização do usuário. É possível ao projetista da interface fornecer o tamanho da tela (e resolução) nas propriedades. No caso da figura 30, a tela poderia ser de um celular, onde o desenvolvedor pode posicionar padrões 2D, indicando onde eles devem ser renderizados na tela assim como seus respectivos tamanhos;
4. Propriedades do padrão selecionado. Cada padrão tem seu próprio conjunto de propriedades que pode ser alterado aqui. No caso da figura 30 as propriedades são referentes à tela de visualização do usuário.

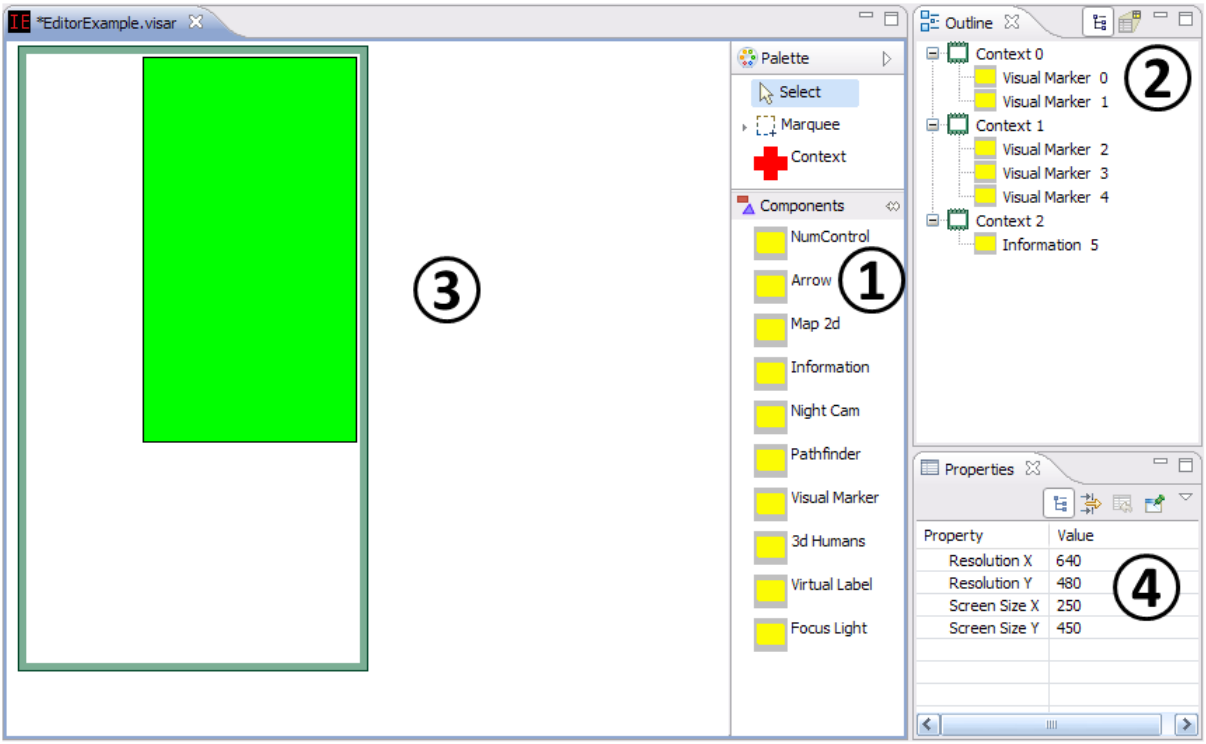

Figura30: VISAR Interface Editor.

4.1.1 Contextos e Cenários

Um aspecto importante das duas ferramentas sendo descritas neste capítulo é que qualquer contexto considerado pelo desenvolvedor pode ser usado para adaptar a interface do usuário, não ficando restrito aos três contextos ditos como principais em sistemas típicos de RA: localização, iluminação e usuário. Para isso foi criada uma estrutura que permite aos desenvolvedores gerar cenários que representam contextos e selecionar o que se deseja mostrar para o usuário em cada um desses cenários (por meio da associação de um conjunto de padrões de interface a cada cenário). Dessa forma é permitida uma modelagem fácil de interfaces dinâmicas. O termo "cenário" é utilizado neste capítulo para definir uma situação específica em que o usuário (ou outra entidade de interesse da aplicação) possa se encontrar. Padrões são definidos na próxima seção.

4.2 *O Framework* VISAR

O principal objetivo do *framework* VISAR é facilitar o desenvolvimento/implementação de interfaces de RA. O *framework* pode ser utilizado em seu potencial completo quando recursos de rastreamento do usuário estão presentes no ambiente de aplicação, como, por exemplo, em prédios inteligentes. VISAR utiliza o conceito de componentes de interface com ca-

racterísticas similares, independente da área da aplicação, assim como também ocorre com sistemas *desktop* e *web*. Estes componentes são os padrões de interface, soluções conhecidas e muito utilizadas para problemas comuns, que expressam um modo reconhecido e aceito de se exibir uma informação [Seffah10]. O exemplo mais comum de padrão de interface na RA é o marcador visual, que é a exibição de um objeto 3D sobre um marcador, quando este marcador está visível para uma câmera.

No VISAR cada padrão se torna um componente gráfico implementado e pronto para ser utilizado, podendo ser controlado através de funções. Assim, a interface do usuário pode ser modelada simplesmente escolhendo-se quais padrões serão exibidos e podendo ser dinamicamente alterada trocando-se os padrões a serem mostrados.

Aplicações que usam o VISAR controlam os padrões em tempo de execução, preenchendo-os com a informação que se deseja mostrar ao usuário. Cada padrão tem sua API (Interface de Programação de Aplicações), que são funções que a aplicação pode usar para gerenciar os padrões, seja alterando alguma informação, seja alterando a posição do padrão na cena.

O VISAR não é uma aplicação, mas sim um *framework* que permite construir e controlar a interface do usuário de uma aplicação de Realidade Aumentada. Portanto ainda é necessário programar uma aplicação, mas o VISAR pode ser usado para acelerar o desenvolvimento, já que todos os componentes gráficos da interface necessários já estão nele programados. Mesmo que um padrão desejado não esteja ainda programado no VISAR, este pode ser adicionado por um desenvolvedor e posteriormente reusado por outros.

A implementação de padrões de interface do VISAR é baseada no padrão MVC (Modelo-Visualização-Controle), em que a interface é dividida em três partes: o modelo, que são os dados dentro de cada padrão, a visualização, que é o modo que estes dados serão projetados para o usuário e o controle, que é responsável por alterar o modelo, forçando assim uma atualização na visualização.

Internamente o VISAR possui o modelo e a visualização. Assim, a aplicação precisa apenas fazer o papel de controle, acessando e alterando o modelo, fazendo com que o VISAR atualize a visualização.

4.2.1 Padrões de Interfaces do VISAR

Padrões de interface já são usados para projetar interfaces tradicionais, mas, até onde é de conhecimento do autor, ainda não houve tentativa de padronização de interfaces de RA de modo a facilitar a criação, o controle e o reúso de interfaces de RA.

De uma maneira geral, os padrões de RA podem ser classificados como estático ou dinâmico. Padrões estáticos são baseados em interfaces convencionais 2D, uma vez que mesmo se tratando de interfaces de RA, ainda é necessário exibir informações mais simples em 2D no visor do dispositivo do usuário. Já os padrões dinâmicos são formados por objetos projetados no ambiente e são divididos pelo tipo de rastreamento necessário para funcionarem: um tipo para rastreamento genérico e o outro específico para visão computacional. Sendo assim, quatro tipos diferentes de padrões foram criados:

- *Estático Normal:* representa padrões que tem a posição fixa em relação ao usuário, normalmente 2D e são fixados no display do usuário, por exemplo, no HUD (*Head up Display*). Suas propriedades são a posição e o tamanho na tela;
- *Estático Rastreado:* representa padrões com posição fixa em relação ao usuário, porém requerem informações de rastreamento do usuário e objetos de interesse na cena para funcionarem. Suas propriedades são as mesmas do tipo anterior;
- *Dinâmico Rastreado:* representam objetos virtuais que serão inseridos no ambiente real e por isso requerem um sistema de localização. Suas propriedades são: a posição (X, Y, Z) no ambiente, o seu tamanho e o ângulo de rotação;
- *Dinâmico com Visão:* representa padrões dinâmicos com rastreamento baseado em visão computacional (normalmente reconhecimento de fiduciais ou marcadores como o ARToolkit). Suas propriedades são as mesmas do tipo anterior mais a imagem do marcador.

Com essa classificação de padrões, é fácil estender a gama de padrões atuais criando novos. Para isso basta declarar o tipo de um novo padrão e parte/todas as funções básicas de funcionamento dele já estarão prontas.

Alguns exemplos de padrões podem ser vistos na figura 31. Estes são:

a) *Rótulo Virtual:* padrão dinâmico rastreado, que mostra um rótulo com informações sobre um determinado alvo (próximo ou no topo do alvo). Sua API tem as seguintes

funções: estabelecer a posição (6DOF – 6 graus de liberdade, ou seja, tanto a posição quanto o ângulo de rotação) e inserir texto no rótulo;

b) *Apontador:* padrão estático rastreado, que mostra uma seta para guiar o usuário na direção do seu destino. Esta é uma seta 3D, portanto também pode apontar para cima e para baixo. O padrão deve saber a posição do usuário e o ponto de destino para então poder calcular a direção em que a seta deve apontar automaticamente e em tempo de execução. Sua API tem somente a função de estabelecer o ponto de destino;

c) *Caminho Virtual:* padrão dinâmico rastreado, que renderiza um caminho virtual que mostra uma rota para o usuário. Este caminho é construído através de um cálculo baseado no modelo 3D (CAD) do ambiente e vai apenas da posição do usuário até a próxima esquina ou porta (para que assim ele não seja mostrado em locais fora da visão do usuário, sobrecarregando a tela com informações desnecessárias). Sua API tem somente a função de estabelecer o ponto de destino;

d) *Marcador Visual:* padrão dinâmico com visão (o mais comumente encontrado em interfaces de RA), que renderiza um objeto 3D no topo de um marcador. Sua API tem as seguintes funções: estabelecer a imagem do marcador e a posição (6DOF) do objeto 3D em relação ao marcador.

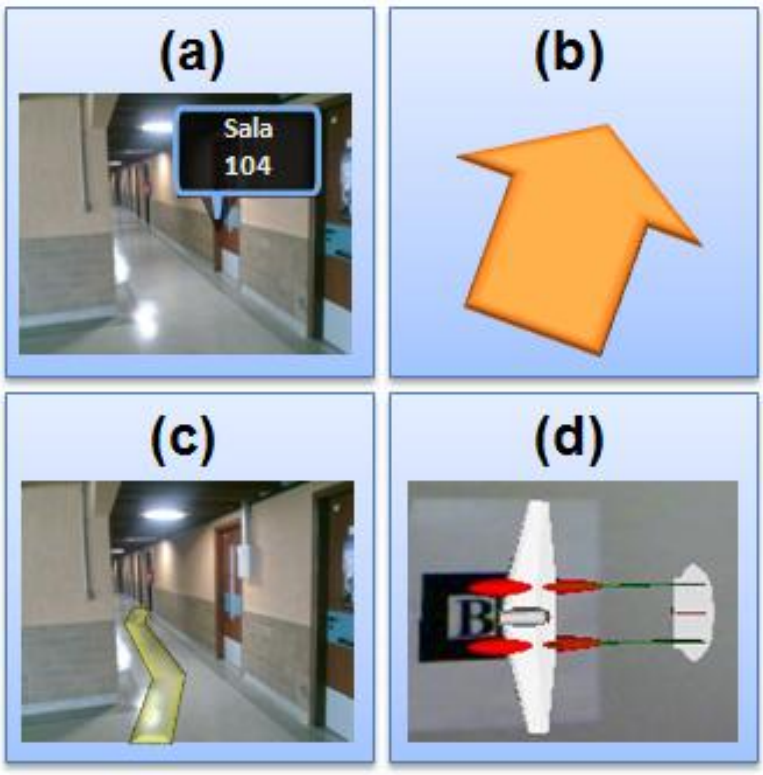

Figura31: Exemplos de padrões de interface.

Outros exemplos de padrões de interface que poderiam ser criados incluem:

- *Humanóide:* padrão dinâmico rastreado, que é um modelo 3D de um corpo humano inserido no ambiente para mostrar onde estão as pessoas próximas do usuário que ele não consegue ver (por exemplo, dentro de uma sala diferente de onde está o usuário). Com um único padrão qualquer número de humanos 3D pode ser gerado. Sua API tem as seguintes funções: gerar um novo humano 3D, onde é necessário saber a posição inicial do humano e alterar a sua posição, o que pode ser feito por um gerenciador de rastreamento que alimenta o padrão passando a posição de cada humano. Um exemplo pode ser visto na figura 33;
- *Luz de Foco:* padrão dinâmico rastreado, que cria uma luz virtual para ser projetada sobre um objeto (real) de interesse, sendo que a luz pode ser mais brilhante quanto maior for o interesse no objeto. Sua API tem as seguintes funções: estabelecer a posição da luz, estabelecer o brilho e a cor da luz. Um exemplo pode ser visto na figura 34;
- *Controle Numérico:* padrão estático normal, que mostra um valor mensurado do ambiente, além de uma imagem que representa esse valor visualmente e mais cinco barras para ajudar a visualizar quão alto ou baixo está o valor. Sua API tem as seguintes funções para estabelecer: o valor numérico, a quantidade de barras pintadas, a cor das barras e a imagem que representa o valor medido. Um exemplo pode ser visto na figura 36;
- *Mapa 2D:* padrão estático rastreado, que mostra a planta baixa do ambiente. Elementos chave na cena, como a posição de um evento para onde o usuário está se direcionando ou a posição de outras pessoas no ambiente, podem ser representados dentro desse mapa. Este padrão pode ser visto na figura 36, onde os pontos azuis são bombeiros no ambiente e a zona vermelha é o local de ocorrência do incêndio. Sua API tem as seguintes funções: colocar um novo evento no mapa com a respectiva posição, tamanho e cor do evento, alterar tamanho e cor do evento, apagar um evento e atualizar a posição deste evento (por exemplo, no caso de uma pessoa se movendo dentro do ambiente);

- *Ponto de Caminho* (*waypoint*): padrão dinâmico rastreado, que renderiza um conjunto de pontos chave que servem como um identificador de caminho para o usuário. Diversos pontos podem ser criados com esse padrão para formar o caminho, mas somente um é renderizado por vez, portanto somente quando o usuário alcança um ponto é que o próximo é mostrado. Funciona como uma fila: o último ponto estabelecido é o último que o usuário verá. Sua API tem somente as funções de determinar um novo ponto com sua posição e apagar o primeiro ponto (quando o usuário passar por ele).

Todas as funções relacionadas a posição de padrões dinâmicos são normalmente conectadas diretamente a um gerenciador de rastreamento que possa alimentar os padrões com a localização (e ,portanto, não são mencionadas na API). Por outro lado, a posição de padrões estáticos é normalmente controlada diretamente pela aplicação. Isto ocorre por que a posição de padrões dinâmicos é um ponto no ambiente real que deve ser rastreado, enquanto a posição de padrões estáticos é somente um ponto na tela do dispositivo.

Cada padrão tem uma funcionalidade única para a aplicação. Alguns têm uso similar, mas com diferente aparência e tipo, como por exemplo, os padrões “Apontador” e “Caminho Virtual” (fig. 31 (b) e (c)) que servem de guias para o usuário porém o “Caminho Virtual” necessita de um rastreamento muito preciso para funcionar. Outros padrões que tem aparência similar, mas uso diferente, como o “Mapa 2D” e o “GPS”, onde o primeiro serve para ambientes internos e o último para externos.

Utilizando estes conceitos, qualquer componente gráfico novo que solucione um problema comum de interface pode se tornar um padrão de interface e, se desenvolvido em uma estrutura padronizada (como o modelo aqui proposto de padrões de interface), este componente pode ser reutilizado por outros, poupando tempo de desenvolvimento.

Os jogos virtuais costumam utilizar intensamente a Realidade Aumentada em seu ambiente virtual para auxiliar os jogadores e, portanto podem ser fonte de inspiração de padrões. Nas figuras 33, 34 e 35são mostrados três dos padrões citados anteriormente em uso em jogos.

Fig. 32: Padrão *Caminho Virtual* sendo usado no jogo *Killing Floor,* para o computador.

Fig. 33: Padrão Humanóide sendo usado no jogo *Metal Gear Solid 4: Guns of the Patriots,* para o Playstation 3.

Figura 34: Padrão Luz de Foco sendo usado no jogo *Borderlands,* para o computador, Playstation 3 e Xbox 360.

4.2.2 Funcionamento do VISAR

O funcionamento do VISAR se inicia com o carregamento, pela aplicação, do documento XML criado pelo VISAR-IE, que contém a descrição da interface (os padrões e cenários que pertencem à interface). A aplicação envia para o VISAR o nome do dispositivo em que a interface deve ser mostrada e o VISAR renderiza os padrões do cenário inicial na tela.

O VISAR suporta a aplicação com funções para controlar a interface (APIs). O caminho dos dados, da aplicação para o VISAR e depois para a renderização no display do usuário, pode ser visto na figura 35.

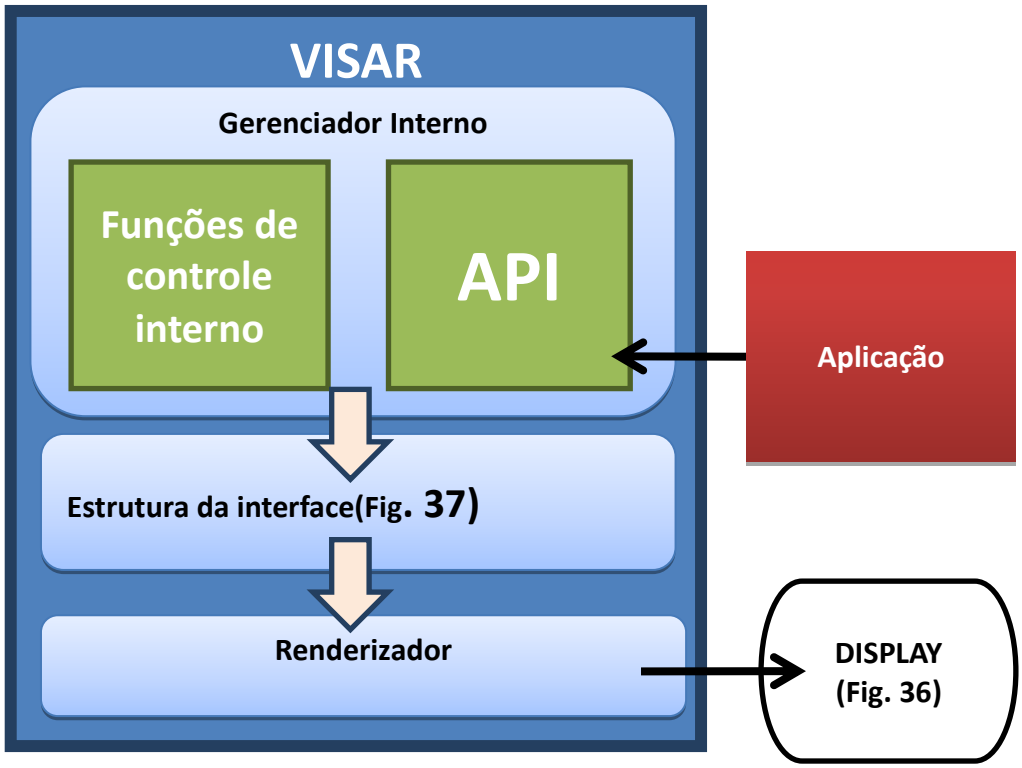

Figura 35:Funcionamento e arquitetura interna do VISAR. Desde a aplicação mandando uma mensagem até o *display* onde a interface é renderizada.

Após a renderização do primeiro contexto, o VISAR espera por um de dois tipos de comunicação por parte da aplicação: ativação ou desativação de um cenário ou alteração no conteúdo de um padrão (que atualiza informações na interface). Os cenários ativos têm todos os seus padrões renderizados, portanto, ativar um cenário faz o VISAR renderizar seus padrões, enquanto que desativar faz o oposto. Assim, é possível mudar a interface em tempo de execução de forma fácil e rápida, desde que o projeto da interface tenha previsto o que deve ser mostrado para o usuário em cada cenário.

Já a alteração no conteúdo de um padrão acontece quando a aplicação deseja atualizar o valor de alguma informação na interface. A figura 36 mostra um exemplo em que a aplicação detecta um aumento na temperatura de 60°C para 85°C. A aplicação então notifica o *framework* deste novo valor, através da API do padrão (padrão do tipo Controle Numérico) que atualiza a interface (tela da direita na figura 36).

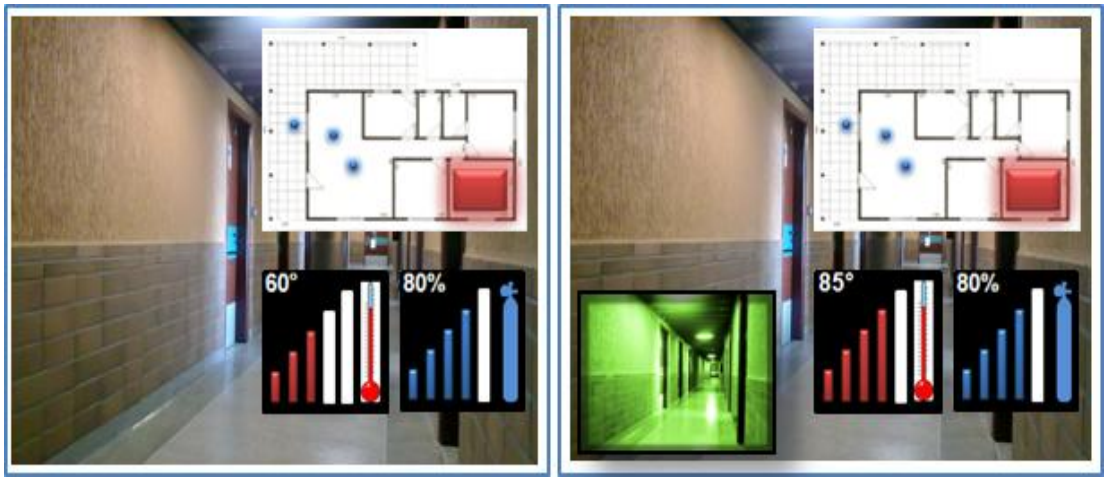

Figura 36: Exemplo de interface criado pelo VISAR, em tempo de execução.Na tela da esquerda o cenário "NORMAL" está ativo; na tela da direita o cenário "ESCURO" está ativo.

Conforme visto na seção anterior, como a RA é baseada no rastreamento do usuário e na projeção de objetos virtuais no local correto, o VISAR deve estar conectado a um gerenciador de rastreamento que calcule a posição em que cada padrão deve ser renderizado.

Cada padrão tem seu próprio requisito de rastreamento, portanto o Gerenciador Interno do VISAR (Fig. 35) é responsável por registrar cada requisito no gerenciador de rastreamento, para que o padrão seja alimentado com a localização atualizada sobre a qual seus objetos virtuais devem ser renderizados. Por exemplo, usando o ARToolkit como gerenciador de rastreamento, um padrão que deseja renderizar um objeto 3D no topo de um marcador deve receber a matriz de transformação necessária para isso, portanto neste caso (já que o ARToolkit não suporta que padrões se registrem para receber informações de rastreamento) o Gerenciador Interno é responsável por obter esta matriz e passá-la para o padrão.

Além do rastreamento, o Gerenciador Interno controla a sobreposição de padrões 2D. Se dois cenários estiverem ativos e ambos possuírem um padrão que fica na mesma posição na tela, o *framework* é programado para detectar esta situação e realocar a posição do padrão com menor prioridade, por exemplo, utilizando o algoritmo de gerenciamento dinâmico de espaço de Bell e Feiner [Bell00]. Essa prioridade é atribuída aos cenários no VISAR-IE durante a fase de modelagem.

Um exemplo de aplicação simples construída no VISAR é um sistema para bombeiros em um ambiente de emergência. Esse sistema inicia no cenário "NORMAL", onde são exibidas para o usuário a temperatura da sala em que ele se encontra, a quantidade de oxigênio restante

no galão e a planta baixa do local. Já quando o usuário entra em uma sala totalmente escura, a aplicação, que é responsável por detectar a luminosidade, ativa o cenário “ESCURO” no VISAR, que então mostra uma visão de câmera de visão noturna. Além disso, quando a temperatura sobe de 60°C para 85°C, a aplicação notifica este novo valor ao VISAR, assim como também solicita que mais uma das barras do medidor sejam pintadas. Este exemplo pode ser visto na figura 36, onde os padrões foram aumentados propositalmente para melhorar a visualização e a estrutura da interface pode ser vista na figura 37.

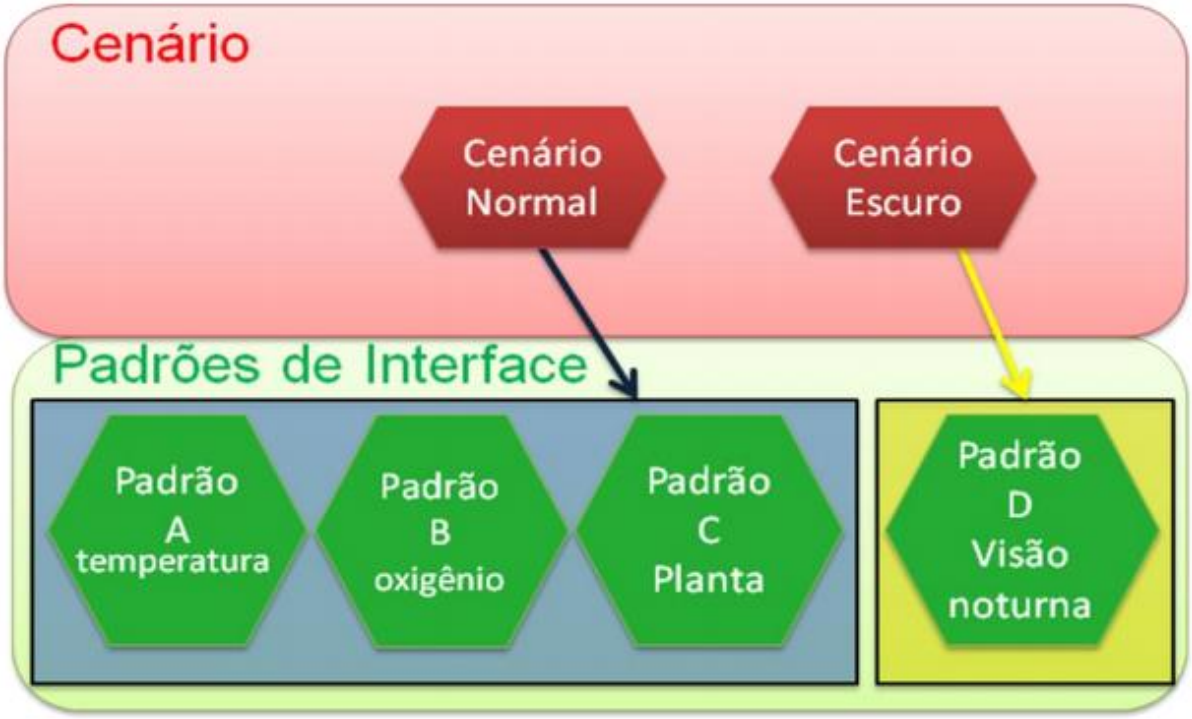

Figura 37: Modelo da estrutura da interface contendo 2 cenários/contextos e 4 padrões.

O VISAR tem uma estrutura de dados que é uma lista de cenários, cada com sua própria lista de padrões, suportando, portanto múltiplos cenários com múltiplos padrões. Os cenários seriam a modelagem de cada adaptação a diferentes contextos ou situações interpretados pelo sistema, cada um tendo sua “prioridade” e “estado”. A prioridade pode ser usada para tomar uma decisão quando dois cenários ativos tem padrões que se sobrepõem (2D) na tela. Já o estado serve para indicar se o cenário está ativo ou inativo.

Para os testes realizados até o momento, câmeras foram conectadas a um computador, mas um HMD estará sendo adquirido pelo laboratório WINDIS. Um sistema de duas dimensões é usado para referenciar a posição na tela de objetos 2D, de modo que a troca de dispositivos de visualização é trivial de ser feita.

4.2.3 Arquitetura de Sistema do VISAR

A arquitetura do *framework* VISAR foi baseada no modelo de Reicher et al. [Reicher03a] e pode ser vista na figura 38. Neste modelo, arquiteturas de RA são baseadas em seis subsistemas (criados a partir do padrão de arquitetura MVC). O trabalho de mestrado sendo aqui apresentado se concentra principalmente no subsistema de apresentação, responsável pela disponibilização de informação para o usuário.

Os subsistemas de Rastreamento, Modelo do Mundo e Interação são representados na cor cinza, por não serem objetivos principais de desenvolvimento deste projeto. O subsistema de Rastreamento recebe dados do Modelo do Mundo, correspondente à informação geométrica do ambiente real e virtual. Após isso, diversas tecnologias podem ser aplicadas para rastrear a posição do usuário e objetos de interesse e estas são fundidas para calcular a posição dos objetos virtuais. Essa informação é passada para o *framework* VISAR diretamente, como já foi comentado anteriormente, para diminuir o atraso. O subsistema de Rastreamento também atualiza o modelo geométrico do mundo virtual quando necessário. Existem várias implementações desses subsistemas, como o Ubitrack, para fundir dados de rastreamento dinamicamente.

O subsistema de Contexto tem a função de capturar e interpretar contextos do ambiente. O serviço de interpretação envia informação já processada para a aplicação e utiliza informações do modelo real do mundo para informar corretamente onde cada evento está ocorrendo. Esse subsistema foi implementado no laboratório WINDIS [Beder11].

O subsistema de Aplicação é onde está a maior parte da lógica dos desenvolvedores. Um aspecto importante a ser considerado na arquitetura sendo apresentada é que a aplicação recebe informações interpretadas e as utiliza através de duas funções do VISAR, que podem ativar ou desativar um cenário no VISAR para adaptar toda a interface ou atualizar as informações e conteúdos da interface.

O subsistema de Interação é responsável por receber comandos de entrada do usuário, que podem ser transformados pelo seu gerenciador para funções da API do *framework* VISAR, atualizando a interface do mesmo modo que a aplicação. Além disso, quando a entrada influencia a posição ou tamanho de um objeto virtual, ela também é repassada para o modelo de mundo virtual. Alguns exemplos de dispositivos de interação são o teclado e mouse, controle de videogames com capacidades de rastreamento (wii ou psmove), objetos tangíveis, e interfaces naturais como gestos e fala.

Por fim, o subsistema de Apresentação é onde o *framework* VISAR atua. As estratégias normais de apresentação foram substituídas por ele, que pode ser usado para criar, controlar e atualizar a interface do usuário.

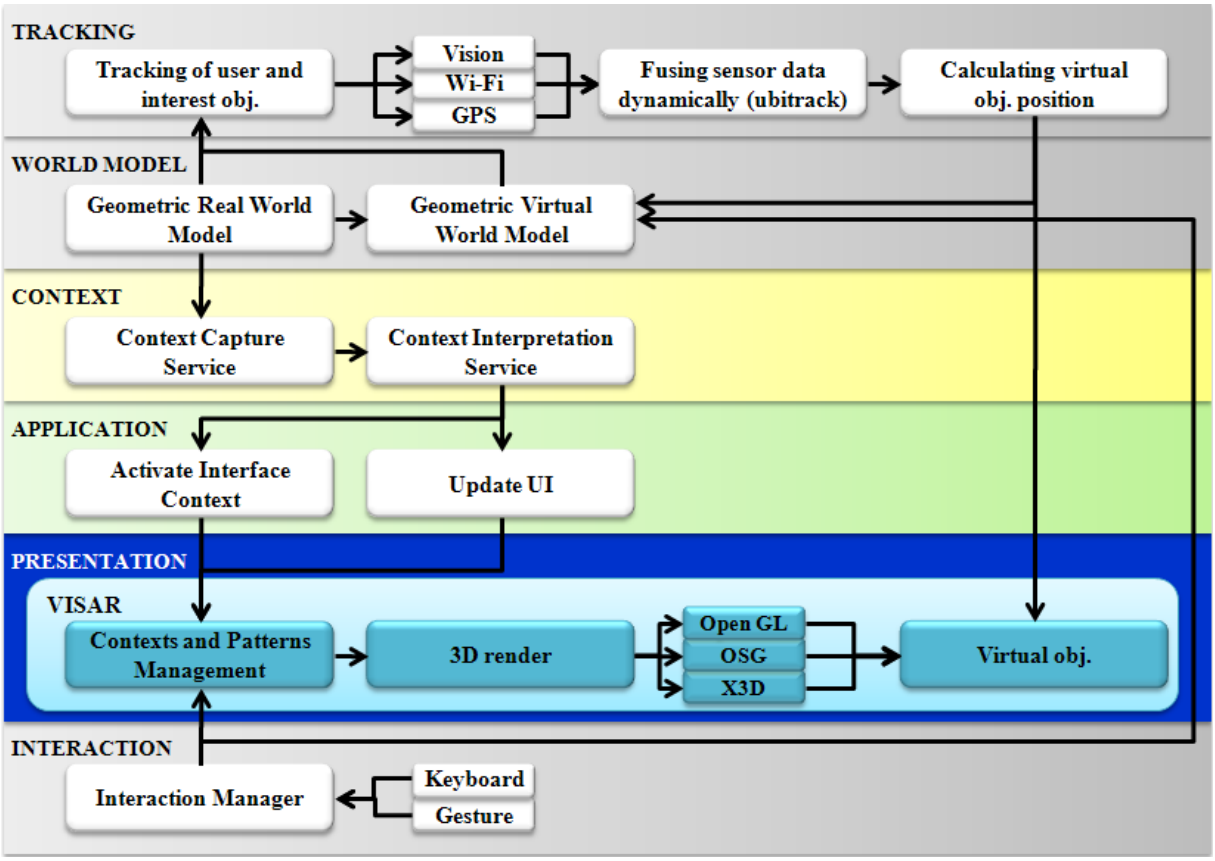

Figura38: Arquitetura de sistemas que utilizam o VISAR.

4.2.4 Interações do usuário com o VISAR

O VISAR é responsável apenas pela interface do usuário, portanto não suporta captura direta de entradas dos usuários. Porém, os padrões podem ter na sua API funções que suportam interações, para que assim a aplicação capture as entradas do usuário e as transforme em comandos para o VISAR.

Exemplos de interações variam de alterar propriedades físicas dos padrões (posição, tamanho, ângulo) a alteração de propriedades visuais (brilho, contraste, transparência), ou ativação de um comando (toque um objeto e ele se move).

4.2.5 Interação entre VISAR-IE e VISAR

A interação entre o VISAR-IE e o VISAR é feita através de documentos XML. Conforme já visto anteriormente, todos os padrões implementados no *framework* devem ter uma representação no editor, para que o projetista da interface possa utilizá-los. Atualmente não existe um modo fácil de acrescentar padrões ao editor, mas isso pode ser feito com um conhecimento básico da tecnologia na qual ele foi construído (Eclipse GEF).

Através do editor, a interface pode ser modelada, criando-se cenários para a aplicação e alocando os padrões a serem mostrados neles. Após projetar e configurar a interface no editor, esta é armazenada em um documento XML. O documento que contem a interface carrega todas as informações configuradas no editor que são importantes para o VISAR, como por exemplo: uma lista de cenários com seus nomes, prioridades e padrões que pertencem a eles, com cada padrão tendo um nome, tipo, descrição e propriedades como tamanho e posição na tela, dentre outras.

A figura 39 mostra um trecho de interface em formato de documento XML salvo no VISAR-IE e que gerou a estrutura da figura 37. Este documento é carregado no VISAR, que armazena os dados em sua estrutura interna (fig. 37) e espera por chamadas de funções da aplicação com atualizações a esses dados enquanto renderiza os padrões na tela.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <context name="0">
3     <priority>2</priority>
4     <pattern name="Visual Marker 0">
5        <properties description="airplane">
6           <markername text="airplane.x3d"></markername>
7           <size>10</size>
8        </properties>
9     </pattern>
10    <pattern name="Visual Marker 1">
11       <properties description="tank">
12          <markername text="tank.x3d"></markername>
13          <size>20</size>
14       </properties>
15    </pattern>
16 </context>
17 <context name="1">
18    <priority>1</priority>
19    <pattern name="Visual Marker 2">
20       <properties description="flag1">
21          <markername text="flag.x3d"></markername>
22          <size>5</size>
23       </properties>
24    </pattern>
```

Figura 39: Arquivo XML que descreve a interface com cenários e padrões.

4.3 Gerenciador de Rastreamento

Conforme visto nas seções anteriores, o VISAR é somente um *framework* de interface, portanto ele não faz rastreamento, mas na RA a interface necessita saber a posição do usuário para ser corretamente renderizada no ambiente (registrada). Portanto é recomendado o uso de um gerenciador de rastreamento que possa calcular e informar automaticamente, ou através da aplicação, ao VISAR, a posição do usuário, poupando esforço e tempo do desenvolvedor em realizar tal tarefa.

O ARToolkit pode ser usado como um gerenciador de rastreamento simples, somente rastreando marcadores na cena e informando suas posições ao VISAR. Porém um gerenciador especializado de rastreamento, como o Ubitrack, que foi descrito na seção 3.4 do Capítulo 3, é capaz de usar diferentes tecnologias de rastreamento e fundir os seus dados em tempo de execução.

O Ubitrack é um *framework* que auxilia aplicações de RA no rastreamento, coletando dados de sensores em tempo de execução e fundindo seus dados de localização, fornecendo à aplicação somente a informação necessária para gerar a matriz de projeção para RA. Sua estrutura de grafo de relação espacial representa a relação espacial entre todos os objetos de interesse na cena, incluindo o usuário. Sua função é atualizar esta estrutura e informar a aplicação. Desse modo, a aplicação sempre terá uma matriz de transformação/projeção precisa de todos os elementos da cena, gerando RA registrada (mais sobre o Ubitrack na seção 3.4).

Esta abstração de gerenciador de localização permite ao VISAR atuar com diferentes tecnologias de rastreamento, tornando-o independente de tecnologias específicas. O ideal é que o sistema de rastreamento se comunique diretamente com o VISAR, permitindo uma troca de dados mais rápida entre rastreadores e padrões, diminuindo assim o atraso do sistema e, portanto, o erro de registro.

4.4 Estudos de casos

Três estudos de caso foram modelados e implementados como parte deste trabalho de mestrado: 1) superfície multitoques integrada a RA; 2) *Web browser* com *drag-and-drop* para

a RA; 3) Sistema de RA de apoio a bombeiros em ambientes de emergência. Os três estudos de caso são descritos a seguir.

4.4.1 RA tangível em mesa multitoque

Esta aplicação foi implementada em uma mesa multitoque tangível[6], construída no laboratório WINDIS, como parte do projeto INCT-SEC. Nesta aplicação, o usuário deve controlar um avião para executar uma ação sobre um alvo. Na mesa um mapa é colocado como tela de fundo, que mostra a posição inicial do avião. O usuário pode então controlar o caminho que deve ser sobrevoado pelo avião e que é identificado através de pontos chave (*waypoints*), sendo o último *waypoint* considerado o alvo nesta aplicação.

Os comandos na mesa (inserção de *waypoints*, ações e o alvo) são realizados através de objetos tangíveis[7] (posicionam-se os objetos sobre o mapa na posição desejada). Como é possível redimensionar e mover o mapa, isto faz com que os objetos tangíveis saiam de suas posições originais. Para solucionar este problema a RA foi utilizada para representar os objetos tangíveis, de forma que eles sejam usados somente como um "carimbo" (colocados e removidos) para determinar suas posições no mapa. Depois dos objetos serem retirados, eles são representados virtualmente, em 3D, através de RA (sendo necessário um HMD ou celular para visualização do virtual).

Figura40: Mesa tangível multitoque com RA.

Para rastrear a posição da mesa, quatro marcadores foram utilizados (um em cada canto da mesa), para a RA poder ser registrada corretamente sobre a mesa e gerar uma visualização

[6] Superfície multitoque que permite interação colaborativa entre múltiplos usuários.

[7] Objetos reais utilizados como elementos de interação na superfície multitoque.

3D dos objetos de interesse. A figura 40 mostra esta aplicação executando, onde pode ser vista a mesa com seus quatro marcadores, a interface da mesa (um mapa do ambiente) e um avião reproduzido através da RA pelo *framework* VISAR.

Nesta aplicação, assim que o usuário posiciona o objeto tangível, a mesa multitoque informa ao VISAR o ID do objeto tangível utilizado e a posição onde este objeto está na mesa. A cada atualização no mapa, caso ele seja movido ou redimensionado, uma nova posição do objeto é passada para o *framework* atualizá-lo. Caso um objeto seja retirado da mesa pelo usuário, por exemplo, um *waypoint* já ativado, a mesa informa ao VISAR qual objeto foi retirado. Cada objeto foi modelado como pertencendo a um cenário. Quando um objeto é utilizado ou desativado, o VISAR ativa ou desativa o cenário correspondente ao objeto.

4.4.2 Sistema de Realidade Aumentada com "*Drag-and-Drop*"

Esta aplicação consiste de uma loja de decoração da internet, de onde móveis e objetos de decoração podem ser selecionados *(Drag)* e posicionados e visualizados em diferentes locais de uma casa *(Drop)* por meio de RA. Para esta aplicação, uma parceria foi feita com o pesquisador Sebastian Feuerstack[8]. A parte inicial do sistema funciona como uma loja padrão de móveis, onde o usuário escolhe suas compra e as coloca no carrinho. Depois disso o usuário pode utilizar um controlador (como *mouse*, Wii-Controller ou gestos) para realizar o *Drag-and-Drop* dos móveis/objetos comprados entre o *browser* e o ambiente real.

Para a visualização do ambiente aumentado é necessário colocar uma câmera em posição fixa com visualização do monitor. É feita então uma auto-calibração do sistema para capturar a posição do monitor relativo à câmera. Na figura 41 é mostrado o resultado dessa auto-calibração, que gera um quadro de sobreposição sobre o monitor. Este quadro representa a fronteira entre o virtual e o real. Quando o objeto, dentro do *browser*, é arrastado para fora deste quadro, ele se torna um objeto de Realidade Aumentada.

[8] Dr. Sebastian Feuerstack é pós doutorando no Departamento de Computação da UFSCar.

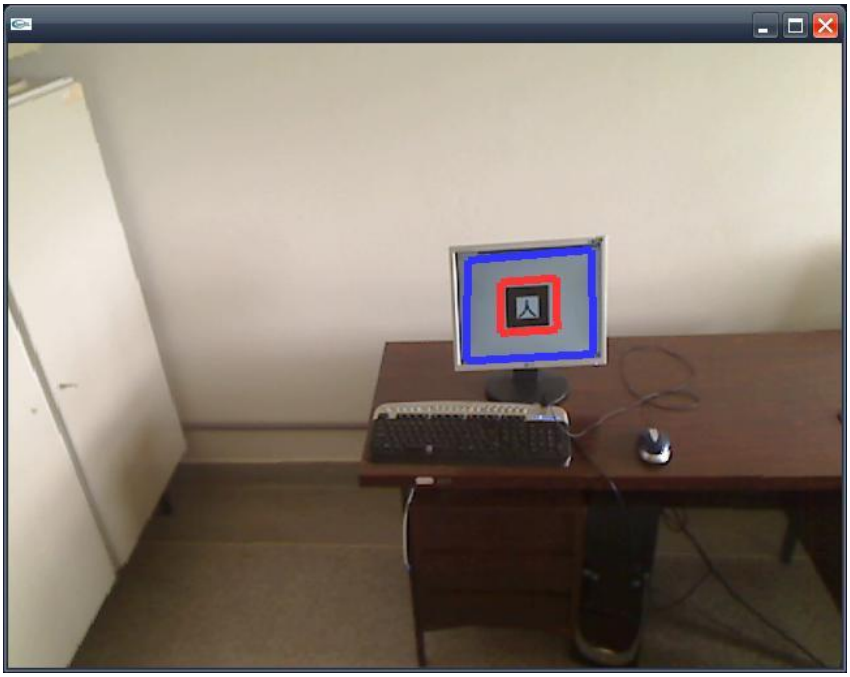

Figura41: Auto-calibração das coordenadas do sistema utilizando o próprio monitor para projetar o marcador.

A figura 42 mostra um móvel comprado sendo arrastado. Assim que o mouse passa o limite da caixa *"Select your products",* a visualização do *browser* muda para a da figura 43, onde o usuário passa a ver o vídeo de sua câmera com a Realidade Aumentada. Nesse modo de visualização o mouse se torna um ponteiro virtual 3D (ponto vermelho da figura 43) e os móveis são posicionados no ambiente pelo usuário usando a técnica *drag-and-drop*, sendo que quando um objeto é arrastado de volta para dentro do quadro que envolve o monitor, a visualização volta para a da figura 42.

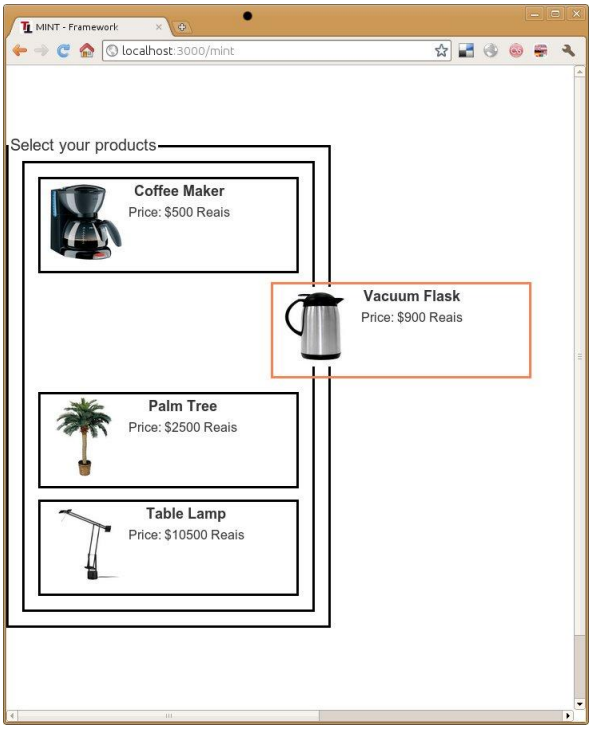

Figura42: Loja de mobília executando em um *web browser*, com um objeto sendo arrastado.

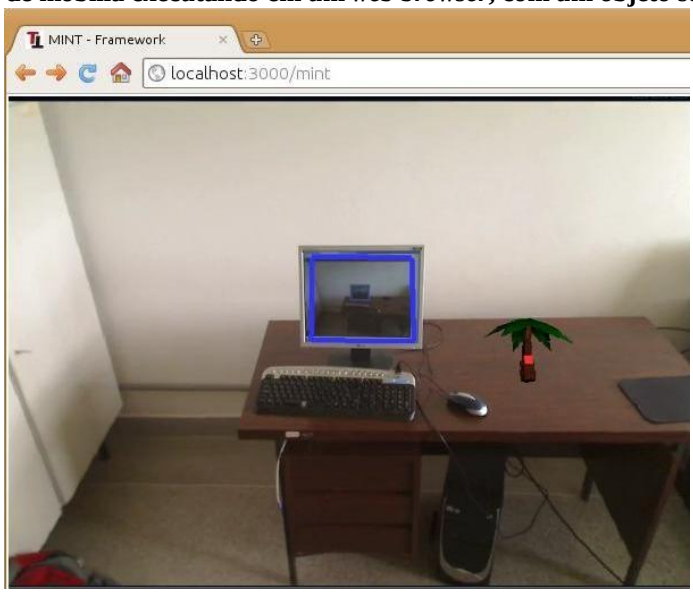

Figura43: Aplicação executando no *browser* no momento em que o usuário arrasta um objeto (planta) para o ambiente real.

A figura 43 mostra a aplicação executando, quando o usuário já posicionou diversos móveis/eletro-domésticos pelo ambiente real (luminária, cafeteira, planta e bule de café).

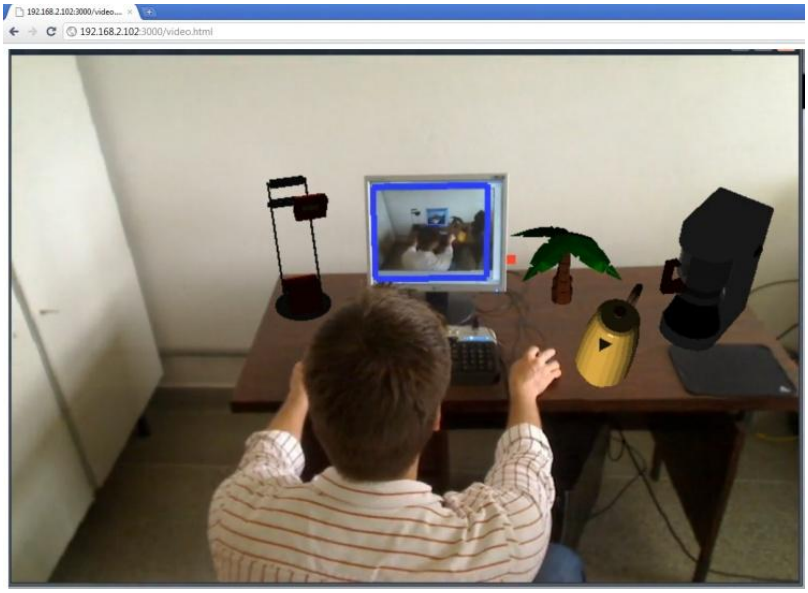

Figura44: Usuário utilizando a aplicação com vários móveis/eletro-domésticos já posicionados no ambiente.

4.4.3 Sistema de RA de apoio a bombeiros em ambientes de emergência (*Augmented Firefighter*)

O terceiro estudo de caso consiste de um sistema de RA de apoio a bombeiros em situações de emergência (tipicamente em ambientes internos), no qual diferentes interfaces e cenários (contextos) são possíveis. Este sistema, denominado *Augmented Firefighter* (AF), possui alto grau de complexidade e, devido à ausência dos recursos necessários para a complexa tarefa de rastreamento dos usuários em ambientes internos, a aplicação não pode ainda ser totalmente implementada, tendo sido somente modelada.

O AF tem como objetivo a adaptação da interface a diferentes cenários possíveis de serem encontrados em ambientes de emergência. Estes cenários podem refletir, por exemplo, tarefas a serem executadas pelo bombeiro, situações particulares de emergência, vítimas no ambiente, dentre outros.

O *middleware* do WINDIS [Beder2011]é responsável por receber informações do ambiente, por meio de sensores e repassar para o sistema de interpretação, que infere o tipo de fenômeno ocorrendo, que pode ser desde um simples aumento de temperatura a uma identificação de explosão iminente. Esta informação é repassada para o AF, que recebe também dados sobre as redes de localização e sua acurácia, ordens das equipes de comando e informações de estado físico do ambiente. O sistema poderá executar em um HMD com capacidade de pro-

cessamento ou em um *notebook*, com óculos para visualização de RA (HMD), uma câmera fixada ao capacete, além do equipamento necessário para monitorar a saúde do bombeiro e o aparelhamento para localização.

A entrada de ordens de comando para o bombeiro em ação é feita pelo comando superior do bombeiro no momento da operação. Como o sistema AF é voltado para ações e operações de equipes em ambientes de emergência, assume-se que o comando superior da operação tenha um sistema próprio para envio de ordens para os usuários do AF, como uma mesa de comando tangível, similar à citada no estudo de caso anterior.

Em um caso de incêndio, se este ocorre em uma sala fechada por um período prolongado de tempo, a porta/janela do local não pode ser aberta, pois isso faria o oxigênio entrar rapidamente o que causaria uma explosão (conhecida como *backdraft*). Neste caso a RA pode auxiliar mostrando que a porta/janela não pode ser aberta, ou quais portas/janelas não podem ser abertas no caso de ser mais de uma. Uma versão simples e inicial do sistema é mostrada na Figura 45.

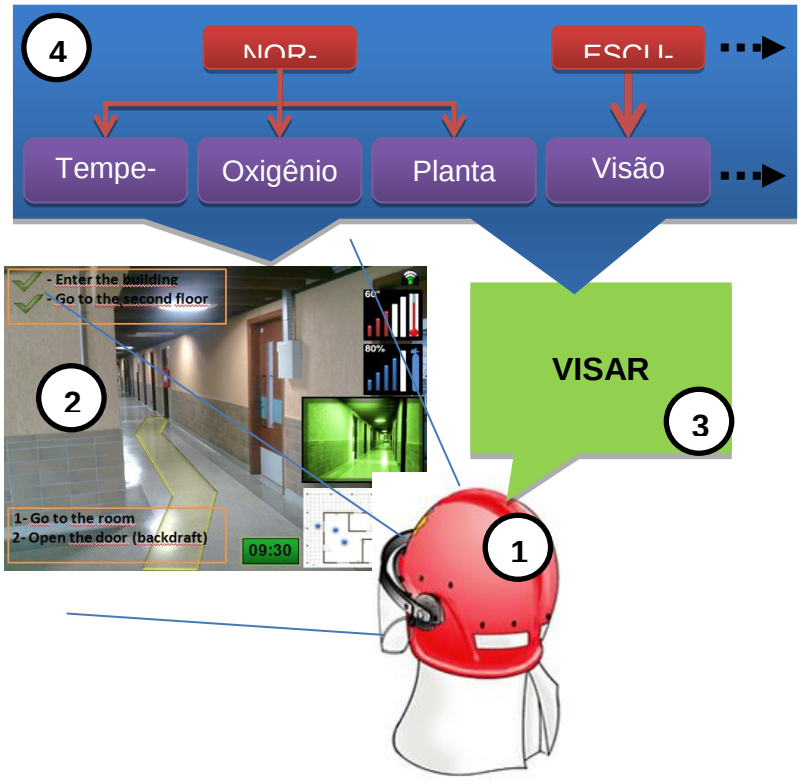

Figura 45: Augmented Firefighter.

Na Figura 45, a parte um (1) é o capacete (HMD) que tem, além do visor onde é projetada a interface, o *hardware* necessário para gerar RA e suportar este sistema, como processa-

dor, placa gráfica e *display* ótico, assim como alguns sensores como o de temperatura, de localização inercial (6DOF) e o suporte a redes sem fio.A parte dois (2) é a interface que o bombeiro vê no capacete. A mesma interface é mostrada na figura 46 com mais detalhes.A parte três (3) é o *framework* VISAR, que executa no capacete através de uma aplicação que o controla. O VISAR é responsável por criar a interface da parte 2 e manter a estrutura da parte 4, que, por sua vez, é a estrutura interna do *framework* onde os padrões e cenários estão alocados. Esta estrutura é refletida na interface do usuário.

A figura 46 mostra vários possíveis padrões de interface. Porém, em uma situação real eles são selecionados por cenários e poucos devem aparecer simultaneamente para não obstruir a visão do bombeiro ou sobrecarregá-lo com informações. Os padrões de interface presentes na figura são listados abaixo:

- Lista de tarefas completadas e a serem feitas;
- Visão de câmera de visão noturna (ativada apenas se o ambiente estiver escuro);
- Mapa 2D do ambiente, mostrando o local de ocorrência do fogo e também a localização de outros bombeiros;
- Valor da temperatura no local, assim como a quantidade de oxigênio restante no galão do bombeiro;
- Horas ou cronômetro com o tempo decorrido desde o início da ação;
- Grau de conectividade do equipamento com a rede sem fio que está alimentando a aplicação com os dados do ambiente;
- Caminho virtual mostrando a rota até o destino;
- “X” em portas que não devem ser abertas;
- Representação 3D de pessoas (no caso vítimas) no ambiente e suas localizações.

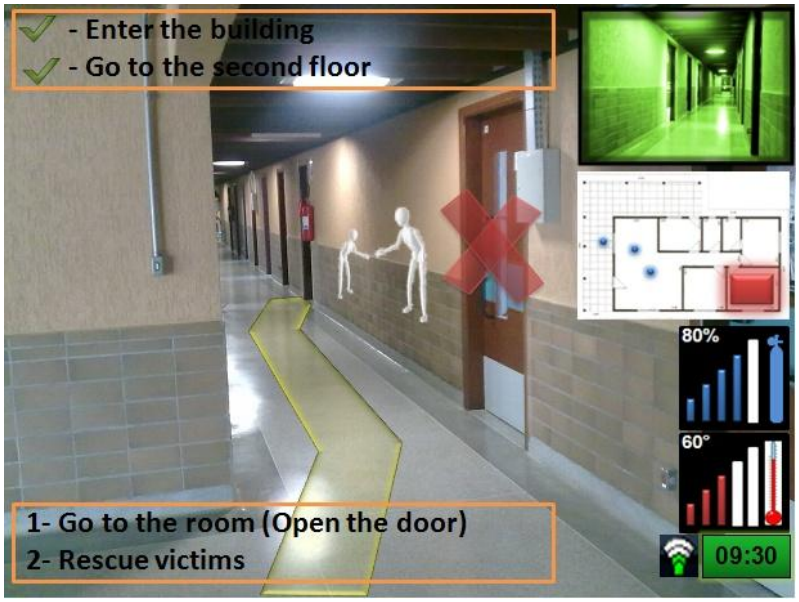

Figura46:Interface do *Augmented Firefighter*.

4.4.3.1 Localização e Rastreamento do sistema

Para a viabilização do sistema AF descrito acima, um sistema de localização é necessário. Como grande parte das ações dos bombeiros ocorre em ambientes internos, uma técnica de localização e rastreamento de baixo custo foi desenvolvida como parte deste trabalho de mestrado, para melhorar o registro, que prescinde de uma infraestrutura prévia. A técnica é explicada no Anexo II.

4.4.3.2 Mapeamento dos Contextos

Para o sistema AF, diferentes contextos foram identificados. O mapeamento destes contextos para o AF é mostrado na tabela 3, no Anexo III. Esta tabela será transformada em um método de modelagem para representação e ficará instanciada em um *middleware* com serviço de interpretação de contexto (como o WINDIS [BEDER2011]). Assim o sistema só precisa consultar o *middleware* para obter resultado dos contextos desejados.

4.4 Considerações Finais

Este capítulo apresentou as ferramentas VISAR-IE e VISAR para modelagem, implementação e visualização de sistemas de RA. Três estudos de caso criados com essas ferramentas foram apresentados: Integração de RA com interfaces tangíveis; *Drag-and-drop* utilizando RA e a modelagem de um sistema complexo de apoio a bombeiros no combate ao fogo. Para

este último estudo de caso, diferentes tecnologias foram utilizadas para melhorar a precisão da localização em ambientes internos utilizando recursos do sistema Ubitrack. O próximo capítulo descreve as conclusões.

.

5 CONCLUSÕES

Apesar do alto potencial de aplicação de RA nas mais diversas áreas, várias limitações ainda precisam ser superadas para que a RA se torne amplamente difundida, dentre elas: forte acoplamento dos projetos de RA ao *hardware* dos dispositivos; complexidade dos mecanismos que correlacionam as informações de contexto ao mundo real, dificultando e tornando cara a criação e difícil a manutenção de interfaces de aplicações de RA; necessidade de desenvolvimento de dispositivos mais eficazes e de mais fácil uso para visualização e interação do usuário com os objetos virtuais; necessidade de desenvolvimento de ferramentas para projetos de interfaces de RA cientes de contexto; possibilidade de calibração de equipamentos em tempo de execução; mecanismos precisos e eficientes de rastreamento do usuário e de objetos de interesse, dentre outros. Mais ainda, além da existência de um número limitado de sistemas e soluções para o desenvolvimento de RA ciente de contexto, estas soluções são limitadas a contextos tradicionais como localização ou iluminação.

Este trabalho apresentou um *framework* de interfaces de usuário de Realidade Aumentada e um editor para desenvolvimento ágil de interfaces adaptativas, com as seguintes características inovadoras: acelera o desenvolvimento do projeto e o processo de implementação de interfaces de RA, através de padrões de interface, permitindo a criação de interfaces para uma ampla variedade de aplicações; facilita o reúso de interfaces; permite a modelagem de adaptações de interfaces de RA que possam ocorrer em tempo de execução, em resposta a qualquer modificação nos contextos do ambiente; torna possível que especialistas na área da aplicação criem as interfaces do usuário, sem a necessidade de conhecimentos de programação, ao utilizar uma ferramenta de autoria (VISAR-IE).

Assim como é importante propor soluções inovadoras para problemas existentes, é importante também conhecer as limitações das soluções propostas. Algumas delas incluem: 1) embora a solução proposta ofereça uma forma de modelar as interfaces por cená-

rios/contextos, esta não tem a capacidade de interpretá-los e, portanto passa essa carga para as aplicações; 2) o *framework* não oferece suporte direto à entrada de dados por usuários, gerando o mesmo problema do caso anterior, em que a aplicação deve se tornar responsável por isso; 3) a arquitetura do sistema (componentes) pode acabar gerando um atraso inconveniente para aplicações de RA, já que o *framework*, que vai renderizar a interface é o ultimo a receber as informações dos componentes de rastreamento e entrada de dados; 4) a modelagem dos cenários no editor não auxilia o desenvolvedor da aplicação a entender o momento em que cada um deve ser usado. Por exemplo, no caso de um especialista modelar a interface, ele tem que manter contato com os programadores da aplicação para instruí-los sobre quando cada interface deve ser mostrada para o usuário; 5) no editor, somente os componentes de interface 2D podem ser posicionados, portanto ainda falta uma técnica para posicionar os componentes 3D na cena de forma fácil pelos desenvolvedores.

Estas limitações poderiam ser superadas, futuramente, das seguintes formas: para (1) e (2), fornecer ferramentas adicionais que realizem essas funções e que tenham sido testadas e funcionem bem em conjunto com o *framework* (como o Ubitrack para rastreamento e o *middleware* do WINDIS para interpretação de contextos); para (3) adotar e otimizar um gerenciador de rastreamento para ser usado com o *framework*, assim o atraso não vai gerar um alto erro de registro; para (4) utilizar uma metodologia de desenvolvimento que una desenvolvedor com especialista; e para (5) estender o editor para suportar posicionamento de objetos 3D no local onde devem ficar no ambiente.

Em relação à lista de objetivos definida na seção 1.2 e tendo em vista o sucesso na implementação dos estudos de caso, acreditamos que as ferramentas propostas e desenvolvidas como parte deste projeto podem contribuir para superar alguns dos desafios no desenvolvimento de interfaces de RA. As ferramentas propostas facilitam a implementação de interfaces de RA, utilizando componentes para desenvolver e criar um ambiente para modelagem de interfaces. Como forma de divulgação desta contribuição o código fonte das ferramentas desenvolvidas será disponibilizado para *download*. Uma maior divulgação também ajudaria a instigar desenvolvedores a criar interfaces que respondam a contextos.

5.1 **Contribuições**

Quatro artigos foram produzidos como resultado desta dissertação, sendo que um já foi

publicado e três foram submetidos para eventos e revistas nacionais e internacionais.

Artigo publicado:

1. Oliveira, A. C. M. ; Araujo, R. B. ; Cavalheiro, C.A.M. ; Moreira, J. "Framework Ciente de Contexto para Visualização de Interfaces de Realidade Aumentada". In: Simpósio de Realidade Virtual e Aumentada, 2010, Natal. Anais do XII Simpósio de Realidade Virtual e Aumentada, 2010.

2. Feuerstack, S. Oliveira, A. C. M. ; Araujo, R. B."Model-based Design of Interactions that can bridge Realities - The Augmented "Drag-and-Drop". In: Simpósio de Realidade Virtual e Aumentada, 2011, Uberlândia. Anais do XIII Simpósio de Realidade Virtual e Aumentada, 2011.

Artigos Submetidos:

1. Oliveira, A. C. M. ; Araujo, R. B. "Creation and Visualization of Context Aware Augmented Reality User Interfaces with VISAR". Submetido ao evento: AVI 2012 Advanced Visual Interfaces.

2. Oliveira, A. C. M. ; Araujo, R. B. "Criação e Visualização de Interfaces do Usuário Cientes de Contexto para Realidade Aumentada com o VISAR". Submetido a Revista do IEEE America Latina.

Foi ministrado o seguinte minicurso no decorrer do projeto:

- Oliveira, A. C. M. "Realidade Aumentada: Fundamentos, Desafios e Desenvolvimento de Aplicações". In: XVIII Congresso de Iniciação Científica da UFSCar.

As seguintes ferramentas e aplicações foram geradas como resultado deste trabalho: *framework* VISAR, editor VISAR-IE e as aplicações loja da *web* com *drag-and-drop* e RA na mesa colaborativa multitoque.

5.2 **Trabalhos Futuros**

Como trabalho futuro foi planejado expandir o editor VISAR-IE, tornando possível posicionar não somente padrões 2D na tela, mas também padrões 3D na cena (no ambiente), adicionando o modelo 3D do ambiente no editor. Desse modo o desenvolvedor vai poder ajustar a posição de padrões 3D já na fase de modelagem (atualmente essa tarefa é realizada durante a fase de implementação). Também é possível estender o VISAR e aprimorá-lo, melhorando a

conexão entre o gerenciador de rastreamento (talvez adotando um gerenciador fixo). Novos padrões também podem ser desenvolvidos, fazendo o VISAR atender a um maior número de aplicações.

REFERÊNCIAS BIBLIOGRÁFICAS

Ahad04 Ahad, R. Hossain, S. “Augmented reality and its challenges”, *SICE 2004 Annual Conference*. vol. 2, Agosto 2004, pp. 1041 - 1043.

Araujo08 Araujo, R. Rocha, R. Campos, M. Boukerche, A. “Um Serviço de Interpretação de Contexto para Redes de Sensores Sem Fio no Domínio do Gerenciamento da Emergência”, *Em: II Workshop on Pervasive and Ubiquitous Computing - WPUC* (Evento paralelo ao SBAC-PAD), Campo Grande, 2008, Anais do II Workshop on Pervasive and Ubiquitous Computing - WPUC, 2008.

Artoolkit09 ARToolKit. Software Free e Open Source. Disponível em: <http://www.hitl.washington.edu/artoolkit/> Acesso em 23.02.2010.

ARWalker11 http://www.nttdocomo.com/

Azuma93 Azuma , R. “Tracking Requirements for Augmented Reality,” *Communications of the ACM*, vol. 36, no. 7, Julho 1993, pp. 50-51.

Azuma97 Azuma, R. “A Survey of Augmented Reality”, *Presence: Teleoperators and Virtual Environment*, vol. 6, no. 4, Agosto 1997, pp. 355-385.

Azuma99 Azuma, R. Lee, J. Jiang, B. Park, J. You, S. Neumann, U. “Tracking in Unprepared Environments for Augmented Reality Systems”, *Computers and Graphics*, vol. 23, no. 6, Dezembro 1999, pp. 787-793.

Azuma01 Azuma, R. Baillot, Y. Behringer, R. Feiner, S. Julier, S. MacIntyre, B. “Recent Advances in Augmented Reality,”. *IEEEComputer Graphics and Applications*, vol. 21, no. 6, 2001,pp. 34-37.

Azuma03 Azuma, R. Furmanski, C. “Evaluating Label Placement for Augmented Reality View Management”, *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality,*2003,pp.66.

Azuma06 Azuma, R. Neely, H. Daily, M. Leonard, J. “Performance Analysis of an Outdoor Augmented Reality Tracking Systemthat Relies Upon a Few Mobile Beacons”, *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2006, pp. 101-104.

Baldauf07 Baldauf, M. “A survey on context-aware systems”, *International Journal Of Ad Hoc And Ubiquitous Computing*, vol. 2, no. 4, 2007, pp. 263-277.

Becker08 Becker, B. Huber, M. Klinker, G. “Utilizing RFIDs for Location Aware Computing”, *Proceedings of the 5th international conference on Ubiquitous*

Intelligence and Computing, Noruega, 2008, pp. 216-228.

Beder11 Beder, D. M., Araujo, R. B. "Towards the Definition of a Context-Aware Exception Handling Mechanism."*Workshop on Exception Handling in Contemporary Software Systems* (EHCoS), São Jose dos Campos, SP, 2011

Bell 00 Bell, B. Feiner, S. "Dynamic Space Management for User Interfaces", *ACM Symp. on User Interface Software and Technology*, 2000, pp. 238-248.

Bell01 Bell, B. Feiner, S. Hollerer, T. "View Management for Virtual and Aug mented Reality", *Proceedingsof the 14th annual ACM symposium on User interface software and technology*, Estados Unidos, 2001, pp. 101-110.

Bimber05 Bimber, O. Raskar, R. "Spatial Augmented Reality Merging Real and Virtual Worlds", A K Peters LTD, 2005.

Bradesco09 http://www.bradesco.com.br/html/content/hotsite/presenca_iphone/#/sobre-o-aplicativo

Broll05 Broll, W. Lindt, I. Ohlenburg, J. Herbst, I. Wittkamper, M. Novotny, T. "An Infrastructure for Realizing Custom-TailoredAugmented Reality User Interfaces",*IEEE Transactions on Visualization and Computer Graphics*, vol. 11,

Broll05 Broll, W. Lindt, I. Ohlenburg, J. Herbst, I. Wittkamper, M. Novotny, T. "An Infrastructure for Realizing Custom-TailoredAugmented Reality User Inter faces",*IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 6, Novembro 2005, pp. 722-733.

Broll08 W. Broll; J. Herling; L. Blum; "Interactive Bits: Prototyping of Mixed Reality Applications and Interaction Techniques through Visual Programming", *IEEE Symposium on 3D User Interfaces*, 2008, pp. 109-115.

Brooks96 Brooks, Frederick P. Jr. "The Computer Scientist as Toolsmith II", *CACM 39*, vol. 39, Março 1996, pp. 61-68.

Brown97 Brown, P. Bovey, J. Chen, X. "Context-aware applications: from the labora tory to the marketplace", *Personal Communications IEEE*, vol. 4, no. 5, 1997, pp. 58-64.

Buchmann08 Buchmann, V. Billinghurst, M. Cockburn, A. "Directional Interfaces for Wearable Augmented Reality", *Proceedings of the 9th ACM SIGCHI New Zealand Chapter's International Conference on Human-Computer Interac tion: Design Centered HCI*, Nova Zelândia, 2008, pp. 47-54.

BuildAR08 BuildAR. Software Free e Open Source. Disponível em: < http://www.hitlabnz.org/wiki/BuildAR/> Acesso em 23.02.2010.

Chen04 Chen, H. "An Intelligent Broker Architecture for PervasiveContext-Aware Systems", PhD Thesis, University of Maryland, 2004.

Christiansson00 Christiansson, P. "Knowledge Representations and Information Flow in the IntelligentBuilding", *Proceedings of the Eighth International Conference on Computing in Civil and Building Engineering (ICCCBE-VIII)*, 2000.

Coelho03 Coelho, E. MacIntyre, B. "High-Level Tracker Abstractions for Augmented Reality System Design", *Em International Workshop on Software Technolo gy for AR Systems (STARS '03)*, Outubro 2003.

Coelho04 Coelho, E. MacIntyre, B. Julier, S. "(osgAR): A Scenegraph with Uncertain Transformations", *Em Proc. Third Int'l Symp. Mixed and Augmented Reali ty (ISMAR '04)*, Novembro 2004, pp. 6-15.

Dey00 Dey, A. Abowd, G. "Towards a Better Understanding of Context and Con text-Awareness", *Em Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000)*, Abril 2000.

De Crescenzio11 De Crescenzio, F.; Fantini, M.; Persiani, F.; Di Stefano, L.; Azzari, P.; Salti, S.; "Augmented Reality for Aircraft Maintenance Training and Operations Support", *In: IEEE Computer Graphics and Applications*, vol. 31, no. 1, 2011, pp.96-101.

DiVerdi07 DiVerdi, S., Höllerer, T., "GroundCam: A Tracking Modality for Mobile Mixed Reality",*IEEE Virtual Reality*, Março 2007, pp. 75-82.

Educause Disponível em: http://net.educause.edu/ir/library/pdf/ELI7007.pdf.

Fua07 Fua, P. Lepetit, V. "Vision based 3D tracking and pose estimationformixed reality",*EmergingTechnologies of Augmented Reality Interfaces and De sign*, 2007,pp. 43-63.

Fuchs98 Fuchs, H. Livingston, M. Raskar, R. Colucci, D. Keller, K. State, A. Crawford, J. Rademacher, P. Drake, S. Meyer, A. "Augmented reality visualization for laparoscopic surgery" *In: Medical Image Computing and Computer-Assisted Interventation — MICCAI'98*, 1998, pp. 934-943.

GhostWire11 www.ghostwiregame.com

Hollerer01a Höllerer, T. Feiner, S. Hallaway, D. Bell, B. Lanzagorta, M. Brown, D. Juli er, S. Baillot, Y. Rosenblum, L. "User Interface Management Techniques for Collaborative MobileAugmented Reality",*Computers &*Graphics, vol. 25, no. 5, 2001, vol. 1496, pp. 799-810.

Hollerer01b Höllerer, T. Feiner, S. Hallaway, Tinna , N. Feiner, S. "Steps Toward Ac commodating Variable Position Tracking Accuracy in aMobile Augmented Reality System", *In Proc. AIMS'01*, 2001, In In Proc. AIMS'01 (2001), pp. 31-37.

Holloway95 Holloway, R. L. "Registration Errors in Augmented Reality Systems", 1996.

Huber07 Huber, M. Pustka, D. Keitler, P. Echtler, F. Klinker, G. "A System Architec ture for Ubiquitous Tracking Environments", *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, PP.1-4.

iARM11 iARM; relatório técnico, acesso em: http://spill.tanagram.com/downloads/iARM_Final_Report.pdf

Indulska03 Indulska, J. Sutton, P. "Location management in pervasive systems", *CRPITS'03: Proceedings of the Australasian Information Security Work shop*,2003, pp.143–151.

Julier00 Julier, S. Baillot, Y. Brown, D. Lanzagorta, M. "Information Filtering for Mobile Augmented Reality", *IEEE Computer Graphics and Applications*,

vol. 22 , no. 5, 2002, pp. 12-15.

Kato99a H. Kato; M. Billinghurst; B. Blanding; R. May, "ARToolKit", Technical report, Hiroshima City University, 1999.

Kato99b Kato, H. Billinghurst, M. "Marker Tracking and HMD Calibration for a Vid eo-based Augmented Reality Conferencing System",*Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, 1999, pp.85.

Keitler10 Keitler, P. Pustka, D. Huber, M. Echtler, F. Klinker, G. "Management of Tracking for Mixed and Augmented Reality Systems", *The Engineering of Mixed Reality Systems, Human-Computer Interaction Series*, Londres, 2010, pp. 251-273.

Klein07 Klein, G. Murray, D. "Parallel Tracking and Mapping for Small AR Work spaces", *Proceedings of the 2007 6th IEEE and ACM International Sympo sium on Mixed and Augmented Reality*, 2007, pp.1-10.

Kinect10 http://www.xbox.com/pt-br/kinect

Korpipää03 Korpipää, P. Mäntyjärvi, J. Kela, J. Keränen, H. Malm,E. "Managing Con text Information in Mobile Devices", *IEEE Pervasive Computing*, vol. 2, no. 3, 2003, pp. 42-51.

Kulas04 Kulas, C. Sandor, C. Klinker, G. "Towards a Development Methodology for AugmentedReality User Interfaces", *Proc. of the International Workshop exploring the Design and Engineering of Mixed Reality Systems*, 2004.

Lafortune90 Lafortune, J. Lecours, M. "Measurement and Modeling of Propagation Lossesin a Building at 900 MHz", *IEEE Transactions on Vehicular Tech nology*, vol. 39, no. 2, 1990, pp. 101-108.

LaMarca05 LaMarca, A. Hightower, J. Smith, I. Consolvo, S. "Self-Mapping in 802.11 Location Systems", *UbiComp 2005: Ubiquitous Computing, 7th Interna tional Conference*, Japão, 2005, pp. 87-104.

Lederman05 F. Ledermann; D. Schmalstieg; "APRIL A High-Level *Framework* for Creating Augmented Reality Presentations", *In Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*, 2005, pp. 187-194.

Lee08 Lee, W. Woo, W. "Exploiting Context-awareness inAugmented Reality Ap plications", *ISUVR2008*, 2008, pp. 51-54.

Liarokapis04 Liarokapis, F. White, M. Lister, P. "Augmented Reality Interface Toolkit", *Proc. International Symposium on Augmented and Virtual Reality*, IV04-AVR, Londres, 2004, pp. 761-767.

Liu06 Liu, A. Hile, H. Kautz, H. Borriello, G. Brown, P. Harniss, M. Johnson, K. "Indoor Wayfinding: Developing a Functional Interface forIndividuals with Cognitive Impairments", *Proceedings of the 8th international ACM SI GACCESS conference on Computers and accessibility*, 2006, pp. 95-102.

Livingston08 Livingston, M. Ai, Z. "The Effect of Registration Error on Tracking Distant Augmented Objects", *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2008, pp. 77-86.

MacIntyre00 MacIntyre, B. Coelho, E. “Adapting to dynamicregistration errors using level of error (LOE) filtering”, *EmInternational Symposium on Augmented Reality (ISAR ’00)*, Outubro 2000, pp. 85–88.

MacIntyre02 MacIntyre, B. Julier, S. “Estimating and Adapting to Registration Errors in Augmented Reality Systems”, in*IEEE Virtual Reality (VR 2002)*, Março 2002.

MacIntyre04 MacIntyre, B. Gandy, M. Dow, S. Bolter, J. “DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences”, *Em Proceedings of the 17th annual ACM symposium on User Interface Software and Technolo gy*, 2004, pp. 197–206.

Matthew96 John, V. Matthew, C. George, W. Randy, P. “3D Magic Lenses”, *Em proce edings ACM Symposium on User Interface Software and Technology*, 1996, pp. 51-58.

Mendez06 Mendez, E. Kalkofen, D. Schmalstieg, D. “Interactive Context-Driven Vi sualization Tools for Augmented Reality”, *Em Proceedings of ISMAR 2006*, pp. 209-218.

Mendez07 Mendez, E. Schmalstieg, D. “Adaptive Augmented Reality Using Context Markup and Style Maps”,*Em Poster Proceedings International Symposium on Mixed and Augmented Reality*, Japão, Novembro 2007, pp. 267-268.

Milgram94 Milgram, P. Kishino, F. “A Taxonomy of Mixed Reality Virtual Dis- plays”, *IEICE Transactions on Information and Systems E77-D*, 1994, pp. 1321-1329.

Naimark05 Naimark, L. Foxlin, E. “Encoded LED System for Optical Trackers”, *Pro ceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2005, pp.150-153.

Newman04 Newman, J. Wagner, M. Bauer, M. MacWilliams, A. Pintaric, T. Beyer, D. Pustka, D. Strasser, F. Schmalstieg, D. Klinker, G. “Ubiquitous tracking for augmented reality”, *Em Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR’04)*, EUA, Novembro 2004.

Newman06 Newman, J. Barakonyi, I. Andreas, S. Schmalstieg, D.”Wide-Area Tracking Tools for Augmented Reality”, *Advances in Pervasive Computing*, 2006, pp. 143-146.

Nilsson09 Nilsson, S. Johansson, B. Jonsson, A. “Using AR to support cross- organisational collaboration in dynamic tasks”, *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, 2009, pp. 3-12.

Oh09 Oh, S. Woo, W. “CAMAR: Context-aware Mobile Augmented Reality in Smart Space”, *Em Proc. of IWUVR2009*, 2009, pp. 48-51.

Park99a Park, J. You, S. Neumann, U. “Natural Feature Tracking for Extendible Ro bust Augmented Realities”, *Proceedings of the international workshop on Augmented reality : placing artificial objects in real scenes: placing artifi cial objects in real scenes*, 1999, pp. 209-217.

Park99b Park, J. Jiang, B. Neumann, U. “Vision-based pose computation:robust and accurate augmented reality tracking”,*Proceedings of the 2nd IEEE and*

ACM International Workshop on Augmented Reality, 1999, pp. 3.

Park08 Park, Y. Lepetit, V. Woo, W. "Multiple 3D Object Tracking for Augmented Reality", *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2008, pp. 117-120.

Parrot10 http://ardrone.parrot.com/parrot-ar-drone/usa/

Peternier06 Peternier, A. Vexo, F. Thalmann, D. "Wearable Mixed Reality System In Less Than 1 Pound", *Proceedings of the 12th Eurographics Symposium on Virtual Environment*, 2006, pp.35-44.

Petersen09 Petersen, N. Stricker, D. "Continuous Natural User Interface: Reducing the Gap Between Real and Digital World", *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, 2009, pp. 23-26.

PSMove10 http://us.playstation.com/ps3/playstation-move/

Pustka06a Pustka, D. "Construction of data flow networks for augmented reality appli cations", *In Proceedingsof the DritterWorkshop Virtuelle und Erweiterte Realit¨at der GI-Fachgruppe VR/AR*, Alemanha, 2006.

Pustka06b Pustka, D. Huber, M. Bauer, M. Klinker, G. "Spatial relationship patterns: Elements ofreusable tracking and calibration systems",*Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Re ality*, 2006, pp. 88-97.

Putska08 Pustka, D. Klinker, G. "Dynamic Gyroscope Fusion in Ubiquitous Tracking Environments", *Proceedings of the 7th IEEE/ACM International Sympo sium on Mixed and Augmented Reality*, 2008, pp. 13-20.

Reicher03a Reicher ,T. MacWilliams , A. Brügge , B. Klinker, G. "Results of a Study on Software Architectures for Augmented Reality Systems", *Proceedings of the The 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2003, pp. 274.

Reicher03b Reicher, T. MacWilliams, A. Bruegge, B. "Towards a System of Patterns for Augmented Reality Systems", *In International Workshop on Software Tech nology for Augmented Reality Systems*, 2003.

Reitmayr01 Reitmayr, G. Schmalstieg, D. "OpenTracker – An OpenSoftware Architecture for Reconfigurable Tracking based onXML",*In Proceedings of the ACM Symposium on Virtual Reality Software & Technology (VRST)*, Canada,2001, pp. 47-54.

Robertson02 Robertson, C. MacIntyre, B. "Adapting to Registration Error in an Intent-Based Augmentation System", *Em Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology (UIST 2002)*, França, Outubro 2002, pp.27–30.

Robertson09 Robertson, C. MacIntyre, B. Walker, B. "An Evaluation of Graphical Con text as a Means for Ameliorating the Effects of Registration Error", *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 2, Abril 2009, pp. 179-192.

Seffah10 A. Seffah, "The evolution of design patterns in HCI: from pattern languages

to pattern-oriented design", in Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems, 2010, pp. 4-9.

Seidel92 Seidel, S. Rappaport, T. "914 MHz Path Loss Prediction Models forIndoor Wireless Communications inMultifloored Buildings*", IEEE Transactions on Antennas and Propagation*, vol. 40, no. 2, 1992, pp. 207-217.

Schall09 Schall, G. Wagner, D. Reitmayr, G. Taichmann, E. Wieser, M. Schmalstieg, D. Hofmann-Wellenhof, B. "Global Pose Estimation using Multi-Sensor Fu sionfor Outdoor Augmented Reality", *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, 2009, pp. 153-162.

Schilit94 Schilit, B. Theimer, M. "Disseminating active map information to mobile hosts", *IEEE Network*, vol. 8, no. 5, 1994, pp. 22-32.

Schmalstieg02 Schmalstieg, D. Fuhrmann, A. Hesina, G. Szalavári, Z. Encarnaçäo, L. Ger vautz, M. Purgathofer, W."The Studierstube Augmented Reality Project." *Presence: Teleoperators and Virtual Environments*, vol. 11, no. 1, 2002, pp. 33–54.

Schreiber08 Schreiber, W."Augmented Reality Applications within Industry. " *ISMAR, Workshop on Industrial Augmented Reality: Needs and Solutions*, 2008.

Seichter08 Seichter, H. Looser, J. Billinghurst, M. "ComposAR: An intuitive tool for authoring AR applications", *Em 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2008, pp.177-178.

Sprite10 http://game.sprite.com.br/game2zero/

State96 State, A. Hirota, G. Chen, D. Garrett, W. Livingston, M. "Superior Augmented Reality Registration by IntegratingLandmark Tracking and Magnetic Tracking", *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, PP. 429-438.

Steggles07 Steggles, P. Gschwind, S."The Ubisense Smart Space Platform", *Advances in Pervasive-Computing, Adjunct Proceedings of the Third International-Conference on Pervasive Computing*, vol. 191, 2007.

Strang04 Strang, T. Linnhoff-Popien, C. "A Context ModelingSurvey", *First Interna tional Workshop on Advanced Context Modelling, Reasoning and Manage ment (UbiComp)*, 2004.

Swan05 J.E. Swan II; J.L. Gabbard; "Survey of User-Based Experimentation in Augmented Reality", *in Proceedings of 1st International Conference on Virtual Reality*, Estados Unidos, 2005, pp. 22-27.

TokStok09 http://www.plantavirtual.com.br/website/

Vitzthum06 A. Vitzthum; "SSIML/AR: A Visual Language for the AbstractSpecification of AR User Interfaces", *IEEE Symposium on 3D User Interfaces*, 2006, pp. 135-142.

Wii06 http://wii.com/

Wikitude08 http://www.wikitude.org/en/

Winograd01 Winograd, T. “Architectures for context”, *Human-Computer Interaction (HCI) Journal*, vol. 16, no. 2, 2001, pp.401–419.

Zhou08 Zhou, F. Duh, B. Billinghurst, M. “Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR”, *In proceedings of ISMAR*, Setembro 2008, pp. 193-202.

ANEXO I. Código fonte de duas classes do *framework* VISAR

Context.h

Representa a classe dos cenários/contextos e é uma lista (vetor) com cada um dos padrões que pertencem ao cénario:

```
#ifndef  CONTEXT_H
#define CONTEXT_H

#include "interface_pattern.h"
#include "context.h"
#include <cstdlib>
#include <iostream>
#include "pattern_text.h"

class context
{
   protected:

       bool status; //1 on, 0 off;
       char *name;
       int priority;
       interface_pattern **pat_list;
       int num_pat;

    public:

       context(){};

    void init(int num_pat, int pri, char *nam)
{
     pat_list = new interface_pattern*[num_pat];
    priority=pri;
    name=nam;
    this->num_pat=num_pat;
    status = false;
}

~context()
{
  free(pat_list);
}

bool active()
{
  return status;
```

```
}
void activate()
{
   status = true;
for(int i=0;i<num_pat;i++)
{
      if(!pat_list[i]->active)
         pat_list[i]->active=true;
   }
}

void deactivate()
{
   status = false;
   for(int i=0;i<num_pat;i++)
       pat_list[i]->active=false;
}

void draw(bool visible, double trans[3][4])
{
    for(int i=0;i<num_pat;i++)
        pat_list[i]->draw(visible,trans);
}


void alloc_pat(int num, interface_pattern *x)
{
   pat_list[num]=x;
}

};

#endif
```

Pattern_vision_drag.h

Utilizado na aplicação Sistema de Realidade Aumentada com “drag and drop” – 4.4.2 :

```
#ifndef  PATTERN_VISION_DRAG_H
#define PATTERN_VISION_DRAG_H

#include "interface_pattern.h"
#include "dinamic_vision.h"
#include "VISAR.h"

class pattern_vision_drag :public dinamic_vision
{
   protected:



   public:
    int  marcador_antigo;
    float somadorX, somadorY, sobe, posx, posy, posz, rotation[4],
```

```
rotx,roty,rotz, scale[3];
    double xx, yy, zz;

    pattern_vision_drag(){};

    void init(char *vrml, float X, float Y, float Z)
    {
        vrml_name = vrml;
        type = DV;
        active = false;
        posx = X;
        posy = Y;
        posz = Z;
        fullscale = 1.0;
        rotation[0] = 0.0 ;
        rotation[1] = 0.0 ; //gira em torno do eixo x
        rotation[2] = 0.0 ; //gira em torno do eixo y
        rotation[3] = 0.0 ; //gira em torno do eixo z
        scale[0] = 1.0 ; //aumenta em x
        scale[1] = 1.0 ; //aumenta em y
        scale[2] = 1.0 ; //aumenta em z
    }

    void set_position(float x, float y, float z)
    {
        posx = x;
        posy = y;
        posz = z;
    }

    void change_position(float x, float y)
    {
        posx = posx + x;
        posy = posy + y;
    }

    void set_z(float z)
    {
        posz = posz + z;
    }
    void change_rotation(double r1, double r2, double r3, double r4)
    {
        rotx = r2 ;
        roty = r3 ;
        rotz = r4 ;
        if(rotx)
        {
            rotation[0] = rotation[1] + r1 ;
            rotation[1] = rotation[0];
        }
        if(roty)
        {
            rotation[0] = rotation[2] + r1 ;
            rotation[2] = rotation[0];
        }
        if(rotz)
        {
            rotation[0] = rotation[2] + r1 ;
            rotation[2] = rotation[0];
        }
```

```
    }


    void change_scale(double s1, double s2, double s3)
    {
        scale[0] = scale[0] + s1 ;
        scale[1] = scale[1] + s2 ;
        scale[2] = scale[2] + s3 ;
    }

    void draw(bool visible, double trans[3][4])
    {
    marcador_antigo = mark_visivel;
     GLdouble p[16];
    GLdouble m[16];

     if(visible){

     arglCameraFrustumRH(get_cam(), 10.0, 10000.0, p);
    glMatrixMode(GL_PROJECTION);
    glLoadMatrixd(p);
    glMatrixMode(GL_MODELVIEW);
    // Viewing transformation.
    //glLoadIdentity();
    // Lighting and geometry that moves with the camera should go here.
    // (I.e. must be specified before viewing transformations.)
    //none
     arglCameraViewRH(trans, m, fullscale);
    glLoadMatrixd(m);

    glTranslated( posx, posy, posz );
    if (rotation[0] != 0.0) {
         glRotated(rotation[0], rotx,roty,rotz);

     }
     glScaled(scale[0], scale[1], scale[2]);
    // All lighting and geometry to be drawn relative to the marker goes
here.
    //fprintf(stderr, "About to draw object %i\n", i);

    arVrmlDraw(vrml_id);

    }
    }
};
#endif
```

ANEXO II. Localização e Rastreamento do sistema de RA para ambientes de emergência

Uma técnica de localização e rastreamento de baixo custo para o *Augmented Firefighter* foi desenvolvida como parte deste trabalho de mestrado. A técnica consiste da fusão de quatro tecnologias. A primeira é a comunicação sem fio *Wi-Fi,* já integrada a equipamentos e veículos dos bombeiros, o que torna-se possível levar pontos de acesso (Acess Points - APs) para o local da emergência. Combinando esses APs com um receptor de transmissões sem fio fixado nos bombeiros que entrarem no local da emergência (ambiente interno), podem-se realizar medições da força do sinal enviado pelo emissor (pontos de acesso) e recebido pelos receptores (bombeiros com algum equipamento de recepção portátil, como *palmtops*). Dessa forma, é possível executar triangulação da posição dos profissionais dentro do ambiente. Para melhorar a eficiência desse processo, a planta digitalizada do local da emergência é utilizada para fornecer informações sobre o número de paredes entre os APs e os receptores. Os seguintes modelos matemáticos podem ser utilizados para calcular a posição dos bombeiros. Para a triangulação, foram utilizados modelos matemáticos que consideram o número de paredes entre o emissor e o receptor de sinais, mostrados a seguir.

O sinal com perda é calculado da seguinte maneira [LaMarca05]:

$$ss = ss0 - 10 \times n \times \log_{10}(d/d0) \textit{(1) (desconsidera paredes)}$$

Nessa expressão, ss é a força do sinal esperado, $ss0$é a força que o sinal teria em um espaço livre a distância $d0$do emissor. A distância entre receptor e emissor é d e a constante baseada nas características particulares do aparelho emissor e do ambiente físico é representada por n, que varia tipicamente entre 2 e 5.

A segunda fórmula é para calcular o sinal em um ambiente livre de obstáculos [Lafortune90]:

$L_F = 20\ \log_{10}(\lambda/4\Pi d)$(2) (sem nenhum obstáculo)

Nesse caso, n é igual a 2 e leva-se em consideração o comprimento de onda λ.

O cálculo do sinal em um ambiente com paredes e divisórias de plástico é o foco da terceira técnica [Seidel92]:

$$\overline{PL}(d)[dB] = 20.0 \times \log_{10}\left(\frac{4\Pi d}{\lambda}\right) + p \times AF(soft\ partition)[dB] + q \times AF(concrete\ wall)[dB]$$

(3) (considera paredes)

Comparando esta fórmula com a (2), verifica-se que houve o acréscimo do número de paredes de concreto q e partições de plástico p, sendo o AF (perda do sinal) para partições de plástico 1.39 dB e para paredes de concreto 2.38 dB, respectivamente.

Através dessas expressões pode-se triangular a posição do usuário. A figura 47 mostra três APs em torno do local da emergência. Através da potência dos sinais recebidos, o receptor consegue mensurar a distância que está de cada AP. Com base nesses dados, determina-se o raio do alcance e, fundamentado nisso, estipula-se que a posição do receptor é sobre a borda do círculo. Em outras palavras, o ponto em que as três circunferências se encontram marca a localização do receptor. Aumentando-se o número de APs, tende-se a aumentar a precisão dessa triangulação, mas o erro médio ainda é alto (3-5 metros) [Peternier06] e portanto, essa tecnologia serve somente como uma base para o rastreamento, informando, por exemplo, em qual cômodo o usuário está.

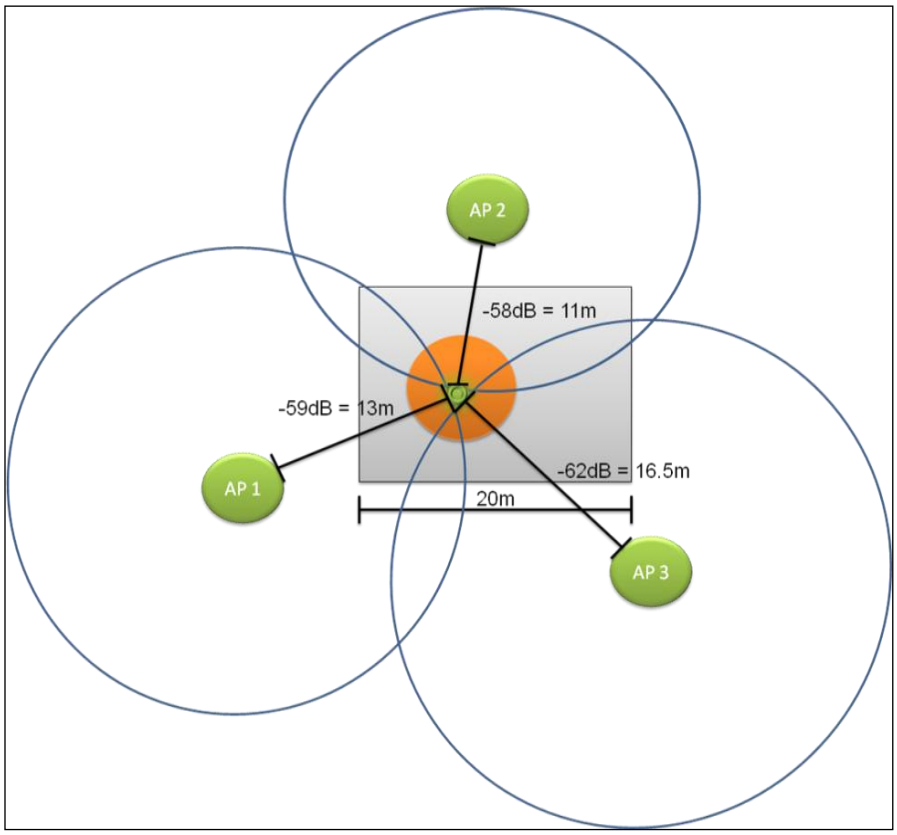

Figura 47: Rastreamento através do *Wi-Fi*.

A segunda tecnologia usada no desenvolvimento da proposta do presente trabalho é o rastreamento de marcadores. Como cada bombeiro tem em seu capacete um pequeno marcador, podendo ser um LED ou um A.R.T. Tracker[9], câmeras de segurança do local podem ser utilizadas para capturar a posição da equipe.

Outra forma de rastreamento com marcadores é a terceira das técnicas empregadas neste projeto. Uma câmera colocada no capacete dos bombeiros estará filmando o ambiente e através de marcadores espalhados pelo ambiente em placas de sinalização de emergência, é obtida a posição do bombeiro. As duas últimas tecnologias citadas oferecem uma precisão de localização maior que a primeira, mas funcionam apenas em ambientes previamente estruturados. A quarta técnica é um sensor inercial 6DOF para detectar a orientação e movimentação do bombeiro, assim como rápidas variações na orientação.

Como o *framework* VISAR utiliza um gerenciador de localização, Ubitrack, o grafo de relação espacial (SRG) da aplicação, com as quatro tecnologias da técnica de localização, foi modelado, como mostra a figura 48.

[9] Equipamento de rastreamento para Realidade Aumentada e realidade virtual, http://www.ar-tracking.de/.

A parte azul indica o rastreamento do marcador embutido no capacete do bombeiro. A parte verde representa a triangulação da posição do bombeiro através de redes sem fio. A vermelha o rastreamento do bombeiro pela câmera no seu capacete, através de marcadores espalhados pelo ambiente em placas de sinalização. O nó "sensor 6DOF" representa o sensor inercial e o nó "plano da imagem", o uso da posição do bombeiro para gerar a RA.

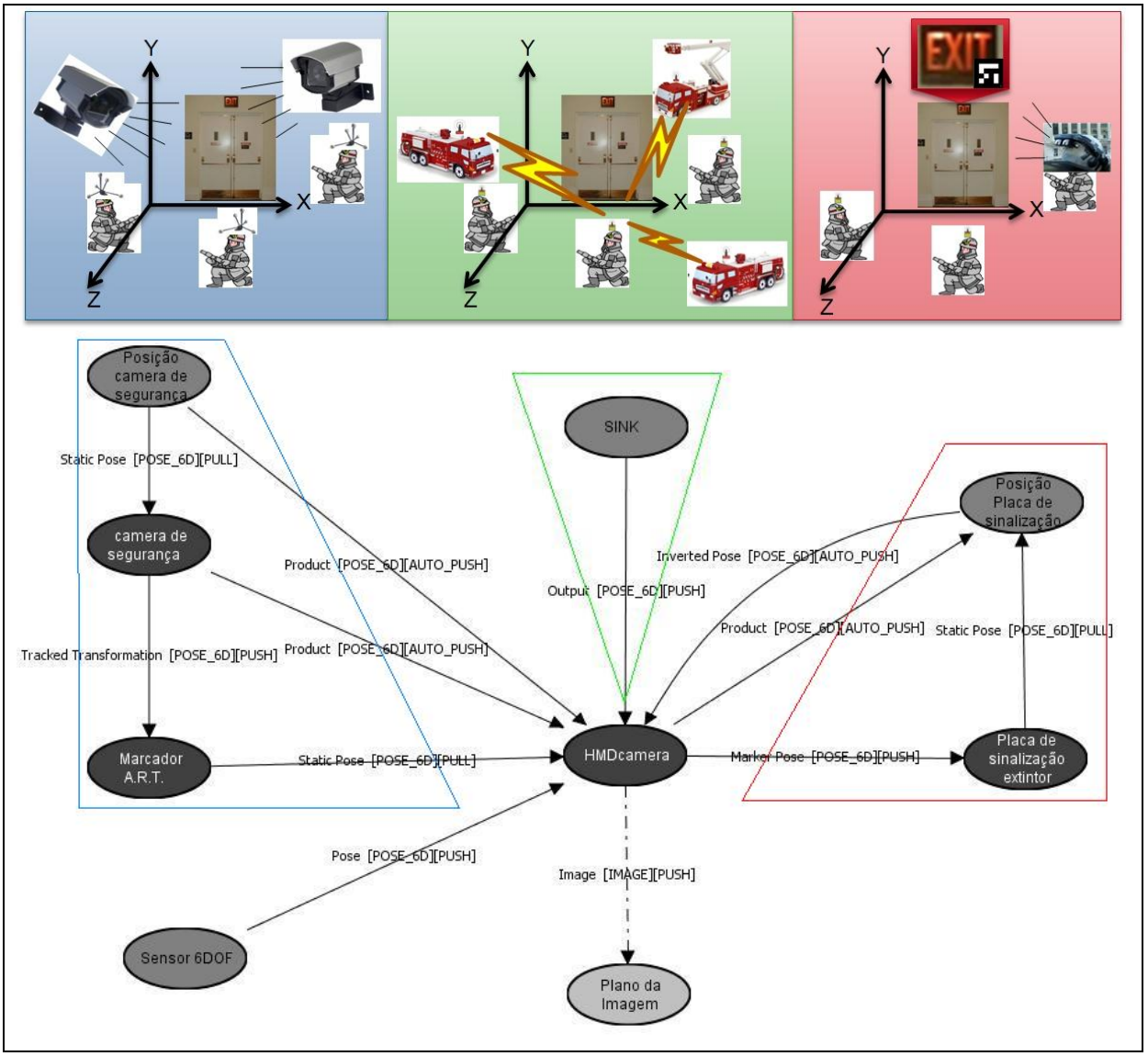

Figura 48: Grafo da Relação Espacial (SRG) da aplicação de resposta a emergência utilizando Ubitrack.

ANEXO III. Mapeamento dos contextos do sistema de RA para ambientes de emergência

As colunas na tabela de mapeamento representam:

- entidade: foco de observação da pergunta;
- pergunta: requisição feita pelo sistema quando deseja receber uma informação;
- contexto: área de observação que se encontra os dados necessários para responder a pergunta;
- dados/informações: dados necessários para se chegar a resposta;
- fonte dado/informação: fonte (equipamento, sensor, etc.) da qual o dado pode ser obtido;
- situação de contexto: conjunto de interpretações necessários para se chegar a resposta;
- resposta: informação devolvida a aplicação sob determinada pergunta.

Tabela 3:mapeamento dos contextos do sistema de RA para ambientes de emergência.

Entidade	Pergunta	Contexto	Dados/Informações	Fonte dado/informação	Situação de Contexto	Resposta
Usuário	Quem está utilizando o sistema?	Pessoa	1. Identificação do usuário		Acesso direto ao dado.	Nome do usuário
	Qual a tarefa designada para o usuário?	Pessoa	1. Identificação do usuário 2. Tarefa do usuário		Sabendo a identificação do usuário conseguimos descobrir a tarefa que ele está executando.	Tarefa do usuário
Ambiente onde se localiza o usuário	Qual a luminosidade do ambiente?	Ambiente	1. Luminosidade	1. Células solares, fotodiodos, fototransistores, tubos foto-elétricos, CCDs, radiômetro de Nichols, sensor de imagem	Acesso direto ao dado.	% de luz no local
	Qual a posição do bombeiro no ambiente?	Pessoa	1. Distancia Triangulação 2. Posição Imagem 3. Movimento 4. Orientação	1. WLan Card, *acess point* 2. Câmera, marcador 3. Acelerômetro 4. Giroscópio, bussola	Fusão dos dados de localização por redes sem fio, rastreamento através de câmeras, acelerômetro e giroscópio.	Posição em 6DOF (X, Y, Z) + 3 graus de orientação
	Qual a fase do incêndio?	Emergência	1. Temperatura 2. Grau de ocupação da fumaça 3. Quantidade de oxigênio	1. Sensores (termômetros, termopares, termístores, termostatos) 2. Sensor de fumaça foto-elétrico 3. Sensor de oxigênio, sonda lambda	A interpretação dos dados permite conhecer a fase do incêndio.	%Não há; %Fase inicial; %Queima livre; %Queima lenta

	Qual o risco de Boil Over?	Emergência	1. Liquido miscível com água e inflamável 2. Temperatura no tanque	1. Etiqueta RFID 2. Sensores (termômetros, termopares, termístores, termostatos)	O risco de Boil Over existe quando um tanque com um liquido miscível com água e inflamável está em alta temperatura.	Sim / Não Tem ou não possibilidade de ocorrer um Boil Over
	Qual o risco de BLEVE?	Emergência	1. Liquido inflamável 2. Temperatura no tanque 3. Pressão no tanque	1. Etiqueta RFID 2. Sensores (termômetros, termopares, termístores, termostatos) 3. Manômetro	O BLEVE ocorre quando a pressão e a temperatura de um liquido inflamável sobem acima de um limite.	0-100% Possibilidade de ocorrer um BLEVE
	Qual o risco de Backdraft?	Emergência	1. Fase do incêndio	1. Combinação entre vários sensores, já descrita	O Backdraft ocorre quando a fase do incêndio é queima lenta e um local fechado é aberto.	0-100% Possibilidade de ocorrer um Backdraft
		Ambiente	1. Local Fechado			
	Qual o risco de Flash Over?	Emergência	1. Fase do incêndio	1. Combinação entre vários sensores, já descrita	O Flash Over ocorre quando a fase do incêndio é queima livre, o local está fechado e o ponto de ignição dos materiais é atingido.	0-100% possibilidade de ocorrer um Flash Over
		Ambiente	1. Local Fechado 2. Ponto de ignição dos materiais	1. 2. Etiqueta RFID		
	Qual a quantidade de gás no local?	Emergência	1. Gás natural (metano), GLP,solventes, hidrogênio, álcool, gasolina, mon. de carbono, amônia	1. GSA-1000, AHG 982B	Acesso direto ao dado.	0-100%/ 25 ppm
	Qual o radioatividade no local?	Emergência	1. Radioatividade	1. Contador Gêiser, dosímetro	Acesso direto ao dado.	0-100%

2º andar	Há risco de incêndio no 2º andar?	Emergência	1. Fase do incêndio	1. Combinação entre vários sensores, já descrita	Analisa-se a fase do incêndio presente no local estipulado.	% de possibilidade de incêndio
		Ambiente	1. Localização	1. Combinação entre vários sensores, já descrita		
	Quem enviar para o 2º andar?	Pessoa	1. Pressão 2. Batimento cardíaco 3. Localização	1. Medidor digital de pressão arterial de pulso 2. Medidor digital de pressão arterial de pulso 3. Combinação entre vários sensores, já descrita	Conhecendo a situação de saúde atual dos bombeiros e a localização deles em relação às condições do edifício, decide-se quais enviar.	Identificação do(s) bombeiro(s) mais próximos e aptos
		Emergência	1. Fase do incêndio	1. Combinação entre vários sensores, já descrita		
		Ambiente	1. Condição estrutural física. 2. Saídas de emergência. 3. Característica dos materiais presentes	1. 2. Etiqueta RFID 3. Etiqueta RFID		

Área monitorada da emergência	Quem está em risco?	Pessoa	1. Pressão 2. Batimento cardíaco 3. Localização 4. Postura	1. Medidor digital de pressão arterial de pulso 2. Medidor digital de pressão arterial de pulso 3. Combinação entre vários sensores, já descrita 4. Câmera de segurança, giroscópio.	Conhecendo o estado de saúde das pessoas envolvidas na situação (participantes e especialistas) e podendo monitorar isso, o estado da infraestrutura física e as condições de evolução dos riscos (inclusive perigos de *Boil Over*, BLEVE, *Back-draft* e *Flash Over*), sabe-se quais pessoas estão em maior risco.	Identificação da(s) pessoas(s) em maior risco
		Emergência	1. Fase do incêndio 2. Local Fechado 3. Liquido miscível com água e inflamável 4. Temperatura no tanque 5. Pressão no tanque 6. Ponto de ignição dos materiais	1. Combinação entre vários sensores, já descrita 2. 3. Etiqueta RFID 4. Sensores (termômetros, termopares, termístores, termostatos) 5. Manômetro 6. Etiqueta RFID		