UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

SIMBOLOGIA EM REALIDADE AUMENTADA MÓVEL

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

Gonçalo Filipe Moreira da Silva

Dissertação orientada por:
Prof. Doutor Maria Beatriz Duarte Pereira do Carmo
e co-orientado pelo Prof. Doutor Ana Paula Pereira Afonso

2015

Agradecimentos

Quero, em primeiro lugar, agradecer à minha família e em especial aos meus pais e irmã, pelo apoio, paciência e valores que me transmitiram ao longo da vida, que permitiram que eu conseguisse chegar até aqui e finalizar o meu percurso académico com sucesso.

Um obrigado às professoras Beatriz Carmo e Ana Paula Afonso, pelo suporte, dedicação e disponibilidade demonstrados ao longo do percurso deste projeto e por serem sempre acessíveis e bem dispostas durante as reuniões. Agradeço também ao Tiago Gonçalves, por ter sido sempre tão prestável durante o projeto. Em simultâneo, agradeço ao Departamento Informática, pelas condições e apoio prestado.

Quero tambem agradecer à minha namorada, Rita Vieira, e ao meu melhor companheiro académico e grande amigo, Hugo Duarte, pela paciência, ajuda e camaradagem que tornaram sempre esta longa jornada mais fácil e tranquila.

Por fim, a todos aqueles que participaram nos testes de usabilidade, pelo tempo dispendido, comentários e sugestões fornecidas, que puderam culminar no melhoramento do meu trabalho.

E ainda, a todas as pessoas cujo nome não é aqui mencionado, mas que de forma directa ou indirecta, contribuiram para o meu sucesso académico ou para o avanço deste projeto.

A todos, um muito obrigado.

Aos meus pais

Resumo

A Realidade Aumentada (RA) em dispositivos móveis é uma área em crescente expansão. Na última década, tem-se assistido a um desenvolvimento acentuado de aplicações que fazem uso desta tecnologia de modo a acrescentar informação útil ao utilizador relativamente ao ambiente que o rodeia.

No entanto, a utilização da RA em dispositivos de pequenas dimensões coloca alguns desafios. Por exemplo, ao visualizar pontos de interesse, não existe controlo prévio sobre o seu número e localização, podendo ocorrer a sobreposição de vários símbolos. Por outro lado, a adição de informação à imagem real não tem que ficar confinada aos objetos presentes no campo de visão do utilizador. A inclusão de pistas sobre a existencia de objetos relevantes fora do campo de visão (objetos *off-screen*) é um contributo importante para a navegação pelo espaço de informação. Outro aspecto a ter em conta é a capacidade de expressar a relevância da informação, ou seja, adaptar a representação de acordo com as preferências do utilizador.

Este trabalho propõe soluções para os problemas mencionados, concretizadas no protótipo IAR. Este protótipo inclui técnicas para reduzir a sobreposição de simbologia, representar a relevância dos objetos e incluir pistas para objetos *off-screen* em ambientes de realidade aumentada móvel.

A avaliação com utilizadores revelou que as técnicas propostas pelo IAR para sinalização *off-screen*, representação de relevância e tratamento de sobreposições tornaram a navegação dos utilizadores mais simples e eficiente. A técnica de visualização que usa todas as funcionalidades implementadas foi a preferida pelos utilizadores e revelou o melhor desempenho, enquanto que a técnica de visualização que faz apenas uso da RA sem qualquer funcionalidade foi a que obteve pior classificação e a que mostrou resultados mais ineficazes.

Palavras-chave: realidade aumentada, dispositivos móveis, sinalização *off-screen*, relevância, sobreposição

Abstract

Augmented Reality (AR) on mobile devices is an expanding area. In the last decade, we have seen a huge growth in the development of applications that use this technology to add useful information to the user concerning the environment that surrounds it.

However, the use of AR in small devices is challenging. For example, when visualizing points of interest, there is no prior control over their number and location, thus, the symbols may overlap. On the other hand, the addition of information to the real image does not have to be confined to objects in the user's field of view. The inclusion of clues about the existence of relevant objects outside the field of view (off-screen objects) is an important contribution for navigation through the space of information available to user. Another aspect to keep in mind is the ability to express the information's relevance, that is, adapt the representation according to user's preferences.

This project proposes solutions for the mentioned problems, achieved in the IAR prototype. This prototype includes techniques to reduce symbols overlapping, represent relevance of objects and include clues for *off-screen* objects in mobile augmented reality environments.

It was conducted an evaluation with users that revealed that the techniques proposed by the IAR for off-screen signalization, relevance representation and symbols overlapping treatment have made the navigation more simple and efficient. The visualization technique that uses all the features implemented was the preferred and presented a better performance, while the visualization technique that just makes use of AR without any added functionality was the one that presented more ineffective results.

Keywords: augmented reality, mobile devices, off-screen signalization, relevance, overlapping

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A visualização de pontos de interesse (POI) através de aplicações de realidade aumentada (RA) em dispositivos móveis tem vindo a tornar-se comum, de forma a ajudar os utilizadores nas suas tarefas diárias.

Existem, no entanto, problemas por resolver neste tipo de aplicações. Este documento apresenta o IAR (*Interest Augmented Reality*), um sistema de visualização de POI para dispositivos móveis num ambiente de RA, cujo principal objectivo é a exploração de técnicas de visualização que permitam minimizar algumas das limitações atualmente existentes.

Este trabalho foi realizado no ano lectivo 2014/2015, no âmbito do Projeto de Mestrado em Engenharia Informática do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa. Foi desenvolvido no Laboratório de Modelação de Agentes atualmente integrado no BioISI.

1.1 Motivação

A RA consiste em sobrepor, em tempo real, à imagem do mundo físico, capturado pela câmara do dispositivo, elementos virtuais. Para uma experiência mais rica, os elementos virtuais devem movimentar-se no ecrã, acompanhando o movimento do dispositivo nas mãos do utilizador [13]. Com o crescimento massivo do uso de *smartphones* e *tablets* que integram vários sensores tais como *GPS*, acelerómetro, magnetómetro, giroscópio, entre outros, estão reunidas as condições necessárias para o desenvolvimento de aplicações de RA para estes dispositivos. Um exemplo deste padrão emergente consiste no aumento da popularidade de aplicações na área do turismo, que sinalizam POI em dispositivos móveis, tais como o Layar [3] ou o Wikitude [11]. Estas aplicações, para além de mostrarem POI num ambiente de RA são usadas para obter variadas informações através de elementos textuais e visuais colocados sobre os mesmos [59]. Neste tipo de aplicações, é fundamental conseguir mostrar a informação de forma adequada, permitindo uma análise e compreensão que estimulem uma reacção apropriada e intuitiva por parte do utilizador.

No entanto, quando as aplicações de RA são usadas em dispositivos móveis, apresentam várias restrições. Um dos principais problemas é o utilizador estar limitado ao campo de visão da câmara, enquanto que a informação se encontra em 360º à sua volta [55]. Desta forma, a informação fora do campo de visão da câmara não é visível no ecrã, limitando drasticamente a perceção do utilizador sobre aquilo que o rodeia, obrigando-o a rodar verticalmente e horizontalmente até encontrar os objetos pretendidos. De forma a mitigar este problema, é necessário conseguir fornecer informação no ecrã relativamente à existência de objetos fora da área visível (objetos *off-screen*). Outra restrição dos dispositivos móveis é o ecrã pequeno, que aumenta a probabilidade da ocorrência de sobreposições entre os elementos virtuais que se traduzem em sérios problemas de navegação e perceção da informação mostrada. Por fim, apresentar a informação de forma a que utilizador identifique o que é ou não relevante de acordo com as suas preferências, permite que a sua tarefa de navegação seja mais rápida e intuitiva e o seu objetivo concluído de forma mais eficiente.

1.2 Objetivos e Contribuições

Este trabalho tem como principais objetivos conceber soluções para aplicações de RA móveis que permitam fornecer indicações sobre a existência de objetos *off-screen*, adaptar a simbologia de um ponto de acordo com a sua relevância para o utilizador, e por fim, implementar mecanismos de tratamento de sobreposições entre elementos gráficos. Desta forma, pretende-se disponibilizar mecanismos que permitam auxiliar o utilizador na navegação e exploração de POI em ambientes de realidade aumentada móvel.

As principais contribuições deste trabalho são resumidas nos seguintes aspetos:

- Análise e discussão dos problemas correntes atuais na área da RA em ambientes móveis: visualização *off-screen*, representação de relevância e sobreposição de simbologia;
- Proposta de soluções que permitam solucionar os problemas mencionados;
- Construção do protótipo IAR que concretiza as soluções propostas;
- Execução de testes de utilizador que validam o IAR e determinam a sua utilidade e eficácia na navegação em ambientes de RA móvel.

Para além destas contribuições, foi publicado um artigo na conferência EPCGI (Encontro Português de Computação Gráfica e Interação) sobre o trabalho desenvolvido [39]. Outro artigo derivado deste trabalho foi submetido para a conferência *Eurographics*.

1.3 Metodologia de Desenvolvimento

O desenvolvimento do sistema IAR seguiu um método iterativo centrado no utilizador, no qual, se aperfeiçoam iterativamente as soluções propostas baseadas na recolha de informação proveniente dos utilizadores e testes informais internos.

Apresentam-se de seguida as principais etapas:

- O primeiro passo consistiu na avaliação da bibliografia;
- Numa segunda etapa, e tendo em conta os problemas identificados, propuseram-se novas soluções;
- Seguiu-se a fase de desenvolvimento das soluções para a representação da relevância, sinalização *off-screen* e tratamento de sobreposições;
- Os protótipos foram testados de forma a obter informações úteis. De acordo com os resultados, foram aperfeiçoados os aspetos problemáticos e iniciado um novo ciclo de desenvolvimento.

1.4 Organização do Documento

Este documento está organizado da seguinte forma:

- Capítulo 2 - Conceitos e Trabalho Relacionado: é explicado o conceito de RA e os seus benefícios. Em seguida são analisados vários trabalhos e técnicas desenvolvidas no âmbito da sinalização *off-screen*, tratamento de sobreposições e representação de relevância em ambientes de RA móvel.
- Capítulo 3 - Aplicação IAR: é descrito o processo de desenho para a conceção da aplicação IAR que permita satisfazer os objetivos propostos, desde os esquemas iniciais até ao *mockup* final. Posteriormente, é explicada a forma como foram implementadas as soluções pensadas, que tecnologias se usaram e de que forma o protótipo foi estruturado.
- Capítulo 4 - Avaliação e Resultados: neste capítulo são apresentados os testes de usabilidade realizados para a validação do IAR e debatidos os respetivos resultados.
- Capítulo 5 - Conclusões e Trabalho Futuro: são apresentadas as principais contribuições do trabalho desenvolvido e discutidos os aspetos que permanecem em aberto. Finalmente, são analisadas também possíveis evoluções do trabalho realizado.

Capítulo 2

Conceitos e Trabalho relacionado

Neste capítulo são abordados os conceitos e técnicas relacionadas com o trabalho proposto. A secção 2.1 descreve o que é a realidade aumentada, como funciona e de que forma é usada para melhorar a vida dos utilizadores. Ainda nessa secção são descritos os vários tipos de ecrãs utilizados e as técnicas de *tracking* existentes. As secções 2.2, 2.3 e 2.4 abordam respetivamente, os problemas de sinalização *off-screen*, representação de relevância e sobreposições, e as técnicas que têm vindo a ser usadas e/ou propostas para os resolver.

2.1 Realidade Aumentada

2.1.1 Conceito de Realidade Aumentada

Para melhor entendimento do conceito de Realidade Aumentada, deve pensar-se na realidade como um espetro contínuo onde os extremos são compostos pelo mundo real (tudo é real) e mundo virtual (tudo é virtual), como sugerido por P. Milgram *et al.* [52]. Entre estes dois extremos é possível criar ambientes mistos, mais especificamente, é possível criar um mundo virtual com alguns elementos reais, designada Virtualidade Aumentada (VA), ou um mundo real com alguns elementos virtuais presentes, designada Realidade Aumentada (RA) (Figura 2.1).

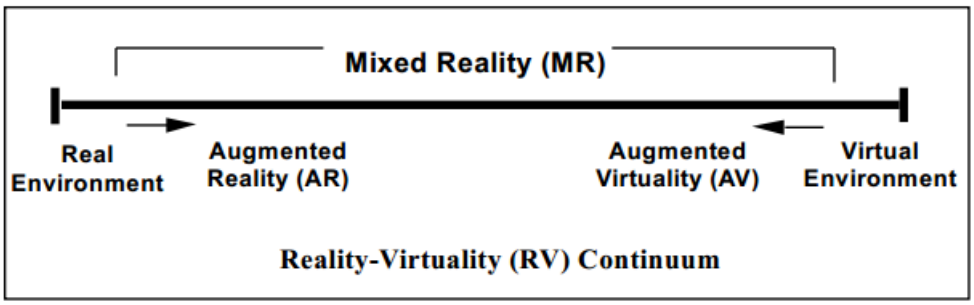


Figura 2.1: Representação do espetro contínuo da realidade [52].

De acordo com Azuma [13], pode definir-se RA como um sistema que cumpre os seguintes requisitos:

- Combina objetos virtuais com o ambiente real;
- Interaje e processa em tempo real;
- É concebido a três dimensões.

De forma geral, é um ambiente misto que adiciona à imagem capturada do mundo real elementos virtuais e que, simultaneamente, responde aos estímulos dados pelo utilizador (ex: mudança de orientação, inclinação, localização) em tempo real. A combinação e alinhamento da parte virtual com a parte real é feita a cada instante e, se o sistema estiver bem construído, será transmitida ao utilizador a ideia de que objetos reais e virtuais coexistem no mesmo espaço. Um exemplo disso, é o jogo ARQuake (Figura 2.2), apresentado por Piekarsky *et al.* [60], que é jogado no exterior e combina as imagens capturadas do mundo real com personagens virtuais que parecem coexistir no mesmo espaço que o utilizador, tornando o jogo extremamente envolvente.

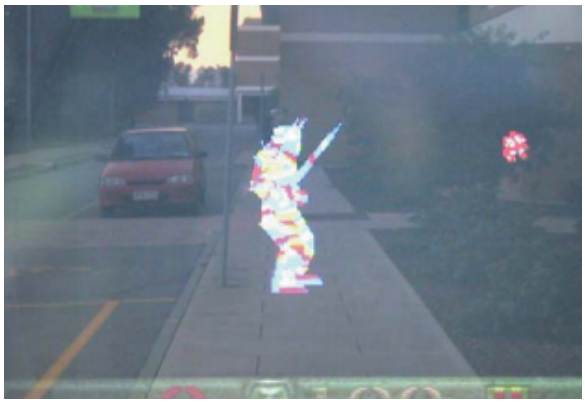


Figura 2.2: Jogo que combina o espaço real com elementos virtuais através da RA [60].

A importância e utilidade da RA prende-se com o facto desta permitir disponibilizar informações úteis e interativas ao utilizador relacionadas com o que este observa, através dos objetos virtuais colocados sobre a imagem real. Desta forma, é possível que o utilizador aumente significativamente o conhecimento sobre aquilo que vê, e que navegue e interaja mais facilmente com o mundo real, ajudando-o a concretizar tarefas do dia a dia de forma mais rápida e eficiente [13].

Apesar de não ser uma técnica recente, o seu potencial tem vindo agora a ser reconhecido e, por isso, a RA encontra-se em expansão e atualmente já existem aplicações que recorrem a esta tecnologia nos mais diversos setores, dos quais se podem nomear: montagem e construção, manutenção e inspecção, navegação e *path-finding*, turismo, trabalhos na área da geografia, jornalismo, arquitetura e arqueologia, modelação urbana, entretenimento, medicina, treino militar e combate, gestão de informação pessoal e *marketing* [45].

2.1.2 Tipos de Sistemas de Realidade Aumentada

Os sistemas de RA podem ser classificados de acordo com o tipo de ecrã que é utilizado. Os ecrãs (*displays*) classificam-se como sendo sistemas de formação de imagem que requerem um conjunto de componentes óticas, electrónicas e mecânicas para gerar imagens algures no caminho ótico (*optical path*) entre os olhos do observador e o objeto físico a ser aumentado [16].

Os sistemas de RA podem ser construídos para vários tipos de ecrãs. Os ecrãs usados podem classificar-se segundo o local a partir do qual estes se utilizam, ou pela forma de apresentação visual da RA ao utilizador (Figura 2.3). Se forem classificados a partir da primeira forma, existem três tipos diferentes: adjunto à cabeça (*Head-Attached*), seguro pelas mãos (*Hand-Held*) ou ainda frente ao utilizador sem que este o esteja a suportar (*Spatial*). Na segunda forma de classificação é possível distinguir os seguintes tipos: visualização através de vídeo (*Video See-Through*), visualização através de superfícies transparentes (*Optical See-Through*), ou visualização através de uma projeção direta dos elementos virtuais no mundo real (*Projective*) [75].

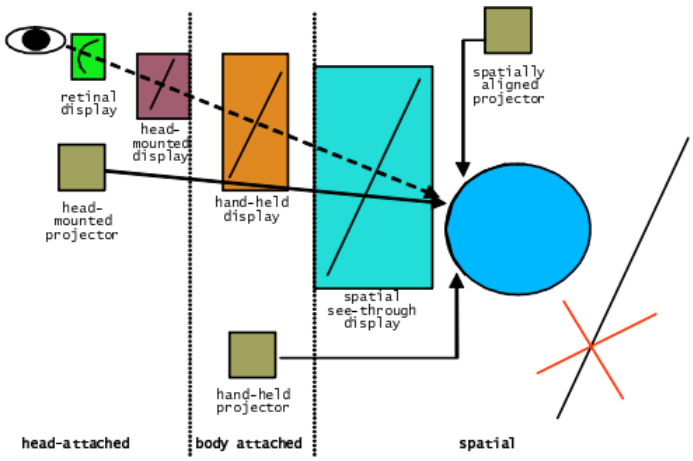


Figura 2.3: Esquema representativo dos tipos de classificação de ecrãs para sistemas de RA [16].

De seguida, é feita uma análise detalhada dos vários tipos de ecrãs existentes de acordo com as duas classificações.

Video See-Through

Os ecrãs *Video See-Through* caraterizam-se por mostrar ao utilizador o ambiente exterior através de vídeo capturado em tempo-real por uma câmara que é projetado num ecrã, juntamente com os elementos virtuais gerados [13]. Ou seja, o utilizador não está em contato direto com o mundo real mas sim com um meio intermédio (monitor de vídeo) entre ambos (Figura 2.4).

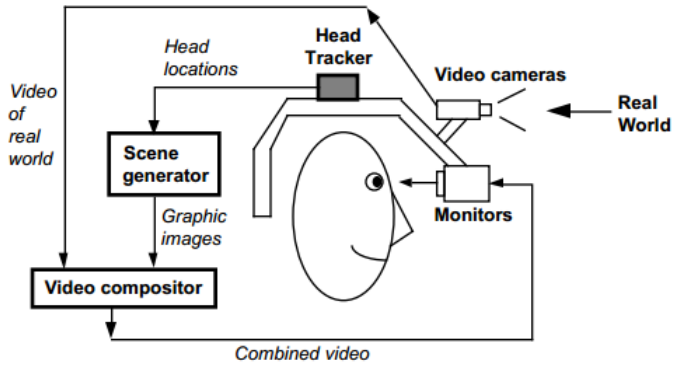


Figura 2.4: Diagrama concetual de um ecrã do tipo *Video See-Through* HMD [13].

Este tipo de ecrãs apresenta algumas vantagens, nomeadamente, a facilidade de integrar ou remover elementos virtuais no ambiente real e a forma como é possível controlar a luminosidade e contraste dos elementos virtuais de acordo com o ambiente real, uma vez que a sequência de imagens capturadas do mundo exterior pode ser facilmente manipulada [75]. É ainda possível obter um campo de visão mais amplo e aplicar técnicas de *tracking* (ver 2.1.3) através da análise do vídeo [13].

No entanto, como desvantagem, a resolução do ambiente real captado é restringida à câmara, sendo inferior à da fóvea humana, podendo tornar-se menos confortável para o utilizador [13].

Optical See-Through

Nos ecrãs *Optical See-Through*, o utilizador observa o mundo real através do ecrã uma vez que este é semitransparente. De seguida, são gerados elementos virtuais que são projetados no ecrã, e consequentemente, sobrepostos ao ambiente real (Figura 2.5) [13]. Nesta técnica, a junção do mundo real com o virtual não acontece no ecrã, mas sim na própria retina do utilizador.

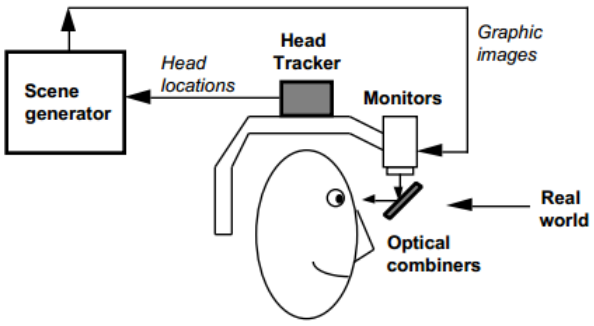


Figura 2.5: Diagrama concetual de um ecrã do tipo *Optical see-through* HMD [13].

Como principais vantagens, nestes ecrãs a resolução apresentada é a da fóvea do utilizador [13], são de baixo custo e seguros, uma vez que é impossível o utilizador perder o

contato com o mundo físico e sofrer do efeito de paralaxe [75].

Por outro lado, trazem também algumas desvantagens. São necessários mais recursos para o *tracking*, uma vez que não é possível fazer o tratamento de imagens capturadas [75]. Além disso, são pouco indicados para utilizações no exterior, pelo facto dos elementos virtuais se apresentarem semitransparentes, o que implica uma baixa definição dos mesmos. O campo de visão também é limitado e a visualização dos elementos virtuais pode não estar sincronizada com o ambiente real observado [13].

Projective

Os ecrãs *Projective* definem-se por não necessitarem de nenhuma superfície intermédia para acomodar os olhos do utilizador, projetando elementos virtuais sobre o mundo real de forma direta. Assim, torna-se possível cobrir grandes superfícies para um amplo campo de visão [75].

Contudo, este tipo de ecrãs é indicado para se usar exclusivamente no interior devido ao baixo brilho e contraste das imagens projetadas [75].

Head-Attached

Os ecrãs *Head-Attached* caracterizam-se por estar adjuntos à cabeça do utilizador. Dentro deste tipo de ecrãs encontramos os *Video See-Through* HMDs, os *Optical See-Through* HMDs, os *Head-Mounted Projective Displays* (HMPD) e os *Retinal Displays* [75].

Os *Retinal Displays* caracterizam-se por utilizarem *lasers* semicondutores de baixa energia para enviar luz modulada diretamente na retina do olho humano. São indicados para o uso exterior uma vez que apresentam uma grande luminosidade e contraste e um baixo consumo de energia. As suas desvantagens prendem-se com o facto de só poderem ser apresentadas imagens monocromáticas (a vermelho) uma vez que ainda não existem *lasers* de baixa energia azuis e verdes económicos, o comprimento de foco é fixo e não existem versões estereoscópicas [16].

Os HMPD são ecrãs também do tipo *Projective*. Funcionam com o princípio de redirecionar o *frustum* de direção com um espelho divisor de feixes. Isto para que as imagens que são transmitidas em superfícies retro-refletivas estejam localizadas em frente ao utilizador. As superfícies refletivas estão cobertas por milhares de micro *corner cubes* que têm como única função ótica a de refletir de volta a luz pela sua direção incidente. Assim, estas superfícies refletem imagens com maior luminosidade do que as superfícies que difundem luz (Figura 2.6) [16].

Verifica-se, no entanto, que os *Head-Attached Displays* sofrem de problemas a nível tecnológico e ergonómico, tornando-se assim ineficaz o seu uso efetivo em todas as áreas de aplicação [16].

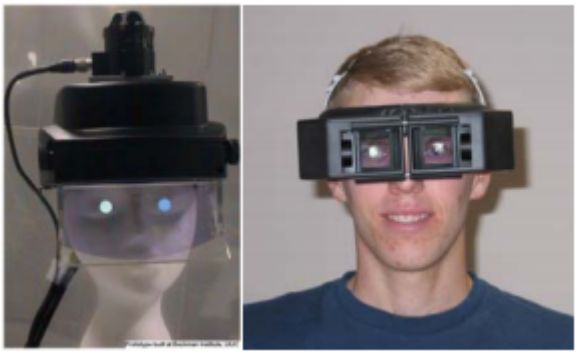


Figura 2.6: Exemplo de um protótipo de ecrã do tipo *Head-Mounted Projector* [16].

Hand-Held

Os ecrãs do tipo *Hand-Held* caracterizam-se por pertencerem a dispositivos móveis com dimensões reduzidas que permitem ser agarrados pelo utilizador e ser vistos à distância de um braço (Figura 2.7). Nesta categoria de ecrãs podem incluir-se os *Hand-Held Video/Optical See-Through Displays* e os *Hand-Held Projectors* [75]. Apesar da mobilidade, o utilizador está constantemente condicionado por não ter as suas mãos totalmente livres para realizar outras tarefas simultaneamente [16].

Figura 2.7: Exemplo de um ecrã do tipo *Hand-Held* [75].

Este projeto utiliza ecrãs do tipo *Hand-Held Video See-Through*, ou seja, todos os protótipos e soluções desenvolvidas em *tablets* vão capturar imagens em tempo real através da câmara e em seguida gerar objetos virtuais que vão ser sobrepostos à imagem capturada.

Estes dispositivos são de baixo custo e incorporam uma variedade de *hardware* já integrado, como a bússola, GPS, câmara de vídeo, acelerómetro e giroscópio, que permitem realizar tarefas de *tracking*.

No entanto, estes dispositvos são muito limitados a nível de processador quando é feito um uso intensivo deste para o *rendering* das imagens. Além disso, o ecrã possui um tamanho muito limitado o que impossibilita o foco numa grande área além de dificultar a disponibilização de muita informação de forma clara. Por fim, os *chips* sensores de imagem das câmaras integradas nestes dispositivos não estão ainda preparados para este tipo de funções e apresentam uma qualidade limitada para tarefas de processamento de

imagens [16].

Spatial

Ao contrário dos ecrãs *Hand-Held* ou *Head-Attached*, os ecrãs *Spatial* separam a maior parte da tecnologia do utilizador e integram-na no ambiente. Esta categoria de ecrãs caracteriza-se por ser colocada estaticamente no ambiente, o que faz deles extremamente escaláveis, bons para apresentações e exibições a grandes públicos com interação limitada [75]. Nesta categoria de ecrãs, os objetos virtuais são projectados no mundo real, por intermédio de uma superfície semitransparente que se situa entre o observador e o mundo físico. Pode-se incluir os ecrãs do tipo *Screen-Based Video See-Through*, *Spatial See-Through* e *Projective* [75].

2.1.3 Técnicas de *Tracking*

A grande diferença entre os sistemas de RA orientados para dispositivos móveis e os restantes, é a mobilidade. Com essa facilidade em mover o dispositivo, e uma vez que os elementos virtuais têm de estar bem alinhados com o mundo físico, torna-se evidente o desafio de determinar de forma contínua a posição e localização o dispositivo e em que direção está orientado, de forma rigorosa. Ao resolver esse desafio, será possivel determinar a posição e orientação que deverá ter o elemento virtual a desenhar no ecrã.

Parâmetros de Avaliação de *Tracking*

Para o sucesso dos sistemas de RA, é essencial criar um ambiente interativo e imersivo que responde como esperado aos *inputs* do utilizador. Este processo é crucial na área de RA, e quanto mais rigoroso for, mais exatos serão os resultados finais obtidos e melhor será a experiência final usufruída pelo utilizador [18]. O processo de *tracking* deve ter em conta os seguintes parâmetros de desempenho:

- Precisão: diferença entre a posição tridimensional real do dispositivo e a reportada pelo *tracker*;
- *Jitter*: representa as variações nos valores reportados pelo *tracker* quando o dispositivo está estacionário;
- *Drift*: representa a degradação da precisão ao longo do tempo;
- Latência: indica o atraso entre uma acção e a resposta do *tracker*;
- Taxa de atualização: número de medidas reportadas pelo *tracker* a cada segundo.

Além da elevada precisão, o processo de *tracking* deve ser realizado com uma elevada taxa de atualização e com baixa latência e *jitter*. A elevada precisão garante que os valores obtidos para a posição e orientação do utilizador são os mais corretos possíveis, e por isso ao usar-se essa informação o elemento virtual irá ser colocado no sítio exato. A elevada taxa de atualização e baixa latência, garantem que o sistema reage rapidamente aos movimentos do utilizador. Por fim, deve-se sempre tentar reduzir o *jitter* ao máximo, de maneira a que os objetos virtuais não se movam caso o utilizador esteja imóvel. Se todos estes parâmetros forem tidos em conta, será possivel proporcionar um ambiente robusto, interativo e confortável [58], caso contrário, haverá *lag*, ou seja, os objetos virtuais vão ainda estar a arrastar-se depois do utilizador ter parado o movimento.

Estimar Posição e Orientação do Dispositivo

Um desafio chave para criar sistemas de realidade aumentada imersivos é conseguir uma mistura credível e verdadeira entre os elementos virtuais e o mundo físico. À medida que o utilizador se movimenta, os elementos virtuais devem manter a sua posição alinhada com o mundo de acordo com a perspetiva do utilizador. Este alinhamento, depende totalmente de um *tracking* preciso da posição e orientação da câmara do dispositivo móvel, relativamente ao ambiente envolvente [77].

Para mostrar elementos virtuais alinhados sobre o ambiente real é necessário entender o ambiente e seguir o movimento relativo do observador de preferência com seis graus de liberdade (6DOF). 6DOF referem-se à liberdade de movimento de um corpo rígido no espaço tridimensional, sendo estes compostos por três variáveis para a posição (x, y e z) e três ângulos para orientação (*yaw*, *pitch*, e *roll*, ou seja, desvio de direção, inclinação de longitude e inclinação transversal) (Figura 2.8) [75].

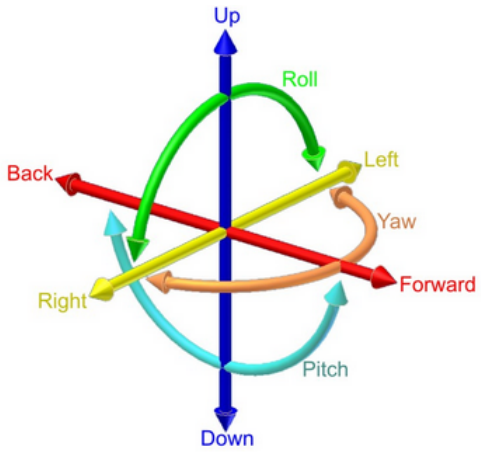


Figura 2.8: Os seis graus de liberdade (6DOF) [9].

De seguida, são analisados as duas grandes abordagens utilizadas para o processo de *tracking*.

Vision-based: Um sistema de *tracking vision-based* é responsável por estimar a posição e orientação da câmara através da análise da imagem capturada pela mesma [49]. A análise de imagem pode ser feita para detetar marcas e implica:

- Reconhecimento de marcas visuais a partir de uma imagem presente no cenário;
- Localização da posição da marca relativamente ao sistema de coordenadas da imagem;
- Dadas n correspondências entre pontos de objetos 3D e as respetivas imagens, é possível estimar a orientação e posição da câmara.

Estas marcas visuais, também designadas marcas fiduciais, *landmarks* ou *markers*, normalmente facilitam todo o processo. São imagens colocadas no ambiente real, fáceis de reconhecer e que proporcionam uma forma credível e fácil de explorar para estimação da posição e orientação da câmara [35]. As marcas dividem-se ainda em ativas ou passivas, sendo ativas quando emitem algum tipo de sinal (ex: magnético, luz) que pode ser capturado por um sensor suportado pelo dispositivo que o utilizador carrega, ou passivas quando são definidas como um padrão que pode ser facilmente identificado no ambiente real através de algoritmos de análise de imagem (Figura 2.9) [18].

Apesar destas marcas facilitarem o processo e o tornarem mais preciso e robusto, exigem algum controlo e intrusão no ambiente para a colocação das marcas, o que pode ser difícil de satisfazer em ambientes exteriores (ex: património natural) [56].

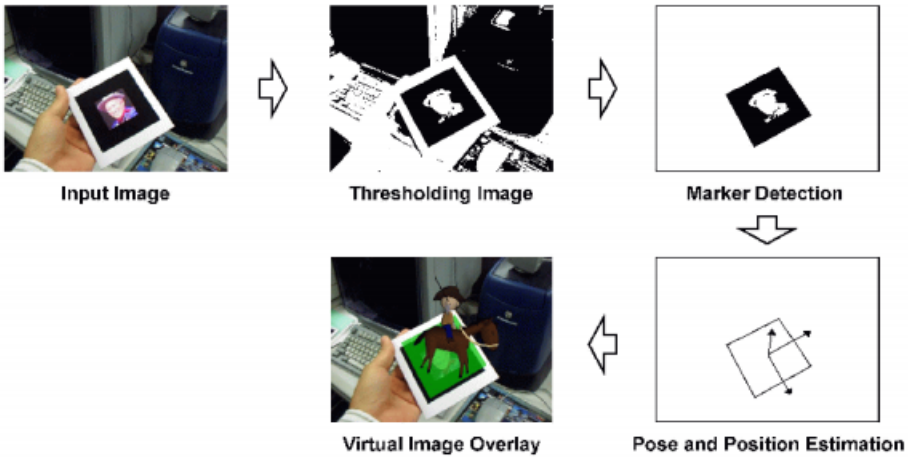


Figura 2.9: Processo de deteção de um *marker* e adição de um elemento virtual de acordo com a posição e orientação da câmara [35].

Outra abordagem designada por *markerless*, não faz uso de elementos distinguíveis que são identificados isoladamente do ambiente envolvente mas sim de elementos pertencentes ao ambiente envolvente para usar como marca, não sendo necessário colocar qualquer marca intrusiva no mundo real *a priori* [73, 15, 31]. Assim, se quisermos fazer

aparecer um elemento virtual sempre que o utilizador vê uma janela, com uma abordagem com *markers* seria necessário colocar um em cada janela do mundo físico, o que seria impensável. Por outro lado, na abordagem *markerless* cada janela seria reconhecida pela aplicação por algoritmos de análise de imagem sem quaquer marca associada.

As técnicas *vision-based* são muito usadas pela sua precisão elevada. No entanto, há uma notória falta de robustez, *delay* do sistema, e elevado processamento computacional. Além disso, como as câmaras dos dispositivos móveis trabalham com taxas de atualização de imagem baixas, a análise de imagem não é apropriada para medir mudanças abruptas de posição ou orientação, pois pode causar falhas e instabilidade [77, 18].

Sensor-based: Sistemas de *tracking* baseado em sensores (ou inercial) calculam a mudança relativa na posição Δp_n e orientação Δq_n a partir da aceleração e velocidade angular do dispositivo. Sabendo a posição p_0 e orientação q_0 iniciais, a posição p_n e orientação q_n atuais podem ser posteriormente determinadas (Figura 2.10) [49]. Estes sensores inerciais, trabalham em conjunto com o GPS e a bússola magnética, presentes nos *tablets* e *smartphones*, para criar um sistema de *tracking* completo.

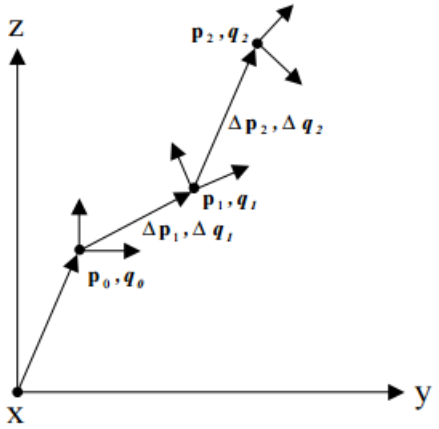


Figura 2.10: Processo de *tracking* inercial que determina a posição e orientação atual através das variações de velocidade angular e aceleração [49].

Os sensores inerciais, presentes nos dispositivos móveis, são usados para *tracking* de movimento, contrariamente à abordagem anteriormente analisada. Estes sensores possuem uma taxa de atualização muito elevada, o que os torna apropriados para registar variações rápidas de orientação e posição. Todavia, os sensores inerciais apenas registam acelerações e velocidades, pelo que estes valores precisam ser computados e integrados para produzir valores finais de posição e orientação. Além disso, os sensores acumulam erros de medição ao longo do tempo devido às latências intermitentes, perturbações no ambiente, e imprecisão inerente ao sistema, o que pode culminar em posições e orientações finais pouco precisas [77, 66, 65].

Métodos de *Tracking* e Ambientes de Utilização

Atualmente existe uma grande variadade de métodos de *tracking*, que segundo Bostanci *et al.* [18], se dividem em quatro grandes categorias: interior (*indoor*), exterior (*outdoor*), estratégias híbridas (*fusion strategies*) e estratégias emergentes (*recent approaches*) (Figura 2.11).

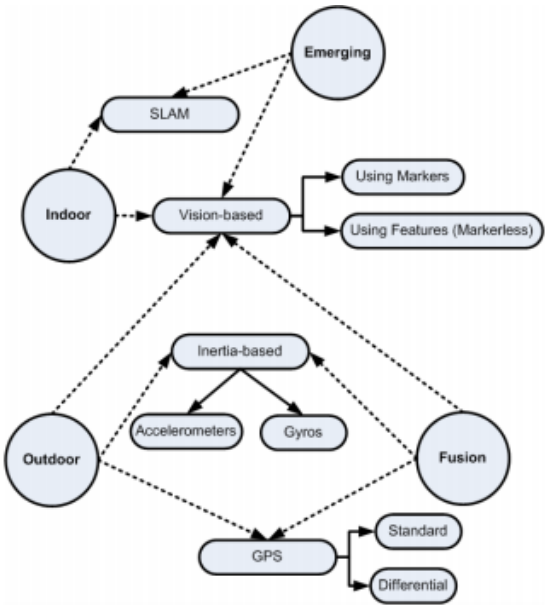


Figura 2.11: Esquema representativo dos vários métodos de *tracking* em RA [18].

Indoor: O *tracking indoor* refere-se ao processo de *tracking* em ambientes contidos dentro de edifícios, em que o ambiente é muito mais condicionado e controlado. Este tipo de *tracking* é normalmente conseguido através de dois métodos: *outside-in* e *inside-out*. No primeiro método, o sensor é fixo num local específico no ambiente envolvente e o utilizador carrega dispositivos nos quais as marcas fiduciais são montadas. Como o nome sugere, o sensor é colocado num sítio à parte do utilizador (*outside*) e identifica as marcas que estão colocadas nos dispositivos que o utilizador usa (*in*). O segundo método funciona de forma inversa, o utilizador carrega o sensor que identifica as marcas que estão colocadas no meio envolvente. Ambos permitem obter a posição da câmara relativamente às marcas [18].

Existem ainda muitos outros métodos de *tracking indoor* recorrendo a sensores magnéticos ou de ultra-som, no entanto, estes são demasiado dispendiosos e de uso complexo. O GPS, apesar de ser uma ótima opção para *tracking* em ambientes exteriores, em ambientes interiores fica com o seu sinal condicionado, o que gera fraca precisão no posicionamento fornecido [18].

Consequentemente, para fazer o *tracking* em ambientes fechados a melhor opção é recorrer a uma abordagem *vision-based*, em que se concebem algoritmos de análise de

imagem para detetar as marcas presentes no ambiente.

Outdoor: Os ambientes exteriores, em contraposição aos interiores, dizem respeito aos ambientes externos aos edifícios, o que os torna menos previsíveis e ilimitados em termos de localização e orientação.

Uma vez que o ambiente não é controlado, apresenta diversos desafios a nível de legibilidade e além disso, marcas pré-definidas possivelmente poderão não ser permitidas o que pode dificultar o processo [18]. Neste tipo de ambientes é necessária uma melhor adaptação dos elementos virtuais a apresentar, uma vez que este tipo de ambientes está sujeito a grandes flutuações da luz natural devido às variadas condições meteorológicas e a inúmeros tipos de diferentes texturas do ambiente real. A luz natural presente no exterior durante o dia pode mesmo chegar a variar entre 1 e 100.000 lux em curtos períodos de tempo [37].

O GPS, apesar de usualmente apenas dar a posição com uma imprecisão com um limite de cerca de 20 metros [51], fornece um sistema simples de *tracking* para ambientes exteriores, que em conjunto com alguns sensores inerciais, permitem calcular a posição e orientação do utilizador [25]. Normalmente, o GPS é usado em simultâneo com acelerómetros e giroscópios que detetam movimentos de translação e rotação, respetivamente, por parte do utilizador. Podem ainda ser usadas redes sem fios (*wireless*), ou métodos de análise de imagem. Os métodos de análise de imagem são particularmente desafiantes no exterior, devido às condições climatéricas, no entanto, de forma a solucionar o problema surgiram abordagens que tentam guardar imagens do ambiente ao longo do ano, fazendo depois um *matching* entre a imagem obtida pela câmara do dispostivo usado pelo utilizador e as imagens guardadas daquele local [18].

Como já anteriormente mencionado, a percetibilidade dos elementos virtuais gerados é degradada, ou por vezes mesmo impossibilitada, nos casos em que a cor e a luminosidade de fundo criam conflitos a nível visual e percetual com a cor ou o contraste dos elementos virtuais. Uma estratégia para mitigar este problema é, em tempo-real, efetuar activamente uma adaptação dos elementos virtuais às diversas condições do ambiente exterior [45] [38].

Resumindo, quando a aplicação é feita para ser usada no exterior, recorre-se a uma abordagem *inertial-based*, onde se usa o GPS juntamente com outros sensores inerciais para calcular a posição e orientação do utilizador.

Fusion strategies: Os métodos de *tracking* de fusão baseiam-se na utilização de vários sensores em simultâneo com o intuito de aumentar a precisão. Os métodos de fusão podem ser classificados como baixo acoplamento ou alto acoplamento. Em sistemas de baixo acoplamento, os sensores atuam separamente e executam cálculos de forma independente. Já nos sistemas de alto acoplamento, os cálculos são realizados em conjunto

por vários sensores para gerar uma estimativa de posição única e melhorada [18].

A técnica *visual inertial tracking* é muito popular por ser um sistema de alto acomplamento que faz o uso de técnicas de análise de imagem em conjunto com uso de sensores. Esta técnica estima a posição e orientação da câmara a partir das análise das imagens capturadas pela câmara, e simultaneamente, a partir dos valores obtidos pelo GPS e por um conjunto de sensores inerciais. Todavia, a técnica de análise de imagens não é robusta com transformações 3D e o cálculo demasiado dispendioso. Por outro lado, nos sensores podem ocorrer erros de calibração e as margens de incerteza podem resultar numa acumulação de erros de posição e orientação a longo prazo. Desta forma, uma computação mais rápida pode ser conseguida com o uso dos sensores inerciais, e o *drift* destes é corrigido pelos algoritmos de análise de imagem.

Concluindo, a abordagem *vision-based* é adequada quando usada com pouca velocidade e aceleração, ao contrário da abordagem *inertial-based*, que tem problemas quando o utilizador está continuamente imóvel. Quando técnicas de análise de imagem e sensores inerciais são usadas em conjunto, fornecem uma computação mais rápida e um resultado mais preciso, equilibrando-se mutuamente e culminando num efeito de simbiose [18].

Emergent approaches: Muitas vezes, as abordagens *markerless* assumem a existência de um modelo tridimensional do espaço que permite saber em que zona exata onde deverá aparecer uma marca [17]. Quando esse conhecimento profundo sobre o cenário não existe, os ambientes nos quais a RA pode decorrer são muito restritos.

O SLAM, *simultaneous localization and mapping*, é o exemplo de uma técnica emergente que assume a existência de um modelo tridimensional inicial do cenário, e que o melhora à medida que o dispostivo navega no mesmo através de *scans* constantes que permitem aprender e melhorar o modelo inicialmente fornecido. Assim, esta técnica que é usada para *tracking*, permite saber a localização do utilizador relativamente ao espaço de forma muito precisa, e consequentemente, permite o acrescento de elementos virtuais de forma muito real. O *SLAM*, já começa a ser usado em sistemas de RA para permitir o seu uso em larga escala em ambientes exteriores [33].

2.2 Visualização *Off-Screen*

Para navegar corretamente, as pessoas precisam planear os seus movimentos com base no conhecimento espacial que têm sobre o ambiente envolvente e que guardam como um mapa mental. Por isso, se o suporte dado pelas *interfaces* de navegação for insuficiente, as pessoas desorientam-se e perdem-se [19].

Um dos principais problemas da RA é que apenas fornece informação sobre os POI que se encontram dentro do campo de visão da câmara. Desta forma, não permite ao utilizador ter perceção sobre os POI que estão fora desse campo de visão, obrigando o

utilizador a navegar extensivamente e a integrar a informação capturada dos diferentes pontos de vista [68, 19]. O problema agrava-se quando falamos de RA orientada a dispositivos móveis, pois a câmara possui uma abertura de câmara inferior que aumenta a probabilidade do ponto não estar a ser capturado por este, e além disso o ecrã pequeno dificulta a leitura de informação [55].

Figura 2.12: Objeto *off-screen* não sinalizado (circulo vermelho) [8].

A Figura 2.12 ilustra o caso em que existe um objeto *off-screen* não sinalizado no ecrã. Vamos supôr que o objeto não sinalizado é aquele que o utilizador está a tentar encontrar. É evidente, que sem uma indicação que dê pistas ao utilizador sobre a localização do POI *off-screen* o processo de procura será muito mais demorado e ineficiente, pois terá que tentar rodar o dispositivo em vários sentidos e amplitudes até o encontrar. Assim, se existisse algum tipo de sinalização o utilizador saberia o que está na sua direção, e em simultâneo, o que existe à sua volta.

De seguida, são analisadas algumas técnicas atualmente existentes em contextos 2D e em RA, usadas para mitigar o problema descrito.

2.2.1 Abordagens Baseadas em Ampliação

Uma das etapas mais complexas do processo de criação de técnicas para visualização *off-screen* é a forma como se estrutura e apresenta essa informação ao utilizador no pouco espaço disponível no ecrã. Por um lado, é necessário conseguir fornecer ao utilizador informação detalhada, e por outro, informação geral que o contextualize para que este possa explorar a informação disponível eficazmente. Assim, o espaço disponível no ecrã torna-se crítico e determinante nas decisões para a criação de uma *interface* que satisfaça essas condições [21].

De seguida, são descritas soluções baseadas em técnicas de ampliação: *overview + detail*, *pan and zoom* e *focus + context*. Estas abordagens não apresentam pistas para objetos *off-screen* mas permitem alargar a informação apresentada ao utilizador. Além disso, algumas técnicas de visualização *off-screen* em RA tem por base estes conceitos.

Overview + Detail

A abordagem *overview + detail* fornece duas vistas separadas em simultâneo, uma de contexto e outra de detalhe. Assim, o mesmo espaço de informação é visualizado de forma distinta, em duas vistas diferentes (Figura 2.13) [29]. Em termos de tamanho, a vista de contexto é quase sempre mais pequena que a detalhada, e está normalmente colocada a um canto do ecrã, mas não há qualquer norma definida relativamente a este aspeto [21]. Na verdade, o tamanho e posicionamento das vistas deverá ser definido de acordo com as tarefas que se esperam ser executadas pelos utilizadores [61].

Figura 2.13: Exemplo de *overview + detail*: vista detalhada sobreposta no canto inferior direito da vista contextual [27].

Apesar de ser uma estratégia bastante eficiente em *desktops*, esta abordagem tende a falhar em dispositivos móveis. Devido ao espaço limitado, torna-se muito difícil relacionar as duas vistas, e pode ser até impossível colocá-las no ecrã em simultâneo. Por outro lado, não é fácil para os utilizadores relacionarem ambas as vistas, mesmo para pessoas que estejam familiarizadas com mapas [27].

Pan and Zoom

A técnica de *panning* (panorâmica) e *zooming* (mudança de escala) é usada em espaços de informação demasiado grandes para serem convenientemente apresentados numa única janela [30]. Desta forma, são fornecidas funcionalidades ao utilizador que lhe permitem selecionar a porção de espaço que querem visualizar, deixando tudo o resto como *off-screen* [23]. Essas funcionalidades incluem: *panning*, que torna possível aos utilizadores mudarem a informação visível através de translações horizontais e verticais, e *zooming*,

que modifica a escala a que a informação está a ser visualizada, permitindo aos utilizadores verem com maior ou menor detalhe as regiões de interesse [**?**]. A Figura 2.14 apresenta os diagramas espaço-escala [36] que ilustram as variações de escala segundo o eixo vertical e as operações de panorâmica nos planos horizontais. Na figura apresentada, a janela de visualização (a) é deslocada em torno do diagrama para obter todos os possíveis pontos de vista. Em (b) é executada uma operação de *zoom* na sobreposição dos circulos, em (c) uma operação de *zoom out* para incluir toda a imagem original, e por fim, em (d) a janela é arrastada para uma parte específica da imagem.

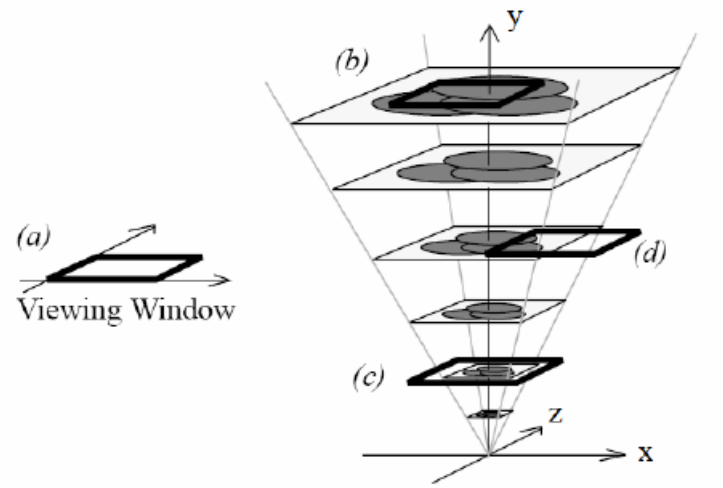


Figura 2.14: Exemplo de *pan* e *zooming* [29].

Ainda que esta técnica seja muito comum, e por isso a maioria dos utilizadores estejam familiarizados com ela, possui algumas limitações quado utilizado em dispositivos móveis [42]. As principais limitações prendem-se com o facto de causarem desorientação pelo movimento visual excessivo da informação ao fazer *zoom* ou *panning*. Uma pequena operação, pode causar demasiada variação na forma como o utilizador vê a informação, acabando por confundi-lo [30].

Focus + Context

Os métodos anteriormente discutidos para gerir vistas detalhadas e contextuais de informação possuem vistas separadas por espaço (*overview + detail*) ou tempo (*pan and zoom*). A abordagem *focus + context*, por outro lado, integra a vista de contexto e de detalhe numa só, onde todas as partes são visíveis em simultâneo [29]. Técnicas baseadas nesta abordagem normalmente mostram uma ou mais áreas de detalhe com conteúdo não distorcido, rodeadas de áreas de contexto que estão distorcidas para caberem no espaço de ecrã disponível (Figura 2.15) [23]. Torna-se assim possível, colocar grandes quantidades de informação num ecrã pequeno sem necessidade de interações de *zooming*.

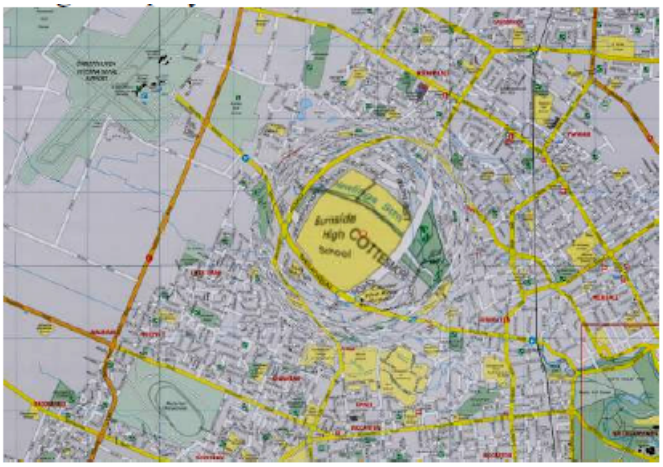


Figura 2.15: Exemplo de *focus + context*: a região central é ampliada e envolvida no contexto envolvente [29].

O facto de ambas as vistas estarem enquadradas no mesmo espaço e ao mesmo tempo (não há separação espacial ou temporal), contribui para a fácil comparação entre ambas. A principal desvantagem desta técnica é que relembrar as localizações dos objetos é difícil, devido à mudança constante da área de detalhe [48].

2.2.2 Técnicas de Visualização *Off-Screen* em Mapas

Nesta secção são apresentadas algumas técnicas para a visualização de objetos localizados fora da área visível no ecrã em mapas 2D.

Contextual Cues

As técnicas *off-screen* usadas em mapas consistem em adicionar elementos gráficos, destinados a ajudar o utilizador a localizar regiões de interesse fora da área visível. Tipicamente, isto é conseguido mostrando formas abstratas nas bordas do ecrã, como referências visuais para pontos *off-screen* [23]. Estes elementos podem ser setas, arcos, ou qualquer outra forma que seja capaz de dar pistas para objetos fora da área visível no ecrã (Figura 2.16).

Não há desvantagens específicas desta técnica, uma vez que essas dependem do elemento gráfico utilizado para a representação *off-screen* e não da abordagem utilizada. Apresentam-se em seguida formas diferentes de apresentar pistas para objetos *off-screen*.

Setas

Uma forma muito comum de indicar a existência de um objeto fora da área visível no ecrã, é através de setas [32]. As setas são colocadas junto às margens do ecrã, e apontam na direção do objeto a ser sinalizado (Figura 2.17).

Figura 2.16: Exemplo de *contextual cues*: uso de setas para indicar a existência de objetos *off-screen* [22].

Figura 2.17: Representação de pontos *off-screen* com a técnica de Setas [42].

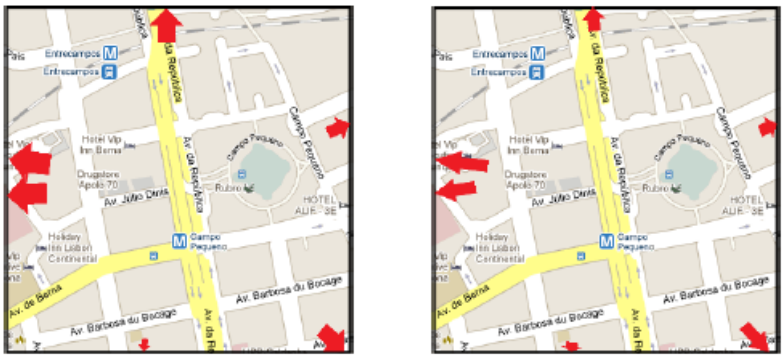
Apesar de indicar a direção do objeto, a seta por si só não fornece qualquer informação sobre a distância ou relevância do mesmo. Todavia, a nível da representação da distância já foram propostas algumas abordagens. Burigat *et al.* [22] propós a utilização de um código de cores e/ou utilização de uma informação textual de forma a permitir ao utilizador inferir a distância. Por outro lado, foi também proposto a utilização da transparência para representar a distância, sendo o objeto mais distante aquele cuja seta possui maior transparência.

A técnica das setas é muito utilizada no domínio dos jogos para alertar os jogadores sobre a presença de determinados objetos fora da área visível no ecrã [42].

Scaled and Stretched Arrows

A técnica de Scaled and Stretched Arrows é uma variação proposta por Burigat *et al.* à técnica das Setas [22]. Recorrendo a estas técnicas é possível fornecer informação sobre a distância e direção de um objeto *off-screen* sem recorrer a qualquer outro atributo que não a morfologia da própria seta. A *Scaled Arrows* faz variar o tamanho da seta, enquanto que a *Stretched Arrows* varia o seu comprimento. Desta forma, sendo o tamanho

e comprimento inversamente proporcionais à distância a que o objeto representado se encontra, quanto maior ou mais comprida for a seta, mais perto o objeto *off-screen* está do utilizador (Figura 2.18).



(a) (b)

Figura 2.18: Técnicas (a) *Scaled Arrows* e (b) *Stretched Arrows* para representação de objetos *off-screen* [42].

Halo

A técnica Halo proposta por Baudisch [14], consiste no desenho de circunferências em torno dos objetos *off-screen* com um raio suficientemente grande para que o arco seja visível no ecrã, mas não exageradamente grande para não se tornar demasiado intrusivo para o utilizador (Figura 2.19). Os arcos são desenhados junto às margens do ecrã, e a partir deles a localização do centro da circunferência é facilmente induzida, permitindo saber a localização do objeto que está a ser sinalizado. Por outro lado, a posição do arco fornece informação sobre a direção do objeto. Quanto mais distante está um objeto *off-screen*, maior será o comprimento do arco visível no ecrã. O Halo associa ainda a transparência à distância do objeto, e por isso, quanto maior for a transparência, maior é a distância ao objeto representado.

Apesar desta técnica conseguir representar a distância e direção, possui alguns problemas, nomeadamente a sobreposição de arcos e o corte parcial dos mesmos quando situados nos cantos do ecrã. Esta técnica quando utilizada com um elevado conjunto de POI, pode tornar-se complicada e confusa, uma vez que a distinção entre arcos torna-se pouco percetível.

City Lights

A técnica *City Lights* fornece informação sobre a direção de objeto *off-screen*, sem a necessidade de apontar para o mesmo [78]. Para esse efeito, esta técnica desenha pequenos rectângulos coloridos nas margens do ecrã. Caso o objeto não visível se situe num canto, é desenhada a ponta de uma seta através de dois rectângulos unidos.

Figura 2.19: Representação de pontos *off-screen* com a técnica Halo [42].

Figura 2.20: Técnica City Lights para a representação de objetos *off-screen* [42].

Apesar de simples, esta técnica possui algumas restrições. Por um lado, não fornece qualquer informação sobre a distância, e além disso, quando comparada à técnica das setas, revelou ser menos precisa relativamente à representação da direção [22].

EdgeRadar

A técnica EdgeRadar foi proposta para a representação da localização de objetos em movimento [43]. Inspira-se na abordagem *overview* + *detail*, sendo as margens do ecrã utilizadas para representar toda a área não visível, e a parte central reservada para objetos visíveis. No espaço das margens, os objetos *off-screen* são representados com pequenos símbolos. A localização dos símbolos nesse espaço é usada para o utilizador inferir a direção e distância até ao objeto fora da área visível (Figura 2.21).

Ainda que esta técnica tenha demonstrado ser vantajosa na medida em que consegue reduzir o número de sobreposições, por ser uma técnica pouco intrusiva, apresenta algumas limitações. Por um lado, não é possível obter informação explícita relativamente à localização do objeto sinalizado. Por outro lado, o facto dos cantos serem zonas de intersecção entre margens, faz com que estes representem uma fatia maior da área não visível, o que pode tornar a avaliação da direção e distância confusa, especialmente

Figura 2.21: Visualização de objetos *off-screen* com auxílio da técnica de EdgeRadar [43].

quando se comparam objetos sinalizados nos cantos com objetos sinalizados nas margens.

Sparkle

Desenvolvido por Boll *et al.* [54], o Sparkle foca-se no uso de uma superfície de LEDs que emite luz, que permite indicar a localização de pontos *off-screen* sem a necessidade de colocar indicações no ecrã. Ou seja, até agora todas as técnicas de sinalização de POI fora da área visível colocavam pistas no próprio ecrã para os encontrar. Pelo contrário, o Sparkle mantém o ecrã totalmente disponível para a visualização dos pontos *on-screen* e usa uma superfície luminosa exterior para sinalização *off-screen*.

Para conseguir este efeito, o dispositivo é colocado sobre uma superfície de LEDs emissores de luz de diferentes cores, que assumem diferentes intensidades e formas que codificam parâmeteros como a distância, o que permite localizar facilmente os pontos *off-screen* (Figura 2.22). Também a relevância do POI foi algo a ser codificado através da luz emitida.

Esta técnica provou ser robusta, diminuindo a carga cognitiva e aumentando a perceção sobre o que rodeia o utilizador de forma intuitiva, tendo obtido bons resultados. Porém, é uma técnica que exige a existência de um suporte de LEDs em conjunto com o dispositivo, sendo apenas viável em situações muito concretas e não no dia a dia do utilizador comum.

Deste modo, conclui-se que o Sparkle é uma técnica que poderá ser usada quando se pretende disponibilizar totalmente o ecrã do *tablet* para pontos *on-screen*. Esta técnica demonstrou diminuir a carga mental necessária para perceber o que está à volta do utilizador de forma eficaz e ser capaz de fornecer boa usabilidade.

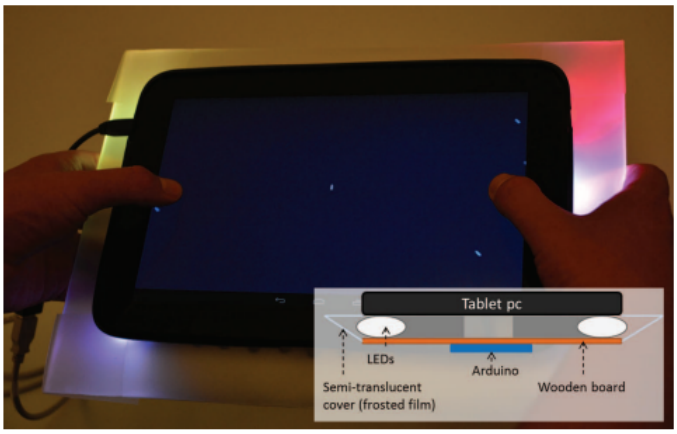


Figura 2.22: Sinalização de pontos *off-screen* com recurso à iluminação LED [54].

2.2.3 Técnicas de Visualização *Off-Screen* em RA

Esta secção aborda técnicas de visualização *off-screen* utilizadas em ambientes de Realidade Aumentada.

Mini-Mapa

A técnica mini-mapa segue uma abordagem *overview* + *detail*, fornecndo uma vista detalhada com a imagem real do que é observado pelo utilizador no momento e uma vista global 2D representativa do mundo visto de cima (*top-down*), estando o utilizador no centro [68]. Existem ainda 2 linhas que mostram claramente o FOV (*field-of-view*, campo de visão) do utilizador, o que torna possível perceber claramente onde se encontra, o que é que está a ver, e que objetos fora do FOV o rodeiam [68]. As vistas podem ser mostradas em paralelo ou sequencialmente [79], contudo, segundo o contexto de visualização 3D proposto por Schinke *et al.* [68], o utilizador vê a imagem do mundo real no ecrã e sobreposta a esta, uma pequena janela com a vista *top-down* (Figura 2.23).

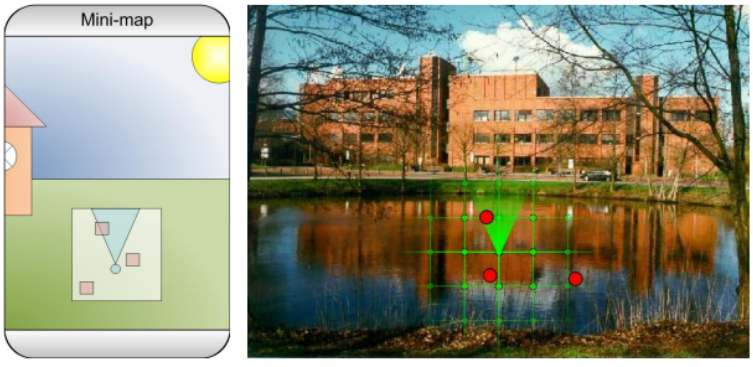


Figura 2.23: Sinalização de pontos *off-screen* com recurso ao Mini-Mapa [68].

Todavia, esta técnica exige ao utilizador uma constante conversão da vista 2D apresentada para a vista real 3D para que a representação do mapa corresponda ao mundo físico.

Apesar deste ser um processo mental automático dos utilizadores, pode causar alguma demora e confusão quando usado com um largo conjunto de POI. Ademais, a limitação desta técnica para representar alturas torna desconfortável a sua utilização, pois não sendo possível saber a que altura está um objeto, o utilizador é obrigado a rodar o dispositivo verticalmente até encontrar o ponto pretendido [12].

Ainda assim, o mini-mapa continua a ser uma técnica largamente utilizada, principalmente em cenários virtuais, pela sua simplicidade e eficácia em ambientes com um número de objetos limitado.

Setas 2D

Um estudo feito por Henze *et al.* [44] sugere a possibilidade de representar a existência de objetos fora da área visível no ecrã no contexto da realidade aumentada através de setas. Quando o utilizador aponta o seu dispositivo para o mundo real, os POI que estão fora do ecrã são assinalados com setas que indicam a direção de pontos *off-screen* (Figura 2.24).

Tal como a técnica anteriormente analisada, esta não é capaz de representar alturas.

Figura 2.24: Uso de setas 2D para a visualização de objetos *off-screen* [44].

Setas 3D

De forma a resolver os problemas associados ao mini-mapa, as setas 3D têm sido frequentemente utilizadas como meio de sinalização *off-screen* [28]. Esta técnica consiste na utilização de setas desenhadas tridimensionalmente, que apontam para objetos *off-screen* (Figura 2.25) [19].

Apesar de solucionar o problema da representação de alturas e não ser necessária qualquer conversão mental, visto que tanto o mundo físico como as setas estão representados tridimensionalmente, quando utilizada com um largo conjunto de POI esta técnica apresenta níveis de oclusão elevados. Assim, torna-se muito difícil seguir as indicações fornecidas, pois a sobreposição entre setas torna-as ilegíveis [28]. Algumas alternativas foram propostas, como por exemplo a diminuição do tamanho das setas de forma a reduzir

Figura 2.25: Sinalização de pontos *off-screen* com Setas 3D [68].

a probabilidade de interseção com as restantes. Porém, neste caso é perdida a informação que seja dada através da morfologia da seta, por exemplo no caso da distância do utilizador ao ponto ser representada pelo comprimento da seta. Outro problema é o facto da vista fornecida pela técnica estar em conflito com o eixo de referência humano, o que pode tornar difícil seguir o movimento proposto pelas setas e até confundir o utilizador relativamente à direção indicada pelas mesmas [24].

Esta técnica mostrou ser tão efetiva como o mini-mapa em ajudar os utilizadores enquanto se deslocaram a andar, e mostrou melhores resultados quando estes se deslocaram de avião [47].

AroundPlot

A técnica AroundPlot, proposta por Jo *et al.* [48], adota uma abordagem *focus + context*, e por isso combina a vista detalhada e a vista geral numa só, o que contribui para a fácil comparação entre as vistas em tarefas de movimento e mudança de direção. A vista geral é composta pelo mundo real enquanto que a vista detalhada é constituída por barras colocadas junto às bordas do ecrã onde aparecem símbolos representativos de objetos *off-screen*.

Uma das principais vantagens é o facto da barra onde aparece um ponto *off-screen* indicar o sentido da rotação que deve ser efetuado para que este se torne vísível (*on-screen*), induzindo um movimento intuitivo, claro e completamente alinhado com o eixo de referência humano ao contrário das setas 3D. Por outro lado, em comparação ao radar, permite representar alturas através das barras existentes em cima e em baixo, no ecrã. Por fim, como estas barras são demasiado estreitas para representarem uma parte tão grande do mundo real (tudo o que não é visível), surge uma funcionalidade de ampliação dinâmica das mesmas consoante o movimento do utilizador. Se o utilizador quer que um ponto colocado na barra direita se torne visível, este terá que se rodar para a direita, e nesse momento, esta será ampliada (ampliação dinâmica) de forma a que os pontos se possam dispersar mais livremente, e simultaneamente, proporcionando uma ideia mais

precisa de quando este irá tornar-se visível, o que torna esta técnica escalável.

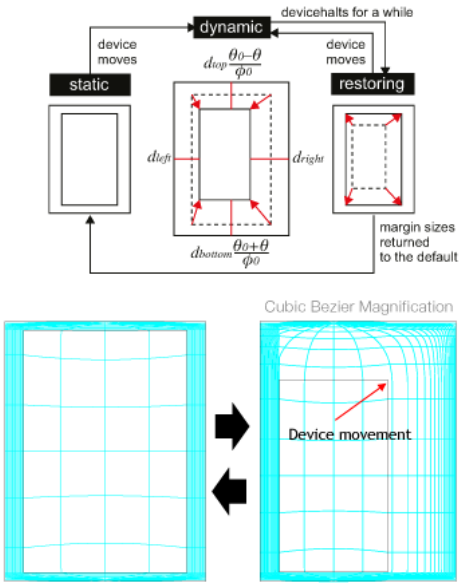


Figura 2.26: Ampliação dinâmica de acordo com o movimento do dispositivo [48].

Contudo, quando usada com muitos POI, esta técnica torna as barras demasiado preenchidas o que torna a leitura da informação sobre os objetos *off-screen* confusa, e ainda que a ampliação dinâmica dissolva grande parte desse problema, os cantos continuam a ser uma zona com elevada densidade de POI, porque pertencem simultaneamente a uma área horizontal e a uma área vertical. Por fim, o AroundPlot pode exigir alguma concentração no momento de passagem de um ponto *off-screen* para *on-screen*, pois quando o ponto está a passar a linha de transição, o ponto real entra em simultâneo dentro do ecrã, o que pode acontecer em zonas distantes do ecrã no caso da barra estar ampliada.

Esta técnica foi comparada à Setas 3D e ao Mini-Mapa, e num teste com um largo conjunto de POI foi claramente mais eficaz. No entanto, num teste com um pequeno conjunto de POI, as diferenças não foram significativas.

SidebARs

O SidebARs foi proposto por Siu *et al.* [71] com o intuito de fornecer suporte e apoio aos bombeiros que possuem fracas condições de comunicação, tentando aumentar a sua perceção relativamente aos recursos que os rodeiam. Os autores desta técnica fazem referência a todas as anteriormente mencionadas, afirmando que nenhuma delas faz agrupamento de POI por tipo. O SidebARs surge na tentativa de colmatar essa falha.

O protótipo do SidebARs usa duas barras laterais, cada uma delas contendo vários símbolos representativos dos objetos *off-screen*, semelhante à técnica anteriormente analisada. Porém, existe uma outra funcionalidade, o agrupamento de POI consoante o tipo (camiões, estações de polícia, hidrantes, etc.). Cada agrupamento de POI é representado

por um símbolo, que mostra um número indicativo do número de objetos existentes daquele tipo, e a distância até chegar ao mais próximo. Por fim, este protótipo dá ainda a possibilidade de configurar quais os tipos de POI que devem estar visíveis na aplicação, pois o que é relevante para um bombeiro pode não ser para outro, dependendo da sua função. Desta forma, o SidebARs além de fazer a sinalização *off-screen* permite aos bombeiros filtrar e encontrar os objetos relevantes com maior rapidez e eficiência (Figura 2.27).

Figura 2.27: *Mockup* da interface do SidebARs [71].

Contudo, os autores não fazem qualquer referência à representação de alturas e as indicações *off-screen* não fornecem nenhuma informação sobre quando é que o objeto vai entrar na área visível, tornando impossível para o utilizador antecipar esse momento e abrandar a rotação.

De forma geral, esta técnica permite o acesso rápido e eficaz dos bombeiros a um determinado tipo de recurso, melhorando o processo de emergência.

2.3 Representação de Relevância

Apesar da evolução dos dispositivos móveis, os ecrãs continuam a impôr restrições severas ao nível da visualização e interação, particularmente aquando da exploração de grandes quantidades de dados geográficos [41]. Uma vez que o protótipo a ser construído ao longo deste projeto deverá permitir a visualização de POI através da colocação de símbolos num ambiente de RA, e tendo em conta que cada símbolo pode representar um tipo de ponto com importâncias diferentes pode-se tornar difícil para o utilizador distinguir quais os POI relevantes.

Filtrar a informação do ecrã de acordo com a sua importância pode tornar-se demasiado complexo se os objetos mais relevantes não "saltarem à vista" comparativamente aos restantes, principalmente devido à capacidade de atenção limitada do ser humano [72]. É importante, no meio de tanta informação navegável, conseguir salientar a informação mais importante ao utilizador para que este possa processar a informação de forma mais eficiente, e consequentemente, navegar de forma mais fácil [64].

No caso da Figura 2.28, a aplicação de RA mostra vários POI que representam restaurantes. Mais especificamente, no momento em que esta imagem foi retirada o utilizador estava a visualizar 7 POI em simultâneo. Apesar de não serem muitos POI, é provável que o utilizador demore algum tempo até perceber se existe algum que realmente lhe interesse entre os 7 visualizados. Isto porque não existe qualquer mecanismo que faça sobressair a informação relevante para o utilizador de acordo com as suas preferências, o que o obriga a olhar para cada um dos símbolos individualmente para obter informações. Imagine-se que o utilizador quer encontrar, preferencialmente, restaurantes de Sushi. Nesse caso, o restaurante *Blowfish Sushi To Die For* deveria estar de alguma forma salientado, para que a correspondência entre os restaurantes e as preferências do utilizador fosse intuitiva e quase instantânea.

No caso de não existir nenhum restaurante de Sushi, é importante continuar a mostrar ao utilizador quais os POI que continuam a ser provavelmente os mais importantes. Por exemplo, a distância ou avaliação do restaurante são possíveis critérios de decisão para o utilizador, pois os POI que estão mais perto de si ou mais bem avaliados serão possivelmente os restaurantes com maior potencial a serem visitados. Dessa forma, o utilizador além de percecionar que não existia nenhum restaurante de Sushi, saberia quais as alternativas com maior relevância. Na verdade, na Figura 2.28 a distância é mostrada textualmente, mas isso exige que o utilizador leia o texto para perceber a que distância está um ponto e a memorizar essa informação para a comparar com as distâncias dos restantes. É por isso que é importante a existência de um mecanismo visual (e não textual), que torne esse processo automático e espontâneo no utilizador.

Figura 2.28: Aplicação de RA que mostra Restaurantes como POI.

Vejamos agora o exemplo ilustrado na Figura 2.29, que identifica 3 tipos diferentes de POI: compras, restaurantes e outros edifícios (ex. teatros, hotéis) e a cada tipo de POI está associado um símbolo diferente. De facto, a adaptação da simbologia ao tipo do POI é uma ajuda preciosa para que o utilizador possa filtrar visualmente a informação. Contudo, todos os símbolos ”saltam”igualmente à vista do utilizador, o que continua a causar alguma confusão no ecrã. O ideal, seria ter um mecanismo visual que permitisse ao utilizador focar-se de forma espontânea e intuitiva nos símbolos mais relevantes, sem ser preciso olhar para todos e analisá-los. Se soubermos que o utilizador procura restaurantes, então o símbolo que lhes está associado deveria estar de alguma forma destacado visualmente relativamente aos outros, e não apenas com uma imagem diferente.

Idealmente, a nível semântico espera-se que os símbolos sejam capazes de identificar o tópico a que pertence o objeto, e além disso, que expressem o grau de interesse de cada POI para o utilizador.

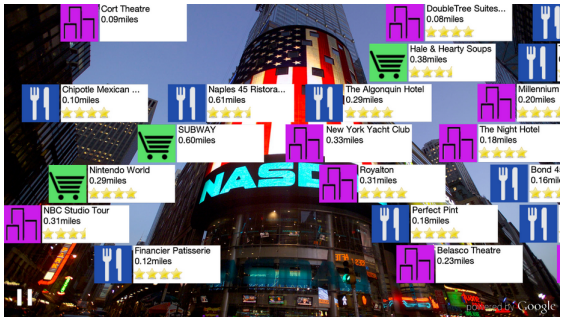


Figura 2.29: Aplicação de RA que mostra diferentes tipos de POI.

Na literatura existem diversas técnicas para solucionar o problema da relevância. De seguida, aborda-se a que foi seguida por este trabalho.

O HaloDot, proposto por Tiago *et al.* [40], é uma técnica que provém da técnica Halo original, já analisada neste documento. Distingue-se da técnica original por conseguir representar a relevância.

Esta técnica usa a cor como atributo para a representação da relevância de um POI. As duas principais razões para o uso da cor são: frequente utilização da cor para a

representação de diferentes significados, capacidade da mesma para despertar a atenção dos utilizadores. Relativamente à primeira razão, e de acordo com Silva *et al.* [70] a cor está frequentemente associada a diferentes significados como, por exemplo, a temperatura, em que o vermelho representa o quente e o azul o frio. A segunda razão provém de estudos efetuados por Wolf *et al.* [76], que após um estudo feito com vários utilizadores, concluiu que a cor é um dos atributos que mais desperta a atenção dos mesmos.

Pelas razões apontadas, o HaloDot utiliza a cor para representar a relevância dos objetos fora da área visível. Isso é feito utilizando uma analogia *warm-cold*, em que os POI mais relevantes são representados por cores quentes, e os menos relevantes por cores frias. Como os seres humanos têm uma perceção reduzida relativamente à perceção métrica da variação de cores, simplificou-se esta representação para suportar apenas 3 cores [50]. Por isso, POI relevantes são representados através de HaloDots vermelhos (cor quente), e os POI menos relevantes com HaloDots azuis (cor fria), e os POI com uma relevância intermédia com HaloDots roxos (cor tépida), uma vez que esta se situa entre o vermelho e o azul na escala RGB (*Red, Green, Blue*). O resultado final desta abordagem é demonstrado na Figura 2.30.

Figura 2.30: Técnica HaloDot para representação da relevância através da cor [42].

No entanto, visualizando apenas a cor pode haver o problema de, eventualmente, comprometer a visibilidade de objetos mais relevantes comparativamente a outros menos relevantes. Isto porque apesar das cores distintas, os objetos menos relevantes ”saltam”igualmente à vista do utilizador relativamente aos mais relevantes.

De forma a solucionar este problema, decidiu adicionar-se o fator transparência a esta representação. Assim, a transparência está associada à relevância, considerando um nível máximo e mínimo de transparência, de acordo com a relevância do objeto. Deste modo, um objeto mais relevante será sempre mais visível que um objeto menos relevante, independentemente da sua distância ou outro qualquer parâmetro a ser considerado, o que torna tudo mais simples e intuitivo para o utilizador. A escala de cor transparência e a técnica HaloDot no seu estado final podem ser visualizadas na Figura 2.31.

Em suma, a técnica aqui analisada permite uma representação eficiente pois, além de permitir a identificação da relevância dos POI através da cor, permite ao utilizador

(a) (b)

Figura 2.31: Técnica HaloDot para representação da relevância através da cor e transparência [42]. (a) Escala de cor-transparência usada no HaloDot (b) HaloDot para representação da relevância através da cor e transparência.

abstrair-se dos resultados menos relevantes através da transparência, o que reduz bastante a carga cognitiva necessária para analisar um mapa com vários POI.

2.4 Tratamento de Sobreposições

Uma das maiores limitações na visualização de informação nos dispositivos móveis é o tamanho do ecrã. Uma vez que o objetivo deste trabalho é a visualização de POI através da colocação de símbolos num ambiente de RA, e tendo em conta que o tamanho destes símbolos não é desprezável quando comparado com o tamanho do ecrã, pode concluir-se que a probabilidade destes símbolos ficarem sobrepostos é muito elevada. Deste modo, é também elevada a probabilidade da imagem gerada se tornar complexa e de difícil leitura para o utilizador [62]. Consequentemente, para que se consiga apresentar a informação de forma clara, é necessário recorrer a técnicas que permitam o tratamento de sobreposições.

Repare-se no exemplo presente na Figura 2.32. Esta aplicação de RA, que localiza bancos e caixas multibanco, só torna possível a visualização da informação associada a 5 multibancos, no entanto, olhando para o Radar existente no canto superior esquerdo do ecrã verifica-se que existem muito mais POI dentro do campo de visão mas que não estão visíveis devido à sobreposição entre símbolos. Desta forma, o utilizador não pode perceber a informação disponibilizada, o que torna difícil tomar uma decisão [67].

Atualmente, existem várias aplicações que dão a possibilidade de aplicar filtros, o que leva à redução de símbolos na RA. Todavia, esta aproximação não resolve o problema das sobreposições mas apenas reduz a probabilidade de tal acontecer. De facto, se a distribuição dos símbolos no ecrã não for uniforme, continua a ser possível a existência de símbolos sobrepostos e, consequentemente, a imagem pode ser de difícil compreensão para o utilizador. Este problema tem uma importância acrescida se se constatar que,

Figura 2.32: Aplicação de RA na qual existe um elevado número de sobreposições.

efetivamente, a distribuição de POI não é normalmente uniforme. Um exemplo demonstrativo deste problema reside na localização de restaurantes. Num centro comercial, por exemplo, é frequente existirem muitos restaurantes, pelo que o espaço no mundo real correspondente a apenas um edifício será preenchido por uma dezena ou mais de símbolos correspondentes a restaurantes, provocando sobreposições [62].

A sobreposição impede a perceção clara e correta da informação prejudicando o processo de navegação do utilizador espaço de informação disponível e respetiva tomada de decisão perante o que observa. Por isso, surgiram algumas abordagens que tentam colmatar esta falha.

De seguida é apresentada uma técnica que tenta solucionar o problema descrito e que foi aplicada a mapas 2D.

No sistema MoViSys que permite a visualização de POI, proposto por Paulo Pombinho [62], é abordado o problema das sobreposições através da subdivisão da área de visualização numa grelha sobreposta à mesma, contabilizando-se, de seguida, o número de POI presentes em cada célula. De seguida, podem acontecer duas situações distintas: os POI que estão na mesma célula têm uma relação semântica entre si (são da mesma categoria), ou os POI que estão na mesma célula não têm qualquer relação entre si.

No caso dos POI terem uma relação semântica, o conjunto de dois ou mais símbolos é substituído por apenas um único que representa um grupo de POI, como mostra a Figura 2.33.

Por outro lado, se os POI não têm qualquer relação, o conjunto de dois ou mais símbolos é substituído por um único que representa informação sobre quais as categorias dos POI que o compõem e a quantidade relativa dos mesmos. Este caso está ilustrado na Figura 2.34.

Por fim, é ainda possível desagregar um grupo de POI. Isto é importante uma vez que o utilizador poderá querer ver os POI na sua distribuição original pelo mapa. No entanto, quando o utilizador executa uma desagregação, poderão existir sobreposições entre símbolos, uma vez que estes vão para a sua posição original. Para atenuar este

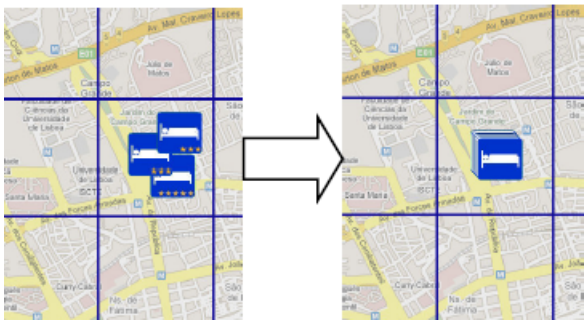


Figura 2.33: Agrupamento de POI da mesma categoria, na mesma célula, num só [62].

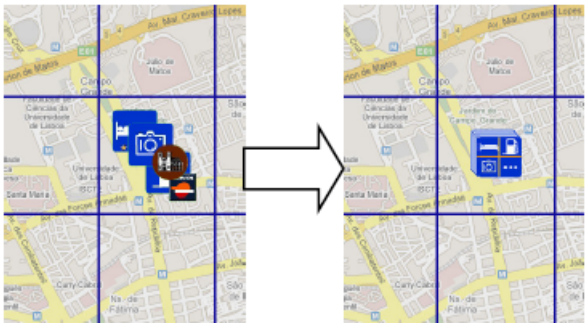


Figura 2.34: Agrupamento de POI de diferentes categorias, na mesma célula, num só [62].

problema, o MoViSys afasta os símbolos para que não se sobreponham e coloca linhas indicativas das suas posições reais. Este mecanismo está ilustrado na Figura 2.35.

Figura 2.35: Desagregação com mecanismo de afastamento para evitar sobreposições dos símbolos [62].

Esta técnica mostrou resolver de forma eficaz as sobreposições existentes, e teve um resultado bastante satisfatório na avaliação executada com utilizadores. Como tal, neste trabalho será utilizada uma abordagem baseada nesta solução.

2.5 Sumário e Discussão

Este capítulo fez uma introdução aos vários conceitos inerentes à RA, apresentou os principais problemas deste tipo de aplicações quando usados em dispositivos móveis e descreveu soluções atualmente propostas para resolver os problemas da visualização *off-screen*, representação de relevância e tratamento de sobreposições.

Como foi possível constatar, as soluções existentes possuem limitações e nenhuma delas é capaz de sinalizar objetos *off-screen*, representar relevâncias e tratar sobreposições simultaneamente em aplicações de RA móveis. Com o objetivo de preencher essa lacuna, na próxima secção é apresentada o protótipo IAR que propõe soluções para resolver os problemas mencionados.

Capítulo 3

Aplicação IAR

Este capítulo apresenta o protótipo da aplicação IAR desenvolvido para concretizar as soluções para os problemas identificados de sinalização *off-screen*, representação de relevância e tratamento de sobreposições. Na secção 3.1 é explicado sucintamente o IAR. Na secção 3.2 é descrito o processo de desenho do IAR, desde as suas fases iniciais até ao resultado final, com justificação das várias decisões tomadas ao longo do projeto. Na secção 3.3 é detalhado o processo de construção do protótipo final, desde a definição do modelo de dados até aos algoritmos concebidos.

3.1 Contexto

O IAR é um sistema RA que permite a visualização de POI em dispositivos móveis.

Neste caso específico, o sistema utiliza dados geográficos fornecidos *a priori* sobre as caixas multibanco existentes em Portugal, e são estes os POI apresentados na aplicação. Pretendendo este sistema ser uma prova de conceito, serão considerados apenas três bancos, Caixa Geral de Depósitos (CGD), Banif (Ban) e Novo Banco (NB), já que são suficientes para demonstrar todas as funcionalidades desenvolvidas.

Desta forma, sempre que se referir o sistema IAR durante restante documento, estar-se-á a falar de um sistema que localiza três tipos de bancos presentes em Portugal (CGD, Ban e NB) num sistema de realidade aumentada móvel.

3.2 Análise e Desenho do IAR

Nesta secção, serão detalhadas as opções tomadas para a conceção das diferentes funcionalidades relativas aos problemas que o IAR se propõe a resolver: representação de relevância, sinalização *off-screen* e tratamento de sobreposições.

3.2.1 Representação da Relevância

Com base na técnica HaloDot analisada anteriormente, e nos estudos que daí surgiram, decidiu-se optar pelo mesmo modelo de representação de relevância. Uma vez que esta técnica mostrou ser muito eficaz, a nível de perceção da relevância de um ponto por parte do utilizador através do uso da cor e transparência, o mesmo comportamento foi adotado ao IAR.

Cada POI é representado por um quadrado. Apesar de ter sido considerada a opção de representar com um círculo, o facto do símbolo vir a conter alguma informação textual tornou esta opção inválida porque restringe o espaço disponível. Além disso, o quadrado é o símbolo maioritariamente utilizado neste tipo de aplicações para representar um POI, e por isso as pessoas estão familiarizadas com o mesmo.

No sistema desenvolvido, a cor está associada à relevância e a transparência à distância de um ponto ao utilizador. Deste modo, será necessário que cada símbolo representativo de um POI (cada quadrado) possua uma cor e grau de transparência de acordo com a sua relevância e distância ao utilizador, respetivamente. Poderia ter-se optado por associar a transparência a outro qualquer parâmetro que não a distância, mas considerou-se esta informação muito importante para o utilizador poder decidir também em termos de distância ao ponto alvo. Tendo em conta que a aplicação de RA deverá ser usada quando o utilizador caminha, a distância será provavelmente um fator que permite decidir entre os POI de igual relevância, qual o que deve seguir. Portanto, o IAR atribui a cada tipo de banco (CGD, Ban e NB) um valor de relevância. Mais especificamente, assumiu-se que os CGD são os POI mais relevantes (representados a vermelho), os Novo Banco os de relevância intermédia (representados a mangenta) e por fim, os Banif os de menor relevância (representados a azul). Quanto à transparência, é preciso notar que, como ilustrado na Tabela 3.1, um ponto muito relevante é sempre mais opaco que um outro pouco relevante, mesmo estando o último muito mais perto do utilizador. Isto acontece porque o objetivo é conseguir fazer o utilizador focar-se na informação mais relevante. Por isso, mesmo que um ponto menos relevante esteja muito mais próximo que um mais relevante, o mais relevante estará mais saliente na aplicação uma vez que apesar deste estar mais distante será aquele a que, provavelmente, o utilizador quer chegar. Por isso, o utilizador só deverá usar a transparência para induzir distâncias entre POI da mesma relevância. Usou-se a mesma tabela como referência para implementar o mecanismo de transparência, variando esta num intervalo de [0, 255], correspondendo o zero à maior transparência.

O resultado final da combinação de cores e transparências pode ser visto na Figura 3.1 que mostra as três cores usadas para representar relevâncias e respetiva transparência consoante a distância ao utilizador. Se o utilizador pretende comparar a distância entre objetos de diferentes relevâncias, então deverá focar-se no tamanho do símbolo, que será tanto maior quanto mais perto este estiver do utilizador. No exemplo da Figura 3.1, é percetível que as caixas multibanco da CGD e NB estão a distâncias muito semelhantes

Tabela 3.1: Distribuição da cor e transparência de acordo com a relevância dos POI [42].

Relevância	Cor	Transparência
Elevada	Vermelho	[190, 255]
Intermédia	Roxo	[64, 189]
Baixa	Azul	[0, 63]

e que a do Banif está mais longe. Além disso, a transparência deste último é maior em comparação às dos restantes por ser o menos relevante. No caso de estudo considerado, a relevância foi atribuída *a priori* consoante o banco. Contudo, poderia ter-se considerado a atribuição da relevância através de uma função de grau de interesse com base em vários critérios, como sugerido por Pombinho *et al.* [63].

É ainda de referir que apesar de não se realizar qualquer adaptação da simbologia de acordo com o ambiente exterior, todos os símbolos possuem um rebordo preto, por ser uma das formas preferidas dos utilizadores para distinguirem os símbolos do fundo, de acordo com um estudo realizado por Carmo *el al.* [26].

Figura 3.1: Exemplo do uso de cores e transparências no protótipo.

3.2.2 Sinalização *Off-Screen*

A solução proposta para a sinalização de objetos *off-screen* segue uma abordagem *focus + context* com uma moldura de sinalização, semelhante ao AroundPlot. A técnica desenvolvida usa cores e transparências para revelar informações sobre os pontos *on-screen* e *off-screen*. É ainda fornecido um Mapa 2D como modo alternativo de visualização e complementar à RA, para que o utilizador possa intercalar entre ambos de forma a encontrar o que pretende de forma mais rápida. Em seguida, será descrito de forma detalhada cada um dos modos de visualização.

Realidade Aumentada

Na camada de RA optou-se por usar uma abordagem *focus + context* com duas barras horizontais e duas barras verticais que formam uma moldura à volta da área visível, inspirada na proposta do AroundPlot. Optou-se por esta abordagem por ser simples, intuitiva,

usar um eixo de rotação alinhado com o do utilizador e ser capaz de permitir a sinalização de objetos não visíveis acima e abaixo da área de visualização. Tal como no AroundPlot, as barras são ligeiramente translúcidas para que não pareçam tão intrusivas, e foram desenhadas de forma a não serem demasiado finas para que os símbolos *off-screen* aí colocados sejam bem visíveis, e não demasiado grossas de forma a ocuparem uma parte muito grande do ecrã, deixando pouco espaço para a visualização do mundo real. O resultado final das barras é ilustrado na Figura 3.2

Figura 3.2: Barras de sinalização.

Para que a sinalização de um ponto fora da área visível no ecrã seja bem sucedida, é necessário conseguir transmitir ao utilizador três informações chave sobre o mesmo: direção, altura e distância. Estas três informações permitem ao utilizador inferir a posição do ponto, e orientar-se de maneira a conseguir encontrá-lo. O desafio é o pouco espaço disponível para a representação dos pontos *off-screen* (espaço da barra de sinalização), que exige uma representação extremamente simples, mas ao mesmo tempo, capaz de fornecer a informação referida.

Antes de explicar as várias representações desenhadas para a sinalização *off-screen*, é preciso relembrar os conceitos de *yaw*, *pitch* e *roll*. O *yaw* é o ângulo de rotação em torno do eixo do *yy*, o *pitch* traduz-se no ângulo de rotação em torno do eixo do *xx*, e por fim, o *roll* é o ângulo de rotação em torno do eixo do *zz* (Figura 2.8). O IAR considera que a origem do eixo referencial *(0, 0 ,0)* é o centro do dispositivo móvel e não do humano, e não considera o *roll* por ser um movimento de rotação pouco natural e expectável de ser executado durante a utilização de uma aplicação de realidade aumentada.

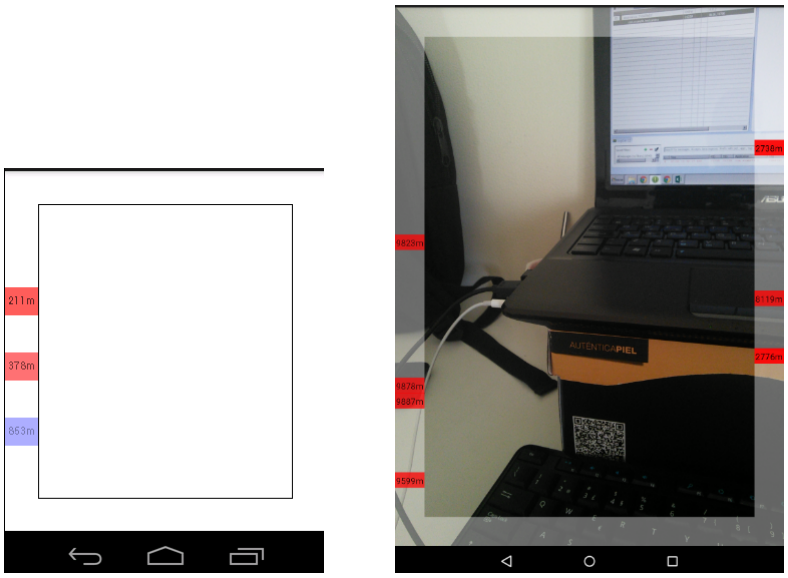
Para a representação de um ponto *off-screen* foi escolhido novamente um quadrado de forma a manter a consistência, e por ser uma figura simples e facilmente adaptável a uma

barra de sinalização. Este símbolo possui uma cor e transparência de acordo com a sua relevância e distância ao utilizador.

A nível de representação da sinalização de um ponto não visível, foi utilizada uma metodologia semelhante à do AroundPlot. Se um objeto está dentro do campo de visão da câmara a nível vertical, isto é, não está a uma altura demasiado elevada ou reduzida que o utilizador não o possa ver, mas está fora do campo de visão da câmara a nível horizontal (ex: atrás das costas), então isso significa que o ponto aparecerá ou na barra direita ou na barra esquerda. Visto que esse mesmo ponto pode tornar-se visível rodando para a esquerda ou para a direita, o símbolo será colocado na barra que indique o movimento de rotação de mais rápido acesso ao ponto. O mesmo raciocínio é aplicado a um objeto que esteja dentro dos limites horizontais, mas fora dos limites verticais. Nesse caso, o objeto *off-screen* está a uma altura demasiado alta ou demasiado baixa e não é capturado pelo campo de visão da câmara, pelo que deverá aparecer na barra superior ou inferior respetivamente. A diferença, é que não foi considerada a possibilidade do utilizador rodar para além dos 180º (câmara traseira apontada para o céu) ou abaixo dos 0º (câmara traseira apontada para o chão), pois esse é um movimento impossível para a anatomia humana.

Com o objetivo de auxiliar o utilizador durante a navegação, é importante que a aplicação disponibilize a informação de forma dinâmica e interativa ao utilizador. Por isso, inicialmente optou-se por fazer o símbolo subir na barra de sinalização (vertical) à medida que a amplitude da rotação (horizontal) necessária para que este se torne visível diminui. Assim, o utilizador saberia que se um ponto está na parte inferior da barra, significa que a amplitude de rotação horizontal necessária para que se torne visível é muito elevada, se está na parte superior da barra, indica que está prestes a entrar dentro do ecrã se este continuar a rodar no sentido indicado. Ademais, para precisar a distância de um ponto ao utilizador (transparência apenas permite induzir um valor) decidiu colocar-se essa informação textualmente no símbolo *off-screen*. Esta ideia foi desenhada e implementada, tal como demonstrado na Figura 3.3.

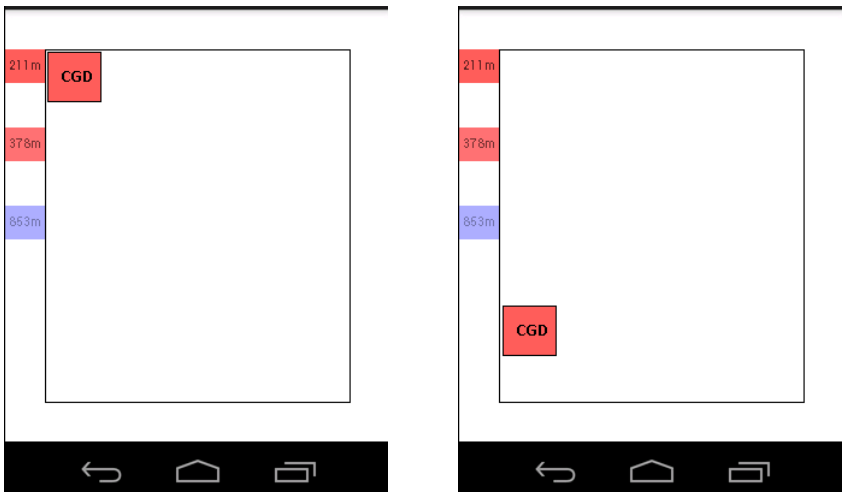
Contudo, esta abordagem apresentou alguns problemas. Após alguns testes internos e com um grupo reduzido de pessoas, concluiu-se que este comportamento era confuso e pouco intuitivo. Segundo os utilizadores, seria expectável que quando o ponto transitasse de *off-screen* para *on-screen*, este mantivesse a mesma altura no ecrã. Todavia, se a altura real do ponto no terreno em relação ao utilizador for baixa existirá um grande desfasamento, ou seja, o ponto será colocado na parte inferior do ecrã o que torna difícil seguir o mesmo no momento em que este se torna visível. A Figura 3.4 ilustra essa situação. O ponto vermelho (CGD, mais relevante) que está a uma distância do utilizador de 211 metros, é o ponto que necessita de uma amplitude de rotação menor (no sentido contrário aos ponteiros do relógio, por estar na barra da esquerda) para entrar na área visível, e por isso, é o que está colocado mais acima na barra de sinalização. O utilizador espera que no momento de passagem de *off-screen* para *on-screen* aconteça o que está representado



(a) (b)

Figura 3.3: Protótipo em que a amplitude de rotação horizontal era dada pela altura do símbolo nas barras direita e esquerda. (a) Protótipo virtualizado no emulador Android (b) Protótipo visualizado num dispositivo móvel real.

em a), mas se o ponto estiver a uma altura baixa relativamente ao utilizador, acontecerá o que está representado em b).

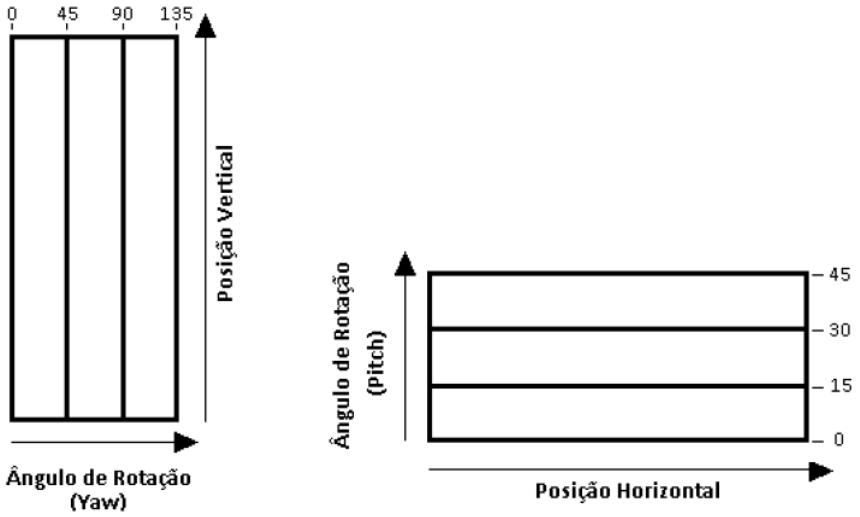


(a) (b)

Figura 3.4: Exemplo do problema da abordagem A (a) Comportamento esperado pelo utilizador (b) Possível comportamento real.

Assim, a utilização da altura na barra de sinalização para representar a aproximação à área visível e não a altura a que o ponto se encontra na realidade conduz a inconsistências na representação, o que provocava confusão aos utilizadores. Decidiu-se por isso modi-

ficar a abordagem para outra que mantivesse o símbolo nas barras à altura real do ponto no terreno relativamente ao utilizador, de forma a manter a coerência no momento de transição. Foi necessário conceber outra forma de indicar a amplitude da rotação necessária para que um ponto se tornasse visível. Dividiu-se verticalmente as barras da esquerda e direita e horizontalmente as de cima e baixo em três partes iguais. A ideia é não usar um rectângulo que ocupe a largura da barra, mas apenas uma das três partes de maneira a indicar a amplitude de rotação necessária.



(a) (b)

Figura 3.5: Desenho das barras de sinalização de modo a permitir a representação de diferentes amplitudes de rotação. (a) Barra vertical direita (b) Barra horizontal inferior.

Começando pela barra que é colocada à direita da área visível representada na Figura 3.5 a), a coluna onde o ponto está posicionado dá a informação ao utilizador do quão próximo este está de se tornar um ponto *on-screen*. Mais especificamente, a coluna permite induzir a amplitude da rotação a ser executada para que um ponto entre para a área visível no ecrã enquanto que a posição vertical indica a altura real do ponto. Assim, como apresentado na Figura 3.5 a), se o ponto está na primeira coluna, ou seja, a que fica adjacente à área visível, significa que necessita de uma rotação em *yaw* no intervalo]0,45] no sentido dos ponteiros do relógio em relação ao eixo vertical do referencial do dispositivo para chegar à zona visível, e assim sucessivamente para as restantes colunas. A Figura 3.6 apresenta os ângulos no sentido correspondente à barra vertical direita. Para a barra vertical esquerda o raciocínio é análogo. Na barra que é colocada abaixo da área visível representada na Figura 3.5 b), aplica-se a mesma estratégia mas de forma inversa, isto é, o posicionamento nas linhas induz a amplitude da rotação em *pitch* necessária para que o ponto se torne visível (Figura 3.7). Os cantos, são uma zona de interseção das barras verticais com as horizontais e por isso ambos os raciocínios se aplicam. Ou seja, quando um ponto está fora do campo de visão verticalmente (ex: demasiado alto) e horizontalmente

(ex: demasiado à direita), este aparecerá representado no canto superior direito, que estará dividido tanto verticalmente como horizontalmente, como ilustrado na Figura 3.8. Note-se que cada divisão vertical (Figura 3.6) corresponde a 15º, enquanto que cada divisão horizontal corresponde a 45º (Figura 3.7). Isto deve-se ao facto de verticalmente, o IAR esperar rotações do utilizador no intervalo [0, 180]º, e horizontalmente, no intervalo [0, 360]º.

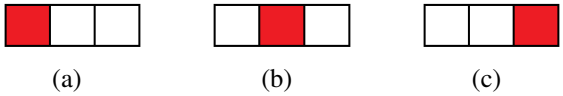


Figura 3.6: Barra vertical direita da moldura de sinalização que mostra um POI que requer uma rotação horizontal de amplitude, respetivamente, (a)]0º,45º] (b)]45º,90º] (c)]90º,135º].

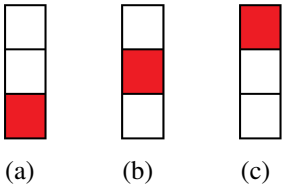


Figura 3.7: Barra horizontal inferior da moldura de sinalização que mostra um POI que requer uma rotação vertical de amplitude, respetivamente, (a)]0º,15º] (b)]15º,30º] (c)]30º,45º].

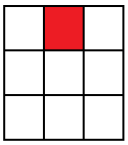
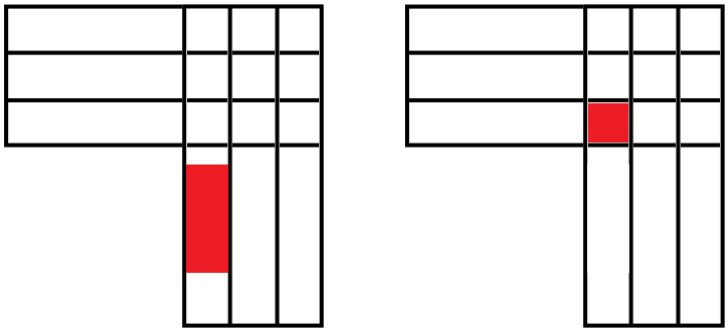


Figura 3.8: Canto superior direito da moldura de sinalização que mostra um POI que requer uma rotação vertical de amplitude, respetivamente,]30º,45º] e horizontal de amplitude]45º,90º].

Por fim, como as barras são divididas em três partes tornaram-se demasiado pequenas para conseguir fornecer a distância ao utilizador textualmente, e por isso esta informação foi removida. Pensou-se em atribuir diferentes tamanhos aos símbolos *off-screen* consoante a distância ao utilizador, tal como acontece com os símbolos *on-screen*. No entanto, esta aproximação não foi seguida e optou-se por colocar todos os símbolos do mesmo tamanho, caso contrário, quando um símbolo grande fosse colocado numa zona de interseção de barras (canto) após uma rotação, perderia a sua forma original devido ao espaço restrito (Figura 3.9).

Portanto, nas barras de sinalização o utilizador apenas consegue induzir a distância por comparação de transparências entre POI da mesma relevância, enquanto que na área



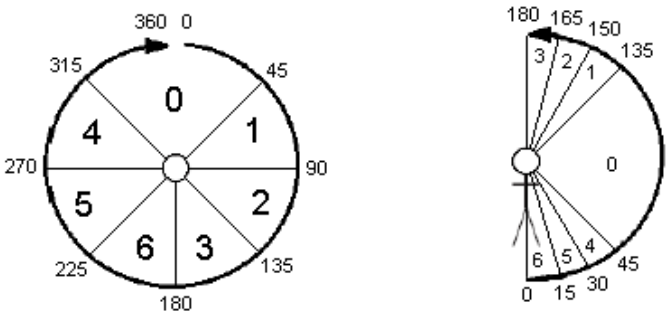
(a) (b)

Figura 3.9: Perda de informação da distância que é dada pelo tamanho, quando um ponto é colocado num canto. (a) Momento antes de rotação vertical (b) Momento após rotação vertical.

visível poderá comparar transparências entre POI da mesma relevância ou tamanhos entre quaisquer POI de diferentes relevâncias (quanto mais perto, maior o símbolo representativo de um ponto).

A colocação dos objetos *off-screen* nas células das barras verticais e horizontais depende da abertura da câmara e da posição do objeto relativamente à direção em que a câmara está a apontar, tanto na horizontal como na vertical. Um dos parâmetros de configuração da aplicação é o ângulo de abertura da câmara, que neste caso foi inicializado a 90º, sendo este o valor que serviu de referência aos ângulos apresentados na Figura 3.10. Relativamente ao plano horizontal, consideraram-se 7 zonas, ilustradas na Figura 3.10 a. Estas zonas são identificadas pelo ângulo entre a direção em que se encontra o objeto e a direção em que está apontado o dispositivo, correspondente ao ângulo 0º: a zona 0 abrange a área visível no ecrã; as zonas 1, 2 e 3 são, respetivamente, a 1ª, a 2ª e a 3ª colunas da barra direita, começando a contar do bordo interior da moldura; e as zonas 4, 5 e 6 são, respetivamente a 1ª, a 2ª e a 3ª colunas da barra esquerda, a contar do bordo interior da moldura. Quanto à disposição dos POI na vertical, seguiu-se uma divisão análoga à considerada para o plano horizontal. Consideraram-se também 7 zonas, em que a zona 0 corresponde à área visível no ecrã e as restantes zonas às áreas das barras superior (zonas 1, 2 e 3) e inferior (zonas 4, 5 e 6) (Figura 3.10 b).

Assim, se um ponto está colocado na barra da direita, o utilizador deve rodar nesse sentido para que este apareça na zona visível do ecrã. A coluna e/ou linha em que o símbolo é colocado funciona como um mecanismo de *feedback*, que permite ao utilizador ir acompanhando o movimento do ponto *off-screen* relativamente à sua orientação à medida que roda. Isto permite que o utilizador seja capaz de antecipar o momento de entrada do ponto na área visível, pois caso o sistema não desse qualquer *feedback*, o processo tornar-se-ia pouco intuitivo e iria transmitir a ideia que nada está a acontecer, enquanto o utilizador rodava o dispositivo.



(a) (b)

Figura 3.10: Rotações permitidas no IAR. (a) Rotação horizontal permitida vista de cima (b) Rotação vertical permitida vista de lado.

O resultado final obtido do mecanismo das barras pode ser observado na Figura 3.11.

Figura 3.11: Sinalização *off-screen* na versão final do protótipo.

Mapa 2D

A RA exige que o utilizador segure no dispositivo de forma firme, o que prolongadamente, pode causar alguma fadiga e desconforto ao utilizador, visto esta ser uma posição

pouco natural [57]. Por esse motivo, pensou-se que a adição de um modo de Mapa 2D complementar à RA poderia trazer benefícios à experiência de utilização. Ao fornecer um modo alternativo de visualização com um Mapa 2D, permitimos que o utilizador use a RA para encontrar o que pretende e se oriente nessa direção, e que de seguida possa mudar para outro modo mais confortável, podendo voltar ao modo anterior sempre que entender. Por outro lado, a RA pode estar a indicar pontos *on-screen* que ainda não são visíveis no mundo real, por existirem obstáculos entre o utilizador e o ponto ou por a distância entre eles ser ainda demasiado grande. Desta forma, alternando para o Mapa 2D, o utilizador poder-se-á situar no terreno e perceber onde estão os POI em relação a ele.

Para este efeito, numa primeira versão criou-se uma outra vista alternativa na aplicação que mostrava um Mapa 2D em que a posição do utilizador era assinalizada com um circulo preenchido a azul. Os POI são assinalados no mapa com raquetes (designados *markers* ou marcadores). Inicialmente, a troca entre o modo RA e Mapa 2D era ativada balançando o dispositivo (*tilt*), evitando que o utilizador tivesse que largar as mãos do mesmo. A vista ilustrada na Figura 3.12 mostra o Mapa 2D nesta versão do protótipo.

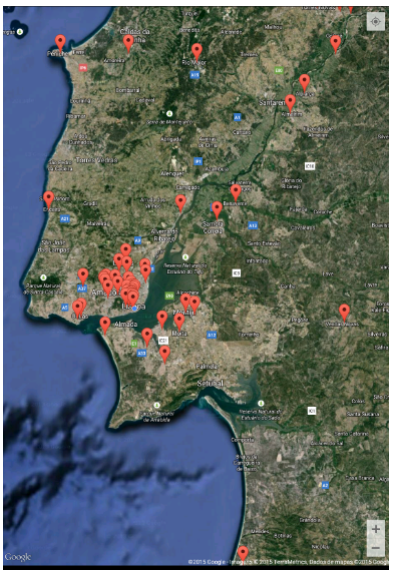


Figura 3.12: Mapa 2D que sinaliza utilizador e bancos no terreno.

Após alguns testes internos com um número reduzido de pessoas, verificou-se que é preferível mostrar o mapa no modo *roadmap* e não *satélite*, por apresentar o mapa com linhas e imagens mais simples e claras. Também os marcadores que sinalizavam os POI geraram alguma confusão, uma vez que todos possuiam as mesmas cores e transparências, ao contrário do modo RA. Os utilizadores esperavam que as mesmas representações se mantivessem de um modo para o outro, e portanto, para evitar essa inconsistência, modificou-se o aspeto por omissão dos marcadores para que assumissem as mesmas cores e transparências que no modo de RA. Por fim, decidiu-se colocar a posição do utilizador no centro do mapa, para mais fácil orientação do utilizador, uma vez que algumas pessoas

revelaram alguma dificuldade para encontrar a sua posição corrente (círculo preenchido a azul) no mapa. Estas alterações culminaram no resultado ilustrado na Figura 3.13.

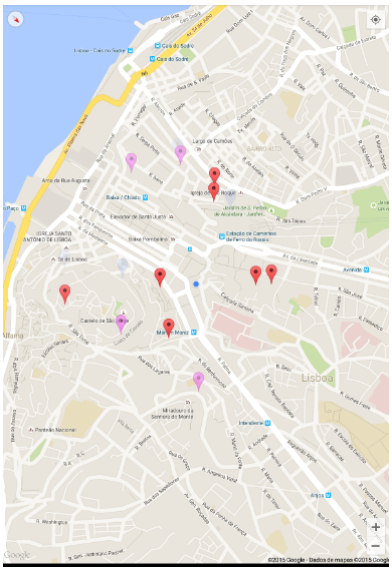


Figura 3.13: Mapa 2D no modo *roadmap* que com as cores e transparências dos bancos em concordância com o modo de RA.

Após as alterações anteriores, persistiram algumas dificuldades na orientação através do Mapa. Na verdade, a RA é capaz de acompanhar o movimento do utilizador e mostrar os elementos virtuais de acordo com a sua orientação, o que não acontece no mapa que é estático e apenas é possível navegar por ele através de operações de mudança de escala e arrastamento. Estes fatores tornam este modo muito mais difícil de usar relativamente ao anterior.

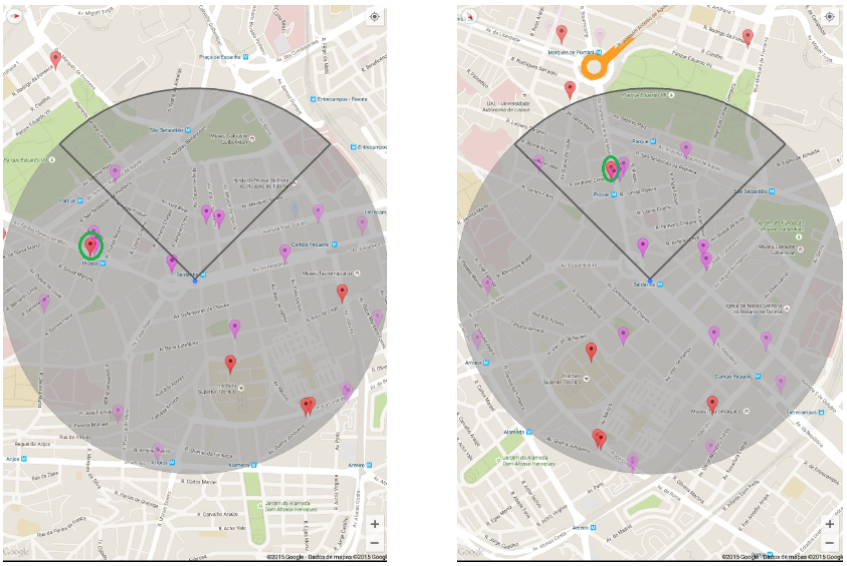
Por esta razão, optou-se por transformar o Mapa num Radar. Primeiro, adicionou-se uma circunferência sobre o Mapa que delimita o raio de pesquisa de POI, de modo a que este saiba que POI estão a aparecer na RA. Em seguida, adicionou-se ao mapa a capacidade de rodar consoante a orientação atual do utilizador, e desenharam-se linhas que delimitam a visão da câmara. Desta forma, criou-se uma forte ligação entre o modo de RA e este, na medida em que ambos respondem a mudanças de localização e orientação do utilizador, e o que é visto na RA corresponde ao que é visto no Mapa. Todos os símbolos *on-screen* na RA estarão dentro do campo de visão no Mapa, enquanto que todos os símbolos *off-screen* na RA, estarão fora do mesmo.

Uma última alteração foi efetuada relativamente à forma como se pode trocar entre os dois modos. Deixou de ser efetuada a mudança por *tilt* uma vez que um clique no ecrã, ou um abano não intencional desencadeava a mudança de modo. Por isso, criou-se um botão no canto superior direito que permite efetuar essa mudança. Por último, pensou-se que seria uma mais valia mostrar alguma informação ao clicar num marcador, e por isso o clique gera um balão que mostra a seguinte informação: tipo, região, morada e telefone.

Esta versão do Mapa 2D pode ser visualizada na Figura 3.14.

Figura 3.14: Mapa 2D capaz de rodar consoante orientação do utilizador e com o campo de visão da câmara delimitado.

Nesta versão, o campo de visão está a ser desenhado sobre o Mapa, e consequentemente, afeta as transparências reais atribuídas aos símbolos, como é visível na Figura 3.14. Por isso o campo de visão deixou de ser preenchido e manteve apenas as linhas de delimitação. O resultado final deste modo pode ser visto na Figura 3.15, que também mostra a capacidade de rotação do Mapa.



(a) (b)

Figura 3.15: Versão final do Mapa 2D. (a) Momento antes de rotação para encontrar o ponto assinalado a verde (b) Momento após rotação com o ponto assinalado a verde já dentro do campo de visão.

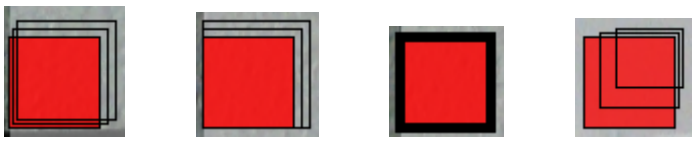
3.2.3 Tratamento de Sobreposições

Um dos problemas recorrentes da RA é a facilidade com que ocorrem sobreposições entre diferentes elementos virtuais, dificultando a visualização das informações fornecidas. É por isso que a aplicação desenvolvida se preocupa em fazer o tratamento das sobreposições, permitindo apresentar a informação de forma organizada e clara ao utilizador. Para resolver o problema é preciso analisar e desenhar soluções para as duas vertentes do problema, as sobreposições que acontecem na área visível e as que acontecem nas barras de sinalização, processo que será detalhado em seguida.

Sobreposições *on-screen*

O processo adotado para o tratamento de sobreposições foi o de agregação de símbolos sobrepostos, à semelhança da abordagem discutida em 2.4. Desse modo, um dos primeiros passos foi definir um símbolo que representasse uma agregação de POI.

Começou-se por desenhar e implementar no protótipo diferentes símbolos que representassem uma agregação de POI, também designados por símbolos em pilha. Durante este processo desenharam-se os símbolos ilustrados na Figura 3.16.



(a) (b) (c) (d)

Figura 3.16: Possíveis símbolos representativos de uma pilha de POI.

Após testes internos na equipa, optou-se pelo símbolo a) da Figura 3.16, um cubo em perspetiva, na medida em que foi considerado o mais intuitivo e esteticamente apropriado à representação de uma pilha de objetos. Fez-se alguns ajustes ao símbolo original, e concluiu-se este processo com o resultado ilustrado na Figura 3.17.

Figura 3.17: Símbolo final representativo de uma pilha de POI

Ademais, é também importante determinar como o utilizador poderá saber quantos POI estão agregados numa pilha. A primeira abordagem foi colocar tantos traços a dividir a pilha quantas sobreposições existissem. Esta opção pareceu ser extremamente intuitiva pois olhando para o ecrã as pilhas com mais POI agregados são claramente mais visíveis, por estarem mais carregadas devido às linhas desenhadas sobre as mesmas (Figura 3.18 a)). No entanto, esta solução revelou ter um problema, porque quando existia um número

de sobreposições considerável tornava-se difícil contabilizar o número de traços (Figura 3.18 b)) ou as sobreposições eram demasiadas e tornavam a pilha totalmente preta devido ao número muito elevado de traços desenhados (Figura 3.18 c)).

(a) (b) (c)

Figura 3.18: Uso de traços para indicar o número de sobreposições representados na pilha. (a) 3 sobreposições (b) 7 sobreposições (c) 14 sobreposições.

Por este motivo optou-se por uma abordagem mais simples e decidiu-se colocar o número de POI agregados como informação textual sobre o símbolo. Além disto, a pilha toma sempre a cor, transparência e tamanho (tamanho apenas no caso de ser *on-screen*) do ponto mais relevante de entre todos os agregados, ou no caso de existirem vários POI de iguais relevâncias, toma a cor e transparência do mais relevante mais próximo. Resumidamente, sempre que a aplicação mostra uma pilha representativa de um conjunto de POI, fornece informações textuais e visuais que permitem ao utilizador saber o número de POI agregados, e, distância e relevância do POI mais relevante e mais próximo de entre os agregados. Este processo de seleção da cor, tamanho e posterior construção da pilha está ilustrado na Figura 3.19.

Para a deteção de dois ou mais POI sobrepostos utilizou-se uma aproximação idêntica à descrita em 2.4, que divide o ecrã do dispositivo em linhas horizontais e verticais de forma a conceber uma grelha. A ideia subjacente é que quando existem dois ou mais POI na mesma célula da grelha, então sabe-se que existem sobreposições a tratar, uma vez que os símbolos que representam esses POI podem ser desenhados praticamente no mesmo sítio do ecrã. É o caso exemplificado pela Figura 3.20, que mostra a grelha do ecrã do dispositivo (invisível ao utilizador) e dois POI cuja posição pertence à mesma célula, o que levará à sobreposição de símbolos quando ambos forem desenhados no ecrã.

Todavia, este mecanismo implica um problema de contantes agregações e desagregações do mesmo conjunto de POI, que tornaria a aplicação confusa e muito difícil de usar. Como explicado, a Figura 3.20 representa uma grelha associada ao ecrã do dispositivo, e por isso conforme o movimento do dispositivo assim a grelha se movimenta. O que acontece é que os POI contidos numa célula da grelha saltam para outra célula quando se move o dispositivo móvel uma vez que estamos a mover a grelha, alterando instantaneamente os conjuntos inicialmente escolhidos para serem agrupados. Este problema está ilustrado na Figura 3.21, que mostra em (a) dois POI na mesma célula e em (b) os mesmos POI em células distintas, fenómeno que aconteceria repetidamente sempre que o dispositivo se movimentasse.

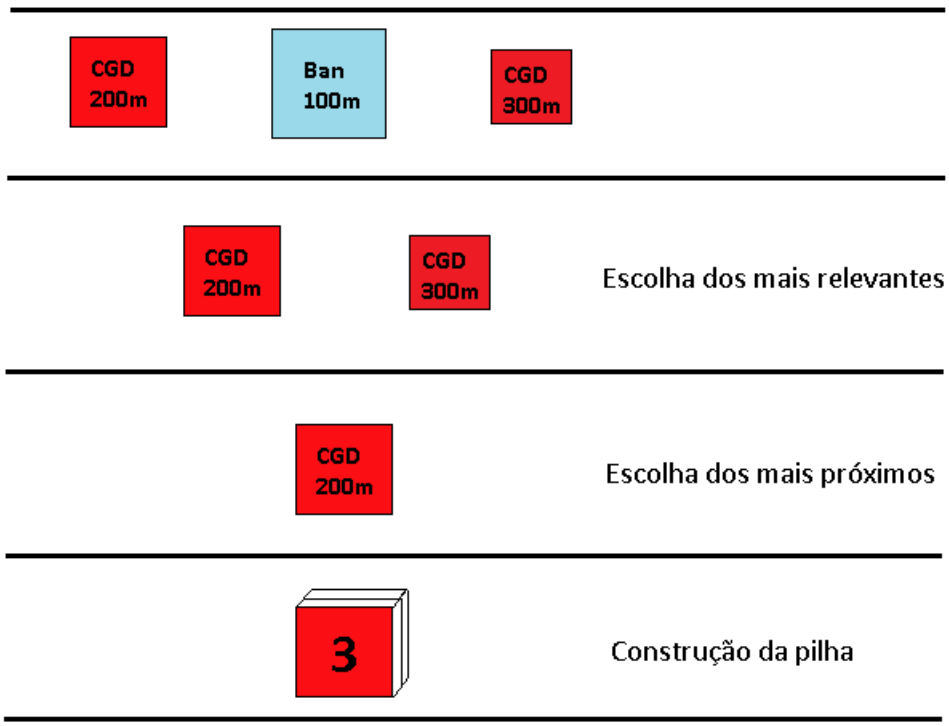


Figura 3.19: Processo de seleção da cor, transparência e tamanho para construção da pilha.

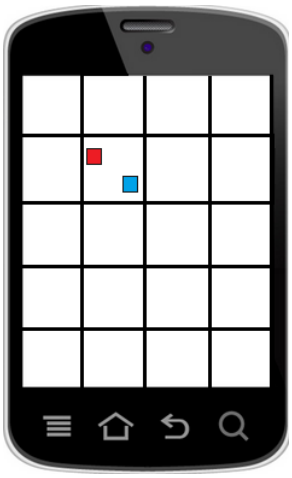
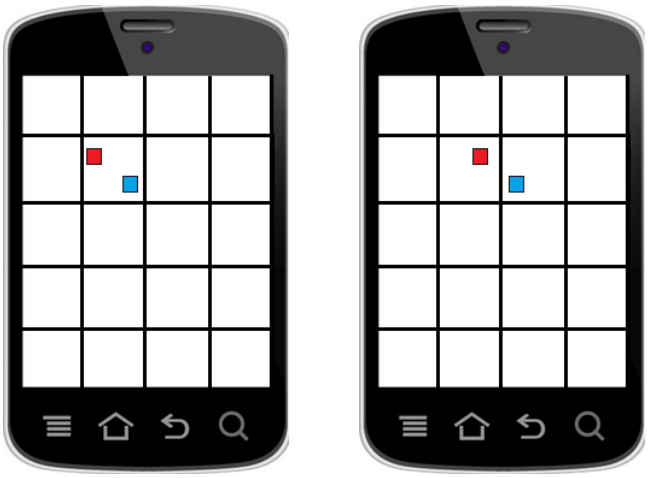


Figura 3.20: Grelha do ecrã com dois POI situados na mesma célula.

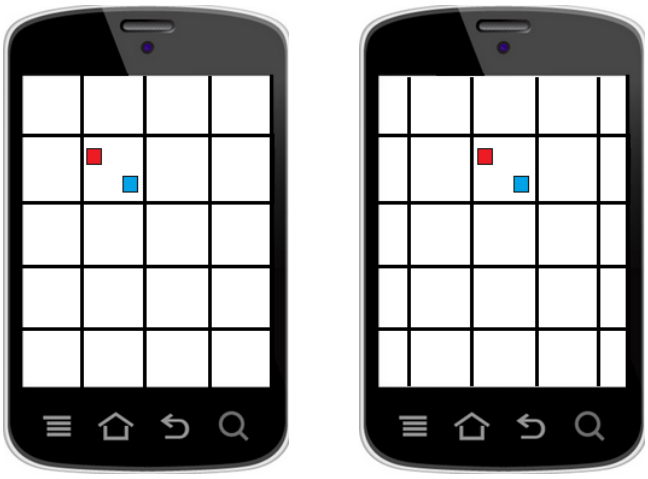
Tendo a abordagem anterior como base, sabia-se que o requisito principal para a solução seria a grelha estar fixa ao mundo e não ao dispositivo, para que quando este se movesse os POI não saltassem para outras células. Por isso, utilizou-se coordenadas geográficas (latitude, longitude, altitude) para definir os limites das secções que dividiriam o mundo real. Portanto, a nova grelha já não era bidimensional mas sim tridimensional, e era capaz de dividir o mundo em setores circulares. Esta nova abordagem que resolveu o problema está exemplificada na Figura 3.22, que mostra que quando o dispositivo se move



(a) (b)

Figura 3.21: Grelha fixa ao ecrã que altera o conjunto inicialmente escolhido para agregar ao rodar o dispositivo. (a) Momento antes da rotação para a esquerda (b) Momento após rotação para a esquerda.

a grelha não se move mas sim o utilizador passa a olhar para outras secções do mundo.

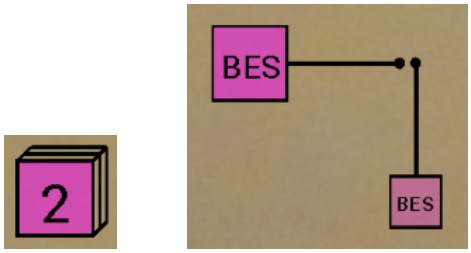


(a) (b)

Figura 3.22: Grelha fixa ao mundo que não altera o conjunto inicialmente escolhido para agregar. (a) Momento antes da rotação para a esquerda (b) Momento após rotação para a esquerda.

Por fim, acrescentou-se a opção de desagregação da pilha. Esta opção é muito importante porque permite ao utilizador ver a representação de cada ponto individualmente na sua localização original. Definiu-se que a desagregação é efetuada sempre que um utilizador clica numa pilha e que os POI pertencentes à pilha se mantém desagregados durante 7 segundos até se agregarem novamente. Quanto ao clique para desencadear a desagregação, pensou-se que seria a forma mais óbvia, lógica e intuitiva para o utilizador.

Já os 7 segundos até os POI serem agregados novamente servem para evitar um segundo clique no ecrã, e consequentemente, evitar o desconforto ou desiquilibrio do dispositivo quando segurado com uma só mão. Decidiu-se ainda que só seria possível ter uma pilha desagregada de cada vez no ecrã, pois se várias pilhas fossem desagregadas seguidamente tornar-se-iam demasiado intrusivas, e possivelmente criando sobreposições com outros símbolos. Ao desagregar uma pilha são colocados círculos pretos nas posições originais de cada um dos POI pertencentes à agregação, que permitem ao utilizador saber a posição real do ponto, e linhas que unem pontos aos símbolos representativos de cada um deles. A função de desagregação pode ser visualizada na Figura 3.23.



(a) (b)

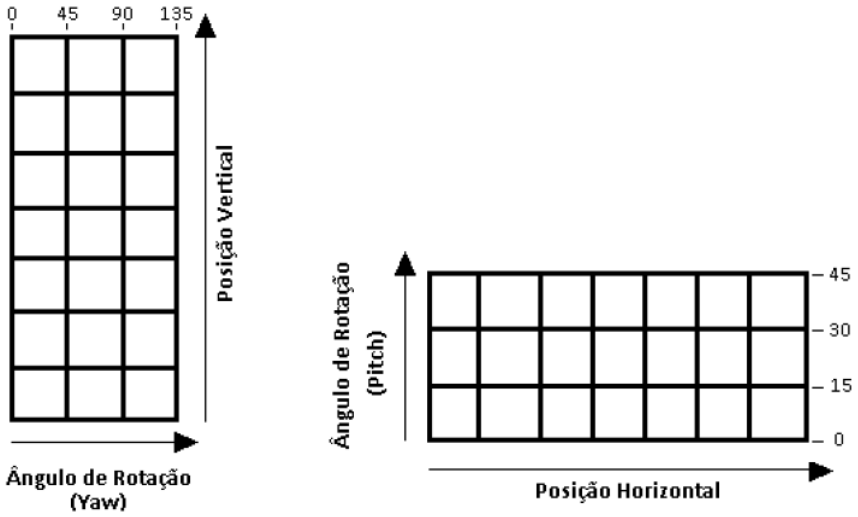
Figura 3.23: Processo de desagregação. (a) Pilha inicial (b) Pilha desagregada.

Sobreposições *off-screen*

O tratamento das sobreposições que ocorrem nas barras de sinalização segue uma abordagem idêntica à aplicada às sobreposições *on-screen*, de maneira a evitar inconsistências ou confusões por parte do utilizador.

O símbolo escolhido para a representação de uma pilha é exatamente igual, um cubo em perspetiva com um número indicativo do número de POI agregados. Também as regras da cor, transparência mantém-se exatamente iguais à exepção do tamanho, que como já explicado não pode ser manipulado por ser perdido quando um ponto está associado a um canto. Por isso, uma vez que o tamanho é igual para todos, relativamente aos pontos *off-screen* só será possível induzir a distância entre POI da mesma relevância através da transparência. A abordagem utilizada na grelha foi igualmente aplicada às barras, o que implicou dividir horizontalmente e verticalmente cada uma das barras, que culminou no resultado ilustrado na Figura 3.24.

É ainda importante realçar que pilhas formadas nas barras de sinalização não podem ser desagregadas, pois não seria cómodo para o utilizador executar essas acções num espaço tão reduzido, nem seria possível associar os POI à sua posição geográfica correta. O sistema de agregação *off-screen* final pode ser visto na Figura 3.25.



(a) (b)

Figura 3.24: Nova versão das barras de sinalização para divisão do mundo em secções. (a) Barra vertical direita (b) Barra horizontal inferior.

Figura 3.25: Agregação de símbolos *off-screen*.

3.3 Arquitetura e Implementação do IAR

Este capítulo destina-se à explicação da arquitetura e desenvolvimento do IAR. Na secção 1.3 é explicada a metodologia adotada para o desenvolvimento do *software*, na secção 3.3.1 é discutida a estrutura da aplicação e os dispositivos alvo dos protótipos. Por fim, na secção 3.3.2 são explicados os algoritmos e técnicas implementadas.

3.3.1 Modelo de Arquitetura

Os protótipos foram construídos na plataforma de código aberto Android SDK 2.2 [2] com recurso à linguagem Java [7], para serem utilizados em dois dispositivos distintos. Um deles é um *tablet* Sony Xperia Z2, com Sistema Operativo Android 5.0.2, com um processador de 2.3GHz, uma resolução de 1200x1920 e um ecrã de 10.1 polegadas. O

outro é um *tablet* Asus Google Nexus 7, com Sistema Operativo Android 5.0.2, com um processador de 1.2GHz, uma resolução de 800x1280 e um ecrã de 7 polegadas. O protótipo final foi também testado num OnePlus One, com Sistema Operativo Android 5.0.2, com um processador de 2.5GHz, uma resolução de 1080x1920 e um ecrã de 5.5 polegadas. Foram realizados testes nestes três dispositivos já que o intuito era que a aplicação funcionasse tanto para dispositivos grandes (Xperia) como para médios (Nexus) e pequenos (OnePlus). Além do Android, para a construção da vista do Mapa 2D e obtenção da localização do utilizador foi necessário usar a GoogleMaps API. Por fim, o desenho dos elementos virtuais (símbolos *on-screen*, símbolos *off-screen*, campo de visão) foi efetuado recorrendo ao Canvas [4], que é um componente gráfico pertencente ao Android. O IDE utilizado durante o desenvolvimento do projeto foi o Eclipse [6].

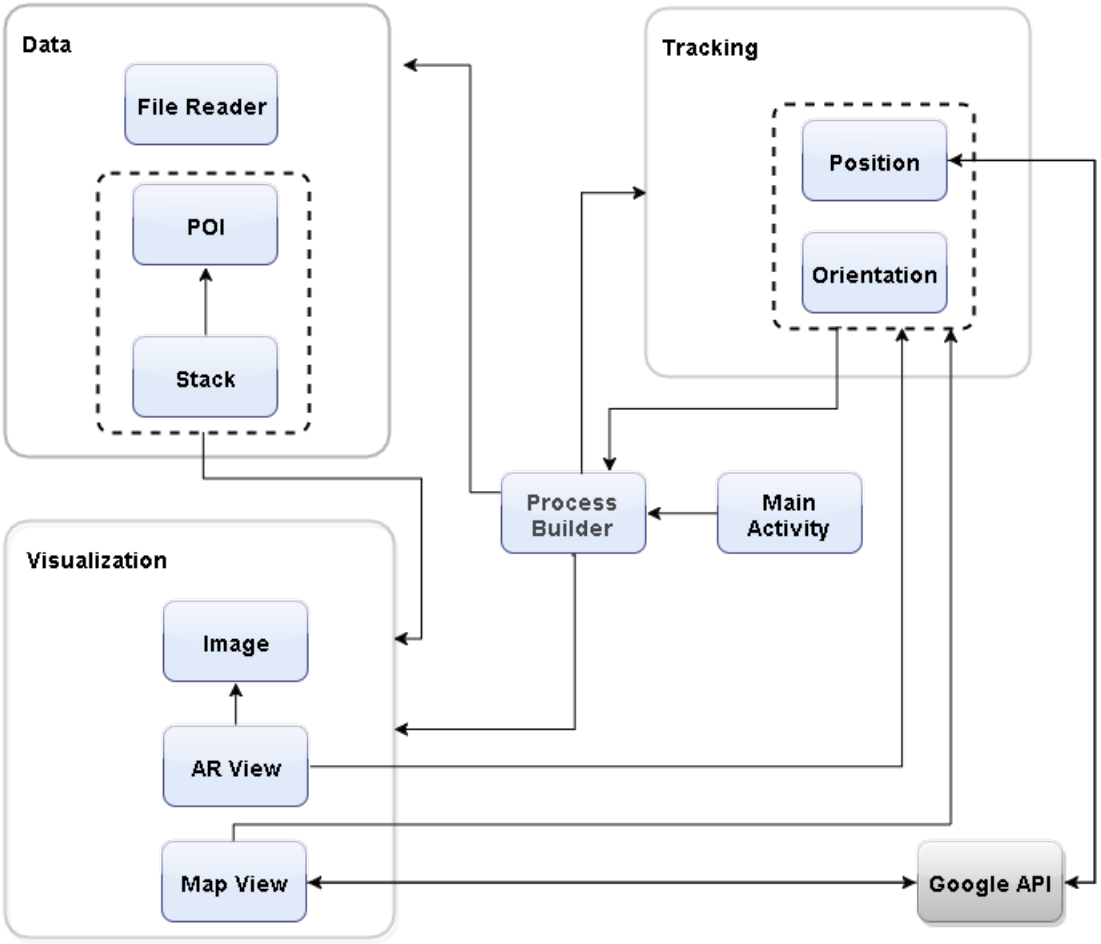


Figura 3.26: Arquitetura do IAR.

A Figura 3.26 mostra a estrutura da aplicação e a forma como os vários componentes desta comunicam. Cada bloco representa uma classe da aplicação, e cada ligação indica uma relação que poderá ser unidirecional (só uma das classes comunica com a outra) ou bidirecional (se existe comunicação em ambos os sentidos). O IAR é constituído por três grandes componentes: *Visualization*, *Tracking* e *Data*.

A aplicação é inicializada com a *Main Activity*, que é a atividade principal onde a aplicação é inicializada. Esta inicializa o *Process Builder*, que é o componente que controla o fluxo principal do programa. O *Process Builder* invoca o *File Reader* para iniciar a leitura dos dados, presentes num ficheiro *csv*, que tem as informações essenciais sobre os bancos (latitude, longitude, altura, relevância), inicializa o processo de captura de imagem, posição e orientação, e por fim, prepara a vista de RA e do Mapa para serem visualizadas. Assim que recebe a indicação do *Position* ou *Orientation* de que a posição ou orientação do utilizador mudaram, desencadeia uma atualização do *Map View* ou da *AR View* consoante o modo escolhido pelo utilizador. O *Process Builder* é o componente responsável por tratar da passagem entre o modo RA e Mapa 2D, e ainda por detetar cliques no ecrã que poderão levar a uma desagregação de símbolos, caso o clique tenha sido feito num símbolo pilha. Desta forma, o *Process Builder* determina o fluxo e comportamento geral da aplicação, comunicando e fazendo a ligação entre todos os outros.

O componente *Tracking*, é o responsável por registar a orientação e posição atuais do utilizador. Para saber a posição do utilizador, o *Position* precisa de comunicar com a API externa do Google, designada *LocationServices*, que vai responder aos pedidos enviados pelo IAR e, consequentemente, comunicar as mudanças de posição do utilizador. Como já dito anteriormente, sempre que há uma mudança na orientação ou posição do utilizador, esta é comunicada ao *Process Builder* para que tanto a vista de RA como a do Mapa sejam atualizadas de acordo com a nova posição e orientação.

O componente *Data* diz respeito à definição da informação. O módulo *POI* define um POI com todas as suas propriedades e funções, e por outro lado, o módulo *Stack* define uma pilha com todas as suas propriedades e funções. O módulo *Stack* comunica com o POI por ser basicamente um conjunto de POI e portanto usa a primeira para a sua própria definição. O *File Reader* como já mencionado, faz a leitura dos dados disponíveis sobre os bancos, criando e guardando os *POI* numa estrutura de dados, que será depois enviada ao *Process Builder* para que estes sejam usados pelos componentes *AR View* e *Map View*.

Por fim, tem-se o componente *Visualization* que é responsável pelo desenho dos elementos gráficos, concretizando os algoritmos necessários para determinar o seu posicionamento no ecrã. O módulo *AR View* começa por comunicar com o módulo *Image*, que obtém a imagem capturada pela câmara do dispositivo e a mostra no ecrã. Depois, efetua os cálculos e o desenho dos elementos associados à vista de RA, enquanto que o módulo *Map View* faz exatamente a mesma coisa mas relativamente à vista do Mapa 2D. Por exemplo, quando uma pilha aparece na área visível no ecrã, é o *AR View* que determina que existem sobreposições, que calcula a posição exata do ecrã em que esta deve aparecer, e determina qual deverá ser a sua cor, tamanho e transparência, e por fim a desenha. Por outro lado, a circunferência e campo de visão presentes no mapa, derivam de cálculos feitos no módulo *Map View*. Os cálculos em ambas as vistas são feitos com base na posição e orientação do utilizador, e daí a comunicação destes componentes com

os módulos *Position* e *Orientation*. Por fim, estes componentes usam e manipulam as definições de dados anteriormente criadas e guardadas dos módulos *POI* e *Stack*, daí a comunicação entre este bloco e estas duas entidades. É ainda de salientar que o *Map View* usa a Google API para que seja possível mostrar um Mapa na aplicação, apesar de não ser usada diretamente, é embutida na definição do *layout* desta vista.

3.3.2 Implementação de Funcionalidades

Tracking da Posição e Orientação

A primeira funcionalidade implementada na aplicação foi o *tracking*, uma vez que é fundamental para a robustez e eficácia do IAR.

O processo de *tracking* da localização é simples, uma vez que a aplicação não precisa de se preocupar na forma como a localização é obtida pois isso é tratado pela Google API. Para o *tracking* da localização poderia ter-se usado a biblioteca Location do Android, mas optou-se por usar a Google API pois tem um menor consumo energético e é considerada mais precisa [1]. Para isso, a aplicação desenvolvida necessita apenas de: conectar-se à Google API; enviar um *request* a pedir a localização exata do utilizador; e tratar a resposta dada ao pedido. A conexão consiste em criar um objeto do tipo *GoogleApiClient*, especificar que se quer efetuar uma ligação com a API *LocationServices*, e por fim, efetuar a ligação com o método *connect*. Em seguida, o *request* é criado e configurado para uma prioridade de alta precisão e para ser atualizado segundo a segundo. Isto é fundamental, pois o objetivo é garantir alta precisão na localização fornecida e uma taxa de atualização elevada, para que o IAR seja fiável e permita uma experiência real e envolvente. Por último, é necessário estar constantemente à espera das respostas aos vários pedidos, e tratá-las. A resposta retorna a localização no formato *(latitude, longitude)*, e o tratamento consiste em perceber se o utilizador mudou ou não de localização. Se mudou é necesssário guardar a nova localização do utilizador e o ecrã da aplicação deve ser atualizado, caso contrário, nada precisa ser feito. As alturas dos POI está estipulada no ficheiro de dados (csv), uma vez que não se conseguiu utilizar a *Elevation API* do *GoogleMaps* para esse efeito, por retornar 0 sempre que o método *getElevation* era utilizado. O processo aqui descrito está representado na Figura 3.27.

Contrariamente, o processo de *tracking* da orientação é mais complexo. Exige o processamento, cálculo e transformação dos valores obtidos através de sensores de *hardware*, em ângulos indicativos das rotações reais do utilizador nos vários sentidos e eixos. Primeiro, é necessário registar os sensores a utilizar através do método *registerListener(...)*, que neste caso são o acelerómetro e o magnetómetro. O acelerómetro presente no dispositivo mede o valor de aceleração (variação da velocidade) aplicada ao longo dos três eixos *(x, y, z)* incluindo o valor da gravidade. Isto é conseguido porque normalmente este acelerómetro é composto por três unidades distintas que estão adjuntas a cada um dos

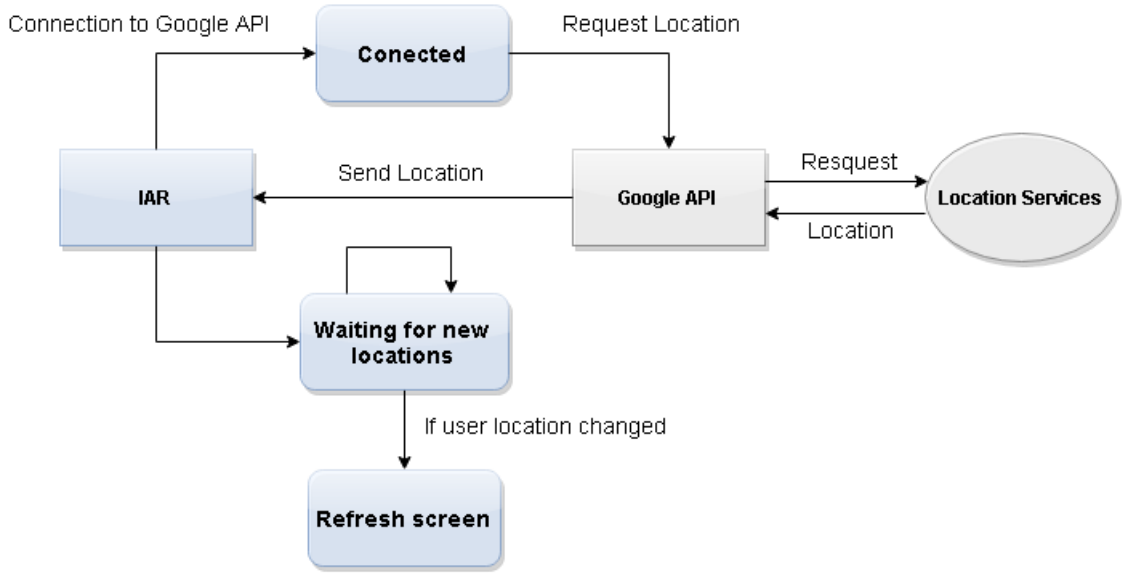


Figura 3.27: Processo de *tracking* da localização.

eixos, e cada uma mede a aceleração ao longo do eixo que lhe está associado. Por fim, sabendo o valor da componente vertical da aceleração ($9.8m/s^2$), é possível calcular o ângulo de rotação do dispositivo com operações trignométricas (Figura 3.28).

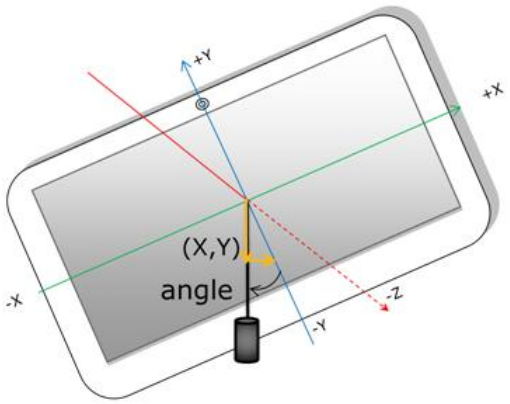


Figura 3.28: Representação das variáveis utilizadas no cálculo da rotação do dispositivo [74].

Por outro lado, o magnetómetro permite monitorizar mudanças no campo magnético da Terra, conseguindo por exemplo identificar em que direção se situa o norte magnético. É fundamental o uso deste sensor, porque caso contrário, saberíamos os ângulos das rotações efetuadas pelo utilizador mas não seria possível enquadrar essas rotações no espaço físico terrestre. Portanto, é a combinação dos valores dados por ambos os sensores que permite o processo completo de *tracking*.

Ao registar os mesmos, o sistema fica à espera que alterações nos valores reportados pelo acelerómetro e magnetómetro se verifiquem, e quando estas ocorrem é gerado um

evento. Este desencadeia as seguintes acções no sistema:

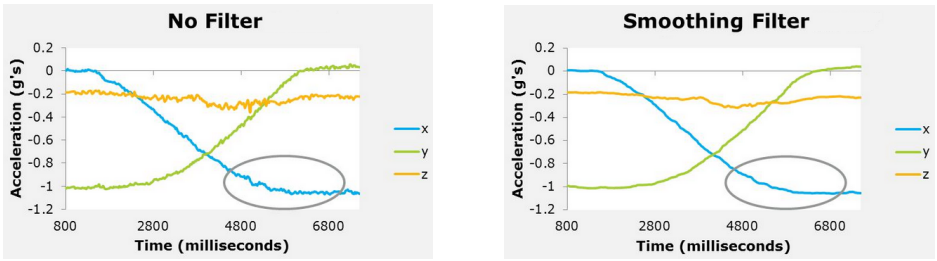
- Obtenção dos dados capturados pelos sensores com o método *getRotationMatrix(...)*;
- Mapeamento entre os eixos do dispositivo e os do mundo real através do método *remapCoordinateSystem(...)*. Neste caso específico, como o IAR foi concebido para ser usado num dispositivo seguro verticalmente é preciso indicar que o eixo do *y* no referencial do dispositivo corresponde ao eixo do z do sistema de coordenadas do utilizador;
- Cálculo da orientação atual do dispositivo com base na matriz de rotação anteriormente obtida, com o método *getOrientation(...)*. A orientação é dada em radianos na forma de *(azimuth, pitch, roll)*, sendo o sentido positivo contrário ao dos ponteiros do relógio.

Dos três ângulos fornecidos, só o *azimuth* e o *pitch* é que vão ser utilizados para o funcionamento da aplicação, uma vez que o *roll* representa rotações pouco naturais e expectáveis no uso deste tipo de aplicações. O *azimuth* retornado encontra-se no intervalo $[-\pi, \pi]$ e o *pitch* no intervalo $[-\pi/2, \pi/2]$, e por isso, foi necessário transformar estes intervalos para $[0, 360]^{\text{o}}$ e $[0, 180]^{\text{o}}$ respetivamente. Primeiro, porque seria muito mais fácil trabalhar em graus ao invés de radianos, visto que em radianos a distribuição dos ângulos nos quadrantes é mais difícil de interpretar. Usou-se uma função da biblioteca de matemática do Java para fazer a conversão de graus para radianos, designada *toDegrees(...)*, e por fim ao *azimuth* somou-se 180^{o} e ao *pitch* 90^{o}. Desta forma, os ângulos estavam preparados para serem usados para posteriores cálculos.

No entanto, quando o processo de *tracking* foi implementado e testado com alguns elementos virtuais no ecrã, surgiu um problema de *flickering* dos elementos virtuais, devido ao ruído dos valores (*noisy data*) reportados pelos sensores. Por exemplo, mesmo quando o dispositivo estava sobre uma mesa, imóvel, os sensores continuavam a capturar variações minímas de aceleração e/ou magnetismo, que causavam mudanças constantes de posição nos elementos virtuais. Para resolver este problema recorreu-se ao algoritmo, designado *lowPassFilter(...)*, capaz de eliminar as frequências muito altas obtidas pelos sensores, e assim suavizar as variações entre os diferentes valores obtidos. Este algoritmo recebe o valor obtido anteriormente v_{antigo} pelo sensor e o novo valor registado v_{novo}, define uma constante λ (à qual se atribuiu um valor de 0.45) que determina o quanto o novo valor deverá afetar o antigo, e aplica a seguinte fórmula para o valor obtido pelo sensor para cada eixo:

$$f(v_{antigo}, v_{novo}) = v_{antigo} + \lambda * (v_{novo} - v_{antigo})$$

Isto permitiu mitigar o problema de *flickering* dos elementos virtuais através da redução de ruído dos valores, como exemplificado na Figura 3.29.



(a) (b)

Figura 3.29: Valores obtidos pelos sensores. (a) Sem filtro (b) Com filtro de baixas frequências [74].

Apesar deste algoritmo resolver maioritariamente o problema, quando o utilizador segurava o dispositivo com as mãos e estava quieto havia uma ligeira variação na posição dos símbolos. Isto acontece porque o utilizador não está realmente imóvel, e o facto de estar a segurar o dispositivo verticalmente numa postura menos confortável não é favorável. O algoritmo anteriormente explicado não resolve este problema, apenas evita variações bruscas nas transições entre os valores otidos pelos sensores. Portanto, foi necessário definir um *delta* (incremento) que impedisse a atualização de valores caso a variação fosse inferior a este. O *delta* definido foi de 3.5º, e portanto só uma variação igual ou superior a esta irá atualizar os valores, e consequentemente, alterar a disposição dos elementos virtuais no ecrã.

Relevância

Como já referido, os símbolos são representados através de duas propriedades gráficas: cor e transparência. Relativamente à cor, o processo é muito simples. No ficheiro de dados *csv* que contém as informações sobre os bancos que são processados na aplicação, a cada banco foi atribuído um número no conjunto 1, 2, 3. O número 1 corresponde à cor vermelha (ponto de elevada relevância), o número 2 à cor magenta (ponto de média relevância), e por fim, o número 3 corresponde à cor azul (ponto de baixa relevância). Assim, quando cada uma das linhas do ficheiro é lida e o ponto guardado numa estrutura de dados como um objeto, essa propriedade fica também definida. O restante processo é evidente, para cada elemento virtual correspondente a um POI atribuímos a cor que lhe foi previamente definida no modelo *RGB*:

- Se a relevância de um ponto é 1, então *RGB(255, 0, 0)*;
- Se a relevância de um ponto é 2, então *RGB(255, 0, 255)*;
- Se a relevância de um ponto é 3, então *RGB(0, 0, 255)*;

Enquanto que a cor é uma propriedade que não se altera ao longo do tempo (nesta aplicação), a transparência por representar a distância de um ponto ao utilizador pode

alterar-se à medida que o utilizador se desloca. Por isso, não pode ser calculada e guardada *a priori*, tem que ser calculada em tempo real. Usou-se a Tabela 3.1 como referência para implementar o mecanismo de transparência, variando esta num intervalo de $[0, 255]$, correspondendo o zero à maior transparência. Para conseguir o comportamento pretendido, criou-se uma função que permitisse interpolar o valor da transparência em função da distância *d* e relevância *r* de um ponto, sabendo para cada relevância o respetivo valor máximo e mínimo da transparência e ainda o raio de pesquisa $r_{pesquisa}$de POI relativamente ao centro (posição do observador):

$$f(r, d) = \left(\frac{t_{max}(r) - t_{min}(r)}{r_{pesquisa}} \right) * d + t_{min}(r) \in [0, 255] \tag{3.1}$$

Por fim, sabendo a cor e transparência corretas a atribuir a um símbolo num dado instante, é preciso definir no Canvas esses atributos para que sejam usados no desenho do mesmo. Para isso, é usado um objeto designado *Paint* que é passado como parâmetro para todos os métodos de desenho do Canvas, que indica como o desenho deve ser feito (ex: cor, espessura, tamanho do texto). Desta forma, o processo deverá seguir o seguinte curso:

- Criação de um objeto do tipo *Paint* e definir a cor e transparência para desenhar com o método *setColor(Color.argb(...))*;
- Efetuar o desenho do símbolo representativo do POI passando o *Paint* anteriormente definido no método *drawRect(...)*.

O processo explicado funciona de forma exatamente igual para símbolos *on-screen* e *off-screen*. Todavia, para aplicar as mesmas transparências e cores nos marcadores do Mapa 2D, de forma a manter a consistência, é necessário adaptar o comportamento pois a API do GoogleMaps funciona de forma diferente do Android. Relativamente à transparência, é preciso converter o intervalo de transparência para $[0, 1]$, pois no *GoogleMaps* a cada marcador só pode ser atribuída uma transparência nesse intervalo. Para definir a cor, não se usa o modelo *RGB* mas sim *HSL*, e é o parâmetro *Hue* que define a cor. De seguida resume-se os passos necessários para traduzir a cor e transparência aplicadas para os marcadores do Mapa 2D:

- Converter o intervalo de transparência $[0, 255]$ para $[0, 1]$, ou seja, efetuar o cálculo $transparencia/255$;
- Definir a cor a aplicar no marcador segundo o modelo HSL: se a relevância é 1 então hue = 0 (vermelho), se é 2 então hue = 300 (magenta), se é 3 então hue = 220 (azul);

- Usar a classe *BitmapDescriptorFactory* para alterar a imagem por omissão do marcador, aplicando o método *defaultMarker(...)* que recebe a cor anteriormente definida como parâmetro;

- Usar a classe *MarkerOptions* para definir a transparência do marcador com o método *alpha(...)*, passando a transparência anteriormente calculada, e usar essa mesma classe para alterar a aparência do marcador com o método *icon(...)*, que recebe o *BitmapDescriptorFactory* previamente definido com a nova imagem do marcador.

Sinalização *On-Screen* e *Off-Screen*

Para representar um POI seja *off-screen* ou *on-screen* é preciso perceber qual a sua posição relativamente ao utilizador no mundo real, de modo a determinar se este está ou não visível num dado momento.

No sistema construído, o cálculo da amplitude e do sentido do ângulo de rotação que deverá ser executado pelo utilizador para que este se alinhe com a direção do POI, é feito do seguinte modo: calcula-se o *azimuth* e a inclinação do POI e do utilizador, e compara-se para determinar onde é que o ponto está relativamente ao utilizador no mundo real. O *azimuth* é o ângulo formado entre o *true north* (norte magnético) e uma outra direção (ex: direção de observação do utilizador ou direção do POI). A inclinação representa o ângulo formado entre o ponto mais próximo do horizonte e um outro. A Figura 3.30 representa a utilização do *azimuth* e da inclinação de um ponto para determinar a sua posição no globo terrestre relativamente ao utilizador.

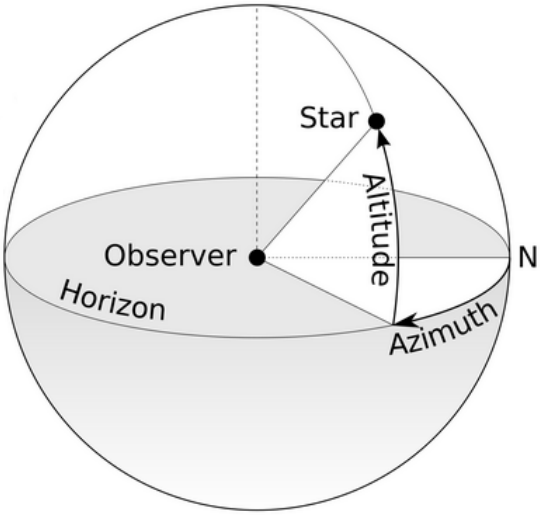


Figura 3.30: Variáveis necessárias para determinar a posição de um ponto na superfície terrestre a partir do *azimuth* e inclinação [5].

Contudo, no IAR é preciso determinar a posição do POI relativamente à direção de observação do utilizador, isto é, a direção para onde aponta o dispositivo móvel. Por

esse motivo, será necessário calcular o *azimuth* e inclinação do POI bem como o *azimuth* e inclinação da direção de observação. Uma vez feitos esses cálculos, será necessário determinar qual o sentido de rotação que torna o ponto visível mais rapidamente. Isso será feito recorrendo ao cálculo do *bearing*, sendo este o conceito que permite saber, dada a localização do utilizador e a do ponto, qual é o ângulo de rotação a adotar relativamente a norte para chegar do primeiro ao último pelo caminho mais curto possível, tendo em conta que a terra não é uma esfera mas uma elipsóide. O *bearing* representa o ângulo entre duas quaisquer direções, e neste caso concreto, entre a direção de observação do utilizador e a direção do POI relativamente ao utilizador. Desta forma, é com base nos dois *azimuth*, da direção de observação do utilizador e do POI, previamente calculados, que se calcula o *bearing*, cujo valor irá determinar a amplitude de rotação necessária para tornar um ponto visível, como representado na Figura 3.31. Como já foi referido, o *azimuth* pertence ao intervalo $[-\pi, \pi]$ e é depois convertido para o intervalo $[0, 360]^{o}$. No entanto, o *bearing* já é retornado pela API Android nesse intervalo.

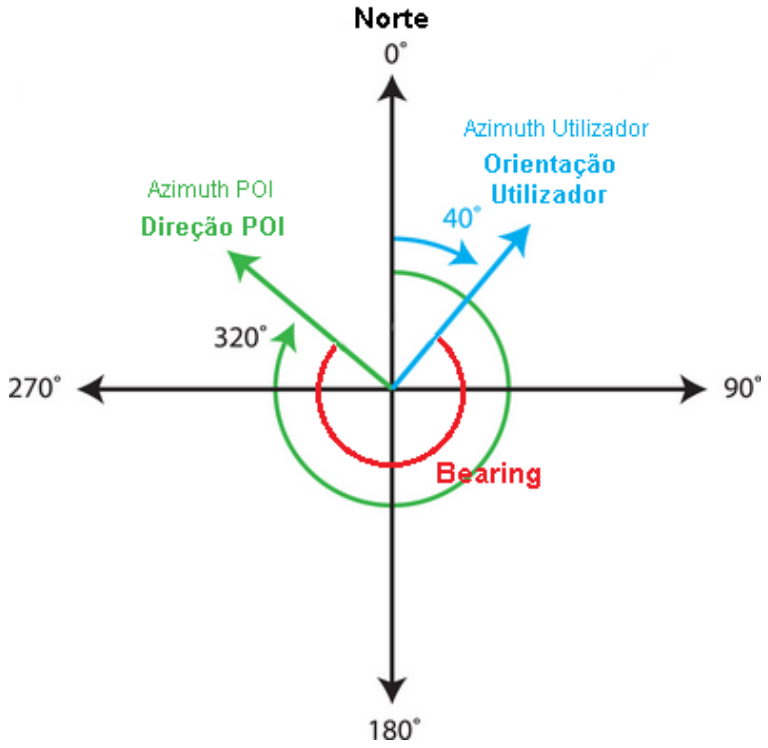


Figura 3.31: *Azimuth* e *Bearing*.

Desta forma, o cálculo da rotação horizontal necessária é feito com base no *azimuth*, *azUtilizador*, da direção de observação do utilizador e *azimuth*, *azPoi*, do POI em causa. O primeiro é obtido através do magnetómetro e acelerómetro do dispositivo móvel, e o último é obtido através do método *bearingTo(...)* disponibilizado na API Android dada a localização do utilizador e a do ponto. A função que faz o cálculo retorna um ângulo positivo no sentido dos ponteiros do relógio, e é dada pela seguinte fórmula:

$$yaw(azUtilizador, azPoi) = \begin{cases} (360 - azUtilizador) + azPoi & \text{se } azUtilizador > azPoi \\ azPoi - azUtilizador & \text{se } azUtilizador \leq azPoi \end{cases} \tag{3.2}$$

Por outro lado, o cálculo da rotação vertical é feito com base na diferença de alturas *h* entre o POI e o utilizador, a distância *d* que os separa e a inclinação do dispositivo, *inclinacaoDisp*, que é calculada através dos sensores já mencionados anteriormente. A função final, funciona de acordo com a seguinte fórmula:

$$inclinacaoPonto(h, d) = \arctan\left(\frac{h}{d}\right) \tag{3.3}$$

$$pitch(h, d) = \begin{cases} inclinacaoDisp & \text{se } h = 0 \\ inclinacaoPonto(h, d) + inclinacaoDisp & \text{se } h \neq 0 \end{cases} \tag{3.4}$$

Por fim, sabendo *yaw y*, *pitch p*, e que a amplitude vertical e horizontal da câmara é de 90º, pode concluir-se se o ponto está ou não na área visível de acordo com a fórmula seguinte:

$$posicaoPonto(y, p) = \begin{cases} \text{on-screen} & \text{se } (315 \leq y \leq 360 \vee 0 \leq y \leq 45) \wedge (45 \leq p \leq 135) \\ \text{off-screen left} & \text{se } 180 < y < 315 \\ \text{off-screen right} & \text{se } 45 < y \leq 180 \\ \text{off-screen bottom} & \text{se } p < 45 \\ \text{off-screen top} & \text{se } p > 135 \end{cases} \tag{3.5}$$

É preciso salientar que, o sistema representado funciona como um conjunto de condições não exclusivas, sendo possível que um ponto seja *off-screen left* e simultaneamente *off-screen top*. Esse é o caso em que o ponto se deverá situar no canto superior esquerdo do ecrã, pois o ponto está à esquerda do utilizador e a uma altura superior à do campo de visão do mesmo.

Relativamente à colocação dos elementos virtuais nas posições corretas do ecrã, o programa contém um evento principal, que está constantemente a ser desencadeado, e é responsável pelo desenho constante dos elementos ao longo do tempo, desginado *onDraw(...)*. Este método é responsável pela atualização dos símbolos (transparência, cor, posição) ao longo do tempo. Este método contém o ciclo principal do programa, que executa os cálculos necessários ao funcionamento da aplicação e desencadeia a atualização da posição dos símbolos gráficos no ecrã. O mundo foi dividido em setores de 9º para construir a grelha, pelo facto de 9 ser submúltiplo de 45 e consequentemente, de 90, 135, 180, 225, 270, 315 e 360, o que facilita os cálculos. Uma explicação detalhada sobre a

construção e funcionamento da grelha será dada mais à frente. Recorde-se ainda que a abertura da câmara é de 90º horizontalmente e verticalmente. É dentro deste ciclo que decorre o algoritmo que define as posições dos POI no ecrã, cujo funcionamento é descrito de seguida:

```
for cada POI existente do
    if está dentro do raio de pesquisa then
        atualizarYawEPitch() ;
        azimuthPOI = obterAzimuthPOI() ;
        inclinacaoPOI = obterInclinacaoPOI() ;
        sector = determinarSectorMundoPOI(azimuthPOI, inclinacaoPOI) ;
        if já existe um POI a ocupar esse setor then
            tratarSobreposição()   /* Detalhado mais à frente */;
        else
            mapaM1.colocarPoiNoMapa(sector)       /* Mapa M1 */;
        end
    end
end
for Cada POI existente no Mapa M1 do
    determinarEstadoVisibilidadePOI(yaw, pitch) ;
    determinarRegiaoVisivelEcra(azimuthUtilizador) ;
    if POI é on-screen then
        coordenadasX = obterXMaxMin() ;
        coordenadasY = obterYMaxMin() ;
    end
    if POI é off-screen left ou right then
        coordenadasX = obterColunaEsquerdaOuDireitaPOI(yaw) ;
        coordenadasY = obterYMaxMin() ;
    end
    if POI é off-screen top ou bottom then
        coordenadasX = obterXMaxMin() ;
        coordenadasY = obterColunaCimaOuBaixoPOI(pitch) ;
    end
    else
        coordenadasX = obterColunaEsquerdaOuDireitaPOI(yaw) ;
        coordenadasY = obterColunaCimaOuBaixoPOI(pitch) ;
    end
    desenharPoiNoEcra(coordenadasX, coordenadasY);
end
```

Para uma descrição mais detalhada do algoritmo, pode consultar-se o Apêndice A.1. Para ilustrar o funcionamento geral do algoritmo, observe-se a Figura 3.32. Inicialmente são calculados os ângulos necessários para mapear o mundo real para o ecrã do dispositivo:

β representa o *azimuth* da direção de observação do utilizador calculado e reportado pelos sensores do dispositivo, e mede 300º. O α representa o *azimuth* do POI, e mede 265º. É

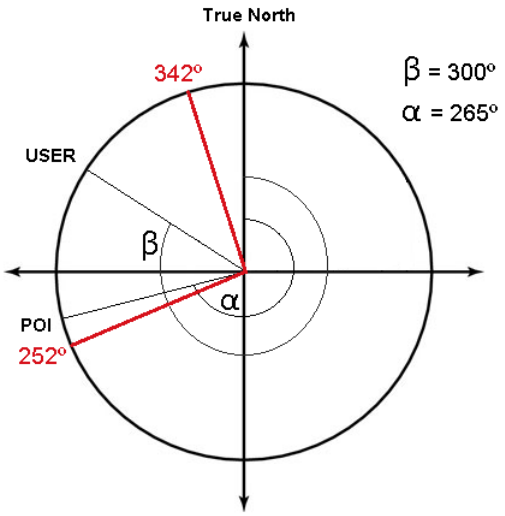


Figura 3.32: Variáveis necessárias para o mapeamento do mundo real para o ecrã do dispositivo.

preciso salientar que o α não varia de acordo com a orientação do utilizador mas sim de acordo com a sua localização geográfica. Por fim, como explicado acima, sabendo que o *azimuth* atual do utilizador é β,

a zona visível seria $[300 - 45, 300 + 45] = [255, 345]$, considerando o campo de visão da câmara de 90º. Contudo, atendendo a que a construção da grelha divide o mundo em sectores de 9º, a área visível no ecrã é ajustada para conter 10 sectores completos, correspondendo ao intervalo [252, 342], cujos limites estão representados a vermelho na Figura 3.32. Em seguida aplica-se a função 3.2 que efetua o cálculo $360 - \beta + \alpha = 360 - 300 + 265 = 325$, cujo resultado será passado como parâmetro para a função 3.5 que irá definir o ponto como *on-screen* horizontalmente, uma vez que $325 \in [315, 360]$. Por fim, como $\alpha \in [261, 270]$, é nesse setor horizontal que o símbolo deverá ser colocado, como demonstrado na Figura 3.33, que identifica a coluna a colocar o ponto com uma cor diferente.

É preciso lembrar que, neste momento, ainda não se sabe se no final do processo o ponto será *off-screen* ou *on-screen*. O facto do ponto ser dado como *on-screen* horizontalmente, não significa que a sua posição final no ecrã seja na área visível. Isto porque o *pitch* poderá ser demasiado baixo ou elevado, tornando-o *off-screen top* ou *off-screen bottom*. Por este motivo, na Figura 3.33 partes das barras de sinalização de cima e de baixo estão pintadas, pois são possíveis zonas finais do POI.

De seguida, procede-se ao cálculo da posição vertical do símbolo. Considere-se o exemplo da Figura 3.34 para os próximos passos.

Primeiro aplica-se a função 3.3 para obter a inclinação entre o ponto e o utilizador. São passados como parâmetros os valores 20 e 76, que representam a diferença de alturas e a distância entre ambos, respetivamente. Aplicando essa função, o sistema irá executar o

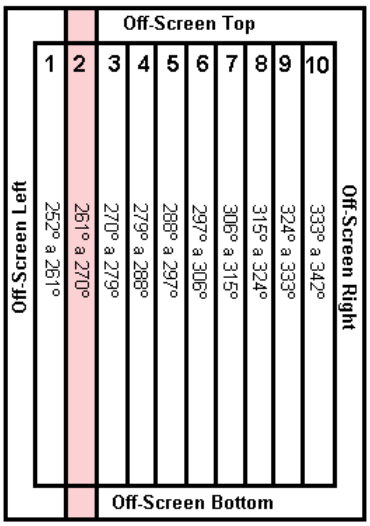


Figura 3.33: Mapeamento dos cálculos efetuados para o ecrã.

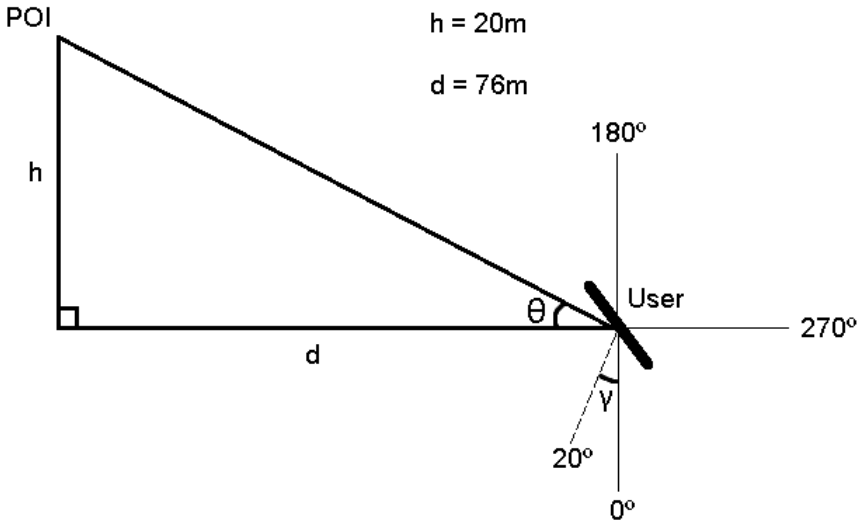


Figura 3.34: Variáveis necessárias para calcular os ângulos utilizados no mapeamento do mundo real para o ecrã do dispositivo.

cálculo $\theta = \arctan\left(\frac{20}{76}\right) = 16^{\text{o}}$ e por fim irá combinar o resultado obtido com a inclinação do dispositivo. Ou seja, da mesma forma que horizontalmente o *azimuth* da direção de observação do utilizador e o *azimuth* do POI precisam ser combinados para computar o ângulo que os separa, também a inclinação do dispositivo precisa ser combinada com a inclinação do POI. Para isso, é utilizada a função 3.4 da seguinte forma: $16 + \theta = 16 + 20 = 36^{\text{o}}$. Relembre-se que, como ilustrado, quando o dispositivo é seguro verticalmente com o ecrã virado para o utilizador, o dispositivo possui uma inclinação de 90º, sendo 180º quando a câmara traseira está apontada para o céu e 0º para o chão.

Por último, a aplicação da função 3.5 indica que o símbolo é, verticalmente, *off-screen top*, uma vez que $36 < 45$. Assim, fazendo a interseção dos cálculos efetuados para o

posicionamento horizontal, com os novos cálculos para o posicionamento vertical, obtém-se o resultado ilustrado na Figura 3.35.

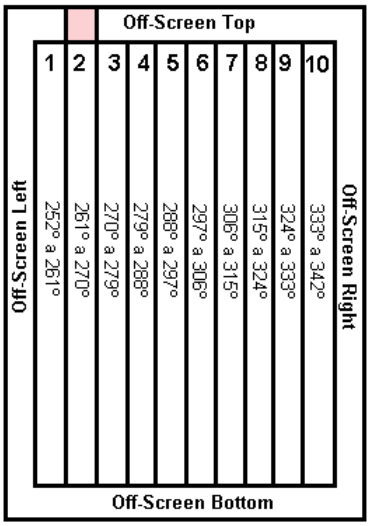


Figura 3.35: Mapeamento final do mundo real para o ecrã.

Modo de Visualização Mapa 2D

Além da sinalização através do modo de RA, existe também um modo de visualização de Mapa 2D. Este modo foi criado com a adição de um *layout* do tipo *MapFragment*, que permite ter uma vista com um mapa do *GoogleMaps*. Para poder usar o mapa, foi necessário gerar uma *API Key* para usar neste projeto. Essa chave foi gerada através do portal *GoogleDevelopers* e colocada no *manifest* da aplicação. Após esta etapa, adicionou-se um pedaço de código que coloca cada POI como um marcador no mapa com as localizações, cores e transparências corretas. Para esse efeito, usou-se o método *addMarker(new MarkerOptions(...))*, onde se passam como parâmetros a sua localização e aparência (cor, transparência). Repare-se que os marcadores só são adicionados inicialmente uma única vez ao mapa, o que se modifica é o terreno que o utilizador vê de acordo com a sua localização e orientação atuais. Posteriormente, desenha-se a circunferência que delimita o raio de busca. Para isso, começamos por mostrar a localização atual do utilizador no mapa com o método *setMyLocationEnabled(...)* e adiciona-se a circunferência com o método *addCircle(new CircleOptions(...))* no qual se passam parâmetros para definir o centro da circunferência, que neste caso será a localização atual do utilizador, e o raio que deverá possuir (em metros). Depois, foi preciso conseguir pôr o mapa a rodar e a mover-se de acordo com a orientação e localização do utilizador. Para tal, usou-se um método, designado *moveCamera(CameraUpdateFactory.newCameraPosition(new CameraPosition.Builder(...)))*, capaz de fazer o mover a câmara do utilizador para a nova localização fornecida, o que permite que a posição do utilizador apareça sempre no centro do ecrã, e simultaneamente, rodar o mapa de acordo com o *azimuth* atual do utilizador.

Este método é chamado sempre que a posição ou orientação do utilizador se modifica. Finalmente, falta desenhar o campo de visão sobre a circunferência previamente desenhada, processo que é feito segundo a seguinte sequência de instruções:

- Obter a matriz de desenho atual da vista com o método *getMatrix()*;
- Aplicar uma rotação de 45º em torno do centro da circunferência (que será o centro do ecrã) nessa matriz com o método *postRotate(...)*. Este ponto é designado *pivot*;
- Construir um *array* com dois pontos que definem uma reta vertical desde o centro da circunferência até ao seu limite;
- Aplicar a matriz de desenho anteriormente modificada ao *array* de pontos com o método *mapPoints(...)*, e guardar os novos pontos gerados;
- Aplicar o mesmo processo mas no ângulo inverso, -45º;
- Por último, quando já calculados os extremos das linhas que delimitam o campo de visão, usa-se o método *moveTo(...)* para iniciar o desenho do mesmo a partir da posição do utilizador. Usa-se o método *lineTo(...)* para desenhar cada uma das linhas, e o *drawArc(...)* para desenhar o arco entre as duas extremidades do campo de visão.

Recorde-se que as rotações feitas são de 45º em ambos os sentidos, uma vez que se considera a abertura total da câmara de 90º. Só desta forma, o que é visto na área visível no modo de RA coincide com o que é visto dentro do campo de visão no modo de Mapa 2D. O processo geral descrito está ilustrado na Figura 3.36. É de salientar que o campo de visão só é desenhado uma única vez no ecrã, pois é imóvel ao longo do tempo. O que se move é o mapa, fazendo com que o terreno dentro dos limites de visão se modifique à medida que o utilizador roda.

Tratamento de Sobreposições *On-Screen* e *Off-Screen*

O tratamento de sobreposições é um processo que consiste em três fases distintas. A primeira fase deverá ser capaz de mapear os POI de um setor para a posição correta no ecrã, ou seja, capaz de converter as coordenadas esféricas *(azimuth, inclinação, distância)* para coordenadas 2D do ecrã. Para isso, é preciso definir a que parte do ecrã corresponde uma determinada parte do campo de visão do utilizador. Ou seja, é preciso definir no IAR de que forma deve mapear cada setor da grelha contida no campo de visão do utilizador para o ecrã. Desta forma, criou-se um método que corre apenas uma vez quando o programa arranca, e que executa alguns cálculos para guardar esse mapeamento. Como já dito na secção anterior, optou-se por dividir o mundo em setores de 9º cada, pelo facto do 9 ser um número submúltiplo da maioria dos ângulos de referência. É importante esclarecer

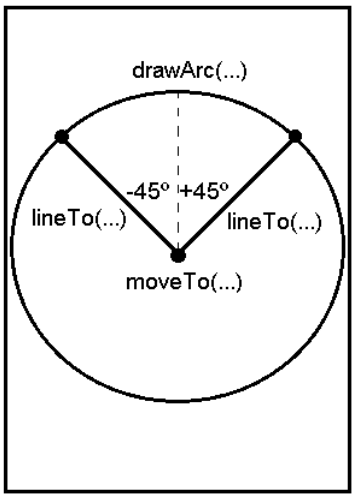


Figura 3.36: Construção do campo de visão no Canvas.

que o mundo é dividido em graus, pois a única maneira de ter a grelha fixa ao mundo é fazer a divisão com base nas coordenadas geográficas *(latitude, longitude, altitude)*, em que a latitude e a longitude representam a amplitude do ângulo entre um dado ponto da superfície terrestre e o equador ou merediano de *Greenwich* respetivamente (Figura 3.37).

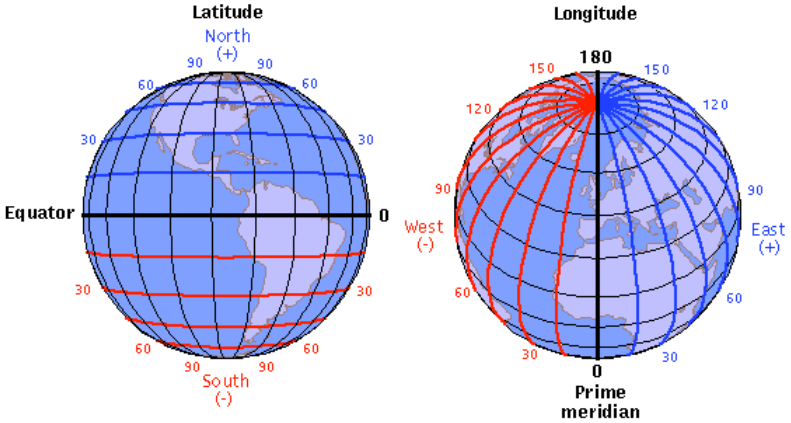


Figura 3.37: Latitude e Longitude [10].

Portanto, lembrando que se considerou a abertura da câmara com 90º verticalmente e horizontalmente, o método que define o mapeamento funciona da seguinte forma:

$$numSetoresVerticais = \frac{90}{9};$$
$$numSetoresHorizontais = \frac{90}{9};$$
$$areaVisivelVertical = alturaEcra - 2 * tamanhoBorda;$$
$$areaVisivelHorizontal = larguraEcra - 2 * tamanhoBorda;$$
$$tamSetorVertical = \frac{areaVisivelHorizontal}{numSetoresVerticais};$$
$$tamSetorHorizontal = \frac{areaVisivelVertical}{numSetoresHorizontais};$$

Sabendo quantos setores correspondem à área visível e o tamanho de cada um deles, pode colocar-se num *array* as coordenadas máximas e minímas de *x* e *y* no ecrã para cada setor. Por exemplo, para o setor mais à direita do ecrã (setor vertical 10), ao centro (setor horizontal 5) pode fazer-se o cálculo da seguinte forma:

$$x_{min} = tamanhoBorda + tamSetorHorizontal * (10 - 1);$$
$$x_{max} = tamanhoBorda + tamSetorHorizontal * 10;$$
$$y_{min} = tamanhoBorda + tamSetorVertical * (5 - 1);$$
$$y_{max} = tamanhoBorda + tamSetorVertical * 5;$$

Os diferentes *arrays* gerados são depois colocados num mapa do tipo *Map<String, ArrayList<Double>>*. A chave utilizada é a concatenação de *i* com *j*, que define a linha e coluna do ecrã em que o ponto deverá aparecer. A Figura 3.38 ilustra o processo aqui descrito.

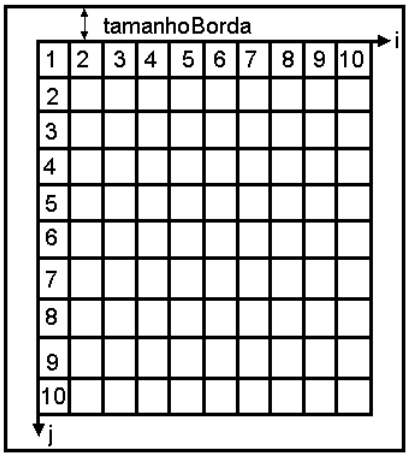


Figura 3.38: Divisão da área visível em setores verticais e horizontais de 9º cada.

Em seguida, na segunda fase, calcula-se o setor que se irá situar cada um dos POI, e inicia-se o tratamento das sobreposições que vão ocorrendo. Esta fase baseia-se na seguinte sequência de passos:

```
for cada POI existente do
    ... ;
    azimuthPOI = obterAzimuthPOI() ;
    inclinaçãoPOI = obterInclinacaoPOI() ;
    sector = determinarSectorMundoPOI(azimuthPOI, inclinacaoPOI) ;
    if já existe um POI A (poiA) a ocupar esse setor then
        if poiB é mais relevante e próximo que poiA then
            poiB.adicionaAPilha(poiA.obterPilha()) ;
            poiB.adicionaAPilha(poiA) ;
            poiA.eliminarPilha() ;
            substituirPoiNoSetor(poiB) ;
        else
            poiA.adicionaAPilha(poiB) ;
            poiB.eliminarPilha() ;
        end
    else
        ... ;
    end
end
```

Para uma descrição mais detalhada deste algoritmo, pode consultar-se o Apêndice A.2. Repare-se que após a execução deste algoritmo, para saber o número de sobreposições existentes num setor, basta aceder ao ponto associado ao setor, obter o número de POI que estão na sua *Stack* e somar 1 (o próprio). A Figura 3.39 descreve o processo acima descrito, para a escolha do ponto mais relevante e mais próximo ao longo do tempo, e consequente tratamento da pilha que lhe está associada. A, B, C e D representam diferentes POI a diferentes distâncias do utilizador (indicadas textualmente) e relevâncias (cor), e o rectângulo representa um setor do mundo. Na Figura 3.39 podemos observar que em (a) o POI A é colocado no setor, em (b) percebe-se que o ponto B é tão relevante como A mas mais distante, logo, o ponto A mantém-se no setor e B é adicionado à sua pilha. Em (c) o ponto A é comparado com C, e pelas mesmas razões que em (b), o ponto A mantém-se e C é adicionado à sua pilha. Por último, em (d) o ponto A é comparado com D, e sendo o último mais relevante que o primeiro, D substitui o ponto A no setor e a pilha de A assim como o próprio A são colocados na pilha de D.

Por fim, decorre a terceira e última fase. Esta serve para colocar o ponto mais relevante e próximo de cada setor na posição correta do ecrã, apresentando-o com o símbolo de pilha caso seja necessário. O algoritmo responsável por este comportamento funciona da seguinte forma:

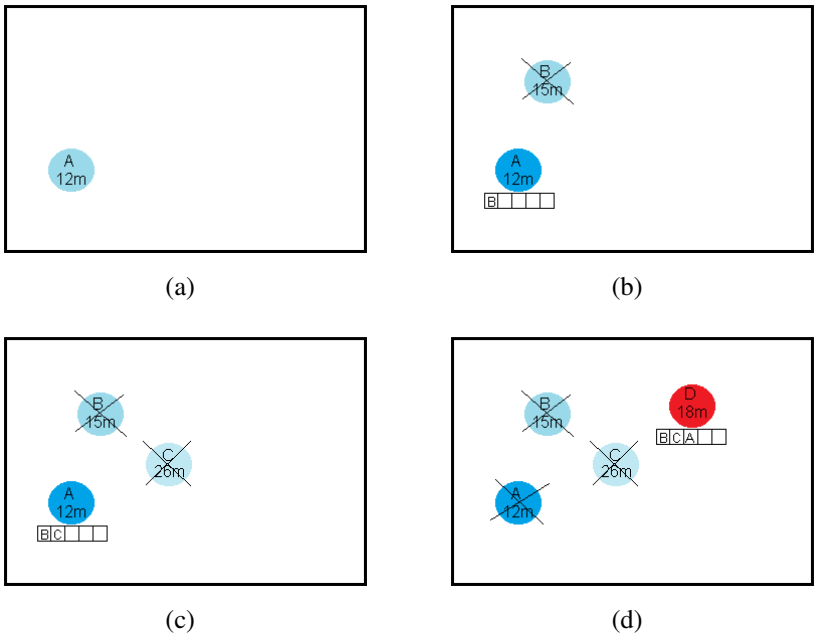


Figura 3.39: Processo de construção da pilha.

```
for cada setor do mundo (Mapa M1) do
    pontoSetor = mapaM1.obterValor(seccaoAtual) ;
    if pontoSetor é totalmente off-screen then
        determinarLinhaColunaMolduraSinalizacao(yaw, pitch) ;
    end
    if pontoSetor é totalmente on-screen then
        mapaM1.obterCoordenadasXeY(yaw, pitch) ;
    end
    if pontoSetor é off-screen verticalmente e on-screen horizontalmente then
        mapaM1.obterCoordenadasX(yaw, pitch) ;
        determinarLinhaMolduraSinalizacao(pitch) ;
    end
    if pontoSetor é off-screen horizontalmente e on-screen verticalmente then
        mapaM1.obterCoordenadasY(yaw, pitch) ;
        determinarColunaMolduraSinalizacao(yaw) ;
    end
end
for cada setor do mundo (Mapa M1) do
    poiDesenhar = mapaM1.obterValor(seccaoAtual) ;
    if poiDesenhar.obterPilha().tamanho ¿ 0 then
        desenharSimboloPilha() ;
        desenharTextoSobreposicoes(poiDesenhar.obterPilha().tamanho) ;
    else
        desenharSimboloPilha() ;
        desenharTextoTipoBanco(poiDesenhar.obterBanco()) ;
    end
end
```

Para uma descrição mais detalhada deste processo, pode consultar-se o Apêndice A.3. Para ilustrar o processo descrito, a Figura 3.40 mostra a sequência de passos até chegar ao desenho da pilha. Para o desenho desta Figura assumimos que o campo de visão do utilizador pertence ao intervalo $[0, 90]$ horizontalmente e $[45, 135]$ verticalmente. A Figura dá continuação à Figura 3.39 mostrada anteriormente para ilustrar o tratamento da pilha.

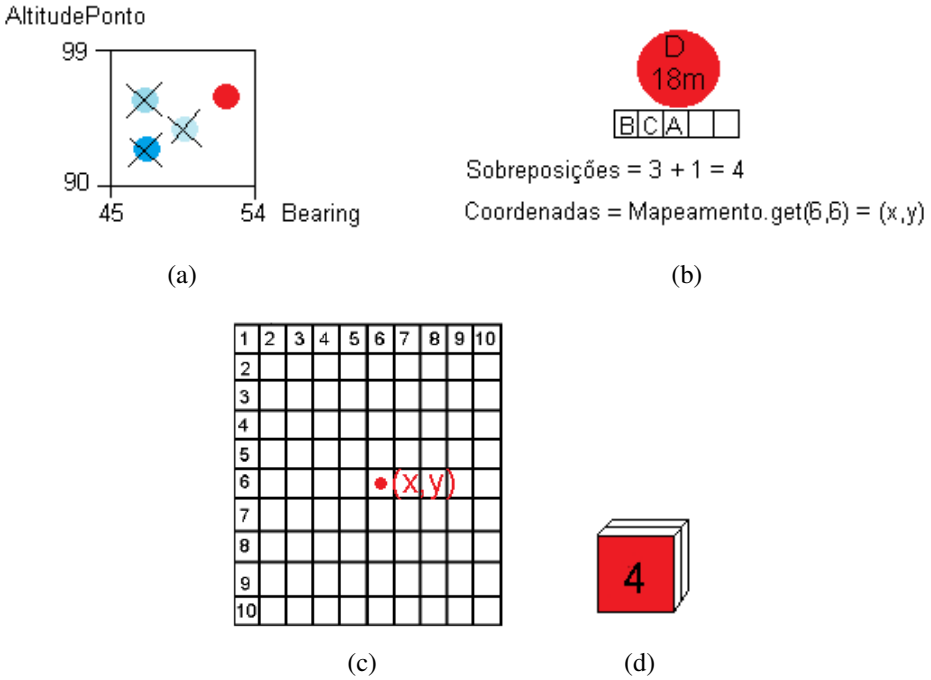


Figura 3.40: Processo de deteção e tratamento de colisões. (a) Obtenção do ponto mais relavante e próximo do sector (b) Cáculo do número de sobreposições e coordenadas do ecrã (c) Mapeamento para o ecrã (d) Desenho da pilha.

A figura começa por mostrar em a) o sector dado pela chave ”45-54-90-99”, ou seja, o sector que corresponde à interseção do setor horizontal dado pelo limite $[45, 54]$ com o setor vertical dado pelo limite $[90, 99]$. Além disso, é possível saber que o símbolo será totalmente *on-screen*, uma vez que $[45, 54] \subset [0, 90] \wedge [90, 99] \subset [45, 135]$. A essa chave, no mapa que representa a posição dos POI no mundo, está associado um único ponto (o mais relevante e mais próximo), e é esse ponto que é obtido. Em b) está representado o cálculo do número de sobreposições e das coordenadas do ecrã em que o POI deverá estar representado. Já a obtenção das coordenadas do ecrã são obtidas a partir do mapa que para cada sector visível possui as coordenadas correspondentes guardadas. Repare-se que o setor $[45, 54]$ corresponde ao 6º setor dos 10 visíveis (horizontalmente), sendo o mesmo válido (verticalmente) para o setor $[45, 135]$. Por isso são passados os parâmetros $i = 6$ e $j = 6$ para a obtenção das coordenadas. Esse mapeamento culmina na imagem c), que define e guarda as coordenadas associadas ao POI a ser representado. Por fim, a imagem d) ilustra o desenho da pilha com o número de sobreposições calculadas.

O processo de desagregação é desencadeado ao clicar no ecrã, sobre um símbolo pilha.

O clique é detetado com o evento *onTouch(...)*, e é neste método que o comportamento da aplicação está definido. O evento recebe as coordenadas do ecrã (x, y) em que o utilizador tocou, e são esses valores que permitem saber se o utilizador clicou sobre uma pilha ou noutro qualquer ponto do ecrã. Para saber se uma pilha foi clicada, compara-se o x com os limites do ecrã que definem cada um dos setores verticais, e o y com os limites do ecrã que definem cada um dos setores horizontais. Quando a condição $limite_{esq} <= x <= limite_{dir} \wedge limite_{sup} <= y <= limite_{inf}$ se concretiza, obtém-se o ponto pertencente à célula respetiva a partir do mapa que define os elementos associados a cada setor do mundo. Obtendo o ponto, basta verificar se é ou não uma pilha, e se for, separá-la. A separação consiste em obter os elementos da pilha através da *Stack* e mapeá-los no ecrã de acordo com as suas posições reais. Repare-se que ao separar, cada ponto individual não é associado a uma posição da grelha criada. Neste caso, é feita uma interpolação entre o *azimuth* do utilizador, *azimuth* do POI e a área visível no ecrã para colocar o símbolo no sitio exacto em que o banco deverá aparecer na imagem capturada pela câmara. Essa interpolação é feita com base no *bearing* horizontal, *bearingH*, e vertical, *bearingV*. Ou seja, esta função usa as diferenças entre os *azimuth* e inclinações do utilizador e do POI, para mapear os POI para a sua posição real no ecrã. A função funciona da seguinte forma:

$$\begin{aligned} xCoord(bearingH) &= \left(\frac{areaVisivelHorizontal}{90} * bearingH\right) \\ &+ \frac{areaVisivelHorizontal}{2} + tamanhoBorda; \\ yCoord(bearingV) &= \left(\frac{areaVisivelVertical}{90} * bearingV\right) \\ &+ \frac{areaVisivelVertical}{2} + tamanhoBorda; \end{aligned} \tag{3.6}$$

É a junção dos resultados que forma o par de coordenadas (x, y), que dita a colocação dos POI, individualmente, no ecrã. Por fim, sempre que uma pilha é desagregada, esta é guardada numa variável e assim mantida durante 7 segundos. Esta variável diz ao programa que a pilha deverá ser representada de forma separada com o auxílio da equação 3.6. Ao fim de 7 segundos essa pilha é apagada da memória, e retorna à sua forma original.

3.4 Sumário e Discussão

Este capítulo destinou-se à apresentação e explicação do funcionamento do protótipo IAR, que visa propor soluções para sinalização *off-screen*, representação de relevâncias e tratamento de sobreposições.

No capítulo seguinte é apresentada a avaliação das soluções propostas e valida-se se, de facto, auxiliam o utilizador durante a navegação num ambiente de realidade aumentada

móvel. O processo de avaliação será descrito desde a fase de planeamento à recolha e análise dos dados registados.

Capítulo 4

Avaliação e Resultados

Com o objetivo de comparar, analisar e perceber se o IAR é uma aplicação que auxilia o utilizador na navegação em ambientes de realidade aumentada móvel, foi efetuado um estudo de usabilidade. Nas secções seguintes, são descritas as várias etapas desde o planeamento da avaliação, construção de protótipos de teste até à análise dos dados e resultados obtidos.

4.1 Plano de Avaliação

4.1.1 Procedimento

De forma a avaliar o IAR e verificar o impacto no desempenho e satisfação dos utilizadores, de cada uma das suas funcionalidades, o teste de usabilidade considera as seguintes visualizações:

- Vis1: Realidade Aumentada;
- Vis2: Realidade Aumentada + Barras de sinalização *off-screen*;
- Vis3: Realidade Aumentada + Barras de sinalização *off-screen* + Tratamento de sobreposições;
- Vis4: Visualização 3 + Mapa 2D;
- Vis5: Visualização 3 + Mapa 2D + Radar.

Na medida em que o objetivo deste trabalho não é a avaliação da representação de relevância, já que a mesma foi avaliada por Gonçalves *et al.* [41], foi assumido para os protótipos que todos têm a reprentação da relevância mas tal não é avaliada.

O teste foi dividido em duas experiências distintas. Na primeira, o utilizador é exposto às visualizações 1, 2 e 3, e numa segunda às visualizações 4 e 5. Na primeira experiência, o objetivo foi avaliar as técnicas propostas para sinalização *off-screen* e tratamento de

sobreposições, e se melhoram a experiência de navegação do utilizador em ambientes de RA móvel.

Na segunda experiência, o objetivo é perceber se a existência de um Mapa 2D num ambiente de RA melhora, de alguma forma, a experiência de navegação, e qual o impacto da existência de um Radar sobre esse mesmo mapa. As visualizações não foram mostradas de forma sequencial, mas segundo a abordagem *Latin Square*, para garantir a aleatoriedade necessária no teste. A grelha de sequência das visualizações pode ser visualizada no Apêndice C.3.

Ambas as experiências consistiram na execução de uma única tarefa que decorre segundo duas configurações diferentes (número de POI), repetida para cada uma das diferentes visualizações pertencentes a cada uma das experiências. Desta forma, cada utilizador deverá executar $2T * 3V + 2T * 2V = 10$ tarefas.

A cada participante apresentaram-se os objetivos do estudo, indicou-se o tempo estimado de duração dos testes (entre 30 a 40 minutos), fez-se um breve questionário sobre os dados demográficos, que pode ser encontrado no Apêndice B, e pediu-se que o utilizador assinasse um documento de autorização para efetuar a experiência. De seguida, foram explicadas as diferentes funcionalidades do IAR, e efetuada uma breve demonstração usando o mesmo. Depois disso, o utilizador dispunha de um tempo para experimentar a aplicação, com todas as funcionalidades, até este se sentir familiarizado e confortável para dar início ao teste. Posteriormente, cada um dos participantes percorreu as várias visualizações tentando completar da melhor forma possível a tarefa proposta, respondendo a algumas perguntas sobre essas visualizações no final de cada experiência. A tarefa é executada no dispositivo em que o IAR foi desenvolvido, e é neste que as várias métricas (preferências, tempo, ângulo de rotação, número de erros e de acessos ao mapa) são registadas. Por fim, após finalização do teste foi realizado um pós-questionário, presente no mesmo documento apresentado no Apêndice B, com o objetivo de obter uma avaliação final numa escala de 1 a 5 das várias visualizações, perceber a estratégia adotada pelo utilizador ao longo das várias técnicas de visualização e receber críticas e sugestões à aplicação.

4.1.2 Tarefa

A tarefa escolhida para ser executada pelo utilizador em todas as visualizações, teve como objetivo averiguar quais as funcionalidades que são uma mais valia para o utilizador na navegação por um dado espaço de POI. Para tal, na tarefa proposta o objetivo do utilizador seria orientar-se na direção do ponto que achasse ser o mais relevante e simultaneamente o mais próximo, e clicar no mesmo. Se o símbolo clicado era o correto, a tarefa era dada como terminada, caso contrário, o utilizador deveria continuar a tentar até acertar. Em cada experiência, a tarefa é executada primeiramente com 15 POI (primeira configuração) para todas as visualizações, e seguidamente, para todas as visualizações com 30 POI

(segunda configuração). Isto permite perceber se a opinião e desempenho dos utilizadores se mantém com o dobro dos POI iniciais, e consequentemente, perceber se o número de POI tem alguma influência no desempenho dos utilizadores.

Para a tarefa proposta ao longo das visualizações foram medidas as seguintes variáveis dependentes: tempo (em segundos) e ângulo de rotação executado pelo utilizador (em graus), e uma classificação relativa à técnica de visualização de acordo com a sua preferência, numa escala de 1 a 5. As variáveis independentes na primeira são: a técnica de visualização com cinco níveis Vis_i e o número de POI nP com dois níveis: 15 e 30.

4.1.3 Hipóteses

Tendo em conta os vários objetivos deste estudo, assim como a tarefa proposta, foram formuladas as seguintes hipóteses:

1. Espera-se que a técnica de visualização 2 que faz uso das margens do ecrã para sinalizar objetos *off-screen* seja preferida pelos utilizadores relativamente à visualização 1, por fornecer informação sobre objetos fora do campo de visão do utilizador aumentando a sua perceção sobre o que o rodeia, e consequentemente, melhorando a experiência de navegação;

2. É expectável que a técnica de visualização 3 que agrega POI em pilhas seja preferida pelos utilizadores relativamente à visualização 2, uma vez que reduz a quantidade de elementos virtuais no ecrã e evita sobreposições de símbolos sem que haja qualquer perda de informação;

3. É expectável que a técnica de visualização 5 que usa um Radar sobre o Mapa 2D seja preferida pelos utilizadores relativamente a todas as outras visualizações e que aumente o seu desempenho consideravelmente, uma vez que permite uma visualização mais precisa da posição e orientação do utilizador relativamente aos POI;

4. Espera-se que a transição de um número de POI reduzido (15 POI) para um número de POI elevado (30 POI), acentue as diferenças de desempenho em concordância com as hipóteses anteriores quando o utilizador tentar completar uma tarefa.

4.2 Protótipo de Teste

Para satisfazer o planeamento de avaliação anteriormente descrito, foi necessário construír um protótipo de teste. Para a sua construção utilizou-se o último protótipo funcional da aplicação, e fizeram-se algumas adaptações.

Começou-se por adicionar um menu de configuração (Figura 4.1), que permite introduzir o nome do utilizador, escolher a técnica de visualização que a aplicação deverá

adotar e determinar se deverão aparecer 15 ou 30 POI (opção *Increase Radius*). Como era necessário garantir que todos os utilizadores estavam expostos ao mesmo conjunto de dados durante o teste, foi necessário construir um programa auxiliar capaz de gerar 10 ficheiros de dados aleatórios (um para cada tarefa) com POI com diferentes localizações e relevâncias. Assim, ao dar início a uma tarefa o conjunto de dados era escolhido aleatoriamente de entre os que ainda não tinham sido utilizados para o utilizador atual. Também um botão de ”Iniciar Tarefa”foi criado, para que o utilizador clicasse no mesmo quando desse início à tarefa, e se iniciasse a contagem do tempo. Além disso, acrescentou-se um mecanismo de *feedback* que indicava ao utilizador se o símbolo clidado era o mais relevante e mais próximo dando a mensagem ”Resposta correta”, ou ”Resposta errada”, caso contrário. O sistema indica também o momento em que a avaliação está completa com a mensagem ”Teste finalizado”, quando todas as tarefas nas diferentes configurações são concluídas com sucesso para todas as técnicas de visualização.

Adicionalmente, foi necessário preparar o IAR para recolher as várias métricas: tempo, ângulo de rotação, erros e acessos ao mapa. Estas métricas começam a ser capturadas quando o utilizador clica em ”Iniciar Tarefa”e terminam quando o utilizador clica no POI correto. Desta forma, todas as acções do utilizador são registadas e guardadas num ficheiro *xml*, para mais tarde fazer a análise de dados.

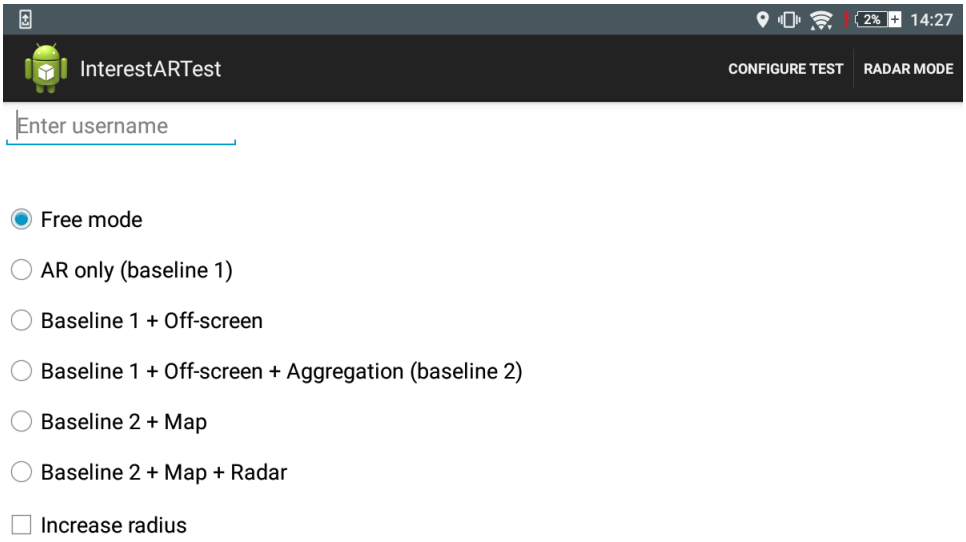


Figura 4.1: Menu de Configuração.

4.3 Estudo com Utilizadores

Esta secção apresenta os resultados do estudo efetuado com os utilizadores. Todos os dados recolhidos, que serão mostrados e analisados nas próximas secções estão represen-

tados sob a forma de tabela no Apêndice C.1 e C.2.

4.3.1 Participantes

Neste teste participaram 18 pessoas, que se apresentaram para este estudo como voluntários sem receber qualquer tipo de compensação monetária. Destas 18 pessoas, 9 eram do género masculino e 9 do género feminino. Existiam 8 utilizadores com idades compreendidas entre os 16 e os 24 anos, 3 utilizadores com idades compreendidas entre os 25 e os 34 anos, 3 utilizadores com idades entre os 35 e os 45 anos, e por fim, 4 utilizadores com mais de 45 anos.

Relativamente às áreas de atuação professional, 8 dos participantes eram da área de informática, 6 da área de gestão/administrativa, 2 da área de saúde, 1 da área de artes e um último ainda a frequentar o ensino secundário. Todos os participantes estavam familiarizados com aplicações de informação georreferenciada, principalmente com o Google Maps, e usam estas aplicações para a localização de POI ou planeamento de percursos. Apenas 15 dos utilizadores estavam habituados a este tipo de aplicações em dispositivos móveis, e maioritariamente, o Google Maps e o MeoDrive foram as aplicações mais referenciadas. Poucos utilizadores referiram dificuldades neste tipo de aplicações, contudo, 4 deles apontaram o tamanho do ecrã como um problema de interação nos dispositivos móveis. Apenas 9 dos participantes estavam familiarizados com técnicas de visualização *off-screen*, principalmente setas e mini-mapa, sendo os jogos o tipo de aplicações mais mencionadas no uso destas técnicas. Por fim, apenas 7 voluntários já haviam experimentado aplicações de RA, sendo o entretenimento a principal razão para o uso deste tipo de aplicações. Destes 7 utilizadores, 2 deles referiram o *tracking* ineficaz e consequente irrealismo como um problema, e 1 dos participantes referiu o consumo elevado de bateria como um impedimento no uso destas aplicações.

4.3.2 Análise de Resultados

Preferências

Após conclusão de ambas as experiências, foi pedido aos utilizadores que classificassem todas as técnicas de visualização de 1 a 5, podendo haver classificações iguais para técnicas de visualização distintas. A Figura 4.2 mostra os resultados obtidos.

Como é observável, a técnica de visualização Vis1 foi a que obteve uma menor classificação (2.0) relativamente à preferência dos utilizadores. A principal razão apontada para isso, foi a necessidade de procurar cegamente devido à inexistência de mecanismos que auxiliassem na procura do mais relevante e mais próximo, como é o caso das barras de sinalização ou do mapa.

A classificação da técnica de visualização Vis2 (3.0) cresce consideravelmente relativamente à primeira, uma vez que, segundo os utilizadores, já existem as barras de

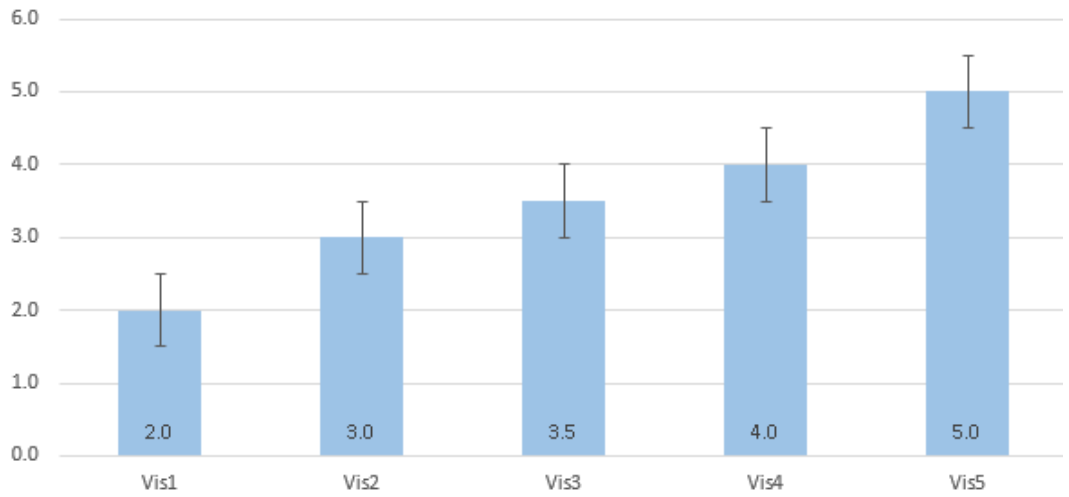


Figura 4.2: Mediana das preferências dos utilizadores, com barras de erro, após conclusão das duas experiências.

sinalização que permitem ter uma melhor perceção sobre o que existe à sua volta, tornando o processo de navegação mais fácil.

A classificação atribuída à técnica de visualização Vis3 (3.5) cresceu ligeiramente relativamente à anterior. Segundo os utilizadores, apesar da funcionalidade de agregação facilitar a leitura de informação não existiu um número suficientemente grande de sobreposições que fizesse com que a mesma se tornasse fundamental.

A avaliação atribuída à técnica de visualização Vis4 (4.0) também teve apenas uma pequena subida relativamente à anterior. Os utilizadores afirmaram que o Mapa os ajuda a orientarem-se e localizarem-se relativamente aos POI, mas que os confunde por não saberem qual a sua orientação atual relativamente ao mesmo.

Por último, a avaliação da técnica de visualização Vis5 (5.0) mostra que esta foi a preferida dos utilizadores. As principais razões salientadas foram a existência do mecanismo de sinalização *off-screen* e tratamento de sobreposições, que permite ter perceção sobre o que existe à volta e nunca ser incomodado com sobreposições no ecrã, e simultaneamente, ter a possibilidade de usar um Mapa com Radar que além de permitir situar o utilizador no terreno relativamente aos POI, permite saber a sua orientação e que POI estão dentro do seu campo de visão num dado instante. Todas essas razões tornaram a navegação mais prática e agradável, segundo o *feedback* dado pelos utilizadores. Assim, estes resultados confirmam as hipóteses 1, 2 e 3 propostas em 4.1.3.

Tempo Médio

Além da análise às preferências dos utilizadores, foi também necessário avaliar várias métricas para verificar se as preferências correspondem, à eficiência demonstrada pelos mesmos em cada técnica de visualização. Os dados foram guardados em ficheiros XML,

como já mencionado anteriormente, e foram posteriormente importados para um ficheiro Excel que permitiu criar alguns gráficos ilustrativos dos dados obtidos.

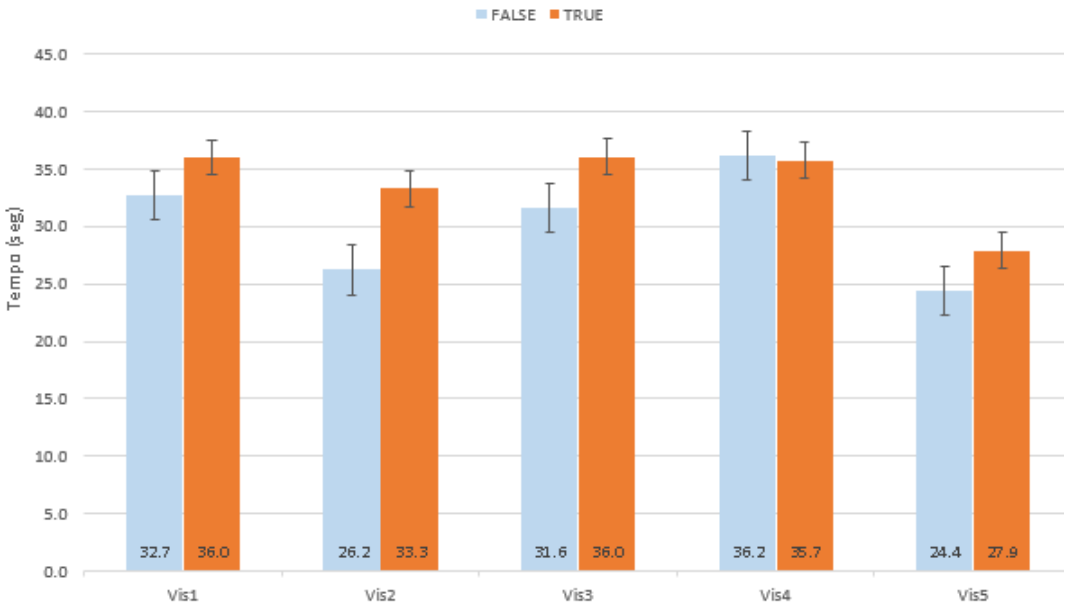


Figura 4.3: Tempo médio, com barras de erro, de execução da tarefa proposta em cada técnica de visualização.

A Figura 4.3 mostra o tempo médio de execução da tarefa proposta ao longo das várias técnicas de visualização, em que as barras azuis (à esquerda) simbolizam a tarefa executada com 15 POI e as laranja (à direita) com 30. Ao olhar para o gráfico, é evidente que a técnica de visualização Vis5 além de ser a preferida dos participantes foi a que permitiu terminar a tarefa em menos tempo. Segundo a opinião expressa pelos utilizadores, é muito fácil encontrar o ponto que está à menor distância no mapa, uma vez que basta comparar a diferença de posições entre o ponto que simboliza a posição atual do utilizador e os vários POI. Os participantes mencionaram que o facto do mapa funcionar como um radar nesta técnica, facilita a escolha na RA do ponto anteriormente visto no Mapa. Os utilizadores afirmaram também que o facto das cores na RA e no Mapa se manterem, são um fator importante na facilidade de uso desta técnica de visualização, caso contrário, teriam que se focar em todos os POI e não apenas nos de cor vermelha (mais relevantes).

As restantes técnicas apresentaram tempos mais elevados, mas que podem ser facilmente justificados. Os participantes reportaram como principal problema da visualização Vis4 a incapacidade de perceber quais os POI que estão dentro do campo de visão num dado instante, o que torna a transição entre o mapa e a RA confusa.

A técnica de visualização Vis1 exige ao utilizador uma procura cega pelos vários POI, o que implica uma necessidade de rodar verticalmente e horizontalmente o dispositivo em zonas onde não existem quaisquer POI, sendo esta a principal dificuldade reportada para esta técnica. No entanto, foi observado durante a experiência que aquando do uso desta

técnica de visualização, os utilizadores rodavam mais rápido, do que nas restantes, por não terem qualquer informação para ler nas barras.

Na técnica de visualização Vis3 houve alguns problemas na análise das pilhas. Alguns utilizadores não perceberam que o tamanho e cor da pilha transmitiam o tamanho e cor do ponto mais relevante e mais próximo de entre os agregados, o que os levou a clicar sucessivamente nas pilhas que apareciam, confusão essa que aumentou o tempo médio para completar a tarefa.

Relativamente à visualização Vis2, que obteve um tempo médio razoável, como observável no gráfico, os utilizadores disseram que a técnica é boa mas que não é comparável com o mapa no que toca à medição de distâncias, pois exige a memorização do tamanho e/ou transparência dos vários símbolos relevantes.

Ângulo de Rotação Médio

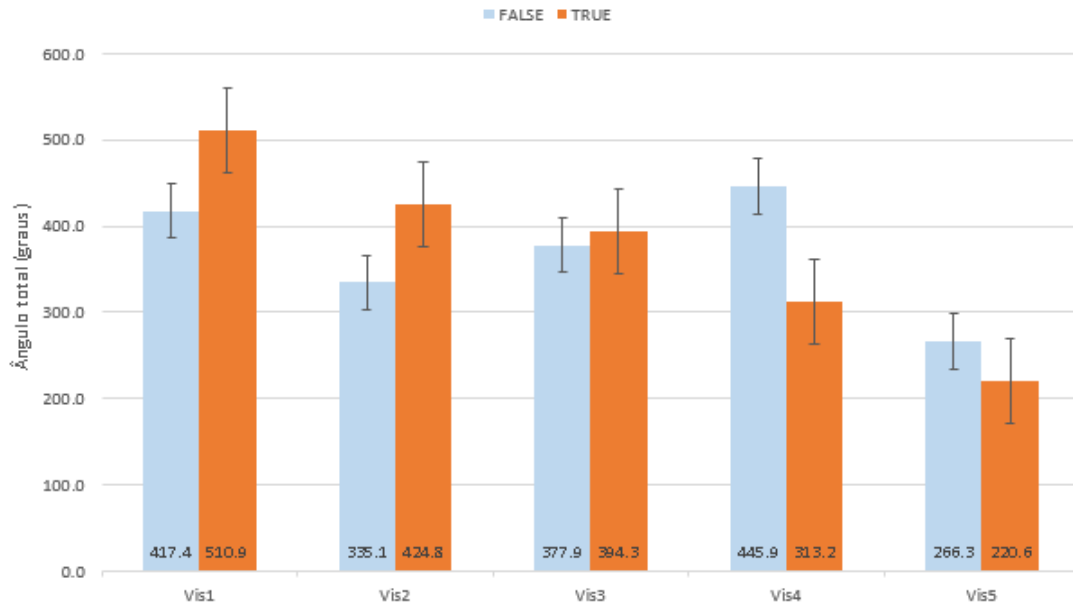


Figura 4.4: Ângulo de rotação médio, com barras de erro, para execução da tarefa proposta em cada técnica de visualização.

A Figura 4.4 mostra o ângulo médio de rotação durante a execução da tarefa proposta ao longo das várias técnicas de visualização, em que as barras azuis simbolizam a tarefa executada com 15 POI e a laranja com 30. Desde logo é percetível que a visualização Vis5, foi claramente aquela que exigiu um menor ângulo de rotação até encontrar o ponto pretendido. De acordo com a opinião dos utilizadores, com esta técnica de visualização é possível encontrar o ponto apenas olhando para o Radar, executando uma única rotação, e voltando à RA, o que culminou no resultado apresentado no gráfico.

Já a visualização que exigiu um maior ângulo de rotação foi a visualização Vis1. Mais uma vez, os utilizadores relataram que a procura cega é um fator problemático.

A visualização Vis4 possui um desfasamento elevado entre a experiência com 15 POI e com 30. Associa-se isto ao facto de terem existido alguns utilizadores mais desatentos (durante a experiência com 15 POI) que se confundiram devido à incapacidade do Mapa 2D para representar alturas. Ou seja, os utilizadores alternavam para o Modo Mapa 2D, orientavam-se na direção do ponto alvo corretamente, mas ao alternarem para o Modo RA para clicarem no mesmo, clicavam incorretamente pois o ponto estava *off-screen top* e não se aperceberam do simbolo desenhado na barra superior. Como a experiência com 15 POIs ocorreu sempre antes da experiência com 30, os utilizadores corrigiram o seu comportamento na segunda.

A visualização Vis3 e Vis4 estiveram bastante equilibradas. Os utilizadores consideraram que são bastante semelhantes em termos de facilidade de uso e que são bastante melhores que a visualização Vis1 mas inferiores às visualizações em que o Mapa 2D está disponível.

Erro Médio

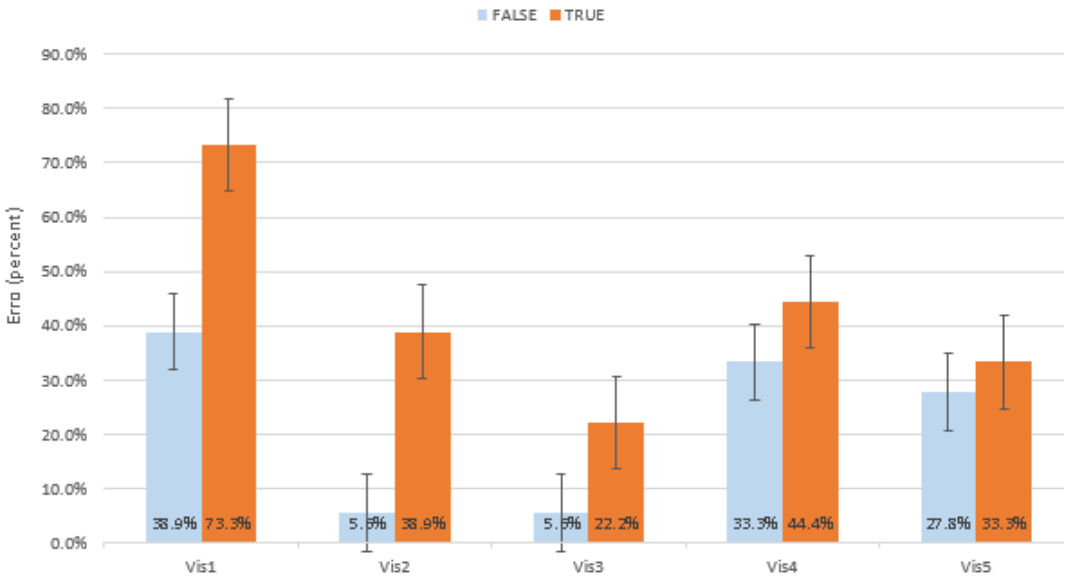


Figura 4.5: Erro médio, com barras de erro, para execução da tarefa proposta em cada técnica de visualização.

A Figura 4.5 mostra o erro médio de execução da tarefa proposta ao longo das várias técnicas de visualização, em que as barras azuis simbolizam a tarefa executada com 15 POI e a laranja com 30. A técnica de visualização Vis1, foi claramente a pior relativamente à quantidade de erros. Segundo o que foi dito pelos participantes, enquanto que em todas as outras técnicas havia informação a ser analisada que ditava o comportamento do utilizador, nesta abordagem o utilizador não tem ajudas e tende a clicar de forma não determinístíca nos POI na expectativa de que aquele seja o correto (mesmo sem ter visto os restantes).

As técnicas de visualização Vis4 e Vis5, geraram mais erros do que o esperado. Tal facto é associado à transição, ainda pouco natural, entre o Mapa 2D e o RA. Como já dito, a transição entre uma vista 2D e uma vista 3D confunde, por vezes, o utilizador menos experiente. Segundo relatado pelos participantes, é muito fácil confiar em demasia no Mapa por ser algo ao qual estão muito habituados, e no entanto, quando voltam à RA o que foi visto anteriormente é representado de forma diferente, o que em ocasiões em que a altura é um fator determinante, pode provocar erros.

Finalmente, as técnicas de visualização Vis2 e Vis3 foram mais eficazes, em especial esta última, pois são as visualizações em que o utilizador tem indicações do que está à sua volta numa só representação. A presença de pilhas é um fator que contribui, inequívocamente, para a redução do erro uma vez que o utilizador pode comparar pilhas entre si e não POI isolados, o que diminui a probabilidade de fazer uma comparação de tamanhos e transparências erradamente.

De acordo com os gráficos apresentados, a maioria das métricas recolhidas (tempo, ângulo de rotação e erros) demonstram piores resultados para 30 POI do que para 15. No entanto, existem três situações em que tal não acontece, mais especificamente, na técnica de visualização Vis4 nas Figuras 4.3 e 4.4, e na técnica de visualização Vis5 na Figura 4.4. Por este motivo, nada se pode concluir em relação à hipótese 4. Isto poderá ser justificado pelo facto da experiência com 15 POI ser sempre efectuada antes da experiência com 30 POI. É também possível que 30 POI não seja um número grande suficiente para acentuar as diferenças de desempenho entre as diferentes técnicas de visualização.

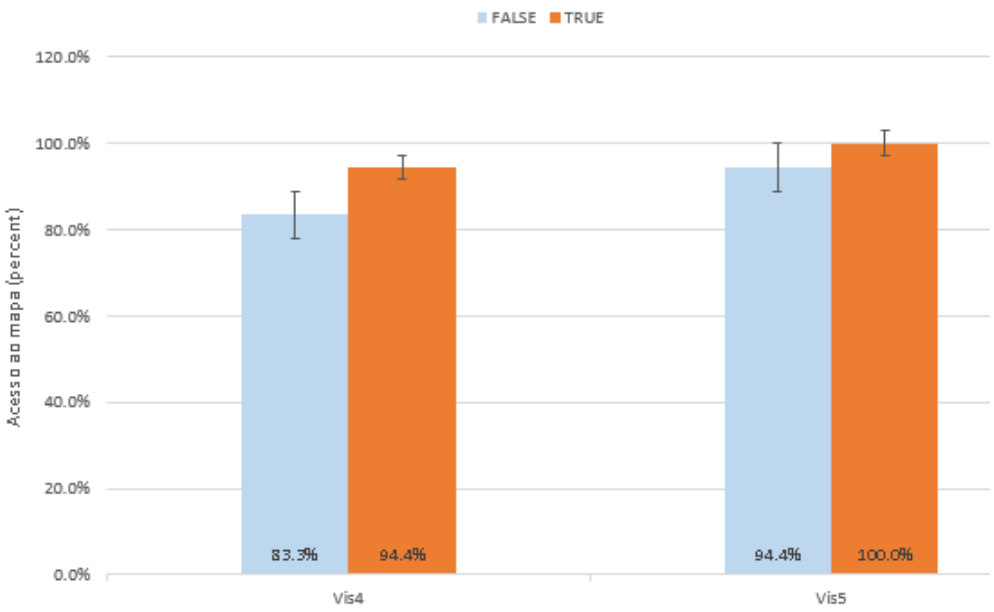


Figura 4.6: Média de acessos ao Mapa 2D, com barras de erro, para execução da tarefa proposta em cada técnica de visualização.

Por último, achou-se que seria importante perceber a quantidade de participantes que

recorreu ao modo de visualização em Mapa 2D. Como demonstrado na Figura 4.6, nas técnicas de visualização Vis4 e Vis5 praticamente todos os utilizadores recorreram ao Mapa 2D, principalmente na experiência com 30 POI.

Opinião dos Utilizadores

Ao longo do estudo, foi sempre dada a oportunidade aos utilizadores de expressarem a sua opinião, tanto de forma oral como por escrito no pós-questionário. Esta secção relata os pontos mais sublinhados pela maioria dos utilizadores.

Um dos principais problemas identificados pelos utilizadores ao longo do estudo efetuado, foi a difícil comparação de tamanhos e transparências dos diferentes símbolos. Os participantes afirmaram que por um lado, as diferenças nas transparências e tamanhos entre objetos a distâncias semelhantes é pouco percetível, e por outro, que quando os símbolos a serem comparados não estão em simultâneo na área visível no ecrã, o processo de comparação torna-se demasiado complicado. Inclusivamente no Mapa 2D, alguns utilizadores afirmaram ser complicado perceber qual o POI mais próximo.

Outro problema referido foi a desorientação causada durante a transição entre os modos RA e Mapa 2D. Alguns utilizadores mostraram-se confusos durante a transição pelo facto da RA representar o mundo tridimensionalmente, e o modo Mapa bidimensionalmente. Ou seja, no modo RA os símbolos na parte superior do ecrã representam os pontos mais altos, enquanto que no modo Mapa 2D representam os pontos mais distantes. Esta diferença origina erros de perceção em alguns utilizadores menos familiarizados com este tipo de aplicações, e consequentemente, piores resultados. Todavia, os utilizadores que não mostraram dificuldades em transitar entre o modo RA e Mapa 2D afirmaram que a combinação entre estes era uma mais valia para a concretização da tarefa proposta.

A espera de 7 segundos até que uma pilha clicada se voltasse a agregar não foi consensual. Alguns utilizadores defenderam que não deveriam ser obrigados a esperar, e que a desagregação se deveria manter enquanto o dedo se mantivesse no ecrã, contrariamente ao que foi implementado. Ainda neste contexto, alguns participantes disseram que ao largar uma mão do dispositivo para clicar numa pilha, era criada alguma instabilidade no ambiente de RA.

De forma geral, os utilizadores consideraram a aplicação útil no contexto da pesquisa de POI em ambientes de RA móvel, uma vez que, segundo os mesmos, os ajuda a encontrar os POI pretendidos de forma mais rápida e eficaz. Consideraram especialmente inovador e importante a combinação entre RA e Mapa 2D.

4.3.3 Sumário e Discussão

Neste estudo, observou-se que a sinalização de objetos *off-screen* melhora a experiência de navegação do utilizador em 3 fatores: tempo, movimento e eficácia. Sempre que existiram indicações para representação de objetos fora da área visível (técnicas de visualização

Vis2, Vis3, Vis4 e Vis5) os resultados foram melhores. Os utilizadores demoraram menos tempo a encontrar o ponto pretendido, rodaram menos e cometeram menos erros. Portanto, a nível de sinalização *off-screen* pode afirmar-se que o IAR cumpriu o esperado, auxiliando o utilizador durante a navegação pelo espaço de POI.

A representação da relevância era algo que não se pretendia avaliar, uma vez que já havia sido estudado e avaliado por Gonçalves [42]. Uma vez que os resultados desta técnica já estavam comprovados na utilização em Mapas 2D, o que se fez foi transportar esta abordagem para um cenário de RA, que segundo a opinião dos utilizadores, foi bem conseguido.

O mecanismo de tratamento de sobreposições não mostrou ser vantajoso em termos de diminuição de tempo ou de rotação necessária até encontrar o ponto pretendido. Contudo, mostrou baixar consideravelmente a taxa de erro dos participantes quando o número de POI aumenta, uma vez que ao evitar sobreposições, apresenta a informação de forma clara e legível. Em simultâneo, permite ao utilizador saber se a pilha lhe interessa, apenas observando o símbolo, sem a necessidade de clicar, algo que nem sempre foi percebido pelos utilizadores.

Deste estudo, pode ainda concluir-se que a junção do Mapa com a RA mostrou ser uma mais valia, na medida em que melhorou consideravelmente a eficácia da pesquisa de POI com relevância para o utilizador. Esta junção torna-se particularmente valiosa, quando existe um Radar sobre o Mapa. A existência de um raio que delimita a visão do utilizador, permite uma transição mais intuitiva entre a RA e o Mapa na medida em que o raio de visão ilustrado no Mapa corresponde à zona visível do ecrã no modo de RA. É esta área (comum a ambas as vistas) que torna mais fácil a transição entre uma vista 3D e uma outra 2D, apesar de não ter sido o suficiente para evitar enganos dos utilizadores, e consequentes erros, aquando da transição. Neste estudo, os participantes não puderam tirar o máximo partido da RA e do Mapa, uma vez que a localização do utilizador foi simulada de maneira a que todos estivessem nas mesmas condições. No entanto, acredita-se que o Mapa poderá ser extremamente importante para o utilizador se situar quando não sabe exatamente onde está, ou quando vê o símbolo do POI representado mas ainda não o vê no mundo físico por estar oculto por outros edifícios ou ainda demasiado longe.

Em suma, as técnicas propostas pelo IAR para sinalização *off-screen*, representação de relevância e tratamento de sobreposições tornaram a navegação dos utilizadores mais simples e eficiente. Os resultados correspondem às preferências dos utilizadores, com a técnica de visualização 5 a ser a preferida e a que obteve melhores resultados durante a utilização da aplicação, enquanto que a técnica de visualização 1 foi a que obteve pior classificação por parte dos utilizadores e a que mostrou resultados mais ineficazes.

Capítulo 5

Conclusões e Trabalho Futuro

Neste capítulo são apresentadas as conclusões finais sobre o trabalho realizado, os respectivos resultados e o possível trabalho futuro.

5.1 Conclusões

Devido ao crescimento significativo do uso dos dispositivos móveis, e com a integração de sensores de posição e orientação, e com câmaras digitais de elevada qualidade de imagem, reuniram-se as condições para desenvolver aplicações de RA para estes dispositivos. No entanto, o ecrã pequeno neste tipo de dispositivos impõe restrições aquando do uso deste tipo de sistemas. Este trabalho, desenvolveu o IAR, uma aplicação móvel para Android que permite explorar técnicas de pesquisa e visualização de informação geo-referenciada.

Os principais problemas que o IAR se propunha resolver eram sinalizar os POI que estivessem fora da área visível no ecrã, representar a relevância de diferentes POI (*off-screen* ou *on-screen*) de acordo com as preferências do utilizador, e ainda, tratar sobreposições de diferentes símbolos. Para resolver o problema da sinalização *off-screen*, o IAR faz o uso de uma moldura onde símbolos gráficos representativos de pontos fora do campo de visão do utilizador. A representação da relevância é feita através da cor e transparência atribuída a cada símbolo, com base na ideia aplicada por Gonçalves [42] na representação da relevância em Mapas 2D. Por fim, o tratamento de sobreposições é efetuado através de uma pilha representativa do conjunto de pontos agregado, que adquire a forma, cor, e transparência do POI mais próximo e relevante. Além disto, existe um Mapa 2D com Radar como modo alternativo, de forma a auxiliar o utilizador a situar-se relativamente aos POI.

Segundo a avaliação efetuada, os resultados foram bastante positivos, e comprovaram a utilidade das técnicas aqui desenvolvidas. A técnica de visualização que juntava o Mapa 2D com Radar e a RA obteve ótimos resultados, apesar da transição entre ambos nem sempre ter sido clara para os participantes. A técnica de sinalização *off-screen* desenvolvida foi bem interpretada e mostrou ser importante no aumento da perceção do

utilizador relativamente ao que o rodeia, e por fim, o tratamento de sobreposições mostrou reduzir bastante a taxa de erro dos utilizadores.

5.2 Trabalho Futuro

Após conclusão do trabalho desenvolvido, existem ainda vários pontos que poderão ser melhorados. Este projeto poderá ser uma base para tais melhorias, de forma a que esta plataforma venha a proporcionar uma experiência de navegação mais confortável e completa em ambientes de RA móvel.

Como já referido anteriormente, um dos principais problemas apontados pelos participantes é a difícil distinção dos tamanhos e cores entre símbolos de relevância e distância semelhantes, em particular, quando esses pontos não se encontram na área visível em simultâneo. Como tal, será interessante tentar explorar outras variáveis visuais que permitam identificar os objetos mais próximos, e novos mecanismos que facilitem a comparação entre símbolos representativos de pontos que estão em direções opostas.

Outro problema mencionado é a desorientação causada durante a transição entre os modos RA e Mapa 2D, uma vez que a RA funciona tridimensionalmente e o Mapa bidimensionalmente. Uma solução interessante, poderá ser a junção dos dois modos numa única vista. Dessa forma, não seria necessário transitar entre modos, o que facilitaria a comparação entre ambos durante a navegação. No Mapa 2D, houve ainda quem tivesse dificuldades em saber qual o ponto que estava mais perto, quando as distâncias eram muito próximas. Um mecanismo que poderá tornar este processo mais simples, será adicionar ao Radar círculos de diferentes raios representativos de diferentes níveis de distância. Ainda relativamente ao Mapa 2D, alguns participantes sugeriram o acrescento de mais informações textuais ao clicar no POI (ex: distância).

A questão da espera dos 7 segundos até a agregação dos símbolos desagregados, é outro fator a melhorar assim como a necessidade de tirar uma das mãos do *tablet* para clicar no ecrã. Seria interessante explorar novas formas de interação do utilizador com a aplicação em que este pudesse manter as mãos no dispositivo.

Por fim, é importante referir que este projeto é uma prova de conceito, que se mostrou útil no apoio durante a navegação num espaço de POI dos utilizadores em ambientes de RA móvel. Uma vez que foi implementado em Android, será interessante explorar as ideias e fundamentos do IAR com o uso de *frameworks* próprias para construção de ambientes de RA (ex: Metaio, Vuforia).

Bibliografia

[1] Android location. Retirado de http://developer.android.com/reference/android/location/package-summary.html. (acedido em 02-10-2015).

[2] Android studio and sdk tools. Retirado de http://developer.android.com/sdk/index.html. (acedido em 03-10-2015).

[3] Augmented reality layar. Retirado de http://www.layar.com/. (acedido em 12-10-2015).

[4] Canvas documentation. Retirado de http://developer.android.com/reference/android/graphics/Canvas.html. (acedido em 03-10-2015).

[5] Counting and measurement. Retirado de http://pics-about.space/altitude-vs-azimuth-astronomy. (acedido em 03-10-2015).

[6] Eclipse foundation open source community website. Retirado de https://www.eclipse.org/home/index.php. (acedido em 03-10-2015).

[7] Java se 7 documentation. Retirado de http://docs.oracle.com/javase/7/docs/api/. (acedido em 03-10-2015).

[8] Mobile augmented reality firms seeking brands and consumers. Retirado de http://www.theguardian.com/technology/appsblog/2011/jun/17/augmented-reality-apps. (acedido em 04-10-2015).

[9] Six degrees of freedom. Retirado de http://batttech.weebly.com/robots-and-artificial-intelligence.html. (acedido em 03-10-2015).

[10] Understanding latitude and longitude. Retirado de https://www.learner.org/jnorth/tm/LongitudeIntro.html. (acedido em 03-10-2015).

[11] Wikitude, the world's leading augmented reality sdk. Retirado de http://www.wikitude.com/. (acedido em 12-10-2015).

[12] Anthony J Aretz and Christopher D Wickens. The mental rotation of map displays. *Human Performance*, 5(4):303–328, 1992.

[13] Ronald T Azuma. A survey of augmented reality. *Presence*, 6(4):355–385, 1997.

[14] P Baudisch and R Rosenholtz. Halo: a technique for visualizing off-screen location. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'03)*, pages 418–488, 2003.

[15] Oliver Bimber and Ramesh Raskar. *Spatial augmented reality: merging real and virtual worlds*. CRC Press, 2005.

[16] Oliver Bimber and Ramesh Raskar. Modern approaches to augmented reality. In *ACM SIGGRAPH 2006 Courses*, page 1. ACM, 2006.

[17] Gabriele Bleser. *Towards visual-inertial slam for mobile augmented reality*. PhD thesis, Verlag Dr. Hut, 2009.

[18] Erkan Bostanci, Nadia Kanwal, Shoaib Ehsan, and Adrian F Clark. User tracking methods for augmented reality. *International Journal of Computer Theory and Engineering*, 5(1):93–98, 2013.

[19] Stefano Burigat and Luca Chittaro. Navigation in 3d virtual environments: Effects of user experience and location-pointing navigation aids. *International Journal of Human-Computer Studies*, 65(11):945–958, 2007.

[20] Stefano Burigat and Luca Chittaro. Visualizing references to off-screen content on mobile devices: A comparison of arrows, wedge, and overview+ detail. *Interacting with Computers*, 23(2):156–166, 2011.

[21] Stefano Burigat and Luca Chittaro. On the effectiveness of overview+ detail visualization on mobile devices. *Personal and ubiquitous computing*, 17(2):371–385, 2013.

[22] Stefano Burigat, Luca Chittaro, and Silvia Gabrielli. Visualizing locations of off-screen objects on mobile devices: a comparative evaluation of three approaches. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 239–246. ACM, 2006.

[23] Stefano Burigat, Luca Chittaro, and Silvia Gabrielli. Navigation techniques for small-screen devices: An evaluation on maps and web pages. *International Journal of Human-Computer Studies*, 66(2):78–97, 2008.

[24] Laura A Carlson-Radvansky and David E Irwin. Frames of reference in vision and language: Where is above? *Cognition*, 46(3):223–244, 1993.

[25] Julie Carmigniani, Borko Furht, Marco Anisetti, Paolo Ceravolo, Ernesto Damiani, and Misa Ivkovic. Augmented reality technologies, systems and applications. *Multimedia Tools and Applications*, 51(1):341–377, 2011.

[26] Maria Beatriz Carmo, Ana Paula Cláudio, António Ferreira, Ana Paula Afonso, and Raúl Simplício. Improving symbol salience in augmented reality. In *GRAPP*, pages 367–372, 2013.

[27] Luca Chittaro. Visualizing information on mobile devices. *Computer*, 39(3):40–45, 2006.

[28] Luca Chittaro and Stefano Burigat. 3d location-pointing as a navigation aid in virtual environments. In *Proceedings of the working conference on Advanced visual interfaces*, pages 267–274. ACM, 2004.

[29] Andy Cockburn, Amy Karlson, and Benjamin B Bederson. A review of overview+ detail, zooming, and focus+ context interfaces. *ACM Computing Surveys (CSUR)*, 41(1):2, 2008.

[30] Andy Cockburn and Joshua Savage. Comparing speed-dependent automatic zooming with traditional scroll, pan and zoom methods. In *People and Computers XVII—Designing for Society*, pages 87–102. Springer, 2004.

[31] Andrew I Comport, Éric Marchand, and François Chaumette. A real-time tracker for markerless augmented reality. In *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 36. IEEE Computer Society, 2003.

[32] Inger Ekman and Petri Lankoski. What should it do?: key isssues in navigation interface design for small screen devices. In *CHI'02 Extended Abstracts on Human Factors in Computing Systems*, pages 622–623. ACM, 2002.

[33] Tom Emrich. Shaping the future of technology with slam. Retirado de `http://www.wikitude.com/blog-shaping-future-technology-slam/`, 5 2015. (acedido em 03-10-2015).

[34] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 590–596. IEEE, 2005.

[35] Pascal Fua and Vincent Lepetit. Vision based 3d tracking and pose estimation for mixed reality. *Emerging Technologies of Augmented Reality Interfaces and Design*, pages 43–63, 2005.

[36] George W Furnas and Benjamin B Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241. ACM Press/Addison-Wesley Publishing Co., 1995.

[37] Joseph L Gabbard, II J Edward Swan, and Deborah Hix. The effects of text drawing styles, background textures, and natural lighting on text legibility in outdoor augmented reality. *Presence: Teleoperators and Virtual Environments*, 15(1):16–32, 2006.

[38] Joseph L Gabbard and J Edward Swan. Usability engineering for augmented reality: Employing user-based studies to inform design. *Visualization and Computer Graphics, IEEE Transactions on*, 14(3):513–525, 2008.

[39] Ana Paula Afonso Ana Paula Cláudio António Ferreira Gonçalo Silva, Maria Beatriz Carmo. Visualização de objetos off-screen em realidade aumentada móvel. *2015*, pages 71–78, Nov 2015.

[40] Tiago Gonçalves, Ana Paula Afonso, Maria Beatriz Carmo, and Paulo Pombinho. Halodot: Visualization of the relevance of off-screen objects, 2011.

[41] Tiago Gonçalves, Ana Paula Afonso, Maria Beatriz Carmo, and Paulo Pombinho. Comparison of off-screen visualization techniques with representation of relevance on mobile devices. In *Proceedings of the 27th International BCS Human Computer Interaction Conference*, page 9. British Computer Society, 2013.

[42] Tiago José Lopes Gonçalves. Visualização da relevância da informação geográfica em aplicações móveis. Master's thesis, Department of Informatics, University of Lisbon, 2011.

[43] Sean G Gustafson and Pourang P Irani. Comparing visualizations for tracking off-screen moving targets. In *CHI'07 Extended Abstracts on Human Factors in Computing Systems*, pages 2399–2404. ACM, 2007.

[44] Niels Henze and Susanne Boll. Evaluation of an off-screen visualization for magic lens and dynamic peephole interfaces. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 191–194. ACM, 2010.

[45] Tobias Höllerer and Steve Feiner. Mobile augmented reality. *Telegeoinformatics: Location-Based Computing and Services. Taylor and Francis Books Ltd., London, UK*, 21, 2004.

[46] Tobias Höllerer, Steven Feiner, Tachio Terauchi, Gus Rashid, and Drexel Hallaway. Exploring mars: developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers & Graphics*, 23(6):779–785, 1999.

[47] Jacek Jankowski and Martin Hachet. Advances in interaction with 3d environments. In *Computer Graphics Forum*, volume 34, pages 152–190. Wiley Online Library, 2015.

[48] Hyungeun Jo, Sungjae Hwang, Hyunwoo Park, and Jung-hee Ryu. Aroundplot: Focus+ context interface for off-screen objects in 3d environments. *Computers & Graphics*, 35(4):841–853, 2011.

[49] Peter Lang, Miguel Ribo, and Axel Pinz. *A new combination of vision-based and inertial tracking for fully mobile, wearable, and real-time operation*. Citeseer, 2002.

[50] Jock Mackinlay. Automating the design of graphical presentations of relational information. *Acm Transactions On Graphics (Tog)*, 5(2):110–141, 1986.

[51] David Marimon, Cristina Sarasua, Paula Carrasco, Roberto Álvarez, Javier Montesa, Tomasz Adamek, Idoia Romero, Mario Ortega, and Pablo Gascó. Mobiar: Tourist experiences through mobile augmented reality. *Telefonica Research and Development, Barcelona, Spain*, 2010.

[52] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. In *Photonics for Industrial Applications*, pages 282–292. International Society for Optics and Photonics, 1995.

[53] Edgar José de Sousa Montez. Visualização de pontos de interesse em dispositivos móveis usando realidade aumentada. 2012.

[54] Heiko Müller, Andreas Löcken, Wilko Heuten, and Susanne Boll. Sparkle: an ambient light display for dynamic off-screen points of interest. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*, pages 51–60. ACM, 2014.

[55] Alessandro Mulloni, Andreas Dünser, and Dieter Schmalstieg. Zooming interfaces for augmented reality browsers. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 161–170. ACM, 2010.

[56] Dmitry Nekrasovski. A comparison of pan and zoom and rubber sheet navigation. Master's thesis, University Of British Columbia, 2006.

[57] Rui Nóbrega, Diogo Cabral, Giulio Jacucci, and António Coelho. Nari: Natural augmented reality interface - interaction challenges for ar applications. pages 504–510, 2015.

[58] SK Ong, ML Yuan, and AYC Nee. Augmented reality applications in manufacturing: a survey. *International journal of production research*, 46(10):2707–2742, 2008.

[59] Harry E Pence. Smartphones, smart objects, and augmented reality. *The Reference Librarian*, 52(1-2):136–145, 2010.

[60] Wayne Piekarski and Bruce Thomas. Arquake: the outdoor augmented reality gaming system. *Communications of the ACM*, 45(1):36–38, 2002.

[61] Catherine Plaisant, David Carr, and Ben Shneiderman. Image-browser taxonomy and guidelines for designers. *Software, IEEE*, 12(2):21–32, 1995.

[62] Paulo Pombinho. Visualizaçao de informaçao geo-referenciada em dispositivos móveis. Master's thesis, Department of Informatics, University of Lisbon, 2008.

[63] Paulo Pombinho, Maria Beatriz Carmo, and Ana Paula Afonso. Evaluation of overcluttering prevention techniques for mobile devices. In *Information Visualisation, 2009 13th International Conference*, pages 127–134. IEEE, 2009.

[64] Tumasch Reichenbacher. *The concept of relevance in mobile maps*. Springer, 2007.

[65] Ronald J Reisman, Steven K Feiner, and David M Brown. Augmented reality tower technology flight test. pages 246–252, 2014.

[66] Gerhard Reitmayr and Tom W Drummond. Going out: robust model-based tracking for outdoor augmented reality. In *Mixed and Augmented Reality, 2006. ISMAR 2006. IEEE/ACM International Symposium on*, pages 109–118. IEEE, 2006.

[67] Ruth Rosenholtz, Yuanzhen Li, and Lisa Nakano. Measuring visual clutter. *Journal of vision*, 7(2):17, 2007.

[68] Torben Schinke, Niels Henze, and Susanne Boll. Visualization of off-screen objects in mobile augmented reality. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 313–316. ACM, 2010.

[69] P Silva, P Pombinho, A Afonso, and T Gonçalves. Rubi: An open source android platform for mobile augmented reality application. In *MAR Workshop, MobileHCI'11*, 2011.

[70] Samuel Silva, Beatriz Sousa Santos, and Joaquim Madeira. Using color in visualization: A survey. *Computers & Graphics*, 35(2):320–333, 2011.

[71] Teresa Siu and Valeria Herskovic. Sidebars: Improving awareness of off-screen elements in mobile augmented reality. In *Proceedings of the 2013 Chilean Conference on Human-Computer Interaction*, pages 36–41. ACM, 2013.

[72] Olivier Swienty, Tumasch Reichenbacher, Simone Reppermund, and Joseph Zihl. The role of relevance and cognition in attention-guiding geovisualisation. *The Cartographic Journal*, 45(3):227–238, 2008.

[73] Veronica Teichrieb, Joao Paulo Silva do Monte Lima, Eduardo Lourenço Apolinário, Thiago Souto Maior Cordeiro de Farias, Márcio Augusto Silva Bueno, Judith Kelner, and Ismael HF Santos. A survey of online monocular markerless augmented reality. *International Journal of Modeling and Simulation for the Petroleum Industry*, 1(1), 2007.

[74] Nathan Totura. Using sensors and location data for cutting-edge user experiences in mobile applications. Retirado de https://software.intel.com/en-us/articles/using-sensors-and-location-data-for-cutting-edge-user-experiences-in-mobile-applications, 10 2012. (acedido em 03-10-2015).

[75] DWF Van Krevelen and R Poelman. A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality*, 9(2):1, 2010.

[76] Jeremy M Wolfe and Todd S Horowitz. What attributes guide the deployment of visual attention and how do they do it? *Nature Reviews Neuroscience*, 5(6):495–501, 2004.

[77] Suya You and Ulrich Neumann. Fusion of vision and gyro tracking for robust augmented reality registration. In *Virtual Reality, 2001. Proceedings. IEEE*, pages 71–78. IEEE, 2001.

[78] Polle T Zellweger, Jock D Mackinlay, Lance Good, Mark Stefik, and Patrick Baudisch. City lights: contextual views in minimal space. In *CHI'03 extended abstracts on Human factors in computing systems*, pages 838–839. ACM, 2003.

[79] Bin Zhu and Hsinchun Chen. Information visualization. *Annual review of information science and technology*, 39(1):139–177, 2005.

Apêndice A

Manual Técnico

A.1 Algoritmo que define a posição de um POI no ecrã

- Percorre-se a lista de POI
  - Verifica-se se o ponto está dentro do raio de pesquisa, comparando o valor obtido pelo método *distanceTo(...)* com o raio previamente definido, e guarda-se essa distância. Este método recebe a localização do utilizador e do ponto e retorna a distância entre ambos.
  - Atualizam-se os valores de *yaw* e *pitch* do ponto realtivamente ao utilizador, através dos cálculos demonstrados nas funções 3.2, 3.3, 3.4;
  - Obtém-se o *bearing* com o método *bearingTo(...)* e a inclinação com a função 3.3;
  - Com esses valores, determina-se os limites do sector do mundo a que o ponto pertence. Por exemplo, se o *bearing* = 12º e inclinação = 60º, então os limites da fatia serão (9,18) para o primeiro, e (54,63) para o último;
  - O POI é colocado no mapa Map<String, PointOfInterest>, que representa as posições dos diferentes POI no mundo, com a chave "9-18-54-63".Poderá já existir um ponto guardado para esta chave, e nesse caso será necessário tratar a sobreposição, mas isto será explicado detalhadamente posteriormente.
- Percorre-se o mapa preeenchido no ciclo anterior
  - Determina-se se o ponto é *on-screen* ou *off-screen*, e no caso de ser *off-screen* se é *left*, *right*, *top*, *bottom* ou se deverá ser colocado num dos cantos. Este cálculo é executado com a função 3.5;
  - Determina-se qual a região visível no ecrã. Para isso, obtêm-se o *azimuth* da direção de observação do utilizador e sabe-se que o campo de visão atual do utilizador será $[azimuth - 45, azimuth + 45]^{o}$;

- Se for *on-screen*
  - Acede-se a uma *lookup table* construída inicialmente do tipo *Map<String, ArrayList<Double>>* para obter o $x_{max}, x_{min}, y_{max}, y_{min}$ que o elemento virtual deverá ocupar no ecrã consoante a célula da grelha a que pertence.
- Se for *off-screen left* ou *right*
  - Acede-se à mesma *lookup table* e consulta-se apenas o y_{max}, y_{min} que o elemento virtual deverá ocupar, pois o x sabe-se que deverá fazer o símbolo ocupar a borda correspondente (esquerda ou direita) e no intervalo certo consoante o *yaw* necessário para que este entre na área visível do ecrã.
- Se for *off-screen top* ou *bottom*
  - Acede-se à mesma *lookup table* e consulta-se apenas o x_{max}, x_{min} que o elemento virtual deverá ocupar, pois o y sabe-se que deverá fazer o símbolo ocupar a borda correspondente (cima ou baixo) e no intervalo certo consoante o *pitch* necessário para que este entre na área visível do ecrã.
- Se estiver fora do ecrã verticalmente e horizontalmente, ou seja, se deve aparecer num dos cantos do ecrã
  - Sabe-se que o símbolo deverá ocupar o canto correspondente e no intervalo certo consoante o *yaw* e *pitch* necessários para que este entre na área visível do ecrã.

A.2 Algoritmo para tratamento de sobreposições

- Percorre-se a lista de POI
  - Obtém-se o *bearing* com o método *bearingTo(...)* e a inclinação do POI através da função 3.3;
  - Com esses valores, determinam-se os limites do sector do mundo a que o ponto pertence. Por exemplo, se o *bearing* = 12º e inclinação = 60º, então os limites da fatia serão (9,18) na horizontal para o primeiro, e (54,63) na vertical para o último;
  - O POI é colocado no mapa Map<String, PointOfInterest>, que representa as posições dos diferentes POI no mundo, com a chave "9-18-54-63";
  - Se já existe um ponto (B) com essa chave
    - Obtém-se o ponto mais relevante, ou no caso dos dois POI possuirem a mesma relevância, o que está mais perto do utilizador;

    - Se o ponto obtido (A) é diferente do que está atualmente no mapa (B)
        - Adiciona-se à *Stack* do POI A todos os POI presentes na *Stack* do ponto B, mais o próprio POI B. A pilha do POI B é eliminada;
        - O mapa é atualizado, e o novo ponto mais relevante e mais próximo substituirá o antigo.
    - Se o ponto obtido (A) é igual ao que está atualmente no mapa (A)
        - Adiciona-se à *Stack* do ponto A o ponto que foi com ele comparado;

A.3 Algoritmo que desenha o POI mais relevante e próximo de cada secção do mundo na posição correta do ecrã

- Definir quais as fatias horizontais e verticais dentro do campo de visão do utilizador no momento, dadas por: $[azimuth-45, azimuth+45]$ e $[inclinacaoDispositivo-45, inclinacaoDispositivo+45]$;

- Percorre-se o mapa criado na segunda fase, que tem associado o ponto mais relevante e mais próximo a cada fatia do mundo

    - Se após o cálculo do *yaw* e *pitch* obtidos pelas funções 3.2, 3.3 e 3.4 e usando a função 3.5 se determina que o ponto é totalmente *off-screen*, então determina-se em que coluna ou linha da respetiva barra de sinalização deve ser colocado a partir desses valores.
    - Se após o cálculo do *yaw* e *pitch* obtidos pelas funções 3.2, 3.3 e 3.4 e usando a função 3.5 se determina que o ponto é totalmente *on-screen*, obtém-se o mapeamento entre este sector e o ecrã através do mapa criado na primeira fase, e guardam-se no ponto as coordenadas do ecrã onde irá ser colocado.
    - Se após o cálculo do *yaw* e *pitch* obtidos pelas funções 3.2, 3.3 e 3.4 e usando a função 3.5 se determina que o ponto é parcialmente *off-screen* e parcialmente *on-screen*, aplicam-se ambos os processos descritos acima.

- Se existem sobreposições

    - Desenhar o quadrado com o método *drawRect(...)*;
    - Desenhar três diagonais adjuntas ao ponto superior esquerdo, superior direito e inferior direito do quadrado, para fazer o quadrado ficar em perspetiva. Para isso utiliza-se o método *drawLine(...)*;
    - Desenhar a junção entre as três diagonais com duas linhas. Novamente utiliza-se o método *drawLine(...)* para esse efeito;

  - O número de sobreposições é adicionado textualmente no centro do quadrado, com o método *drawText(...)*

- Se não há sobreposições

  - Desenhar o quadrado com o método *drawRect(...)*;
  - O tipo do banco é adicionado textualmente no centro do quadrado, com o método *drawText(...)*

Apêndice B

Plano de Avaliação: Validação do IAR

Questionários utilizados nos testes de usabilidade feitos para validar o IAR.

Plano de Avaliação: Validação do InterestAR

Perfil

1. Nome: _____

2. Sexo: M ☐ F ☐

3. Idade: 18 - 24 ☐ 25 - 34 ☐
   35 - 45 ☐ > 45 ☐

4. Ocupação: _____

Pré-Questionário

5. Já utilizou aplicações de realidade aumentada?

   Se sim:

   i. Com que objetivos?

   _____

   ii. Com que frequência as utiliza?

   Nunca usou ☐ Raramente ☐
   Poucas vezes ☐ Algumas vezes ☐
   Muitas vezes ☐

   iii. Qual a aplicação que mais utiliza?

   _____

   iv. Encontra algum tipo de dificuldades na utilização deste tipo de aplicações? Quais?

   _____

   _____

6. Está familiarizado com aplicações que lhe permitam ter conhecimento da existência de objectos/itens fora da área visível no ecrã?

   Se sim:

   i. Como é que tal é indicado?

   _____

   ii. Que aplicações?

   _____

   iii. Com que frequência as utiliza?

Nunca usou	☐	Raramente	☐
Poucas vezes	☐	Algumas vezes	☐
Muitas vezes	☐		

7. Já utilizou aplicações de pesquisa de informação georreferenciada?

   Se sim:

   i. Com que objetivos?

   _____

   ii. Com que frequência as utiliza?

Nunca usou	☐	Raramente	☐
Poucas vezes	☐	Algumas vezes	☐
Muitas vezes	☐		

   iii. Qual a aplicação que mais utiliza?

   _____

   iv. Encontra algum tipo de dificuldades na utilização deste tipo de aplicações? Quais?

   _____

   _____

8. Já utilizou aplicações de pesquisa de informação georreferenciada em dispositivos móveis?

Se sim:

ii. Com que objetivos?

_____

iii. Com que frequência as utiliza?

Nunca usou	☐	Raramente	☐
Poucas vezes	☐	Algumas vezes	☐
Muitas vezes	☐		

v. Qual a aplicação que mais utiliza?

_____

vi. Encontra algum tipo de dificuldades na utilização deste tipo de aplicações? Quais?

_____

_____

Contexto da experiência

- InterestAR é uma aplicação de realidade aumentada

- Relevância é representada pela côr, sendo o vermelho o mais relevante e o azul o menos. Dentro da mesma côr, não existem pontos mais relevantes que outros.

- Distância de um ponto ao utilizador pode ser induzida pela transparência e tamanho de um símbolo. Pontos menos relevantes são sempre mais transparentes que os mais relevantes, por isso só é possível comparar transparências entre símbolos com a mesma relevância.

- Sinalização off-screen é feita pela colocação de símbolos nas bordas laterais do ecrã. A posição do símbolo na borda representa a rotação necessária para que o símbolo entre na área visível do ecrã.

- Agregação on-screen é feita através da formação de pilhas, que reprentam pontos muito próximos entre si. Cada pilha mostra um número, indicativo do número de pontos agregados, e toma a côr e tamanho do ponto mais relevante e mais próximo.
- O mapa 2D é uma alternativa à realidade aumentada, onde o utilizador pode ver os pontos no terreno. O mapa roda de acordo com a orientação do utilizador.

- Existe ainda um Radar sobre o mapa, que permite perceber o ângulo de visão do utilizador e o raio de procura dos pontos.

Antes do início da experiência o utilizador é colocado no modo livre, para que possa ter algum tempo para experimentar e ambientar-se à aplicação. Enquanto este usa a aplicação vão sendo feitas algumas perguntas simples para garantir que o utilizador compreendeu os conceitos explicados.

Experiência 1

Cenário:

O utilizador encontra-se algures no concelho de Lisboa e, usando a aplicação InterestAR no seu dispositivo móvel, orienta-se para Norte apontando o dispositivo nessa direção.

De seguida, o utilizador será exposto a várias visualizações de forma sequencial, tentando executar a mesma tarefa para cada uma delas. As diferentes visualizações vão combinar um conjunto menor ou maior de funcionalidades, nomeadamente: a sinalização de objetos fora do ecrã e agregação de pontos sobrepostos.

Ordem apresentada: _____

Observações: _____

_____

_____

_____

_____

_____

Tarefa 1:

Num cenário com um número reduzido de pontos, encontre, para cada visualização, o ponto mais relevante e mais próximo da sua localização. Oriente-se nessa direção e, de seguida, clique no mesmo.

Questões:

Numa escala de 1 a 5, classifique as visualizações de acordo com a sua preferência:

V1: ___ V2: ___ V3: ___

Tarefa 2:

Num cenário com um número elevado de pontos, encontre, para cada visualização, o ponto mais relevante e mais próximo da sua localização. Oriente-se nessa direção e, de seguida, clique no mesmo.

Questões:

Numa escala de 1 a 5, classifique as visualizações de acordo com a sua preferência:

V1: ___ V2: ___ V3: ___

Experiência 2

Cenário:

O utilizador encontra-se algures no concelho de Lisboa e, usando a aplicação InterestAR no seu dispositivo móvel, orienta-se para Norte apontando o dispositivo nessa direção.

De seguida, o utilizador será exposto a várias visualizações de forma sequenciada, tentando executar a mesma tarefa para cada uma delas. As diferentes visualizações vão combinar um conjunto menor ou maior de funcionalidades, nomeadamente: a possibilidade de utilização de um mapa 2D e de um radar.

Ordem apresentada: _____

Observações: _____

_____

_____

_____

_____

_____

Tarefa 1:

Num cenário com um número reduzido de pontos, encontre, para cada visualização, o ponto mais relevante e mais próximo da sua localização. Oriente-se nessa direção e, de seguida, clique no mesmo.

Questões:

Numa escala de 1 a 5, classifique as visualizações de acordo com a sua preferência:

V4: ___ V5: ___

Tarefa 2:

Num cenário com um número elevado de pontos, encontre, para cada visualização, o ponto mais relevante e mais próximo da sua localização. Oriente-se nessa direção e, de seguida, clique no mesmo.

Questões:

Numa escala de 1 a 5, classifique as visualizações de acordo com a sua preferência:

V4: ___ V5: ___

Questões pós-tarefas

9. Numa escala de 1 a 5, classifique as visualizações de acordo com a sua facilidade de utilização – Justifique.

V1: ___ V2: ___ V3: ___

V4: ___ V5: ___

10. Encontrou dificuldades nas diferentes visualizações? Quais?

_____

_____

_____

11. Que funcionalidades acham que o teríam ajudado a concretizar as tarefas?

_____

_____

_____

12. Qual foi a sua estratégia para conseguir executar as tarefas propostas?

_____

_____

_____

Documento de autorização

Departamento de Informática – FCUL
Contacto: Gonçalo Silva, fc45337@alunos.fc.ul.pt
Julho de 2015

Documento de Autorização

Para experiência de visualização de simbologia em Realidade Aumentada

Eu, abaixo-assinado:

1. Autorizo o uso e tratamento dos dados por mim gerados nesta experiência pelas pessoas responsáveis pela mesma;

2. Comprometo-me a ter uma postura respeitadora relativamente às pessoas envolvidas e à experiência em si.

Declaro ter conhecimento dos meus direitos:

a. À privacidade;
b. De poder desistir livremente da experiência.

_____

(o voluntário)

Apêndice C

Resultados da Validação do IAR

Os documentos aqui presentes apresentam estatísticas feitas a partir dos dados recolhidos do pré-questionário, durante a utilização da aplicação e do pós-questionário.

C.1 Estatística dos Dados Recolhidos no Questionário

Mediana de preferências

Utilizador	Vis1	Vis2	Vis3	Vis4	Vis5
1	1	4	4	3	5
2	2	3	3	4	5
3	2	3	4	4	5
4	2	3	3	4	5
5	5	3	4	4	5
6	1	2	2	3	4
7	4	5	5	4	5
8	2	3	4	4	5
9	2	5	5	4	5
10	1	3	3	3	5
11	3	3	3	2	5
12	2	4	3	4	5
13	2	2	3	1	5
14	2	4	4	4	5
15	3	4	4	4	5
16	2	3	3	5	5
17	2	2	3	4	5
18	3	5	5	4	5
Mediana	**2.0**	**3.0**	**3.5**	**4.0**	**5.0**

Distribuição por áreas de conhecimento

Utilizador	RA	Off-Screen	Geo-Ref	Geo-Ref Mobile
1	1	1	1	1
2	1	1	1	1
3	1	0	1	1
4	0	0	1	0
5	0	0	1	1
6	1	0	1	1
7	0	0	1	1
8	0	0	1	0
9	0	0	1	1
10	0	1	1	1
11	0	1	1	1
12	0	0	1	1
13	1	1	1	1
14	1	1	1	1
15	1	0	1	0
16	0	1	1	1
17	0	1	1	1
18	0	1	1	1
Total	**7**	**9**	**18**	**15**

Distribuição por género e idade

Utilizador	Masculino	Feminino	16-24	25-34	35-45	> 45
1	1	0	0	1	0	0
2	0	1	0	0	0	1
3	0	1	1	0	0	0
4	0	1	1	0	0	0
5	0	1	1	0	0	0
6	1	0	1	0	0	0
7	1	0	1	0	0	0
8	0	1	0	0	0	1
9	0	1	0	0	1	0
10	1	0	1	0	0	0
11	1	0	1	0	0	0
12	0	1	0	0	1	0
13	0	1	0	0	0	1
14	1	0	0	0	1	0
15	0	1	0	0	0	1
16	1	0	0	1	0	0
17	1	0	0	1	0	0
18	1	0	1	0	0	0
Total	**9**	**9**	**8**	**3**	**3**	**4**

Distribuição por área de atuação

Utilizador	Informática	Gestão/Administrativa	Saúde	Artes	Secundário
1	1	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	0	0	0	0	1
5	0	0	0	1	0
6	0	1	0	0	0
7	1	0	0	0	0
8	0	1	0	0	0
9	0	1	0	0	0
10	0	0	1	0	0
11	0	0	1	0	0
12	0	1	0	0	0
13	1	0	0	0	0
14	1	0	0	0	0
15	1	0	0	0	0
16	0	1	0	0	0
17	0	1	0	0	0
18	1	0	0	0	0
Total	**8**	**6**	**2**	**1**	**1**

C.2 Estatística dos Dados Recolhidos Durante a Utilização do IAR

utilizador	tipo	nº pois	tempo	ıngulotota	erros	acessomapa
1	1	15	24.60	257.20	1	0
1	2	15	4.40	0.00	0	0
1	3	15	30.30	252.10	0	0
1	1	30	15.90	111.30	1	0
1	2	30	18.20	160.00	1	0
1	3	30	31.50	107.60	0	0
1	4	15	64.60	657.90	1	1
1	5	15	10.30	70.80	0	1
1	4	30	16.30	275.60	1	1
1	5	30	26.80	17.70	1	1
2	2	15	5.90	34.90	0	0
2	3	15	31.40	168.00	0	0
2	1	15	61.00	448.40	1	0
2	2	30	31.50	172.40	1	0
2	3	30	25.30	93.10	1	0
2	1	30	33.50	195.40	1	0
2	5	15	17.80	156.60	0	1
2	4	15	36.00	392.40	1	1
2	5	30	24.30	252.10	0	1
2	4	30	31.60	303.20	1	1
3	3	15	15.60	309.60	0	0
3	1	15	30.50	373.60	0	0
3	2	15	21.20	107.70	0	0
3	3	30	44.80	467.40	0	0
3	1	30	46.80	348.60	0	0
3	2	30	25.30	401.30	0	0
3	4	15	34.20	350.30	0	1
3	5	15	21.30	305.90	0	1
3	4	30	32.90	320.10	0	1
3	5	30	12.30	105.80	0	1
4	3	15	60.80	686.50	0	0
4	2	15	43.10	413.10	0	0
4	1	15	66.50	716.80	0	0
4	3	30	35.40	403.40	0	0
4	2	30	21.10	431.60	0	0
4	1	30	57.70	865.00	1	0
4	5	15	38.90	403.30	0	1
4	4	15	50.30	892.00	1	1
4	5	30	39.10	225.40	0	1
4	4	30	135.30	728.10	1	1
5	2	15	49.20	840.30	0	0
5	3	15	57.50	770.10	0	0
5	1	15	18.50	533.60	0	0
5	2	30	19.70	363.80	0	0
5	3	30	57.60	787.50	0	0
5	1	30	32.80	510.70	1	0
5	4	15	31.30	587.10	0	1
5	5	15	39.10	763.10	1	1
5	4	30	22.40	334.10	1	1

5	5	30	66.80	636.00	1	1
6	1	15	26.90	470.90	1	0
6	2	15	29.70	291.40	0	0
6	3	15	9.50	259.80	0	0
6	1	30	29.00	771.30	1	0
6	2	30	7.80	104.30	0	0
6	3	30	20.70	119.00	0	0
6	5	15	7.70	186.60	0	1
6	4	15	14.20	152.20	0	1
6	5	30	28.00	263.40	1	1
6	4	30	10.30	137.00	0	1
7	3	15	20.10	321.70	0	0
7	1	15	44.90	342.90	1	0
7	2	15	54.30	373.40	0	0
7	3	30	42.80	313.20	0	0
7	1	30	62.80	474.30	1	0
7	2	30	46.70	298.40	0	0
7	5	15	51.20	297.00	0	1
7	4	15	68.90	329.60	0	1
7	5	30	29.80	167.70	0	1
7	4	30	57.60	262.80	0	1
8	1	15	31.50	339.80	1	0
8	3	15	57.40	648.20	1	0
8	2	15	17.10	325.00	0	0
8	1	30	37.80	695.30	1	0
8	3	30	31.60	657.90	0	0
8	2	30	43.10	434.60	1	0
8	4	15	25.00	299.00	1	1
8	5	15	16.50	110.20	0	1
8	4	30	23.50	72.60	0	1
8	5	30	18.00	62.80	0	1
9	1	15	31.10	250.40	0	0
9	3	15	53.10	345.20	0	0
9	2	15	24.00	130.10	0	0
9	1	30	24.40	236.20	0	0
9	3	30	54.80	352.00	0	0
9	2	30	32.40	177.90	0	0
9	5	15	36.10	196.50	0	1
9	4	15	31.10	160.10	0	1
9	5	30	20.70	45.90	1	1
9	4	30	44.90	95.40	0	1
10	3	15	18.20	466.00	0	0
10	2	15	22.40	630.40	0	0
10	1	15	18.50	625.10	0	0
10	3	30	16.40	417.70	0	0
10	2	30	22.90	710.20	1	0
10	1	30	49.60	902.20	1	0
10	4	15	16.80	276.40	0	1
10	5	15	25.80	309.60	0	1
10	4	30	16.10	399.60	0	1

10	5	30	14.30	207.80	0	1
11	2	15	21.30	414.50	0	0
11	1	15	9.40	152.40	0	0
11	3	15	13.10	164.30	0	0
11	2	30	13.60	386.00	0	0
11	1	30	23.60	469.50	1	0
11	3	30	2.20	7.80	0	0
11	4	15	26.10	314.20	1	1
11	5	15	18.00	171.90	1	1
11	4	30	15.50	31.90	0	1
11	5	30	15.40	145.20	0	1
12	1	15	18.20	368.30	0	0
12	2	15	19.80	612.30	0	0
12	3	15	36.70	598.50	0	0
12	1	30	22.90	372.00	1	0
12	2	30	44.20	386.50	1	0
12	3	30	43.00	571.20	1	0
12	5	30	34.50	377.80	0	1
12	4	30	55.50	415.90	1	1
12	5	15	17.70	279.40	0	1
12	4	15	39.50	1017.50	0	0
13	1	15	23.10	322.10	0	0
13	2	15	23.60	173.10	1	0
13	3	15	12.30	55.70	0	0
13	1	30	22.60	471.30	0	0
13	2	30	32.00	52.30	1	0
13	3	30	18.40	57.80	0	0
13	5	15	19.50	81.40	1	0
13	4	15	6.10	5.50	0	0
13	5	30	38.90	156.40	1	1
13	4	30	17.80	7.40	1	1
14	2	15	42.40	476.40	0	0
14	3	15	43.90	449.70	0	0
14	1	15	31.80	564.40	0	0
14	2	30	42.40	524.70	0	0
14	3	30	64.70	801.40	0	0
14	1	30	58.80	884.00	0	0
14	4	15	77.20	939.70	0	1
14	5	15	21.70	78.40	0	1
14	4	30	17.50	352.10	0	1
14	5	30	24.50	51.20	0	1
15	2	15	5.60	19.70	0	0
15	3	15	36.90	11.90	0	0
15	1	15	17.80	129.70	0	0
15	2	30	15.00	114.60	0	0
15	3	30	27.00	137.90	1	0
15	1	30	41.50	176.50	1	0
15	5	15	13.20	11.70	1	1
15	4	15	28.80	266.60	0	0
15	5	30	16.00	32.50	1	1

15	4	30	37.10	258.20	1	0
16	2	15	15.20	366.00	0	0
16	1	15	32.40	555.70	1	0
16	3	15	16.80	599.70	0	0
16	2	30	46.00	1259.10	0	0
16	1	30	17.80	429.30	0	0
16	3	30	30.60	658.00	1	0
16	5	15	9.80	117.10	0	1
16	4	15	11.90	189.00	0	1
16	5	30	10.30	203.60	0	1
16	4	30	13.20	265.60	0	1
17	3	15	25.60	340.80	0	0
17	1	15	54.40	500.10	0	0
17	2	15	47.80	705.50	0	0
17	3	30	49.10	652.60	0	0
17	1	30	41.80	837.80	0	0
17	2	30	109.30	1503.10	1	0
17	4	15	44.60	806.10	0	1
17	5	15	53.40	1049.10	0	1
17	4	30	68.80	972.90	1	1
17	5	30	54.10	922.30	0	1
18	1	15	48.10	561.80	1	0
18	2	15	25.30	118.80	0	0
18	3	15	29.70	354.90	0	0
18	1	30	28.60	445.60	0	0
18	2	30	28.30	166.00	0	0
18	3	30	52.60	492.50	0	0
18	4	15	44.60	391.00	1	1
18	5	15	21.00	204.10	1	1
18	4	30	27.00	404.40	0	1
18	5	30	28.20	97.40	0	1

visualização	nº pois	média tempo	média ângulo	% erros	% acessos mapa
Vis1	15	32.7	417.4	38.9%	n/a
Vis1	30	36.0	510.9	73.3%	n/a
Vis2	15	26.2	335.1	5.6%	n/a
Vis2	30	33.3	424.8	38.9%	n/a
Vis3	15	31.6	377.9	5.6%	n/a
Vis3	30	36.0	394.3	22.2%	n/a
Vis4	15	36.2	445.9	33.3%	83.3%
Vis4	30	35.7	313.2	44.4%	94.4%
Vis5	15	24.4	266.3	27.8%	94.4%
Vis5	30	27.9	220.6	33.3%	100.0%

C.3 Grelha de Sequência das Técnicas de Visualização Apresentadas

Utilizador	Vis	Vis	Vis	Vis	Vis
1	1	2	3	4	5
5	2	3	1	4	5
3	3	1	2	4	5
8	1	3	2	4	5
11	2	1	3	4	5
10	3	2	1	4	5
6	1	2	3	5	4
15	2	3	1	5	4
7	3	1	2	5	4
9	1	3	2	5	4
12	2	1	3	5	4
4	3	2	1	5	4
18	1	2	3	4	5
14	2	3	1	4	5
17	3	1	2	4	5
13	1	3	2	4	5
16	2	1	3	4	5
2	3	2	1	4	5