



Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

ARHydra - Uma proposta de visualização e redirecionamento de recursos utilizando Realidade Aumentada

Ricardo Felipe Lacerda de Andrade

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Ricardo Pezzuol Jacobi

Coorientador
Fabrício Nogueira Buzeto

Brasília
2012

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Alexandre Zaghetto

Banca examinadora composta por:

Prof. Dr. Ricardo Pezzuol Jacobi (Orientador) — CIC/UnB
Prof. Dr. Pedro de Azevedo Berger — CIC/UnB
Prof. Ms. Tiago Barros Pontes e Silva — DIN/UnB

CIP — Catalogação Internacional na Publicação

Andrade, Ricardo Felipe Lacerda de.

ARHydra - Uma proposta de visualização e redirecionamento de recursos utilizando Realidade Aumentada / Ricardo Felipe Lacerda de Andrade. Brasília : UnB, 2012.

111 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2012.

1. realidade aumentada, 2. computação ubíqua, 3. redirecionamento de recursos

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília-DF — Brasil



Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

ARHydra - Uma proposta de visualização e redirecionamento de recursos utilizando Realidade Aumentada

Ricardo Felipe Lacerda de Andrade

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Ricardo Pezzuol Jacobi (Orientador)
CIC/UnB

Prof. Dr. Pedro de Azevedo Berger Prof. Ms. Tiago Barros Pontes e Silva
CIC/UnB DIN/UnB

Prof. Dr. Alexandre Zaghetto
Coordenador do Bacharelado em Ciência da Computação

Brasília, 8 de outubro de 2012

Dedicatória

Dedico este trabalho a meus pais, que me educaram e ensinaram a batalhar pelos meus sonhos.

Agradecimentos

Hoje, completo mais uma etapa em minha vida agradecendo primeiramente a Deus, pois Ele é a fonte de sabedoria e força que devemos buscar para enfrentar todas as dificuldades, aos meus amigos e familiares que me ajudaram a chegar ao final desse caminho.

Um agradecimento à Rebeca Cavalcante, uma pessoa muito especial em minha vida, pelo seu apoio incondicional em todos os momentos, me ajudando ser a uma pessoa melhor a cada dia.

Agradeço ao Professor Ricardo Jacobi por me orientar neste trabalho, sempre acreditando em minha capacidade e no potencial para desenvolver este projeto.

Não poderia de deixar de fora meu agradecimento ao Fabrício Buzeto por seus conselhos, conhecimentos repassados, apoio e também por suas cobranças que contribuíram para o sucesso deste trabalho.

Resumo

A computação ubíqua tem por objetivo a criação de ambientes inteligentes, conhecidos também por *smart spaces*. O objetivo destes é possibilitar uma interação pró-ativa para o ambiente com seus usuários. Esta pró-atividade deve ser realizada da maneira transparente e minimizando a intrusão nas atividades sendo realizadas. Porém, para que isto ocorra, é necessária a análise das informações disponíveis para que se conheça o contexto do usuário. Estas informações são obtidas através dos dispositivos presentes no ambiente e são necessárias para fornecer a inteligência almejada. Camadas de software denominadas *middlewares* são utilizadas para obter e tratar estas informações delegando as ações as aplicações construídas sobre elas. O *uOS* é um exemplo de *middleware* focado em permitir o acesso aos dispositivos do ambiente.

A *Hydra* é uma aplicação construída utilizando o *uOS* que possibilita ao usuário redireciona seus recursos para outros mais adequados no ambiente. Esta aplicação visa facilitar o usuário em suas atividades fornecendo uma maior gama de opções ao realizá-las. Para isso, a aplicação foca em apresentar de forma textual os dispositivos e recursos disponíveis. No entanto, a tarefa de se localizar estes no ambiente aumenta com a quantidade de dispositivos presentes. A fim de minimizar esta tarefa ao usuário, esse trabalho apresenta uma aplicação denominada de ARHydra (*Augmented Reality Hydra*).

Esta objetiva em prover uma interface de interação aprimorada à *Hydra*, permitindo ao usuário uma maior transparência e facilidade na escolha dos recursos do ambiente. Para isso é feito o uso das técnicas de Realidade Aumentada, permitindo uma integração entre as informações e o ambiente real. A ARHydra faz uso de marcadores, utilizando o QRCode em sua composição, para mapear os dispositivos dentro do ambiente inteligente e a partir deste, obter e apresentar os recursos por ele disponibilizados. Para medir a influência dos dispositivos e do ambiente na execução da ARHydra, serão apresentados os resultados obtidos nos testes efetuados mostrando um comparativo de desempenho da ARHydra entre dois *smartphones* e a influência de aspectos envolvidos no processo de captura da imagem e na composição do QRCode que possam interferir nestes resultados.

Palavras-chave: realidade aumentada, computação ubíqua, redirecionamento de recursos

Abstract

Ubiquitous computing aims at creating smart environments, also known by smart spaces. The goal of these interactions is enable a proactive environment for its users. This pro-activity should be done in a transparent manner and minimizing the intrusion on the activities being performed. However, for this to occur, is necessary to analyse of available information so that know the context of the user. These informations are obtained through the devices in the environment and are necessary to provide the desired intelligence. Layers of software called middleware are used to obtain information and treat these actions delegating applications built on them. The uOS is an example of middleware focused on allowing access to devices in the environment.

The Hydra is an application built using the uOS that allows the user redirect its resources to other more suitable environment. This application aims to facilitate the user in their activities by providing a greater range of options to perform them. For this, the application focuses on presenting in textual devices and resources available. However, the task of locating these environmental increases with the number of devices present. In order to minimize this task to the user, this work presents an application called ARHydra (Augmented Reality Hydra).

This aims to provide an improved interface for the interaction with the Hydra, allowing greater transparency to the user and ease in choosing the environment resources. For this are used the techniques of Augmented Reality, allowing integration between the information and the real environment. The ARHydra uses markers, using QRCode in their composition, to map the devices within the smart space and from this, obtain and provide the resources has made available. To measure the influence of the devices and the environment in the execution of ARHydra, will present the results of those tests showing a comparative performance of the ARHydra between two smartphones and the influence of aspects involved in image capture and in the QRCode composition that may interfere these results.

Keywords: augmented reality, ubiquitous computing, resource redirect

Sumário

1	Introdução	1
2	Realidade aumentada	3
2.1	Aplicações	5
2.2	Equipamentos	6
2.3	Classificação	7
2.4	Marcadores	9
2.4.1	Símbolos bidimensionais	9
2.4.2	Marcadores para Realidade Aumentada	13
2.5	Reconhecimento de marcadores	13
2.6	Trabalhos correlatos	16
2.6.1	<i>CyberCode</i>	16
2.6.2	<i>HELLO</i>	17
2.6.3	SmART World	19
3	<i>ARHydra</i>	21
3.1	O middleware uOS	21
3.2	A aplicação Hydra	22
3.2.1	Recursos	23
3.3	Incrementando a Hydra com a Realidade Aumentada	25
3.3.1	Marcadores na ARHydra	26
3.3.2	Interação no ambiente	27
4	Implementação e testes	30
4.1	Módulo de Reconhecimento	31
4.2	Módulo de Decodificação	32
4.3	Módulo de Integração	34
4.4	Módulo de Apresentação	34
4.5	Hydra <i>Driver</i>	35
4.6	Testes	36
4.6.1	Reconhecimento dos marcadores	37
4.6.2	Resultados	39
5	Conclusão	44
5.1	Trabalhos Futuros	45
Referências		46

Listas de Figuras

2.1	<i>Ambiente de Realidade Misturada [24]</i>	4
2.2	<i>Head Mounted Display [28]</i>	7
2.3	<i>Exemplo de utilização de smartphone na Realidade Aumentada [7]</i>	8
2.4	<i>Exemplos de símbolos unidimensional e bidimensional</i>	10
2.5	<i>Ilustração das propriedades estruturais do QRCode. Adaptado de [29]</i>	11
2.6	<i>Exemplo de alguns marcadores utilizados na realidade aumentada</i>	13
2.7	<i>Etapas do processo de reconhecimento de marcadores na Realidade Aumentada [30]</i>	14
2.8	<i>Obtenção do código identificador do marcador. Adaptado de [15]</i>	15
2.9	<i>Exemplo de um marcador CyberCode [31]</i>	16
2.10	<i>Exemplo do tutor virtual utilizado no HELLO. Adaptado de [27]</i>	18
3.1	<i>Middleware uOS [10]</i>	22
3.2	<i>Sala com computadores</i>	26
3.3	<i>Marcador utilizado na ARHydra</i>	27
3.4	(a) <i>Exemplo da visão do usuário para a busca do marcador do dispositivo.</i> (b) <i>Exemplo da visualização do objeto virtual apresentando os recursos disponíveis</i>	28
3.5	<i>Integração Hydra com o ambiente</i>	29
4.1	<i>Interação entre os módulos</i>	31
4.2	<i>Processos do Módulo de Reconhecimento</i>	32
4.3	<i>Processos do Módulo de Decodificação</i>	33
4.4	<i>Objeto virtual</i>	35
4.5	<i>Listagem dos recursos</i>	36
4.6	<i>Condições para visualização do objeto virtual</i>	37
4.7	<i>Dimensões do marcador</i>	39
4.8	<i>Comparativo de desempenho da aplicação ARHydra entre os dispositivos Motorola Defy e Samsung Galaxy SIII</i>	40
4.9	<i>Comparativo entre as taxas de erro e de não decodificação da aplicação ARHydra entre os dispositivos Motorola Defy e Samsung Galaxy SIII</i>	41
4.10	<i>Taxa de não decodificação para as aplicações ZBar e ZXing</i>	42
4.11	<i>Nível da taxa de não decodificação aplicado sobre a influência da tolerância a falhas do QRCode</i>	43

Lista de Tabelas

2.1	Códigos bidimensionais e suas categorias.	10
2.2	Níveis de correção de erros suportado pelo QRCode [21].	11
2.3	Comparação de vários códigos [13].	12

Capítulo 1

Introdução

Conhecida também como *ubicomp*, a Computação ubíqua tem por objetivo permitir que a interação entre o usuário e as tarefas a serem desempenhadas ocorram de forma transparente. Por essa razão a *ubicomp* também é chamada de Computação Invisível [33, 35, 36]. Essa invisibilidade permite ao usuário focar mais na tarefa a ser desempenhada e não na ferramenta a ser manipulada para sua execução. Para isso, todos os serviços e dispositivos seriam regidos sem a intervenção do usuário [34].

Essa interação inteligente, entre usuários e dispositivos, é realizada através de cenários onde estão integrados vários dispositivos, recursos e *softwares* especializados para a construção deste tipo de ambiente, denominado de *smart spaces* [5]. Essa inteligência é baseada na análise de informações de contexto do ambiente para que serviços possam ser providos de forma pró-ativa, antecipando as necessidades do usuário [14]. Para que a interação ocorra, faz-se necessário a criação de camadas de *softwares*, também denominados de *middlewares*, com o propósito de orquestrar a troca de informações a respeito dos usuários e dispositivos inseridos no *smart space*. Ele abstrai os detalhes relativos às camadas inferiores, como por exemplo, serviços de segurança, comunicação, escalabilidade, heterogeneidade de dispositivos, adaptabilidade e identificação de serviços. Isso permite que aplicações possam interagir com o usuário para que sejam disponibilizados recursos com o propósito de melhor atender suas tarefas e necessidades.

Com a evolução da tecnologia novos recursos foram inseridos nos dispositivos. O ambiente inteligente deve oferecer informações, a respeito dessa variedade de recursos, para auxiliar as aplicações inteligentes na seleção daqueles recursos necessários à execução de uma determinada tarefa. Para isso o *middleware* cria uma instância para cada recurso presente e o disponibiliza no ambiente através de um identificador único. Por outro lado, este identificador pode trazer dificuldades para o usuário associá-lo ao dispositivo que esteja disponibilizando-o, devido a volatilidade e a quantidade de dispositivos inseridos no ambiente inteligente. Por essa razão, faz-se necessário a criação de mecanismos com o objetivo de facilitar a sua localização, visualização e utilização desses recursos.

Como exemplo de aplicação inteligente que utiliza recursos disponíveis no ambiente, o UnBiquitous desenvolveu a Hydra, aplicação que proporciona o redirecionamento de recursos entre dispositivos. Para interagir com os dispositivos, a Hydra utiliza o *uOS*, um *middleware* em desenvolvimento junto a UnB cujo foco é a adaptabilidade de serviços em um ambiente inteligente [6, 10]. A Hydra possui uma interação feita de forma sugerida, apresentando ao usuário somente os recursos que lhe são compatíveis. Ela não implementa

mecanismos que facilite a associação dos recursos sugeridos aos dispositivos de origem.

Para minimizar esse problema foi desenvolvido a aplicação ARHydra (*Augmented Reality Hydra*). Ela utiliza os conceitos provados pela Realidade Aumentada para combinar uma visão do *smart space* e dos recursos disponíveis nele, com o objetivo de criar uma nova forma de identificação desses recursos e prover uma interface mais intuitiva para o redirecionamento dos mesmos. Esta aplicação faz uso de marcadores para identificar um dispositivo e apresentar seus recursos ao usuário através de objetos virtuais, sendo estes visualizados a partir de um *smartphone* utilizando a plataforma Android. Os marcadores são representados através de um código bidimensional, o QRCode, afixado em local visível sobre os dispositivos. Através do código é possível identificar o dispositivo e determinar, através do *uOS*, o conjunto de recursos que ele disponibiliza ao *smart space*. Isto permite ao usuário interagir com o objeto virtual apresentado e prover o redirecionamento dos recursos apresentados.

Este trabalho se encontra organizado da seguinte maneira: o Capítulo 2 fundamenta os conceitos a respeito da Realidade Aumentada apresentando suas aplicações, classificações, marcadores, além de alguns trabalhos relativos a Realidade Aumentada aplicadas no contexto da Computação Ubíqua. O Capítulo 3 mostra uma visão geral do projeto UbiquitOS. A aplicação ARHydra é apresentada após um detalhamento dos conceitos referentes à Hydra. Nela é descrita seus conceitos, marcadores a serem utilizados e sua integração com o ambiente. O Capítulo 4 detalha as etapas de implementação, as soluções utilizadas para reconhecimento e apresentação dos recursos, bem como sua integração com a Hydra. Adicionalmente, neste capítulo também são apresentados os resultados de testes realizados sob a aplicação, bem como a sua discussão. No Capítulo 5 são relacionadas algumas considerações finais sobre este trabalho e sugestões de trabalhos futuros.

Capítulo 2

Realidade aumentada

A Realidade Virtual utiliza a tecnologia com o propósito de simular uma inserção do usuário em um mundo virtual onde ele não conseguiria distinguir o cenário real daquele em que fora inserido, ou seja, o usuário estaria inserido em uma simulação de um mundo complementar sintético [8]. Esse mundo sintético pode imitar as propriedades de alguns ambientes encontrados no mundo real, podendo até exceder os limites da realidade física.

Em contraste com a Realidade Virtual, a Realidade Aumentada permite ao usuário focar em seu ambiente físico e interagir em tempo real entre ambos os mundos. Essa nova forma de interação possibilita ao usuário manipular informações entre esses mundos de uma forma mais natural, ou seja, minimizando a necessidade de um treinamento para sua adaptação [24]. De forma geral, pode-se citar três características principais de aplicações desenvolvidas para a Realidade Aumentada:

- Combinação entre o real e o virtual;
- Interatividade em tempo real;
- Possibilidade de visualização de objetos virtuais em 3D.

A Realidade Aumentada combina uma visão composta por cenas reais (visualizados pelo usuário) com objetos virtuais (gerados com auxílio do computador) para serem apresentadas ao usuário com o objetivo de inserir novas informações virtuais a respeito de objetos físicos visualizados por ele. O reconhecimento dos objetos físicos pode ocorrer através do uso marcadores, onde cada objeto físico possui seu marcador que o identifique e suas informações são apresentadas através de objetos virtuais. Tais informações são projetadas com o objetivo de melhorar a percepção sensorial do usuário com relação aos cenários obtidos pela integração do real com o virtual [19].

A Realidade Aumentada está inserida dentro de um conceito denominado Realidade Misturada. Esta é definida como um ambiente no qual os objetos pertencentes tanto mundo real quanto ao mundo virtual são apresentados de forma simultânea ao usuário. A Realidade Misturada visa complementar aspectos de ambos os mundos através do uso de informações incluídas em elementos virtuais. A figura 2.1 apresenta a divisão desses conceitos.

1. **Ambiente Real:** Representa um ambiente constituído exclusivamente de objetos reais.

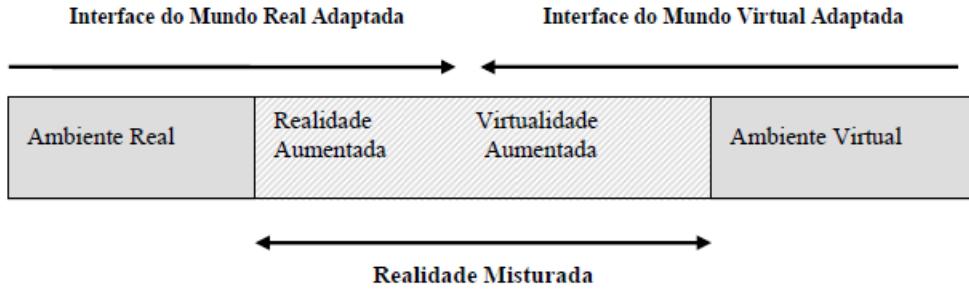


Figura 2.1: *Ambiente de Realidade Misturada* [24]

2. **Realidade Aumentada:** Possui o ambiente real como interface de interação com o usuário, com o propósito de inserir novas informações nele através de objetos virtuais.
3. **Virtualidade Aumentada:** A diferença entre a Realidade Aumentada e a Virtualidade Aumentada está em sua forma de interação. Esta tem a finalidade de inserir elementos reais dentro do mundo virtual e utilizar o ambiente virtual como interface de usuário. Para que isso ocorra, a Virtualidade Aumentada utiliza técnicas computacionais para capturar elementos reais e inseri-los no mundo virtual como objetos virtuais, como por exemplo, uma foto do usuário (objeto real) ser escaneada ou retirada através de uma *webcam* e ser inserida no mundo virtual para compor o objeto virtual.
4. **Ambiente Virtual:** É definido como um ambiente compostos exclusivamente por objetos virtuais, simulando um ambiente completamente sintético.

Como um todo, os sistemas desenvolvidos para a Realidade Aumentada apresentam pontos chaves para seu funcionamento [17]:

1. **Rastreamento:** O sistema é responsável por obter as informações corretas a respeito do posicionamento e orientação do usuário. A recuperação dessas informações torna-se necessária para a apresentação do conteúdo virtual correspondente. Nesta etapa são utilizados equipamentos (seção 2.2) responsáveis pela captura e processamento dessas informações. O estabelecimento desses parâmetros é conhecido como *tracking*.
2. **Registro:** Após o rastreamento ser concluído é feito o registro das informações correspondente a cada objeto reconhecido. O registro deve ser feito de modo que seja preservada a interatividade do usuário para que a relação entre o real e o virtual estejam alinhadas dentro de um mesmo domínio.
3. **Visualização:** A partir do resultado final gerado, esse sistema deve ser capaz de produzir os objetos correspondentes em algum dispositivo que permita ao usuário visualizar tais informações.

O processamento destas imagens é constituído de etapas bem definidas (seção 2.5), tendo como etapa inicial o reconhecimento dos marcadores utilizados pela Realidade Aumentada e, posteriormente, a obtenção do código identificador do marcador. Por fim, a construção do objeto virtual correspondente.

2.1 Aplicações

Tendo em vista o potencial de aplicação da Realidade Aumentada na interação com usuários muitos estudos tem sido realizados neste sentido. Aplicações tem sido desenvolvidas em diversos ramos da atividade humana. A seguir exemplifica-se alguns usos da Realidade Aumentada:

- **Entretenimento**

É nessa área que encontra-se o maior número de aplicações utilizando a Realidade Aumentada, sendo estas desenvolvidas principalmente para a área de jogos. O grande diferencial em utilizar a Realidade Aumentada em jogos é fazer com que seus cenários interajam com o mundo real ao qual o usuário está inserido. Isso possibilita uma interação maior do usuário e o cenário apresentado pelos jogos.

- **Medicina**

O uso da Realidade Aumentada vem auxiliando a medicina em muitos aspectos, desde a visualização partes do corpo até a sua utilização em cirurgias. O *HMD (Head Mounted Display)* é um equipamento bastante utilizado para auxiliar na visualização dos objetos virtuais, possibilitando que o mesmo seja utilizado em cirurgias guiadas por imagens. Deste modo, essa aplicação da Realidade Aumentada auxiliará em um melhor planejamento cirúrgico, contribuindo para uma diminuição dos riscos envolvidos [28, 32].

- **Mercado imobiliário e arquitetura**

Maquetes e *design* de interiores são construídas utilizando Realidade Aumentada com propósito de exemplificar aos consumidores o estado do empreendimento ao término de sua construção. Desta forma, a Realidade Aumentada é bastante utilizada nessas áreas para possibilitar aos compradores visualizar e customizar seus ambientes, possibilitando a modificação da disposição dos objetos e a observação dessas novas disposições.

- **Auxílio na obtenção de informações a respeito de produtos**

Empresas utilizam a Realidade Aumentada com o propósito de oferecer maiores detalhes a respeito de seus produtos. Eles são identificados por marcadores reconhecidos pela Realidade Aumentada. As informações necessárias são extraídas após o reconhecimento desse marcador e um objeto virtual contendo informações a respeito do produto é apresentado ao usuário. No caso de uma rede de supermercados, tais informações poderiam representar descontos, opiniões e ingredientes de produtos anunciados. Utilizando ainda o exemplo anterior, a localização dos produtos dentro do estabelecimento poderia ser mapeada de uma forma com que uma aplicação pudesse auxiliar o usuário a encontrar um determinado produto, através de uma navegação guiada por GPS ou por outros marcadores que guardariam a localização referentes a cada tipo de produto.

- **Educação**

Através dos benefícios providos pela flexibilidade e usabilidade, a Realidade Aumentada foi utilizada na educação com o foco na aprendizagem. Sua utilização vai

além da aprendizagem utilizando somente os livros, ela explora características que até então não eram percebidas no ambiente acadêmico, potencializando sua aprendizagem devido principalmente a interatividade provida pela Realidade Aumentada [9, 22]. Para exemplificar essa aplicação na Realidade Aumentada, objetos dentro do museu britânico foram mapeados utilizando marcadores reconhecidos pela Realidade Aumentada proporcionando aos visitantes obterem informações a respeito dos objetos apresentados [2]. Por outro lado, a Realidade Virtual vem sendo aplicada na educação desde a década de 90. Esses projetos proporcionam a criação de um mundo virtual com o propósito de ensinar e investigar os aspectos relacionados, como por exemplo, da cinemática, eletrostática, estruturas moleculares, estudo da biologia e matemática [23, 26].

- **Turismo**

Essa área tem sido bastante explorada devido a possibilidade de mapeamento de pontos turísticos e disponibilização de informações através desse mapeamento. Essas informações podem ser disponibilizadas de acordo com o perfil do usuário, com objetivo de auxiliá-lo na busca de locais de seu interesse. A localização desses pontos é feito através das coordenadas de localização do usuário, obtidas através de um GPS. Essas informações são cruzadas com posições de longitude, latitude e altitude obtidas de um banco de dados contendo todos os mapeamentos feitos. Na Realidade Aumentada essas posições mapeadas são denominadas de POI's (*Points Of Interest*). Por causa da mobilidade obtida por esses recursos, eles são encontrados principalmente em aplicações voltadas para dispositivos móveis.

Como observado, a Realidade Aumentada é utilizada em diversas áreas com base em uma característica comum entre elas, a possibilidade de interação entre o real e o virtual. A visualização dos recursos providos pela Realidade Aumentada necessita de equipamentos compatíveis. Esses equipamentos proporcionam a captura de imagens, correspondente ao ambiente real do usuário, a construção e apresentação de objetos virtuais ao usuário.

2.2 Equipamentos

Como etapa inicial para o funcionamento da Realidade Aumentada, necessita-se de equipamentos que possuam a funcionalidade de captura de vídeo, como por exemplo *webcam's*. Esses proporcionam a captura do ambiente do usuário e envia os dados para um dispositivo responsável pelo processamento. Dentre os equipamentos utilizados inicialmente na Realidade Aumentada pode-se citar o *HMD* (*Head Mounted Display*). Eles eram utilizados com o objetivo de captura das informações, através de suas câmeras acopladas, posteriormente processavam as informações necessárias e exibia o resultado do processamento ao usuário através de objetos virtuais visualizados em suas telas acopladas ao equipamento.

Este equipamento é fixado na cabeça do usuário podendo ter o formato de um capacete ou de um óculos. Algumas utilidades desse equipamento pode ser encontrada em realizações de simulações computadorizadas e também para a visualização dos objetos virtuais ao qual o âmbito da Realidade Aumentada está inserida [8].

Quando esse equipamento foi proposto, ele possuía algumas desvantagens por ser muito pesado e evasivo. Atualmente, projetos estão sendo desenvolvidos para tentar minimizar essas desvantagens. A figura 2.2 mostra um equipamento *HMD* com características que favoreçam sua utilização. Por outro lado, a grande vantagem desse tipo de equipamento é a possibilidade de imersão do usuário em um ambiente onde ele consiga interagir entre o virtual e o real de uma forma mais natural.



Figura 2.2: *Head Mounted Display* [28].

Atualmente, os *smartphones* estão ganhando cada vez mais espaço dentro da Realidade Aumentada. Com a evolução do *GPU* (*Graphics Processing Unit*) nestes, ocorreu a popularização de seu uso em aplicações voltadas para a Realidade Aumentada. A grande vantagem em sua utilização está na abrangência com que eles são distribuídos e principalmente na mobilidade conseguida através dos mesmos. Este equipamento comporta todos os recursos necessários para sua utilização na Realidade Aumentada:

1. A câmera do celular substitui a *webcam* ou as câmeras acopladas ao HMD;
2. O processamento gráfico é feito na *GPU*;
3. O resultado é apresentado no próprio visor do aparelho, suprindo a necessidade de um monitor ou telas para a visualização do objeto virtual.

Um exemplo de utilização da Realidade Aumentada em *smartphones* pode ser visto na figura 2.3. Nesta, a câmera do *smartphone* captura a imagem do marcador, processa as informações necessárias e exibe um carro como objeto virtual correspondente para aquele marcador.

2.3 Classificação

Com a evolução dos equipamentos, a visualização dos objetos virtuais torna-se cada vez mais real. Através desses equipamentos, os sistemas de Realidade Aumentada podem ser classificados de acordo com a forma como que o usuário vê o mundo misturado. Em [8] é apresentado uma divisão para classificações entre as tecnologias óptica e vídeo. Essas classificações variam de acordo com o tipo de equipamento utilizado e com o tipo de sistema de visualização:



Figura 2.3: *Exemplo de utilização de smartphone na Realidade Aumentada [7]*.

1. Visão direta

Também conhecida como visão imersiva, nesta classificação os objetos virtuais são visualizados na mesma direção com que as cenas reais são capturadas. Esta classificação é utilizada principalmente em equipamentos que capturam as imagens reais, processam as informações necessárias e apresentam objetos ao usuário no mesmo equipamento [32]. Por exemplo, a utilização de equipamentos do tipo *HMD* proporciona ao usuário uma visão direta, tendo em vista que a captura das cenas e a apresentação do objeto virtual foi feito pelo mesmo equipamento. Desta forma o usuário não necessita desviar o foco do mundo real a fim de observar as informações providas pelo objeto virtual.

2. Visão indireta

Ocorre quando a visualização do mundo misturado é feita através de algum dispositivo e a apresentação dos objetos virtuais, correspondentes as cenas, seja feita em um outro dispositivo, ocasionando o desvio da atenção do usuário [24, 32]. Um exemplo para esse tipo de classificação é observado quando a captura das imagens reais é feita através de uma *webcam*, processadas em um computador e o resultado apresentado em projeções ou em monitores.

Uma outra forma de se classificar a Realidade Aumentada baseia-se nos sistemas utilizados por ela. Estes podem ser classificados de acordo com o tipo de equipamento utilizado para captura de cenas reais, processamento e os equipamentos responsáveis pela visualização do objeto virtual. Estas classificações abrangem tanto sistemas que utilizem visão óptica quanto visão por vídeo. Sendo classificados em:

1. Visão direta por vídeo

Neste tipo de classificação, equipamentos *HMD* são utilizados para o recebimento direto da imagem real através de suas câmeras acopladas. As imagens capturadas são

processadas em um gerador de cenas e as informações virtuais geradas por este são apresentadas diretamente ao usuário, através das telas acopladas ao equipamento.

2. Visão direta por óptica

De forma análoga a visão direta por vídeo, esta possui as mesmas etapas para obtenção e geração as informações virtuais a serem apresentadas ao usuário. No entanto, as informações geradas são apresentadas ao usuário através de lentes ou espelhos. Estas são inclinadas e posicionadas para que a projeção das imagens vindas do monitor acoplado seja refletida e redirecionada para os olhos do usuário.

3. Visão por vídeo baseada em monitor

Após a captura das cenas, os objetos virtuais são gerados no gerador de cenas e misturados com as cenas reais no combinador de cenas. As informações geradas são apresentadas ao usuário através de um monitor.

4. Visão por óptica baseada em projeção

Nessa classificação, as imagens dos objetos virtuais são apresentadas sem a necessidade de um equipamento específico. As mesmas são processadas em computador e projetadas em superfícies do ambiente real, por causa da necessidade de superfícies específicas para projeções, essa classificação torna-se mais restrita em comparação com as demais classificações.

2.4 Marcadores

Atualmente, é possível encontrar aplicações que utilizam padrões de símbolos bidimensionais para compor informações e transportar as mesmas de uma forma mais simplificada, a exemplo dos padrões *QRCode*, *PDF147* e *DataMatrix* [16]. Alguns padrões permitem que as informações estejam representadas de forma redundante, pois auxiliam em sua detecção e na correção de possíveis erros. No âmbito industrial, seu campo de aplicação varia de acordo com a necessidade de cada sistema, como por exemplo transportar informações em etiquetas. Outros tipos de marcadores são utilizados para representar dados de localização e reconhecimento de objetos, como visto na Realidade Aumentada.

2.4.1 Símbolos bidimensionais

Esse tipo de código de barras foi desenvolvido por volta de 1987. No código de barras bidimensional as informações são armazenadas tanto na altura quanto na largura, por essa razão possui uma maior capacidade de armazenamento das informações quando comparada aos símbolos unidimensionais. Os modelos de códigos unidimensionais armazenam as informações em apenas em uma dimensão, por causa dessa característica há uma limitação na quantidade de informação ser armazenada por eles, de acordo com cada padrão.

Para exemplificar essas características é apresentada a figura 2.4 ilustrando esses dois tipos de símbolos. O código apresentado na figura 2.4a corresponde a um símbolo unidimensional, a altura desse símbolo (representado no eixo vertical) fornece redundância e legibilidade, possibilitando um tempo de resposta rápido, uma vez que a análise é feita em

Tabela 2.1: Códigos bidimensionais e suas categorias.

Código bidimensional	Categoria
<i>Code 49</i>	<i>Stacked</i>
<i>Data Matrix</i>	<i>Matrix</i>
<i>Portable Data File 417 (PDF417)</i>	<i>Stacked</i>
<i>QRCode</i>	<i>Matrix</i>

uma dimensão. Devido a limitação no armazenamento das informações por ele, geralmente são guardados valores chaves que possibilitem a busca de informações correspondentes ao código em um banco de dados específico. A figura 2.4b apresenta um símbolo bidimensional denominado *DataMatrix*, as informações contidas neste são armazenadas em ambos os eixos horizontal e vertical, otimizando o espaço utilizado por ele, possibilitando a inserção de caracteres alfanuméricos, bem como um mecanismo responsável pela correção das informações obtidas através do processo de decodificação. Em contraste com esse modelo encontrado nos códigos unidimensionais, as informações armazenadas por esse símbolo remete a um conceito de banco de dados portátil e não somente o conceito de chave como apresentado no modelo unidimensional, podendo armazenar uma maior quantidade de informação [16].



Figura 2.4: Exemplos de símbolos unidimensional e bidimensional.

Os códigos bidimensionais podem ser divididos em duas categorias. A primeira denominada *stacked code* e a segunda chamada de *matrix code*. A diferença entre essas categorias está na sua composição. Enquanto que a *stacked code* trabalha com simbologias compostas por uma série de código de barras unidimensionais empilhadas uma em cima das outras, a *matrix code* codifica os dados com base nas posições dos pontos negros definidos dentro de uma matriz. Isso faz com que cada elemento da matriz possua a mesma dimensão, desta forma a decodificação é calculada em relação a posição do elemento preenchido na matriz. A tabela 2.1 apresenta alguns tipos de códigos bidimensionais e a categoria correspondente a cada código listado.

A figura 2.5 exemplifica um código de barras bidimensional da categoria *matrix code*. Esse tipo de código é denominado de *QRCode* (*Quick Response Code*). O *QRCode* foi desenvolvido no Japão, no ano de 1994 com o objetivo de ser um código que possibilitasse

Tabela 2.2: Níveis de correção de erros suportado pelo QRCode [21].

Nível	Porcentagem de informação recuperada
<i>Low</i>	7%
<i>Medium</i>	15%
<i>Quality</i>	25%
<i>High</i>	30%

um bom armazenamento de informações, compacto e que oferecesse uma leitura rápida e de fácil acesso. Este código pode codificar até 7089 caracteres numéricos (somente números), 4296 caracteres alfanuméricicos (letras e caracteres ASCII) ou 2953 *bytes* de dados binários (*bytes* hexadecimal) [21]. Possui um formato quadrático com dimensões variando de 21 x 21 até 177 x 177 células (conforme a versão do *QRCode*), onde cada célula codifica um *bit*. É oferecido quatro níveis de detecção e correção de erros, possibilitando a recuperação de informações de regiões danificadas do código. A tabela 2.2 exemplifica esses níveis e apresenta a porcentagem de recuperação das informações oferecida por cada nível.

O *QRCode* possui características estruturais com o objetivo de facilitar sua identificação e obtenção das informações necessárias para a sua correta decodificação. Tais características estão ilustradas na figura 2.5.

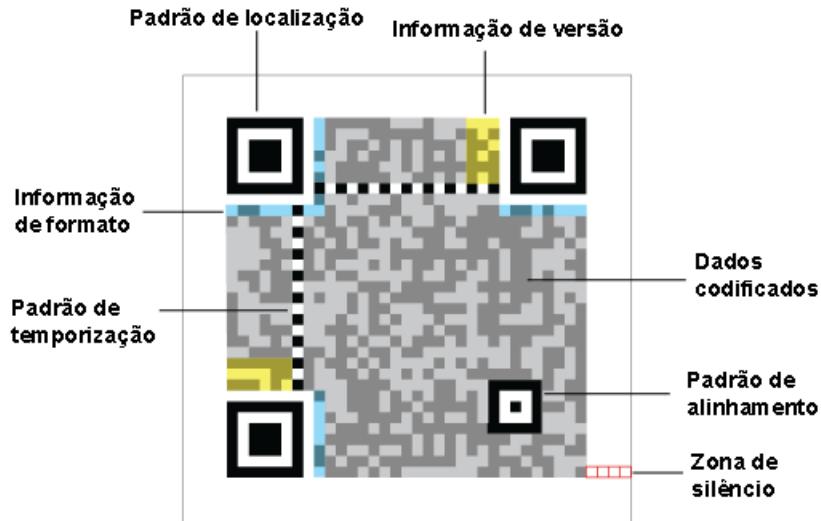


Figura 2.5: Ilustração das propriedades estruturais do QRCode. Adaptado de [29].

- *Padrão de localização*

Esse padrão baseia-se nas posições dos três vértices localizadas nos cantos do símbolo. A partir desses três pontos é possível estimar o quarto canto da imagem,

Tabela 2.3: Comparação de vários códigos [13].

	<i>Barcode</i>	<i>QRCode</i>	<i>DataMatrix</i>
Capacidade de armazenamento	1 byte/4cm	2953 bytes	2335 bytes
Velocidade de leitura	Rápido	Rápido	Lenta
Especificação aberta	Sim	Sim	Sim
Referências internas	0	3	2
Recuperação dos dados	Não	Até 30 %	Até 30 %
Formato	Linear	Quadrado	Retangular

por consequência a posição, tamanho e o centro dos símbolos podem ser detectados, sendo assim utilizado para referência posicional do símbolo. Desta forma, o reconhecimento pode ser feito em todas as direções.

- *Padrão de temporização*

Padrão responsável pela identificação e correção das coordenadas centrais de cada célula. Este padrão é utilizado quando o símbolo está distorcido, possibilitando a identificação do espaçamento entre as células.

- *Padrão de alinhamento*

Por fim, a correção de distorção, especialmente a não linear, é feita através desse padrão. Essa visa corrigir a distorção do símbolo quando este estiver em uma superfície curva ou com o leitor inclinado. Por fim, esta célula facilita a obtenção da coordenada central do padrão de alinhamento.

- *Dados codificados*

É a região responsável pelo armazenamento dos dados codificados.

- *Informação de formato*

Contém a taxa de correção de erro e o padrão de máscara utilizado.

- *Zona de silêncio*

É a margem ao redor do *QRCode* necessário para que o código seja lido corretamente. Possui a medida correspondente a quatro células de largura.

- *Informação de versão*

Responsável pela identificação da versão utilizada pelo *QRCode*, podendo variar a partir da versão 1 (correspondente a 21 x 21 células) até a versão 40 (correspondente a 177 x 177 células). As versões possuem a mesma estrutura, porém a capacidade de armazenamento varia de uma versão para outra.

A tabela 2.3 mostra um comparativo entre diferentes tipos de códigos tanto unidimensionais (*barcode*) quanto bidimensionais (*QRCode* e *Data Matrix*). As quantidade de referência internas apresentadas na tabela dizem respeito a pontos contidos internamente dentro do código cujo propósito seja o auxílio para obtenção do correto posicionamento do mesmo. Também é mostrada a diferença na quantidade de informação armazenada por esses códigos.

2.4.2 Marcadores para Realidade Aumentada

Cada aplicação voltada para a criação e/ou detecção de marcadores pode possuir seus próprios padrões de marcadores e algoritmos para seu reconhecimento. A área de definição de marcadores utilizados ainda está em aberto, com espaço para melhorias de acordo com cada aplicação [30]. Dentro desse universo pode-se citar as aplicações ARToolkit [20], ARTag [15] e ARToolkitPlus [25].

De uma forma geral, para os objetos sejam reconhecidos como potenciais marcadores por essas aplicações, estes necessitam de um formato padrão constituído por uma “moldura” confeccionada com bordas quadradas, de preferência na cor preta, pois algumas aplicações convertem as cores da imagem em uma escala de cinza. A construção e decodificação do conteúdo interior do marcador pode variar de acordo com cada aplicação, possibilitando uma flexibilização na construção do mesmo. Na abordagem da aplicação ARToolkit, o centro desses marcadores pode ser constituído por uma figura. Já no ARToolkitPlus, o interior dos marcadores é preenchido por quadrados de cor preta, podendo formar figuras geométricas diversas. Exemplos de marcadores utilizados por essas aplicações são apresentados na figura 2.6.

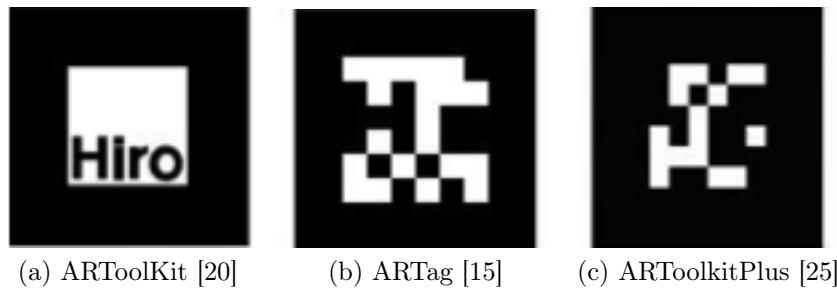


Figura 2.6: *Exemplo de alguns marcadores utilizados na realidade aumentada.*

Fazendo uma comparação entre esses marcadores e o *QRCode* é possível observar que, da mesma forma como acontece com o código unidimensional apresentado na tabela 2.3, estes não possibilitam a recuperação de informações danificadas, mas possibilitam a identificação do erro. Desta forma o não reconhecimento de parte do marcador pode comprometer todo o processo de reconhecimento. Diferentemente do *QRCode*, esses marcadores possuem uma capacidade bastante limitada para armazenamento de informações. Por causa dessa limitação a maioria dos marcadores guardam somente um código identificador. Entretanto, são menos sensíveis a distorções de ângulo devido sua simplicidade.

2.5 Reconhecimento de marcadores

O processo de reconhecimento dos marcadores é dividido em etapas bem definidas. No entanto, estas podem ser implementadas de formas diferentes levando-se em consideração a aplicação utilizada. Esse processo tem o objetivo de identificar objetos com formatos pré-definidos e que possuam códigos de identificação compatíveis com a aplicação. A partir desse reconhecimento é possível estabelecer o posicionamento dos objetos em relação

ao dispositivo responsável pela captura da imagens e apresentar o objeto virtual correspondente sobreposto ao marcador reconhecido. As etapas envolvidas nesse processo são apresentadas na figura 2.7 e divididas da seguinte maneira:



Figura 2.7: *Etapas do processo de reconhecimento de marcadores na Realidade Aumentada [30]*.

1. Procurar marcadores

O fluxo de vídeo obtido por uma câmera é entrada da etapa inicial do processo de reconhecimento. Ele é lido e encaminhado para o sistema de rastreamento. Neste, aplica-se uma operação de detecção de bordas como uma etapa inicial. Faz-se um reconhecimento de cada *pixel* dentro de um segmento para que seja feita um agrupamento de *pixels*, a fim de se achar uma forma quadrangular e reconhecer a figura como um possível marcador. Esse método acaba sendo mais eficaz quando comparado com o método de derivar o marcador a partir de uma intensidade de luminosidade de escala de cinza, devido o seu reconhecimento em ambientes de pouca luminosidade e problemas relacionados com oclusão (parte da visualização do marcador é obstruída por um outro objeto) [15].

2. Encontrar posição e orientação do marcador 3D

Nesta etapa o marcador já foi encontrado na imagem. Para obter uma estimativa da posição do marcador é necessário obter quatro pontos não lineares e identificáveis na imagem. Estes pontos devem aproximar a um formato correspondente a um quadrado. Na situação de identificação das posições baseados na utilização de várias câmeras, envolve a localização do mesmo recurso em duas imagens obtidas por câmeras diferentes. Isso possibilitaria estimar a posição geométrica tridimensional de um recurso. Neste caso para estimar a localização de tal recurso necessitaria combinar as informações de três pontos não lineares. Alguns problemas podem ser detectados, como por exemplo, a distorção de perspectiva ocorrendo quando o plano da tela não é o mesmo plano do marcador [25].

3. Identificar marcadores

É nesta etapa que acontecem mais particularidades para o reconhecimento dos marcadores observadas em diferentes aplicações. Na aplicação ARTag, o conteúdo extraído do interior do marcador é preenchido com uma matriz quadrada 6 x 6 com células de cores preto e branco, aos quais representarão valores binários de 0's e 1's. Essa matriz retornará uma sequência de 36 bits, sendo analisada de quatro formas distintas por causa das quatro rotações possíveis para o marcador.

O código identificador do marcador é constituído por 10 bits, sendo codificado a partir da sequência dos 36 bits obtidos e deixando os 26 bits restantes para a detecção e correção de erros, bem como a unicidade em torno das quatro possibilidades de rotação do marcador [18]. São utilizados os métodos *CRC* (*Cyclical Redundancy Check*) e *Forward Error Correction* para detecção do marcador e extração do seu número identificador correspondente, adicionando o conceito dos operadores lógicos XOR para codificação e decodificação. A correção e reparo dos erros, desalinhamento de fronteira, oclusão, dentre outros, fica sobre a responsabilidade do *FEC* (*Forward Error Correction*). Outros métodos, baseados em probabilidades, também são aplicados para garantir o reconhecimento correto do identificador do marcador em relação aos demais marcadores.

A figura 2.8 exemplifica os passos necessários utilizado pelo ARTag para obter o código de identificação do marcador a partir do marcador reconhecido nas etapas anteriores do *pipeline*.

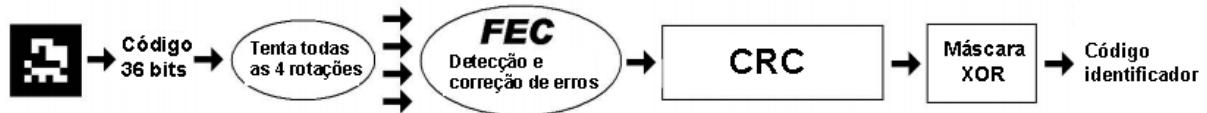


Figura 2.8: Obtenção do código identificador do marcador. Adaptado de [15].

De forma análoga, o ARToolkit extrai o conteúdo do interior do marcador e gera uma matriz quadrada. No entanto, as dimensões dessa matriz são de 16 x 16 ou 32 x 32. Os valores obtidos por essa matriz geram um vetor característico que é comparado com uma biblioteca de vetores característicos conhecidos e geram como resultado um fator de confiança para dizer se o marcador foi reconhecido.

4. Posicionar e orientar objetos

Nesta etapa, os objetos virtuais 3D são identificados e associados de acordo com o marcador identificado na etapa anterior. Os mesmos são alinhados ao marcador pelo posicionamento obtido na etapa 2.

5. Renderizar objetos 3D no frame de vídeo

Por fim é gerado o *frame* de vídeo, contendo o marcador e seu respectivo objeto virtual 3D. Após ser renderizado, o mesmo é repassado para o objeto de visualização de vídeo, podendo ser um monitor, *HMD* ou *smartphone*.

A quantidade de marcadores, aos quais podem ser geradas com uma boa qualidade de reconhecimento varia para cada tipo de aplicação. Essa qualidade de reconhecimento

refere-se a símbolos no interior dos marcadores aos quais são facilmente distinguíveis pelas aplicações. Isso mostra que a quantidade de *bits* obtidos a partir da matriz gerada pelo interior do marcador, pode-se construir diversos símbolos. No entanto, pela isomorfia dos marcadores (rotação do marcador em 0, 90, 180 e 270 graus) muitos símbolos acabam sendo descartados por se tratar do mesmo marcador. No ARTag e no ARToolkitPlus, esses números são bastantes diferentes. Enquanto que no primeiro, 2002 marcadores são reconhecidos facilmente, no segundo esse número cai para 512. Porem, muitos outros símbolos podem ser gerados, mas sem muita garantia de uma boa qualidade em seu reconhecimento. Não foi possível obter as mesmas informações a respeito da quantidade de marcadores que são reconhecidos pelo ARToolkit.

2.6 Trabalhos correlatos

O aumento da mobilidade no ambiente de trabalho de usuário, proporciona uma abstração de diferentes configurações de um ambiente, sendo suportado pelas diversas tecnologias de comunicação sem fio utilizadas na computação móvel. Desta maneira, vários trabalhos estão sendo desenvolvidos utilizando esta computação com o propósito de auxiliar o usuário na solução de seus problemas. Serão apresentados trabalhos que utilizam a computação móvel, aplicadas a ambientes ubíquo.

2.6.1 *CyberCode*

Em [31] é apresentado uma proposta de utilização de marcadores baseados em códigos bidimensionais, denominado de CyberCode. Nesta aplicação as informações são codificadas em um padrão bidimensional para ser reconhecido por dispositivos de baixo desempenho. As informações correspondente aos marcadores são armazenados em um banco de dados, possibilitando assim a alteração dos dados de forma dinâmica, sem a necessidade de alterar o marcador. A figura 2.9 mostra um exemplo do marcador CyberCode.



Figura 2.9: *Exemplo de um marcador CyberCode [31]*.

O uso dos mesmos pode ser combinado com outras tecnologias de rastreamento, com o objetivo de auxiliar o usuário na navegação em um ambiente específico. Desta maneira, eles seriam reconhecidos com o propósito de obtenção da localização do usuário no espaço. Com isso seria possível apresentar as informações ao usuário a respeito dos demais locais próximos a ele.

Uma outra proposta para esses marcadores diz respeito a utilização dos mesmos para o mapeamento dos dispositivos. A própria equipe construiu um dispositivo de fácil manuseio constituído por uma câmera e um visor, denominado de *InfoPoint*. Esse dispositivo capta a imagem correspondente ao marcador, acessa um banco de dados para extrair o conteúdo correspondente e apresenta o conteúdo correspondente no visor do dispositivo. Através desse dispositivo, o usuário é capaz de selecionar um marcador específico e transferir dinamicamente o conteúdo relativo a esse CyberCode para um outro marcador.

Aplicado à computação ubíqua

A integração desse projeto com a computação ubíqua ocorre através da utilização de uma mesa digital. Sobre esta são acopladas câmeras com o propósito de reconhecer os objetos distribuídos em sua superfície. Após o reconhecimento de um novo dispositivo marcado por um CyberCode são apresentadas as informações correspondente ao objeto reconhecido.

Também é apresentada a utilização da mesa através da integração com um *notebook*. O *notebook* é reconhecido através do seu marcador associado, posteriormente é feita uma busca das informações relativas ao dispositivo, como por exemplo seu endereço *IP* e o posicionamento relativo a mesa digital. Desta forma, o *notebook* estará integrado com a rede local juntamente com os objetos físicos reconhecidos pela mesa.

Essa integração possibilita uma troca de informações livre entre os dispositivos. No exemplo anterior, o recurso de tela do *notebook* foi estendido para a mesa digital. Isso possibilita com que o usuário utilize os recursos da mesa digital para interagir com o *notebook*. Por exemplo, caso o usuário sinta a necessidade de mover o cursor do *notebook*, ele poderá utilizar a sensibilidade da mesa para mover o cursor dentro do espaço delimitado como extensão da tela do *notebook*.

2.6.2 *HELLO*

O projeto HELLO (*Handheld English Language Learning Organization*) integra os benefícios provados pela Realidade Aumentada, computação ubíqua e móvel com o objetivo de auxiliar na aprendizagem da língua inglesa [27]. Esse projeto utiliza a tecnologia denominada de *m-learning* (*Mobile Learning*) para que os alunos tenham acesso a informação independente do local físico que eles estejam, flexibilizando e potencializando a aprendizagem. Seu conceito de *u-learning* (*Ubiquitous Learning*), visa a possibilidade do usuário ser inserido em um ambiente onde ele tenha as informações de forma acessível e transparente, com o propósito de flexibilizar e tornar contínuo o processo de aprendizagem. Desta maneira, o usuário obtém vantagens providas pela *ubicomp*, tais como: acessibilidade, interatividade e sensibilidade ao contexto.

O funcionamento do projeto HELLO é dividido em subsistema servidor e um utilitário denominado *u-Tools*. A aplicação servidor fica responsável pelo armazenamento das informações, tais como: materiais didáticos, aulas, banco de dados, provas, dentre outros. A *u-Tools* foi desenvolvida para a plataforma utilizada pelos *PDA's* (*Personal Digital Assistant*), proporcionando assim as funcionalidades de acesso do conteúdo oferecido pela aplicação servidor (através de uma comunicação *Wifi*), a leitura dos códigos de barra bidimensionais e auxílio na comunicação dos usuários. A prova de conceito do projeto

foi realizado em um colégio de ensino médio com o propósito de medição do nível de aprendizagem dos alunos ao término do projeto.

Uso da Realidade Aumentada

Durante uma etapa de aprendizagem, o estudante tinha que obter informações a respeito da execução de uma atividade. Ao se aproximar de uma sala de aula, o aluno percebe a existência de um código de barras bidimensional perto da sala. Obtendo as vantagens providas pelos códigos de barras bidimensionais, o projeto HELLO utiliza o QRCode para obtenção de informação, junto a seus servidores, a respeito da localização do usuário e provê o conteúdo correspondente. O aluno então captura a imagem contendo o marcador e a envia para o processamento nos servidores.

Após o processamento, o servidor envia as informações correspondentes ao usuário (um tutor virtual e os diálogos são apresentados no PDA do usuário) utilizando a Realidade Aumentada. Esse tutor fica responsável pela prática da conversação de acordo com o nível de complexidade da zona que o usuário esteja. Caso o usuário tenha êxito na conversação, o mesmo é encaminhado para uma nova zona até que se complete o percurso. Um exemplo dessa interação pode ser visto na figura 2.10.



Figura 2.10: Exemplo do tutor virtual utilizado no HELLO. Adaptado de [27].

2.6.3 SmART World

SmART World é um *framework* voltado para a computação ubíqua, ao qual utiliza a Realidade Aumentada com o objetivo de criar novos conteúdos virtual e interação com o ambiente inteligente [37]. Outro objetivo desse *framework* está no auxílio da construção de aplicações voltada para a Realidade Aumentada baseada em dispositivos móveis, tornando esse desenvolvimento simplificado. O *framework* é capaz de criar e apresentar objetos virtuais ao usuário, provendo uma interação por meio destes. Sua arquitetura é dividida em:

1. Camada Física

Nessa camada os objetos inteligentes são interligados através de transmissores de rádio frequência, onde estes são ligados um nó central que faz acesso a Camada Intermediária. Esse nó central também fica responsável por reconhecer os demais nós ativos na rede, controlar as informações que são repassadas pelos nós e posteriormente enviar os dados a Camada Intermediária para que as informações correspondentes aos nós sejam atualizadas.

Cada transmissor possui seu código identificador para se comunicar com a rede, essa comunicação é feita através de sensores ou por comunicações interligadas fisicamente. O objetivo da utilização desses transmissores acoplados aos dispositivos possibilita o controle dos mesmos, acrescentando uma inteligência a rede. Desta forma, de acordo com a informação captada pelos transmissores, é possível enviar sinais específicos para os dispositivos. Esse controle será feito através da interface provida pelo *smartphone*, na Camada AR.

2. Camada Intermediária

Essa camada possui a responsabilidade de armazenar e gerenciar todas as informações a respeito do ambiente inteligente e dos usuários e objetos pertencentes a ele. A comunicação entre a aplicação e os demais componentes da rede é feita através de uma interface web que disponibiliza acesso a aplicação e ao banco de dados contido em servidor. O banco de dados armazena informação a respeito do objeto, tais como: posicionamento, modelagem 3D, textos apresentados ao usuário, formato e contexto.

3. Camada AR

Essa camada utiliza uma aplicação voltada para a Realidade Aumentada que é capaz de rastrear e reconhecer os marcadores. Através disso possibilita uma interação entre os objetos virtuais e reais. Essa camada foi projetada para ser utilizada em dispositivos móveis. Por essa razão, o protótipo para essa camada foi desenvolvido para a plataforma Android.

Marcadores e a Realidade Aumentada

O rastreamento e reconhecimento dos marcadores é feito a partir de dois métodos combinados. O primeiro é o reconhecimento dos objetos através do uso de marcadores. Após o reconhecimento, o segundo método é iniciado. Neste, sensores são utilizados para obter a orientação e posicionamento do objeto. Esse posicionamento é gravado na Camada

Intermediária através das coordenadas {x,y,z}. O *framework* possui a funcionalidade de seleção de objetos virtuais através da tela do *smartphone*, possibilitando redefinir o novo posicionamento do objeto virtual selecionado no ambiente inteligente.

Capítulo 3

ARHydra

O *smart space* pode ser composto por uma grande variedade de dispositivos, os quais disponibilizam recursos ao ambiente. Por sua vez, esses recursos devem ser utilizados para suprir as necessidades do usuário e auxiliar na execução de suas tarefas da melhor forma possível. Adicionalmente, o ambiente poderá oferecer ao usuário a possibilidade de seleção do recurso a ser utilizado na execução de uma tarefa. Dentro desse contexto, a aplicação Hydra (apresentado na seção 3.2), utilizando o *middleware uOS* (seção 3.1), possibilita ao usuário escolher e utilizar um determinado recurso disponível no ambiente.

Devido a grande volatilidade e a quantidade de dispositivos dentro do ambiente inteligente, a aplicação ARHydra (apresentada na seção 3.3) utiliza os recursos providos pela Realidade Aumentada e os objetivos do *uOS* auxiliando o usuário na visualização e seleção dos recursos presentes no ambiente inteligente.

3.1 O *middleware uOS*

O *middleware uOS*, projeto do grupo de pesquisa *UnBiquitous* da Universidade de Brasília, tem como objetivo a integração das aplicações no ambiente inteligente e foca na adaptabilidade dos serviços presentes nos mais diversos dispositivos. Desta forma, os serviços passam a ser compartilhados de uma forma menos intrusiva [10]. Conforme apresentado na figura 3.1, o *middleware uOS* atua como uma camada entre as aplicações e *drivers*.

O *uOS* segue os conceitos propostos pela *DSOA (Device Service Oriented Architecture)* para modelagem do ambiente inteligente. Nessa arquitetura são definidos o modelo de comunicação a ser utilizado, implementando algumas características definidas pelo *SOA (Service Oriented Architecture)* [11]. Por causa da diversidade e capacidade de processamento dos diversos dispositivos presentes no *smart space*, foi criado um conjunto de protocolos para a computação ubíqua, implementada sobre o *middleware uOS*, denominada de *uP (Ubiquitous Protocols)*, viabilizando uma maneira simplificada e padronizada para a troca de informações entre esses dispositivos. O *uP* permite uma interação entre os dispositivos, levando-se em consideração a heterogeneidade das plataformas, os diferentes modelos de comunicação e interação com o ambiente. Desta forma, o *uP* possibilita a descoberta de novos dispositivos no ambiente e a representação dos recursos através de *drivers*.

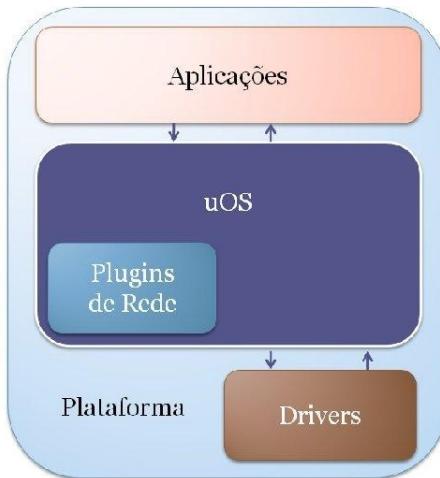


Figura 3.1: *Middleware uOS [10]*.

O *uOS* foi desenvolvido na linguagem JAVA, possibilitando a utilização do mesmo nos mais diversos dispositivos que a ofereçam suporte. Ele se encontra disponível em duas versões: uma *mobile*, sob a plataforma JME (*Java Micro Edition*), e uma seguindo a plataforma JSE (*Java Standart Edition*). Esta última foi também portada para a utilização em dispositivos Android. Viabilizando sua utilização tanto em dispositivos limitados (usando a plataforma *mobile* ou Android) à outros mais capazes.

3.2 A aplicação Hydra

Na visão da *ubicomp*, o *smart space* é composto por diversos dispositivos que fornecem uma gama de recursos e funcionalidades. É plausível de se esperar que muitos destes recursos sejam equivalentes entre si, como a existência de várias telas a disposição. Com tantas opções espera-se que um ambiente inteligente possibilite ao usuário escolher aquela que melhor lhe atende na tarefa sendo realizada. É neste cenário que a aplicação *Hydra* entra em cena. Fazendo uso da plataforma provida pelo *middleware uOS*, esta aplicação permite que o usuário redirecionar os recursos de sua máquina para outros mais adequados no ambiente [6].

Para melhor entender a atuação da *Hydra* no ambiente considere o seguinte exemplo de uso:

O ambiente é dotado dos seguintes dispositivos:

1. Macbook e o *notebook* HP disponibilizam os recursos de *webcam*, saída de vídeo, mouse e teclado;
2. *Laptop* Dell disponibiliza somente os recursos de saída de vídeo, mouse e teclado;
3. *Smartphone* Galaxy SII disponibiliza os recursos de câmera, mouse e teclado;
4. *SmartTV* disponibiliza os recursos de saída de vídeo.

A seguir temos um exemplo de interação neste ambiente utilizando a *Hydra*:

Inicia-se uma reunião cujo propósito é alinhar os conhecimentos a respeito dos projetos do grupo UnBiquitous, discutir as soluções propostas e apresentar os artigos base para os projetos. Os participantes da reunião são Marcelo (dono do notebook HP), Bruno (dono do smartphone Galaxy SII), Ricardo (dono do Macbook) e Fabrício (dono laptop Dell).

Para facilitar a visualização dos tópicos a serem discutidos, Fabrício como líder do projeto, instalou a aplicação Hydra em seu laptop e redirecionou a tela do laptop para a SmartTV. Os demais dispositivos estão utilizando uma instância do middleware uOS, disponibilizando seus recursos ao ambiente.

Fabrício mostra a pauta da reunião. Após a discussão dos primeiros tópicos, Ricardo sente a necessidade de complementar alguns tópicos apresentados. Ricardo então pede ao Fabrício que redirecione o recurso de teclado do laptop Dell para seu Macbook. Após acrescentar os novos tópicos, Ricardo pede ao Fabrício que libere o recurso do teclado, para que seja utilizado o teclado do laptop do Fabrício novamente. Bruno percebe a necessidade de apontar algumas questões nos novos tópicos criados pelo Ricardo e solicita ao Fabrício que o recurso de mouse seja utilizado em seu celular. Com os avanços dos apontamentos feitos pelo Bruno, Marcelo decide mostrar um vídeo que lhe chamou a atenção. Solicita então o uso do recurso de saída de vídeo para que possa exibi-lo na SmartTV. Após o término da exibição do vídeo o grupo encerra as discussões e finaliza a reunião.

Ao final da reunião, a aplicação Hydra proporcionou uma visão diferente no *laptop* Dell pertencente ao Fabrício devido ao redirecionamento dos recursos. Neste, os recursos de saída de vídeo e *mouse* estão direcionados para outros dispositivos no ambiente. Desta forma o *laptop* do Fabrício pode ser visto como uma composição dos recursos que estão distribuídos no ambiente.

Através desse exemplo foi possível observar que os dispositivos foram redirecionados de uma forma manual, partindo de uma necessidade específica. No entanto, a Hydra pode implementar inteligências que simplifiquem essa seleção e o redirecionamento dos recursos.

3.2.1 Recursos

Com o propósito de preservar a invisibilidade do sistema, a interação do usuário com os recursos pode acontecer de três formas distintas:

- Interação direta

Esta interação possibilita ao usuário escolher qual dispositivo será utilizado, levando em consideração as vantagens e desvantagens de cada recurso.

- Interação sugerida

O sistema apresenta, de forma organizada, as possibilidades disponíveis compatíveis com a necessidade do usuário em um determinado momento, sendo esta implementada através de um mecanismo de inteligência.

- Interação automática

Nesta interação, o sistema leva em consideração a sensibilidade do contexto para selecionar, automaticamente, o dispositivo que mais adeque ao propósito de resolução da tarefa intencionada.

A Hydra implementa a Interação Sugerida seguindo os conceitos apresentados pela *DSOA*. Ela oferece suporte para quatro tipos de recursos, sendo estes definidos no ambiente através de *UpDriver's*:

1. *Mouse*

Este *driver* provê uma abstração para comandos enviados por um apontador. Os eventos gerados por esse *driver*, como por exemplo um clique duplo, são enviados ao destinatário através de eventos compostos por mensagens. Estes eventos são reconhecidas na Hydra e a movimentação do ponteiro na Hydra é feita com o uso da classe *Robot*.

São exemplos de serviços disponibilizados por esse recurso:

- posicionamento do ponteiro;
- clique simples;
- duplo clique;
- clique com o botão direito;
- *scroll*.

A implementação desse recurso está disponível para as versões JSE (*Java Standard Edition*) e JME (*Java Micro Edition*).

2. Teclado

Esse recurso possui um comportamento similar ao *mouse*, no que diz respeito envio dos eventos capturados. Este por sua vez, implementa o reconhecimento de eventos para pressionamentos, simples e simultâneo, e liberação de teclas, realizando capturas para:

- Caracteres, maiúsculos ou minúsculos;
- Números, de 0 a 9;
- Comandos de teclado (*shift*, *control* e *alt*);
- Teclas de funções (F1 a F12);
- Botões do teclado numérico.

Devido a diversidade de leiautes configuráveis pelo teclado, os eventos capturados podem ser interpretados pela Hydra de uma forma diferente. Um exemplo desse comportamento pode ser observado no evento de captura da tecla “ç”, para o leiaute do teclado no padrão ABNT2. Neste caso, para o padrão ABNT, o evento que apresenta a tecla “ç” é configurado como uma combinação de eventos de outras teclas. Essas diferenças são esperadas e podem serem corrigidas através da configuração do novo leiaute.

As mensagens recebidas pela Hydra são traduzidos em comandos, que por sua vez são emulados utilizando a classe *Robot*.

3. Tela

Esse recurso utiliza a classe *Robot* para capturar as imagens da tela, ao qual posteriormente será sequencia e transformada em um formato de vídeo adequado para transmissão. A transmissão do *streaming* de dados é feita através do protocolo RTP (*Real Time Protocol*) utilizando o JMF (*Java Media Framework*).

Após o recebimento do conteúdo na Hydra, o *streaming* de dados é interpretado, reconstruído e posteriormente exibido na tela utilizada pela aplicação Hydra.

4. Câmera

De modo análogo a funcionalidade de tela, este *driver* utiliza o protocolo RTP para realizar a transmissão do *streaming* de dados. Adicionalmente, este *driver* disponibiliza os serviços de seleção e ativação de câmeras presentes no dispositivo. Após a detecção das câmeras e da seleção de qual câmera utilizar, a aplicação Hydra receberá o *streaming* de dados enviados pela câmera escolhida e exibirá o conteúdo recebido em uma nova janela na tela utilizada pela Hydra.

Com o suporte para esses conjunto de recursos, a Hydra consegue abranger uma boa gama de cenários de uso. A maioria dos recursos estão classificados em um dos tipos de recursos suportados pela Hydra, possibilitando assim o redirecionamento de diversos recursos presentes no ambiente inteligente.

3.3 Incrementando a Hydra com a Realidade Aumentada

Conforme visto, a Hydra auxilia no redirecionamento de recursos de um dispositivo para outros mais adequados no ambiente. Sua interação é feita de maneira sugerida, de forma que são exibidos para a seleção do usuário apenas aqueles que lhe são compatíveis. Nesta abordagem, cabe ao usuário realizar a ligação entre o nome do dispositivo exibido nas opções e o recurso que deseja utilizar. No entanto, o usuário pode ter dificuldades de associar o nome exibido e ao dispositivo contendo o recurso desejado.

A figura 3.2 mostra uma sala com computadores com características físicas semelhantes, numerados de 1 à 12. Considere que um usuário esteja utilizando o computador de número 4 e deseje utilizar um recurso específico de um outro computador (assuma que todos os computadores estejam disponibilizando seus recursos no ambiente). A Hydra mostrará uma lista de recursos passíveis de redirecionamento e ficará aguardando a seleção de um dispositivo. Suponha que o recurso desejado esteja disponível no computador de número 10 e o usuário o selecione. Nesse momento, o usuário terá que identificar qual computador foi selecionado na lista fornecida pela Hydra. No entanto, como há uma grande similaridade nas características físicas, o usuário terá dificuldades de associar o nome exibido, pela lista da Hydra, ao equipamento selecionado.

A ARHydra (*Augmented Reality Hydra*) visa amenizar esta tarefa fazendo uso de técnicas de Realidade Aumentada. A inserção de informações através de objetos virtuais facilitará na identificação dos dispositivos no ambiente inteligente. Os dispositivos serão mapeados através de marcadores estratégicamente localizados no dispositivo. De modo



Figura 3.2: *Sala com computadores.*

que, novas informações a respeito dos recursos disponibilizados pelo dispositivo fossem apresentadas para o usuário no momento de sua localização.

Esse novo meio de visualização dos recursos proverá uma nova forma de interação entre o usuário e o ambiente inteligente. Desta forma, a utilização das facilidades de redirecionamento de recursos provida pela Hydra pode ser incrementada com uma visualização, e localização, simplificada destes.

3.3.1 Marcadores na ARHydra

Aplicações para a realidade aumentada utilizam marcadores com características próprias, como é o caso das aplicações ARToolkit e ARTag. A maior parte dos trabalhos que utilizam esses marcadores, associam o código de identificação extraído do marcador a um objeto virtual pré-definido. Essa relação causa uma dependência de um mapeamento prévio de todos os objetos virtuais e seus respectivos códigos de identificação, proporcionando uma limitação na interatividade entre novos marcadores.

A ARHydra segue uma estratégia diferente. Utiliza características que favoreçam o reconhecimento e a extração de informações através de seus marcadores, sem a necessidade de uma tabela de mapeamento. Esse comportamento híbrido proporciona uma interatividade na identificação dos marcadores, uma vez que a disponibilidade dos recursos dependem dos dispositivos presentes no ambiente.

A escolha do código QRCode, para identificação dos marcadores, baseia-se na possibilidade de inserção de caracteres alfanuméricos, na rápida leitura e na tolerância a erros oferecidas por esse código bidimensional. Possibilitando assim, uma forma simplificada para a extração da identificação dos dispositivos no ambiente inteligente.

A figura 3.3 exemplifica o marcador utilizado pela ARHydra. Na primeira parte da figura é apresentado um marcador utilizado pela aplicação ARToolkit. Em seguida, um código QRCode responsável por armazenar alguma informação relevante a respeito do

dispositivo. A junção dos mesmos formará um marcador constituído de bordas pretas, aos quais delimitam o marcador e favorecem uma forma rápida de localização, e o QRCode no centro do marcador, responsável pela identificação do marcador.



Figura 3.3: *Marcador utilizado na ARHydra.*

3.3.2 Interação no ambiente

Considere o seguinte exemplo de interação da ARHydra com o ambiente. Suponha um *smart space* composto por:

1. Macbook: disponibiliza os recursos de *webcam*, saída de vídeo, *mouse* e teclado;
2. *Smartphone* Galaxy SII: disponibiliza os recursos de câmera, mouse e teclado;
3. iMac: disponibiliza os recursos de saída de vídeo, *mouse* e teclado.

No macbook terá uma aplicação utilizando o *uOS* disponibilizando seus recursos através de *UpDriver's*, sendo este dispositivo mapeado no ambiente através de um marcador reconhecido pela ARHydra. O *smartphone* ficará responsável pela execução da aplicação ARHydra. Por fim, a aplicação Hydra será executada no iMac.

O uso dos marcadores, alinhado com a mobilidade com que o *smartphone* pode interagir com o ambiente, faz com que a aplicação ARHydra seja utilizada no *smartphone* do usuário. Essa interação possibilita ao usuário uma melhor forma de localização e seleção do recurso disponível.

A figura 3.4 exemplifica o uso de um marcador para identificar o dispositivo no ambiente. Como etapa inicial ilustrada através da figura 3.4a, há uma busca por esse marcador inserido no ambiente, ao qual posteriormente será feito uma tentativa de reconhecimento para obtenção da informação contendo os recursos disponibilizados. Com a conclusão bem sucedida da etapa anterior, os recursos serão apresentados ao usuário, conforme ilustrado na figura 3.4b. Essa exibição é feita utilizando técnicas da realidade aumentada. Por fim, a aplicação ARHydra possibilita ao usuário escolher qual recurso deseja que seja redirecionado para a Hydra.

A interação da ARHydra no ambiente inteligente segue os passos observados na figura 3.5:

1. A Hydra implementa o mecanismo de descoberta de novos dispositivos e os registra os dispositivos encontrados em sua base de dados;



(a) Visualização do marcador

(b) Visualização do objeto virtual

Figura 3.4: (a) Exemplo da visão do usuário para a busca do marcador do dispositivo. (b) Exemplo da visualização do objeto virtual apresentando os recursos disponíveis

2. A ARHydra procura o marcador identificador do Macbook;
3. Faz a reconhecimento e decodificação do marcador encontrado, busca os recursos disponíveis do dispositivo e apresenta esses recursos na tela do *smartphone* com os recursos providos pela realidade aumentada, possibilitando ao usuário selecionar um recurso disponível.
4. Caso o usuário selecione um recurso, a ARHydra faz uma requisição à Hydra para que redirecione o recurso selecionado do Macbook para o uso na Hydra.
5. A Hydra recebe a solicitação vinda do *smartphone*, solicita o registro do recurso selecionado para a aplicação do *uOS* executada no Macbook.
6. A partir desse momento, o recurso do Macbook selecionado pela ARHydra será redirecionado para o iMac.

Com isso, a aplicação ARHydra complementa a interação do usuário com a Hydra, auxiliando na identificação visual dos dispositivos e no redirecionamento dos recursos disponíveis.

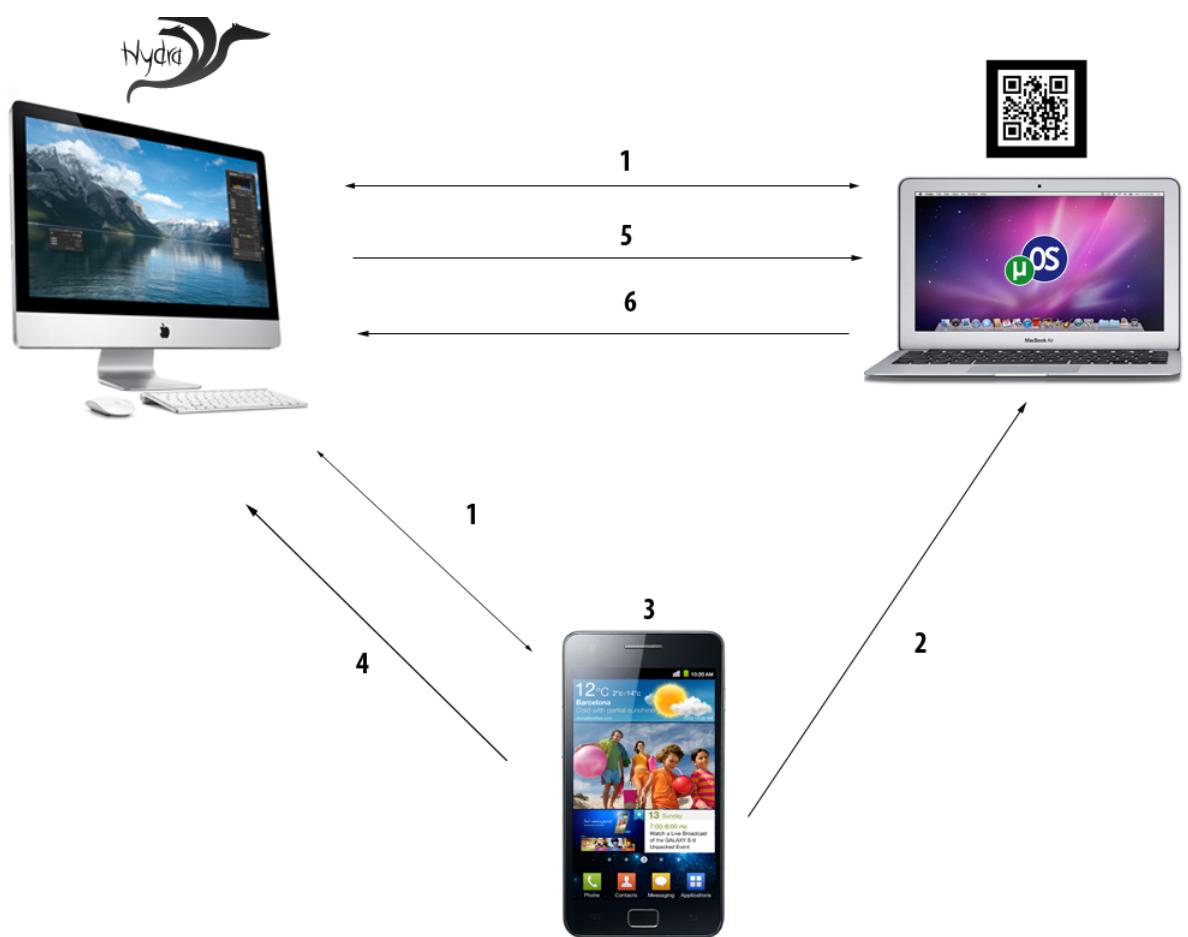


Figura 3.5: *Integração Hydra com o ambiente.*

Capítulo 4

Implementação e testes

A ARHydra tem por objetivo prover uma interface de interação aprimorada à Hydra, de modo que o usuário tenha uma maior transparência e facilidade na seleção dos recursos do ambiente. No capítulo anterior foi detalhado o seu uso, já neste serão apresentados os aspectos relativos a sua implementação. Neste sentido será enfatizada a sua arquitetura e os testes realizados.

A aplicação foi desenvolvida para ser utilizada em *smartphones* dotados de tela sensível ao toque e câmera. Tais dispositivos são adequados para a realidade aumentada devido a mobilidade, além disso permite que o usuário interaja com o ambiente e a informação de forma direta (Visão Direta). Como plataforma foi utilizada o sistema Android que faz uso da máquina virtual Dalvik. Esta escolha permitiu o uso do *middleware uOS*, desenvolvido em Java, sem grandes complicações.

A figura 4.1 apresenta como os quatro módulos da aplicação interagem entre si. O fluxo tem início com a obtenção da imagem capturada pela câmera do celular. Este é repassado ao Módulo de Reconhecimento (seção 4.1) onde são encontrados os marcadores presentes na imagem sendo exibida. Identificado um marcador é então calculado seu centro, bem como a sua orientação.

Conhecendo a posição do marcador na imagem cabe ao Módulo de Decodificação (seção 4.2) extraír a informação presente. Esta consiste de um código identificador do dispositivo representado, sendo utilizado na localização dos recursos disponíveis.

Conhecendo as informações sobre o dispositivo, cabe ao Módulo de Apresentação (seção 4.4) desenhar o objeto virtual na tela. As informações que compõe esse objeto são obtidas a partir da integração entre a Hydra e a ARHydra, tarefa esta sob responsabilidade do Módulo de Integração (seção 4.3). No objeto virtual são exibidas as informações que auxiliem o usuário na sua escolha, sendo estas composta pelo nome do dispositivo e os recursos por ele disponibilizados. Por fim, o Módulo de Integração possibilita ao usuário o redirecionamento ou a liberação de um recurso.

Tanto o Módulo de Reconhecimento quanto o Módulo de Decodificação possuem suas próprias linhas de execução, ocasionando uma interação assíncrona entre esses módulos. Por causa desse tipo de interação, a comunicação entre esses módulos é feita através de eventos, isso propicia que seja mantida a interação junto ao usuário enquanto o processamento é realizado em segundo plano.

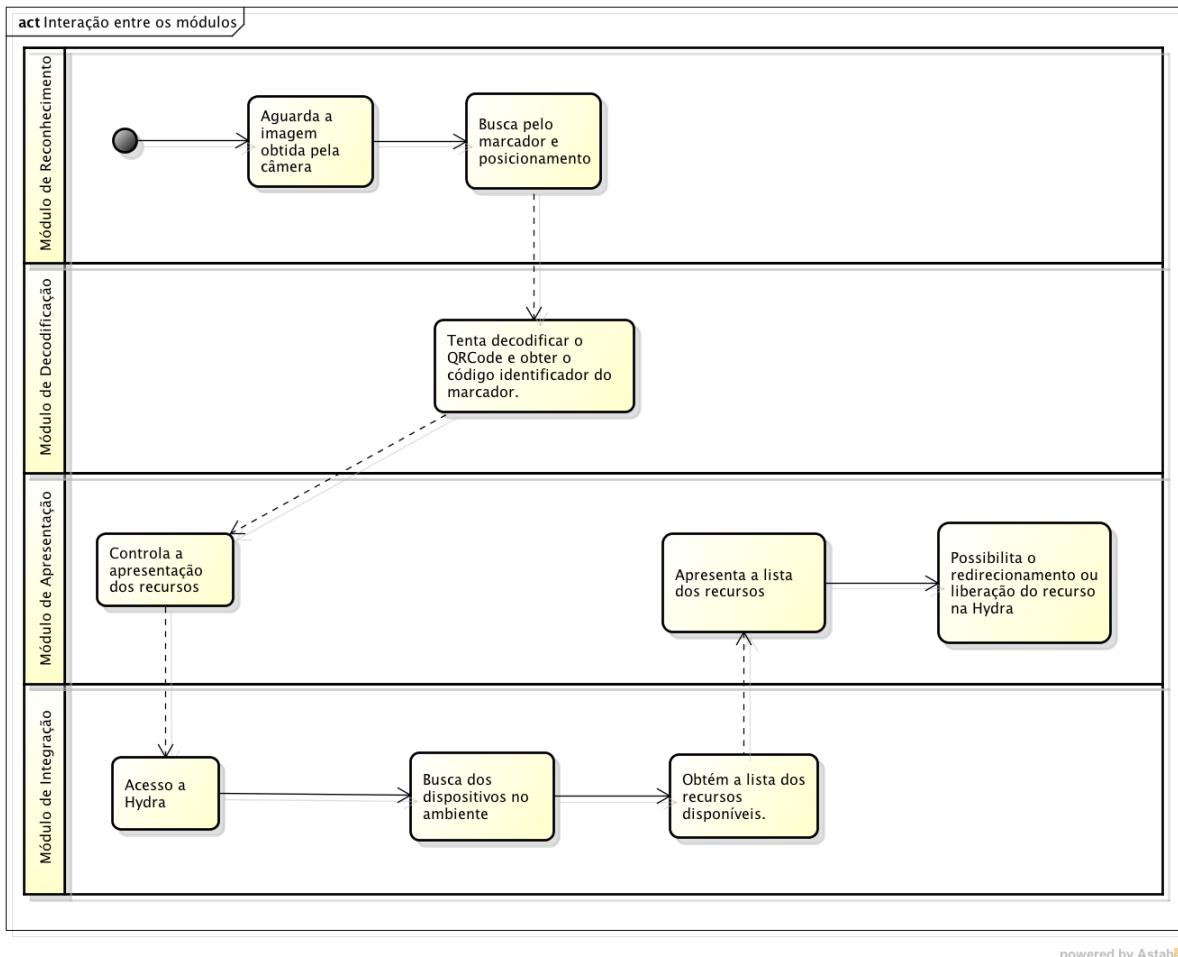


Figura 4.1: *Interação entre os módulos.*

4.1 Módulo de Reconhecimento

Neste módulo é realizada o reconhecimento dos marcadores bem como a obtenção de sua posição na imagem do ambiente. Este é notificado a cada imagem nova capturada pela câmera do dispositivo e de posse desta sua linha de execução inicia os passos apresentados na figura 4.2 e conforme descrito a seguir:

1. A imagem obtida é convertida para escala de cinza com o intuito de se obter uma homogeneidade da coloração dos *pixels*. Desta forma a detecção de padrões ocorre de forma facilitada.
2. Utilizando a biblioteca OpenCV é realizada a correção de perspectiva da imagem, já que o ângulo observado é distinto da figura esperada (vista frontal). Através da mesma biblioteca é realizada a detecção de contornos, com base na borda negra do marcador, e determinação dos quatro pontos que a delimitam.
3. Estima-se o posicionamento do marcador baseado nos quatro pontos não lineares identificados no passo anterior. Nesse procedimento também é obtido o ponto cen-

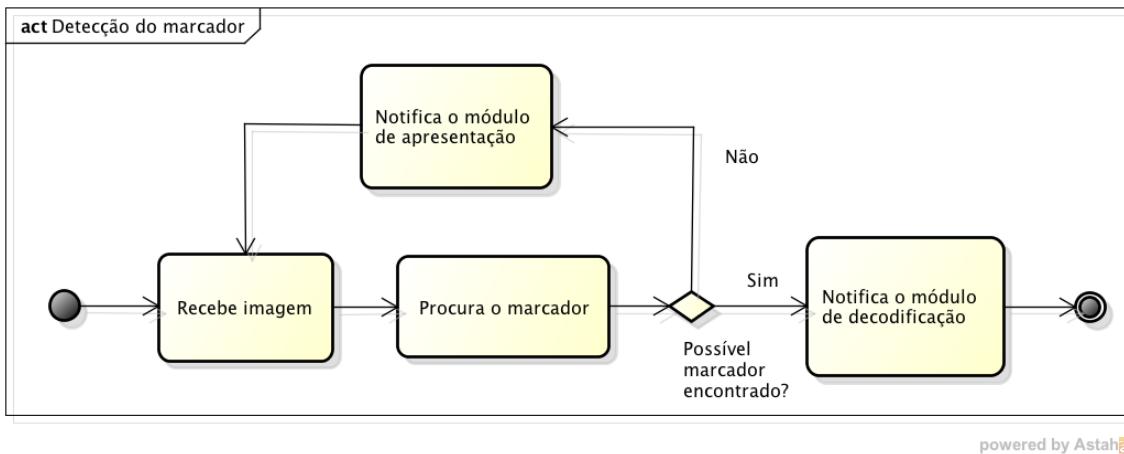


Figura 4.2: Processos do Módulo de Reconhecimento.

tral da imagem, responsável pelo posicionamento correto do objeto virtual a ser sobreposto ao marcador.

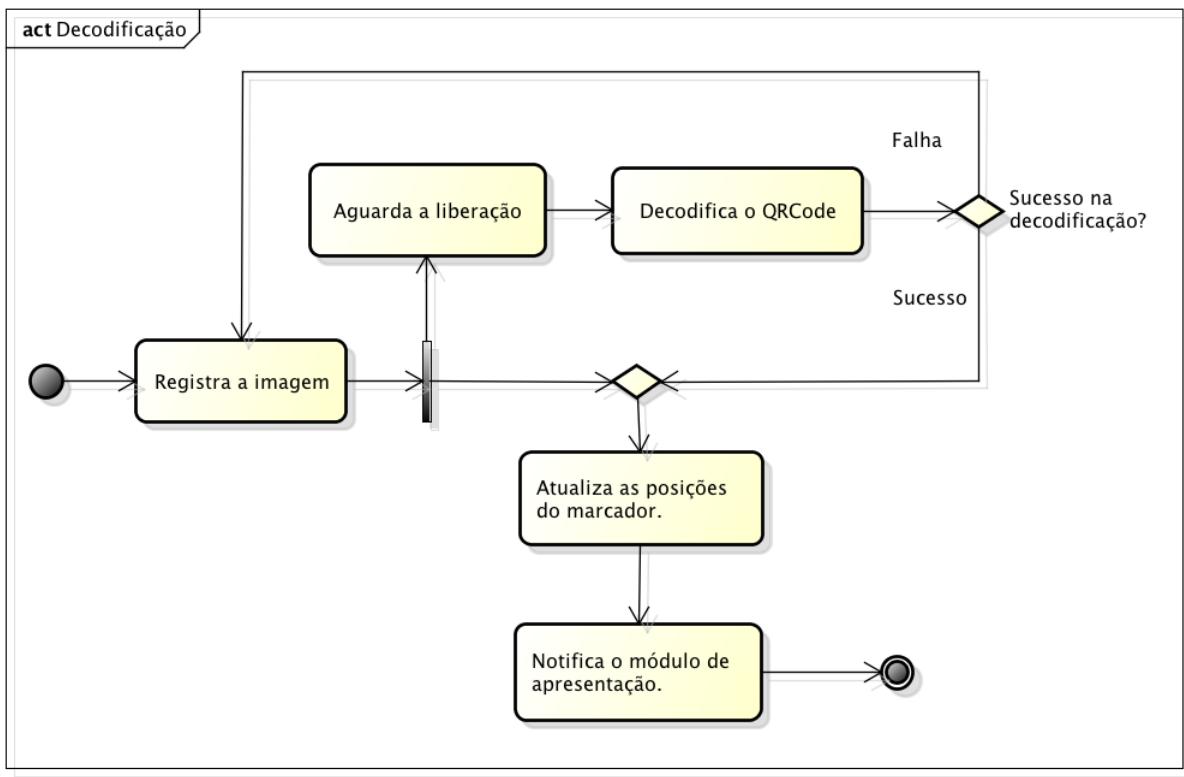
4. Para a obtenção da orientação do marcador foi utilizado o sensor de orientação disponível pelo *smartphone*. A partir dessas informações é possível estimar o ângulo correto de visão do *smartphone*, posicionando o objeto virtual corretamente.

Caso os quatro passos sejam executados com sucesso, tanto o marcador quanto sua posição seja obtida com sucesso, estas informações são repassadas ao Módulo de Decodificação para que se prossiga com o fluxo normal da aplicação. Caso contrário, é feita uma contagem para analisar se o marcador ainda está sendo capturado pela câmera do celular. Se o processo em andamento atingiu a quantidade máxima de tentativas consecutivas sem sucesso, o Módulo de Apresentação é notificado a respeito que nenhum marcador foi encontrado na imagem, desta forma é possível atualizar a visão do usuário. Independente de ter ocorrido um reconhecimento, o módulo voltará a aguardar uma nova imagem para que um novo procedimento seja feito.

Apesar da plataforma Android aceitar aplicações que sejam desenvolvidas utilizando a linguagem Java é possível combinar aplicações Java com aplicações e bibliotecas desenvolvidas em C/C++. Essa integração ocorre através do JNI (*Java Native Interface*) suportada pelo NDK (*Native Development Kit*). Tirando proveito deste suporte, este módulo integra-se com a solução elaborada para o reconhecimento através de uma conexão utilizando o JNI, devido ao fato do processo de reconhecimento utilizar a biblioteca *OpenCV*, implementada utilizando a linguagem C.

4.2 Módulo de Decodificação

Conhecendo a imagem do marcador encontrado cabe a este módulo extrair a informação representada por ele. Para isto são utilizados marcadores QRCode ao qual representa o nome do dispositivo. Desta forma é possível encontrar os recursos (*drivers*) por este disponibilizados. Assim como o Módulo de Reconhecimento, este módulo consiste em



powered by Astah

Figura 4.3: *Processos do Módulo de Decodificação.*

uma linha de execução própria e seu fluxo de execução ocorre conforme representado na Figura 4.3 e descrito a seguir.

O Módulo de Decodificação é criado junto com o módulo de Reconhecimento, no entanto sua execução é iniciada após o recebimento de algum marcador encontrado pelo Módulo de Reconhecimento. O marcador recebido é registrado para que seja feito a atualização das informações a respeito da imagem recebida. Com o registro efetuado com sucesso é dado início ao processo de decodificação do marcador recebido. Pelo fato do processo de reconhecimento ser mais rápido quando comparado ao processo de decodificação fez-se necessário a criação de um controle para que fosse garantido uma correta sincronização na decodificação dos marcadores. Por essa razão, ocorrerá o descarte de qualquer marcador recebido enquanto houver um processo de decodificação em execução.

Foi criado um Gerenciador com o propósito de centralizar todas as informações necessárias, de classes e procedimentos, envolvidas no processo de decodificação. Ele age como uma interface entre o Módulo de Decodificação e as aplicações responsáveis pela decodificação. A aplicação ZBar [3] foi utilizada como ferramenta suporte ao processo de decodificação. Esta ferramenta é implementada utilizando a linguagem C. Por essa razão a integração entre a aplicação ZBar com a ARHydra ocorre através do JNI.

As posições do marcador são atualizadas após o processo de decodificação ser finalizado com sucesso, ou seja, houve êxito na obtenção do código de identificação do marcador analisado. Essa mesma etapa correspondente a atualização das posições é feita quando um marcador é recebido do Módulo de Reconhecimento. Essa tarefa tem um comportamento assíncrono devido o processo de decodificação ser um processo oneroso quando comparado

dos demais procedimentos separadamente. Por essa razão, enquanto houver um processo de decodificação em execução, as posições referentes ao objeto virtual apresentado ao usuário são atualizadas com as informações recebidas pelos marcadores enviadas do Módulo de Reconhecimento. Após a atualização, essas informações são repassadas ao Módulo de Apresentação para que as informações correspondentes ao marcador sejam atualizadas e apresentadas ao usuário.

Por outro lado, caso o processo de decodificação não consiga obter o código identificador correspondente ao marcador analisado, o Módulo de Decodificação enviará uma notificação para o Módulo de Apresentando informando a não possibilidade de detecção do código identificador do marcador, desta forma é possível atualizar a visão do usuário. Após a conclusão dessa notificação, o Módulo de Decodificará aguardará o recebimento da próxima imagem enviada pelo Módulo de Reconhecimento, reinicializando todos os procedimentos até aqui apresentados para este módulo.

4.3 Módulo de Integração

Na composição do objeto virtual é necessário reunir informações relevantes sobre a disponibilidade dos recursos provido pelo dispositivo escolhido. O Módulo de Integração é responsável pela integração da ARHydra com a Hydra com o propósito de criar um canal de comunicação entre essas duas aplicações. Esta integração é feita através de *drivers* de comunicação, conforme estabelecido pela DSOA.

Quando um recurso apresentado pelo dispositivo é selecionado faz-se necessário informar a Hydra qual recurso fora escolhido. Esse canal externo de comunicação entre as aplicações não é fornecida pelo *uOS* de forma nativa. Por essa razão, a arquitetura inicial da Hydra não implementou nenhuma forma para o recebimento de requisições externas. Permite apenas a interação via terminal de aplicação, ou seja, sem acesso por outras aplicações. Desta forma, para o usuário redirecionar algum recurso, ele terá que ir até o dispositivo executando a Hydra e prover o redirecionamento de forma manual. Para realizar essa comunicação foi desenvolvido um *driver* na Hydra que possibilitasse o recebimento de requisições externas. O envio das requisições, bem como o recebimento das respostas realizadas pelas mesmas, decorrentes dessa comunicação, ocorre através do Módulo de Integração.

Para economizar recursos do *smartphone*, a obtenção da informação de quais dispositivos estão disponíveis dentro do ambiente inteligente fica sob responsabilidade da Hydra, ao qual implementará mecanismos para identificação dos dispositivos no ambiente através da utilização de radares. A ARHydra possui um mecanismo de agendamento para que essa informação seja atualizada. Com a periodicidade de 60 segundos, o Módulo de Integração envia informações para a Hydra requisitando quais são os dispositivos que estão ativos dentro do ambiente.

4.4 Módulo de Apresentação

Após o marcador ser reconhecido e identificado, o Módulo de Apresentação fica responsável por apresentar ao usuário todos os recursos disponíveis do dispositivo selecionado. São aceitos os mesmos recursos compatíveis com a Hydra: câmera, *mouse*, teclado ou

tela. Os recursos são visualizados através de um objeto virtual, ao qual é apresentado na tela. Este é representado através de um retângulo onde é exibido o nome do dispositivo e os recursos por ele disponibilizados. Na figura 4.4, podemos observar um caso onde temos o dispositivo iMacDevice que disponibiliza os seus recursos de câmera, *mouse*, tela e teclado.

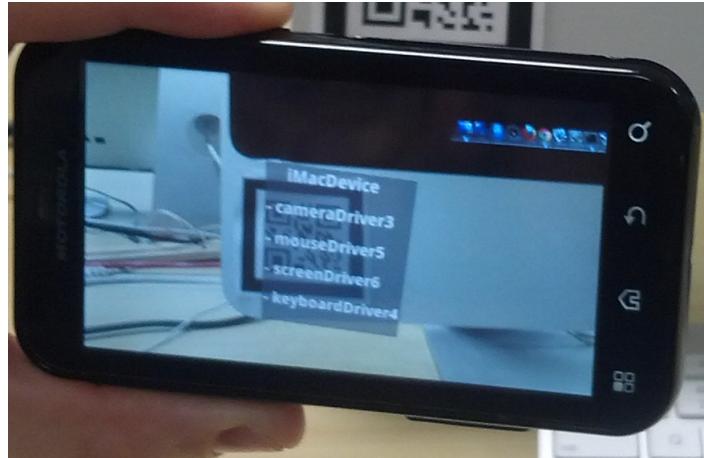


Figura 4.4: *Objeto virtual*.

Para interagir com o dispositivo basta ao usuário tocar a tela sobre o objeto que o representa. Desta forma é exibida uma nova tela onde é possível controlar o uso dos recursos do dispositivo. Possibilitando ao usuário selecionar o redirecionamento ou liberação conforme desejado. A figura 4.5 mostra o detalhamento dos recursos disponíveis ao usuário no dispositivo. Os recursos de teclado e *mouse* já estão sendo redirecionados para a Hydra, enquanto os demais estão disponíveis para serem utilizados pela mesma.

O *framework* DroidAR [1] foi utilizado para facilitar a renderização dos objetos na tela. Para o gerenciamento desses objetos virtuais fez-se necessário a criação de uma classe responsável pela criação, reposicionamento e deleção desses objetos. Essa classe recebe informações dos Módulos de Reconhecimento e Decodificação para determinar a ação a ser executada. A interação desse controle com os demais módulos é apresentado na figura 4.6.

Desta maneira, o controle de exibição dos objetos virtuais possibilita uma correta visualização dos recursos disponíveis pelos dispositivos. De modo que somente os recursos de um dispositivo sejam apresentados por vez.

4.5 Hydra *Driver*

A aplicação Hydra foi concebida para ter seu controle realizado pelo usuário diretamente na máquina de onde os recursos seriam direcionados. Desta forma, não seria possível que outros dispositivos solicitassesem que um novo redirecionamento seja realizado. Para superar esta limitação foi desenvolvido um *driver* denominado *HydraDriver* que disponibiliza as opções de interação da Hydra no *Smart Space*.

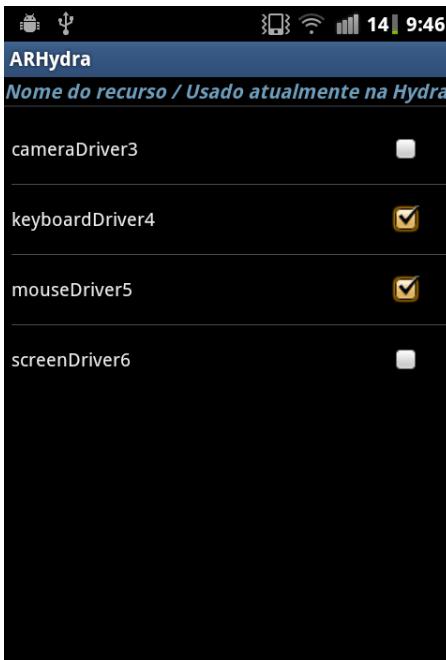


Figura 4.5: *Listagem dos recursos.*

Este recurso visa replicar as mesmas capacidades disponíveis na interface gráfica da aplicação original sendo composto por um conjunto de três serviços síncronos:

1. **Redirecionamento do recurso:** Com este serviço é possível redirecionar um recurso da máquina para um disponível em outro dispositivo.
2. **Liberação do recurso:** Informado o recurso desejado e o dispositivo a ele relacionado, é finalizado o redirecionamento existente entre eles.
3. **Uso do recurso:** Retorna as informações acerca do recurso local. Nestas informações é possível observar se ele está em uso no momento e para qual dispositivo está sendo redirecionado.

4.6 Testes

A operação da aplicação ARHydra está diretamente relacionada ao tempo que esta necessita para levar a informação ao seu usuário. Com o objetivo de medir este tempo foram realizados testes onde se variou a distância do marcador bem como o equipamento utilizado. Os dados foram analisados com relação a execução completa dos três componentes envolvidos na operação (reconhecimento, decodificação e apresentação). Estes testes foram realizados no ambiente do LAICO (LAboratorio de Sistemas Integrados e COncorrentes), situado no Departamento de Ciências da Computação da Universidade de Brasília. Foram utilizados dois *smartphones* distintos, um Motorola Defy e um Samsung Galaxy SIII, de forma a avaliar a influência da qualidade da câmera e do processamento durante o uso. O Motorola Defy possui as seguintes especificações: processador Cortex-A8 de 800 MHz, 512 MB de memória RAM, resolução de tela de 480 x 854 pixels, GPU (*Graphics*

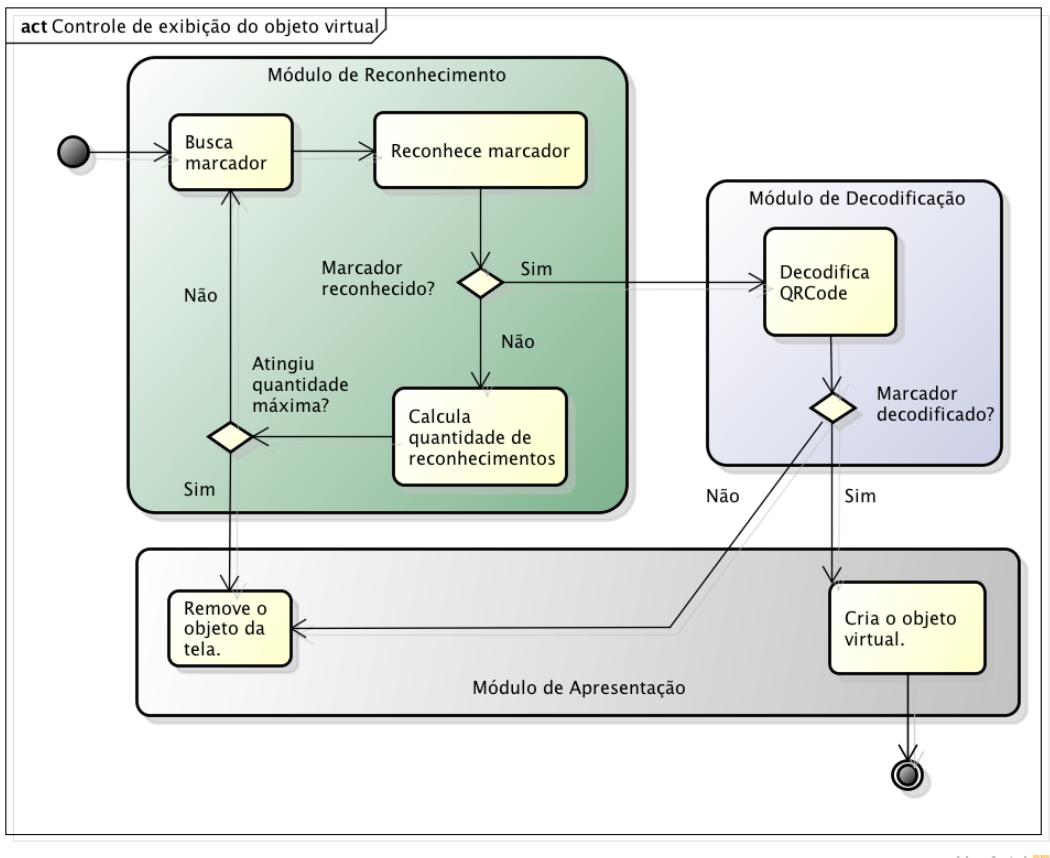


Figura 4.6: Condições para visualização do objeto virtual.

*Processing Unit) PowerVR SGX530 e câmera com resolução de 5MP. O Sistema Operacional testado nesse dispositivo foi o Android 2.3.7 com *firmware CyanogenMod 7.2*. Já o *smartphone* Samsung Galaxy SIII possui processador Quad-core 1.4 GHz Cortex-A9, 1GB de memória RAM, resolução da tela de 720 x 1280 pixels, GPU Mali-400MP, câmera com resolução de 8MP e sistema operacional Android 4.0.4.*

4.6.1 Reconhecimento dos marcadores

Foram executados conjuntos de testes com o objetivo de mensurar o tempo gasto no processo de reconhecimento do marcador proposto. O tempo de reconhecimento é composto pela soma dos tempos gasto na identificação das bordas do marcador, do processo de decodificação do QRCode, da obtenção dos dados (via integração com a Hydra) referentes ao marcador selecionado e da apresentação dos recursos ao usuário. Deste modo, foram propostos testes que realizassem as seguintes medições:

1. **Medição do tempo do primeiro reconhecimento:** Tempo com que o marcador é reconhecido pela primeira vez após a aplicação ARHydra ser inicializada;
2. **Medição do tempo de recorrência:** Tempo com que a aplicação gasta para reconhecer um marcador de forma recorrente, conforme a câmera é movimentada sem que a mesma perca a visualização do marcador;

3. **Medição do tempo de reconhecimento após o marcador não estar mais no campo de visão da câmera do smartphone:** Tempo médio necessário para que a aplicação reconheça um novo marcador após o usuário perder o campo de visão do marcador no *smartphone*.
4. **Taxa de erros:** Este valor representa o percentual das ocorrências com que a aplicação ARHydra deixou de identificar o marcador quando este estava sendo capturado pela câmera do *smartphone*.
5. **Taxa de não decodificação:** Este valor representa a porcentagem média de falha nas decodificações do QRCode inseridos no marcador. As ocorrências nesta taxa são contabilizadas quando um marcador é reconhecido porém não é feita a decodificação do QRCode pelo Módulo de Decodificação. Esse valor varia de acordo com a aplicação responsável pela decodificação, o nível de tolerância a falhas utilizada no QRCode, a qualidade da obtenção das imagens pelo dispositivo e a distância entre o marcador e o *smartphone*.

A aplicação ARHydra oferece suporte, presente no Módulo de Decodificação, para a utilização de diversos aplicativos ao qual disponibilizem o recurso de decodificação de QRCode's. Através desse suporte foram realizados testes que permitiram estabelecer um comparativo de desempenho entre essas diversas aplicações utilizadas. A fim de obter esse comparativo, as aplicações ZBar [3] e ZXing [4] foram utilizadas nos referidos testes.

Uma outra informação importante a ser definida refere-se ao estabelecimento de uma distância máxima para o reconhecimento destes marcadores. Este valor pode variar de acordo com a aplicação utilizada no processo de decodificação, considerando as limitações envolvidas no recolhimento das informações necessárias para a decodificação mesmo quando os QRCode's apresentarem níveis de tolerância a falhas. Desta maneira, para o estabelecimento desse valor, os testes propostos foram executados para diferentes distâncias.

As taxas de erros e de não decodificação estão diretamente relacionadas a qualidade da imagem obtida. No entanto, esta qualidade não diz respeito somente a resolução da câmera utilizada, outros fatores podem influenciar no reconhecimento do marcador. Dentre estes pode-se citar a qualidade das lentes e o algoritmo utilizado na compressão das imagens, possibilitando que seja obtido resultados diferentes para cenários semelhantes, quando estes resultados são coletados utilizando dispositivos dotados de diferentes câmeras. Um outro ponto importante para se garantir esta qualidade está no controle de luminosidade implementado pelas câmeras. Para minimizar esses problemas apresentados é possível ser implementado etapas de pré-processamento da imagem para melhorar a qualidade dessa imagem obtida.

Especificações do Marcador

O marcador foi construído baseando nas especificações apresentadas pelo QRCode (seção 2.4.1) e nas dimensões necessárias voltadas para o módulo de reconhecimento validar o marcador. Também deve ser considerado a inserção desses marcadores no ambiente de forma menos intrusiva possível, mas que suas características possibilitem seu reconhecimento.

A Figura 4.7 apresenta as dimensões adotadas na construção desses marcadores. O QRCode é envolvido por uma moldura, com bordas no valor experimental de 0,8 centímetros, para que o módulo de reconhecimento consiga validar e estabelecer as informações de posicionamento referentes ao marcador e o módulo de decodificação realize com sucesso a decodificação do QRCode.

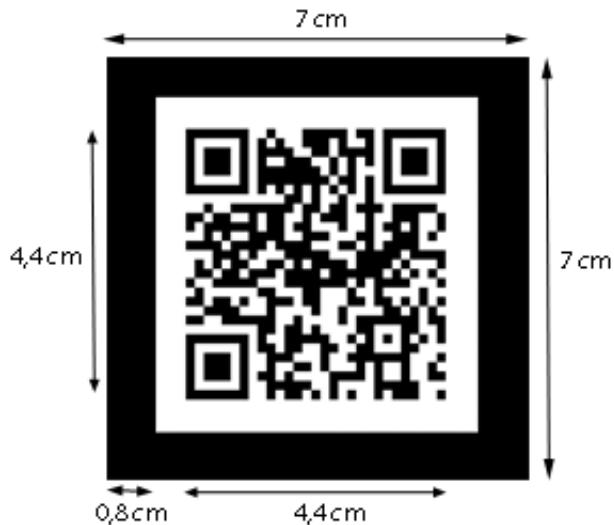


Figura 4.7: Dimensões do marcador.

4.6.2 Resultados

Os conjuntos de testes foram executados a uma distância inicial de 50 centímetros, referente a distância entre o marcador e a aplicação ARHydra executada no *smartphone*, sendo esta aumentada gradativamente em 10 centímetros até atingir o valor de 1 metro. Foram executados os seguintes testes:

- Desempenho e qualidade da obtenção da imagem: Neste teste foi utilizado dois *smartphones* com o propósito de estabelecer a diferença de desempenho da aplicação ARHydra, bem como apresentar o comparativo entre taxas de reconhecimento e não decodificação do QRCode para cada *smartphone* utilizado.
- Suporte a diversas aplicações de decodificação do QRCode: Neste teste foi validado o suporte oferecido pela ARHydra para a utilização de diversos aplicativos cujo propósito seja a decodificação de QRCodes.
- Influência da tolerância a falhas do QRCode: Este permite medir a influência dos níveis de tolerância a falhas aplicados ao QRCode e sua relação com a taxa de não decodificação.

Teste de desempenho e qualidade na obtenção da imagem

Um dos objetivos para este teste é comparar o desempenho da aplicação ARHydra utilizada nos dispositivos mencionados anteriormente, com o propósito de mensurar a diferença nos tempos de reconhecimento do marcador, sendo esta influenciada diretamente pela capacidade de processamento, processador e *GPU*, de cada dispositivo. O outro objetivo consiste em analisar a influência da qualidade da imagem obtida pela câmera, destes dispositivos, estabelecendo um comparativo entre as taxas de erro e não decodificação.

Para a execução do teste foram realizadas vinte medições para a primeira aparição, duzentas medições para as recorrências e cinquenta medições para o reconhecimento ao perder o marcador. Os resultados dessas medições são apresentados nos gráficos das Figuras 4.8 e 4.9.

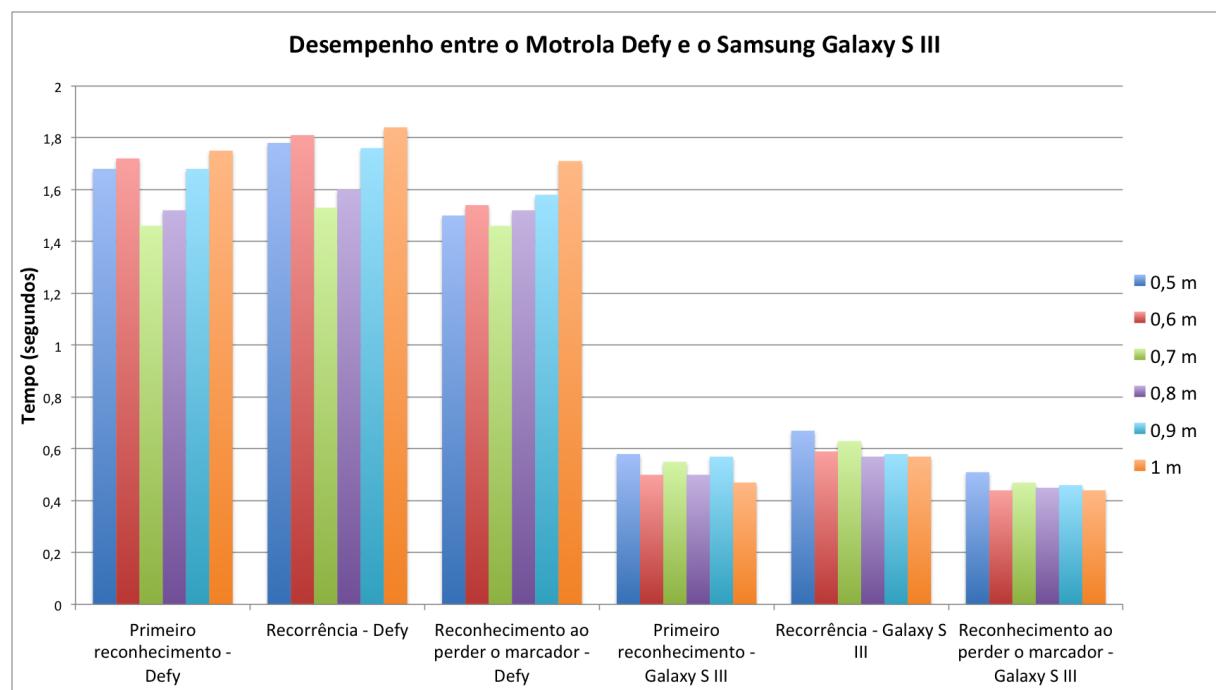


Figura 4.8: Comparativo de desempenho da aplicação ARHydra entre os dispositivos Motorola Defy e Samsung Galaxy SIII.

A Figura 4.8 apresenta o resultado que mede o desempenho dos dispositivos. Ela mostra que o dispositivo Galaxy SIII obteve melhor desempenho, sendo mais rápido no reconhecimento dos marcadores, devido a diferença na capacidade de processamento, processador e *GPU*, presente no processador do Galaxy SIII. Desta forma, quanto maior for a capacidade de processamento do dispositivo menor será o tempo gasto pela aplicação ARHydra reconhecer os marcadores.

Os tempos obtidos no primeiro reconhecimento foram inferiores quando comparados ao tempo gasto na recorrência, em ambos os dispositivos, por causa da baixa quantidade de imagens a serem processadas ou descartadas pelos módulos de reconhecimento, decodificação e apresentação após a captura das imagens ser inicializada. Adicionalmente, a aplicação ARHydra não apresenta um mecanismo de histórico de reconhecimentos ocasi-

onando com que cada reconhecimento concluído com sucesso seja considerado um novo reconhecimento, onerando assim o tempo de recorrência.

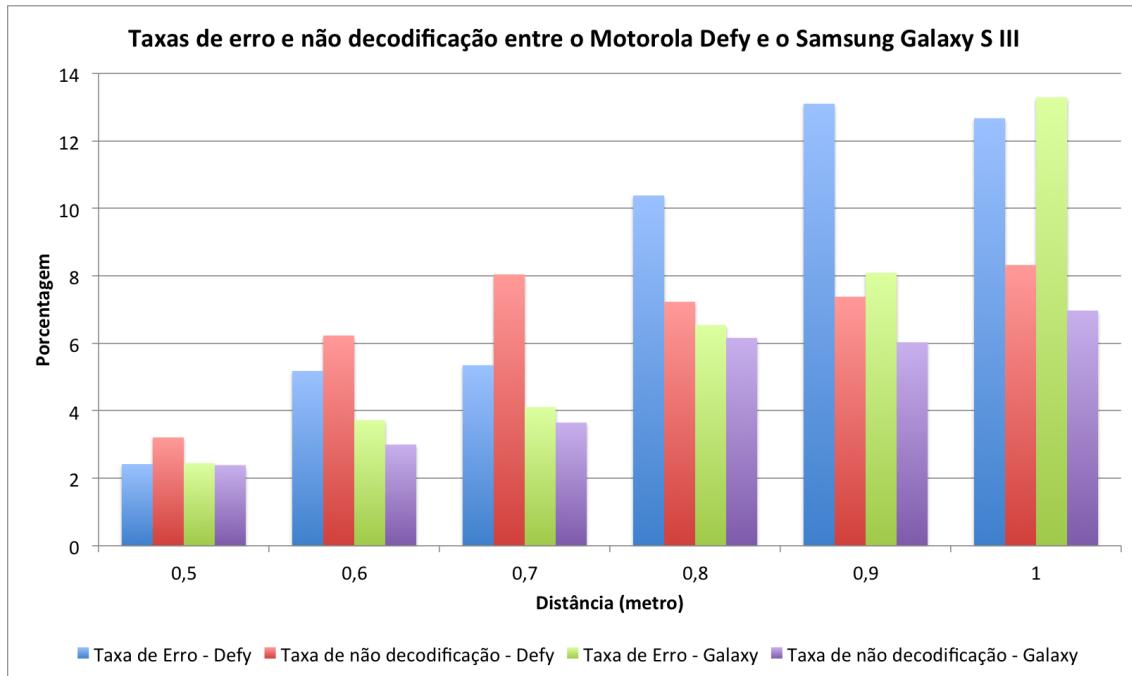


Figura 4.9: Comparativo entre as taxas de erro e de não decodificação da aplicação ARHydra entre os dispositivos Motorola Defy e Samsung Galaxy SIII.

A Figura 4.9 apresenta as Taxa de Erros e Taxa de não decodificação. Novamente o Galaxy SIII mostrou superioridade na maioria das distâncias avaliadas. Esses resultados demonstram a influência da qualidade da imagem capturada em relação as taxas aqui apresentadas. Não levando em consideração as etapas voltadas ao pré-processamento de imagens que possam ser implementadas na ARHydra, conclui-se então que quanto maior for a qualidade da câmera (resolução, qualidade das lentes, algoritmo utilizado na compressão das imagens, controle de luminosidade, etc), juntamente com a melhoria na qualidade dos valores recebidos pelo sensor de orientação do *smartphone*, menores serão as taxas de erro e decodificação apresentadas na ARHydra.

Suporte a diversas aplicações de decodificação do QRCode

A ARHydra permite a integração de diferentes mecanismos para a decodificação de QRCodes a serem utilizados no Módulo de decodificação. Na atual versão da aplicação é permitido utilizar tanto a implementação fornecida pelo ZBar e o ZXing para esta tarefa. Tomando esta flexibilidade como base, foram realizados testes para verificar o comportamento fornecido por cada um destes.

Para estes testes foi utilizado o *smartphone* de melhor capacidade, o Samsung Galaxy SIII, de forma que o nível de tolerância a falhas do QRCode foi ajustado para o modo *Quality*. Neste teste foram obtidas quinhentas medições por posição em cada implementação. Os resultados obtidos podem ser observados na Figura 4.10.

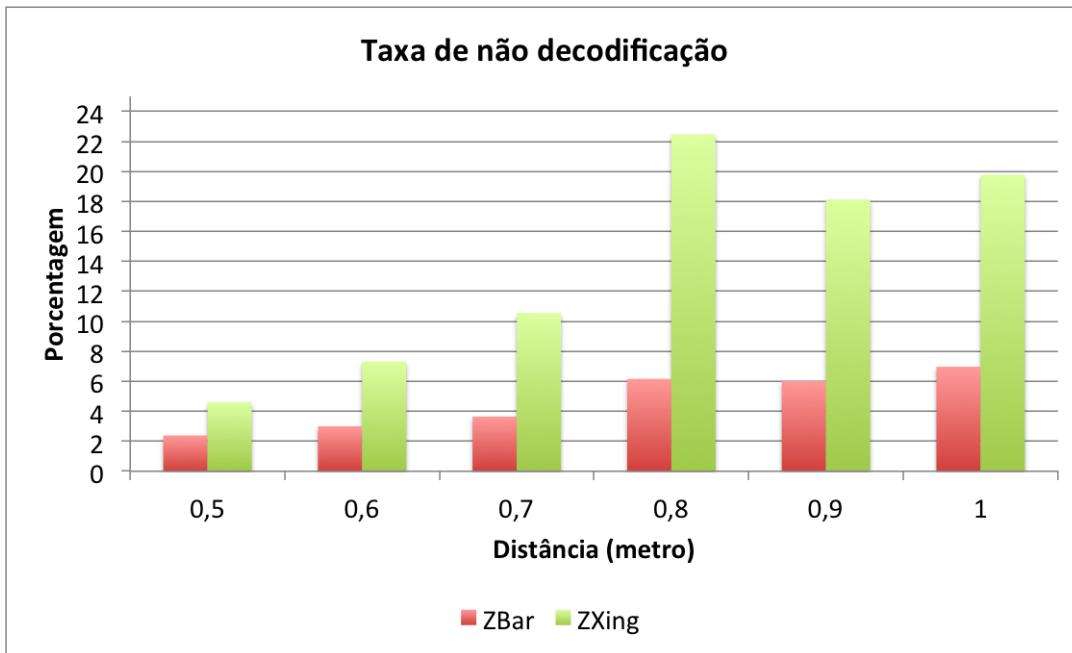


Figura 4.10: *Taxa de não decodificação para as aplicações ZBar e ZXing.*

Os resultados apresentados indicam que a implementação ZBar apresenta taxas de não decodificação mais baixas do que a ZXing. Na maioria das distâncias analisadas, ambas as aplicações apresentaram aumento em suas taxas a medida com que a distância fosse aumentada. A diferença apresentada nos resultados está relacionado ao algoritmo utilizado por essas aplicações, porém não foi possível obter informações a respeito da implementação dos mesmos.

Influência da tolerância a falhas do QRCode

Como visto na seção 2.4.1, os QRCode podem ser compostos por quatro níveis de tolerância a falhas. Este conjunto de testes por sua vez, visa estabelecer um parâmetro de qual nível seria melhor aplicado à aplicação ARHydra, uma vez que a longa distâncias a câmera não consegue capturar os pontos do QRCode, impossibilitando assim sua decodificação.

Para análise dos resultados foram realizadas quinhentas medições para cada distância, sendo esta repetida para todos os níveis de tolerância a falhas. Por fim, o módulo de decodificação foi configurado para utilizar a aplicação ZBar nos testes mencionados, devido suas melhores taxas obtidas no teste anterior.

A Figura 4.11 apresenta os resultados para este teste. Estes níveis possuem percentuais de recuperação a falhas conforme a Tabela 2.2. Os marcadores, quando ajustados aos níveis *Low* e *Quality*, obtiveram melhores resultados comparado aos demais níveis. Embora, as taxas de todos os níveis tenham apresentado aumento à medida que o dispositivo se afastava do marcador em questão. Quando os níveis *Low* e *Medium* são comparados, é possível observar que o formato de seus campos responsáveis pela estruturação do QRCode, mesmo quando a diferença na porcentagem da tolerância a falhas é pequena, interfere na obtenção dos pontos internos ao código, ocasionando uma variação muito

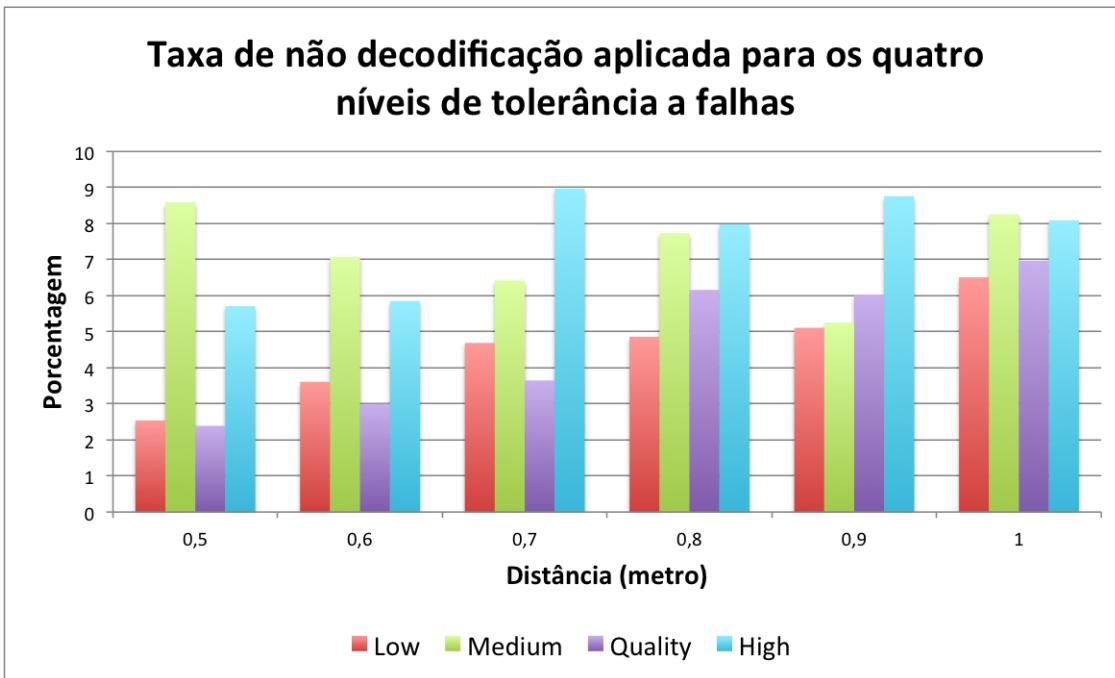


Figura 4.11: Nível da taxa de não decodificação aplicado sobre a influência da tolerância a falhas do QRCode.

grande na taxa de não decodificação. Esse mesmo comportamento é observado para os níveis *Quality* e *High*.

A medida com que é elevado o nível de tolerância a falhas há um aumento da quantidade de informação a ser armazenada para cobrir a porcentagem de informação que pode ser recuperada. No entanto, o espaço ocupado pelo QRCode, no marcador utilizado, permaneceu o mesmo, independente do nível de recuperação a falhas aplicado. Desta forma, quanto maior o nível de recuperação a falhas utilizado mais complexo será a decodificação do marcador utilizado. A partir dos resultados analisados, conclui-se que o nível *Low* de tolerância a falhas é o que melhor se aplica para utilização na aplicação ARHydra para as distâncias analisadas.

Capítulo 5

Conclusão

Com a evolução da computação móvel, os *smartphones* tornaram-se mais presentes em nosso dia a dia devido as crescentes funcionalidades que oferecem aos usuários. Estes, por sua vez, podem ser compartilhados dentro da *ubicomp*, através do acesso aos seus serviços pelos demais dispositivos inseridos dentro do *smart space*. É neste cenário que se insere a construção de mecanismos que simplifiquem a integração e visualização dos recursos. Seu intuito é que o usuário descubra de forma mais simples as opções que se encontram no ambiente a sua volta.

A aplicação ARHydra (*Augmented Reality Hydra*) surgiu a partir dessa dificuldade em se localizar e utilizar os recursos dos dispositivos no *smart space*. Tomando por base os conceitos de adaptabilidade de serviços providos pela DSOA, aplicado ao *middleware uOS*. A aplicação auxilia o usuário a localizar, visualizar e redirecionar os recursos presentes no ambiente inteligente através do uso de recursos da Realidade Aumentada e pela sua integração com a aplicação Hydra.

Foram conduzidos conjuntos de testes cujo objetivo fosse validar a proposta desse trabalho. Como foi possível verificar com os testes apresentados na seção 4.6.2, a aplicação ARHydra operou de forma bastante satisfatória para os resultados apresentados para estes testes. Os resultados apresentaram a influência da qualidade na captura das imagens pelas câmeras. Essa qualidade não é caracterizada somente pela resolução da câmera, mas também por uma série de fatores que possam divergir no resultado mesmo quando aplicados nas mesmas condições de uso. Por essa razão, faz-se necessário aplicação de etapas de pré-processamento de imagens para minimizar este problema.

Adicionalmente, os testes realizados voltados para validação da troca de informação, pelo do Módulo de Integração, entre as aplicações ARHydra e Hydra foram desenvolvidos utilizando o *framework* Junit. Através destes, comprovou-se a correta implementação dos objetivos propostos por este trabalho, facilitando ao usuário encontrar, visualizar e redirecionar um determinado recurso à Hydra em poucos segundos, demonstrando ser uma boa forma de interação com a Hydra. No entanto, os testes demonstraram uma fragilidade da implementação da Hydra para o redirecionamento dos recursos de câmera e tela, implementados utilizando o JMF (*Java Media Framework*), fazendo com que algumas vezes o redirecionamento não ocorra de conforme o esperado.

5.1 Trabalhos Futuros

A aplicação ARHydra tornou possível a interação dos usuários com o *smart space* de forma mais intuitiva, combinando os recursos providos pela Realidade Aumentada e os princípios envolvidos pela *ubicomp*. Essa interação acrescentou mais características ubíquas a Hydra e proporcionou facilidades ao usuário na visualização e seleção dos recursos inseridos dentro do ambiente inteligente, possibilitada através da integração entre dessas duas aplicações.

Com o desenvolvimento da ARHydra, abre-se caminho para melhorias e novos desenvolvimentos. Dentre as melhorias podemos citar:

1. Ampliação no reconhecimento dos marcadores para que a aplicação consiga identificar e interagir com múltiplos marcadores ao mesmo tempo, expandindo as possibilidades observados pelo usuário.
2. Localização e reconhecimento de marcadores dentro de imagens mais complexas, possibilitando a implementação de técnicas de pré-processamento de imagens para aumentar a qualidade do reconhecimento e decodificação do QRCode, podendo minimizar os problemas decorrentes da captura dessas imagens em ambientes com pouca luminosidade.
3. Implementação de um mecanismos que identifique e armazene os marcadores que foram reconhecidos. Atualmente, a não implementação deste mecanismo faz com que cada reconhecimento seja reprocessado a partir do início.
4. Devido a mobilidade dos dispositivos inseridos no *smart space* poderia ser desenvolvido uma integração da ARHydra com tecnologias baseadas em rádio frequência, conforme visto na seção 2.6.3, de forma com que o *uOS* ficasse responsável pelo gerenciamento e atualização do posicionamento dos objetos. Deste modo, o *uOS* disponibilizaria essas informações através de um canal de comunicação com a ARHydra onde essas informações poderiam ser atualizadas a cada novo reconhecimento.
5. Adaptação da aplicação ARHydra para sua utilização com outros tipos marcadores, que não utilizem o QRCode para armazenamento de informações a respeito do dispositivo.

Referências

- [1] DroidAR Augmented Reality Framework. <http://code.google.com/p/droidar/>, 2012.
- [2] The British Museum. <http://www.britishmuseum.org/>, 2012.
- [3] ZBar. <http://zbar.sourceforge.net/>, 2012.
- [4] ZXing. <http://code.google.com/p/zxing/>, 2012.
- [5] Gregory Abowd, Chris Atkeson, and Irfan Essa. Ubiquitous smart spaces. *A white paper submitted to DARPA (in response to RFI)*, 1988.
- [6] Lucas A. Almeida. *Aplicação Hydra para o middleware uOS*. Departamento de Ciências da Computação, Universidade de Brasília, 2011.
- [7] ARToolWorks. Artoolkit for android. <http://www.artoolworks.com/products/mobile/artoolkit-for-android/>.
- [8] Ronald T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, August 1997.
- [9] Mark Billinghurst. Augmented reality in education. *New Horizons for Learning*, (1):339–342, 2002.
- [10] Fabricio N. Buzeto. Um conjunto de soluções para a construção de aplicativos de computação ubíqua. Master’s thesis, Departamento de Ciência da Computação, Universidade de Brasília, 2010.
- [11] Fabricio N. Buzeto, Carlos B. de P. Filho, Carla D. Castanho, and Ricardo P. Jacob. Dsoa: A service oriented architecture for ubiquitous applications. *International Journal on Handheld Computing Research (IJHCR)*, 2(2):47–64, 2011.
- [12] Douglas Chai and Florian Hock. Locating and decoding ean-13 barcodes from images captured by digital cameras. *Information, Communications and Signal Processing, 2005 Fifth International Conference on*, pages 1595–1599, 2005.
- [13] Marlon F. de Alcantara, Marcelo S. Hounsell, and Alexandre G. Silva. Alternative position, orientation and data recognition algorithms for augmented reality markers. In IADIS, editor, *Applied Computing 2011*, pages 557 – 561, Rio de Janeiro, November 2011.
- [14] Paul Dourish. What we talk about when we talk about context. *Personal Ubiquitous Comput.*, 8:19–30, February 2004.

- [15] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, volume 2 of *CVPR '05*, pages 590–596, Washington, DC, USA, 2005. IEEE Computer Society.
- [16] Jerry Gao, Vijay Kulkarni, Himanshu Ranavat, Lee Chang, and Hsing Mei. A 2d barcode-based mobile payment system. In *Proceedings of the 2009 Third International Conference on Multimedia and Ubiquitous Engineering*, MUE '09, pages 320–329, Washington, DC, USA, 2009. IEEE Computer Society.
- [17] Anders Henrysson. Bringing Augmented Reality to Mobile Phones. *Science And Technology*, PhD(1145), 2007.
- [18] Martin Hirzer. Marker detection for augmented reality applications. *Image Rochester NY*, 2008.
- [19] Tobias H. Höllerer and Steven K. Feiner. *Mobile Augmented Reality*, volume Telegeoinformatics: Location-Based Computing and Services, chapter 9. Taylor and Francis Books Ltd., London, UK, 2004.
- [20] Hirokazu Kato and Mark Billinghurst. *ARToolkit User Manual*. Human Interface Technology Lab, University of Washington, 2.33 edition, 2000.
- [21] Hiroko Kato and Keng T. Tan. Pervasive 2d barcodes for camera phone applications. *IEEE Pervasive Computing*, 6(4):76–85, October 2007.
- [22] Hannes Kaufmann and Dieter Schmalstieg. Mathematics and geometry education with collaborative augmented reality. In *ACM SIGGRAPH 2002 conference abstracts and applications*, SIGGRAPH '02, pages 37–41, New York, NY, USA, 2002. ACM.
- [23] Hannes Kaufmann, Karin Steinbügl, Andreas Dünser, and Judith Glück. Improving Spatial Abilities by Geometry Education in Augmented Reality-Application and Evaluation Design. In *7th Virtual Reality International Conference (VRIC - Laval Virtual 2005)*, pages 25–34. Citeseer, 2005.
- [24] Claudio Kirner and Romero Tori. *Fundamentos de Realidade Aumentada*, chapter 2, pages 22–34. SBC, 2006.
- [25] Tijs D. Kle. Integration of the artoolkitplus optical tracker into the personal space station. Master's thesis, University of Amsterdam, Section Computational Science, May 2007.
- [26] Chih-Hsiang Ko, Ting-Chia Chang, Yung-Hsun Chen, and Li-Han Hua. The application of augmented reality to design education. In *Proceedings of the 6th international conference on E-learning and games, edutainment technologies*, Edutainment'11, pages 20–24, Berlin, Heidelberg, 2011. Springer-Verlag.
- [27] Tsung-Yu Liu and Yu-Ling Chu. Handheld augmented reality supported immersive ubiquitous learning system. In *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, pages 2454 –2458, oct. 2008.

- [28] Susanna Nilsson and Björn Johansson. Fun and usable: augmented reality instructions in a hospital setting. In *Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces*, OZCHI '07, pages 123–130, New York, NY, USA, 2007. ACM.
- [29] L2 - The Campaign Technology People. Quick response (QR) codes. http://www.12soft.com/FUSE_T-QR_Codes.aspx.
- [30] Bernardo Reis, João Teixeira, Veronica Teichrieb, and Judith Kelner. Detecção de marcadores para realidade aumentada em FPGA. In *Brazilian Symposium on Computer Graphics and Image Processing*, Rio de Janeiro, 2009. Anais do SIBGRAPI 2009.
- [31] Jun Rekimoto and Yuji Ayatsuka. Cybercode: designing augmented reality environments with visual tags. In *Proceedings of DARE 2000 on Designing augmented reality environments*, DARE '00, pages 1–10, New York, NY, USA, 2000. ACM.
- [32] Tim Suthau, Marcus Vetter, Peter Hassenpflug, Hans-Peter Meinzer, and Olaf Hellwich. A concept work for augmented reality visualisation based on a medical application in liver surgery. In *Medical Application in Liver Surgery, Technical University Berlin, Commission V, WG V/3*, sep 2002.
- [33] Rawesak Tanawongsuwan. Ubiquitous computing (ubicomp). *Human-Computer Interaction*, CS(6751), 1997.
- [34] Mark Weiser. The computer for the twenty-first century. *Scientific American*, 1:94–104, September 1991.
- [35] Mark Weiser. Ubiquitous computing. *Computer*, 26:71–72, 1993.
- [36] Mark Weiser. The world is not a desktop. *Interactions*, pages 7–8, January 1994.
- [37] A. W. W. Yew, S. K. Ong, and A. Y. C. Nee. Smart world - user-friendly mobile ubiquitous augmented reality framework. In Leila Alem and Weidong Huang, editors, *Recent Trends of Mobile Collaborative Augmented Reality Systems*, pages 39–51. Springer New York, 2011. 10.1007/978-1-4419-9845-3₃.