

Universidade da Beira Interior

Departamento de Informática



**Departamento de
Informática**

**Nº 112 - 2019: *Codificação da Cor de Hologramas
Digitais Usando Multivistas***

Elaborado por:

Raquel Sofia Brás Guerra

Orientadora:

**Professora Doutora Maria Manuela Areias da Costa Pereira de
Sousa**

7 de setembro de 2020

Às minhas avós.

Agradecimentos

Não poderia deixar de começar por demonstrar a minha eterna gratidão aos meus pais por todo o apoio que me deram neste atribulado percurso que foi o ensino universitário.

Agradeço à minha professora Orientadora, Maria Manuela de Sousa, pela incansável ajuda, não só neste projeto mas também no meu percurso académico desde as primeira aulas.

De seguida, gostaria de agradecer ao meu melhor amigo Igor Nunes. Ele foi um grande apoio num dos momentos mais difíceis que enfrentei. Sem ele não sei se teria conseguido concluir este projeto.

Agradeço ainda ao professor Simão Melo de Sousa por me ter cedido o laboratório *RELiE And SECure Computation Group* (Release) e o respetivo acesso para poder trabalhar e estudar a qualquer hora. E claro, estou grata por todas as tardes que comigo dispensou no Release com excelentes conselhos e dicas no meu percurso académico.

Por fim, agradeço aos meus amigos Diogo Simões, Pedro Batista, Nuno Fernandes, Lina Jesus e Pedro Cavaleiro por serem aqueles que me acompanharam nestes últimos anos e que não se esqueceram de mim.

Conteúdo

Conteúdo	iii
Lista de Figuras	v
Lista de Tabelas	vii
1 Introdução	1
1.1 Enquadramento	1
1.2 Motivação	1
1.3 Objetivos	1
1.4 Organização do Documento	2
2 Estado da Arte	5
2.1 Introdução	5
2.2 Perspetiva Histórica e Conceitos Base da Holografia	5
2.3 O Holograma	8
2.3.1 Obtenção	8
2.3.2 Reconstrução	9
2.3.3 Representação	13
2.4 Compressão de um Holograma	13
2.4.1 Uma Breve Introdução ao JPEG2000	14
2.4.2 PSNR: uma Métrica de Relação Qualidade-Débito	17
2.5 Conclusões	17
3 Tecnologias e Ferramentas Utilizadas	19
3.1 Introdução	19
3.2 Tecnologias e Ferramentas	19
3.2.1 Software de Reconstrução de Hologramas e sua Transcrição para Python	20
3.2.2 <i>Kakadu Software</i>	20
3.3 Materiais Utilizados	21
3.3.1 <i>Hardware</i>	21
3.3.2 Hologramas	22

3.4	Conclusões	22
4	Etapas de Desenvolvimento e Implementação	25
4.1	Introdução	25
4.2	Reconstrução dos Hologramas	26
4.3	Compressão dos Hologramas Reconstruídos	28
4.3.1	Breve Estudo do <i>Kakadu Software</i>	28
4.3.2	Implementação e Execução do <i>Script</i> de Compressão .	29
4.3.3	Automatização dos <i>Scripts</i>	29
4.4	Determinação da Relação Qualidade Débito	30
4.4.1	Criação de gráficos	31
4.5	Conclusões	31
5	Testes e Resultados	33
5.1	Introdução	33
5.2	Testes	33
5.3	Resultados	34
5.4	Discussão	34
5.5	Conclusões	38
6	Conclusões e Trabalho Futuro	41
6.1	Conclusões Principais	41
6.2	Trabalho Futuro	42
A	Documentação de Funções	43
A.1	Função <i>load_hologram</i>	43
A.2	Função <i>propagate_asm</i>	44
A.3	Função <i>reconst_hologram</i>	45
A.4	Função <i>reconst_16views</i>	46
A.5	Função <i>compress_views</i>	47
A.6	Função <i>cod_jpeg2000</i>	48
A.7	Função <i>dec_jpeg2000</i>	49
A.8	Função <i>psnr</i>	50
	Bibliografia	53

Lista de Figuras

2.1	Ilustração do efeito de difração.	6
2.2	Ilustração do efeito de interferência.	6
2.3	Ilustração do efeito de paralaxe.	7
2.4	Métodos de obtenção de hologramas.	10
2.5	Obtenção de um HGC pela técnica <i>point-source</i>	11
2.6	Obtenção de um HGC pela técnica <i>layer-based</i>	11
2.7	Geometria da difração.	12
2.8	Exemplo de um holograma no formato amplitude-fase	14
3.1	Holograma Dices4k (imagem original)	23
3.2	Holograma DiffuseCar4k (imagem original)	23
3.3	Holograma Piano4k (imagem original)	24
4.1	Diagrama do processo efetuado sobre um holograma para obtenção dos resultados.	27
5.1	Variação da média do PSNR relativo ao holograma dices4k.	36
5.2	Variação da média do PSNR relativo ao holograma diffuseCar4k.	38
5.3	Variação da média do PSNR relativo ao holograma piano4k.	40
A.1	Diagrama de dependências da função <code>reconst_16views</code>	51
A.2	Diagrama de dependências da função <code>compress_views</code>	51

Listas de Tabelas

3.1	Especificações dos hologramas	22
5.1	Parâmetros variáveis utilizados na reconstrução dos hologramas.	34
5.2	Resultados da métrica PSNR para o holograma dices4k.	35
5.3	Resultados da métrica PSNR para o holograma diffuseCar4k.	37
5.4	Resultados da métrica PSNR para o holograma piano4k.	39
A.1	Documentação da função load_hologram	43
A.2	Documentação da função propagate_asm	44
A.3	Documentação da função reconst_hologram	45
A.4	Documentação da função reconst_16views.	46
A.5	Documentação da função compress_views.	47
A.6	Documentação da função cod_jpeg2000.	48
A.7	Documentação da função dec_jpeg2000.	49
A.8	Documentação da função psnr.	50

Acrónimos

ASM	<i>Angular Spectrum Method</i>
CPU	<i>Central Processing Unit</i>
CUDA	<i>Compute Unified Device Architecture</i>
dB	<i>Decibel</i>
DCT	<i>Discrete Cosine Transform</i>
EBCOT	<i>Embedded Block Coding with Optimal Truncation</i>
FTM	<i>Fresnel transform method</i>
GPU	<i>Graphical Processing Unit</i>
HCM	<i>Huygens convolution method</i>
HGC	Holograma Gerado por Computador
ICT	<i>Irreversible Color Transform</i>
JPEG	<i>Joint Photographic Experts Group</i>
JSON	<i>JavaScript Object Notation</i>
kdu	<i>Kakadu Software</i>
MSE	<i>Mean Squared Error</i>
PSNR	<i>Peak signal-to-noise ratio</i>
RCT	<i>Reversible Color Transform</i>
Release	<i>RELiabile And SECure Computation Group</i>
RGB	<i>Red, Green & Blue</i>
RMN	Ressonância Magnética Nuclear
ROI	<i>Region of Interest</i>

SDK	<i>Software Development Kit</i>
SFTF	<i>Spatial Frequency Transfer Funcation</i>
SLM	<i>Spatial Light Modulator</i>
TAC	Tomografia Axial Computurizada
UHD	<i>Ultra-High Definition</i>
XML	<i>Extensible Markup Language</i>

Capítulo

1

Introdução

1.1 Enquadramento

A história da captura, armazenamento e visualização de imagens é extremamente rica e milenar. Marcos importantes destacam-se, sendo do particular interesse no Século XXI os grandes passos dados na imagem digital.

Contudo, a vasta maioria da fotografia tem-se centrado na captura de imagens estáticas em duas dimensões. O interesse na captura e representação de objetos e momentos em três dimensões tem ganho um interesse crescente nas últimas décadas.

A área dedicada ao estudo deste modelo, a **holografia**, carece de vários marcos que já fazem parte do quotidiano da fotografia clássica, nomeadamente normas para a codificação de **hologramas** em formato digital.

1.2 Motivação

Dada a referida ausência de normas no armazenamento e representação da informação, reconstrução e codificação de um holograma, é do interesse da comunidade do JPEG Pleno estudar os codificadores existentes para melhor perceber qual a sua adaptabilidade aos hologramas e quais as modificações necessárias para resolver a falta de normas nos pontos mencionados.

1.3 Objetivos

Tendo em mente a motivação apresentada na secção 1.2, o presente projeto tem por objetivo principal investigar o desempenho do codec JPEG2000 na

codificação de hologramas a cores em multivistas.

Por seu turno, os objetivos secundários — os quais refletem as diferentes fases da investigação — são os seguintes:

1. Implementar um reconstrutor para hologramas com cor;
2. Reconstruir hologramas em vários pontos de vista;
3. Comprimir os hologramas reconstruídos com recurso ao codificador JPEG2000;
4. Avaliar a qualidade das imagens comprimidas face à reconstrução original.

Os objetivos supra-mencionados refletem o objetivo geral de estudar holografia e, assim, expandir o conhecimento na área das tecnologias multimédia.

1.4 Organização do Documento

Apresenta-se de seguida a estrutura do presente documento:

- No 1º capítulo — **Introdução** —, são formulados os objetivos do presente projeto, assim como a motivação deste;
- No 2º capítulo — **Estado da Arte** — são explorados os conceitos essenciais da holografia, assim como é feita uma breve perspetiva histórica e um estudo sumariado da norma JPEG2000;
- No 3º capítulo — **Tecnologias e Ferramentas Utilizadas** — são expostas as tecnologias utilizadas para a investigação levada a cabo no âmbito no projeto (*software* externo e linguagem utilizada para a implementação de *scripts* próprios), assim como os materiais a que se recorreu (em particular os hologramas testados e as especificações dos computadores utilizados);
- No 4º capítulo — **Etapas de Desenvolvimento e Implementação** — é dada a conhecer a estratégia de investigação, detalhando as três partes na qual se dividiu;
- No 5º capítulo — **Testes e Resultados** — procede-se à apresentação dos resultados obtidos, sendo feita uma discussão destes com base em três questões que se levantaram após a sua análise;

- Por fim, no 6º capítulo — **Conclusões e Trabalho Futuro** —, são formuladas as conclusões do projeto, rematando assim os objetivos apresentados na Secção 1.3, e são feitas propostas de estudos para trabalhos futuros.

Capítulo

2

Estado da Arte

2.1 Introdução

O que é a holografia? O que é um holograma? Como se obtêm, codificam e representam? Que formas já foram estudadas de os comprimir? Como se pode avaliar a qualidade da imagem comprimida?

Estas e outras questões impõem-se perante a investigação que se almeja. A sua resposta, dada no presente Capítulo, orientará a definição de uma estratégia de investigação e permitirá avançar a área da holografia com um renovado conhecimento acerca da compressão de hologramas com a norma JPEG2000.

2.2 Perspetiva Histórica e Conceitos Base da Holografia

Entende-se por **holograma** uma gravação física de um padrão de interferência que recorre ao efeito de difração da luz para reproduzir um campo de luz tridimensional. Tal resulta numa imagem a qual retém as propriedades da cena original, entre elas a profundidade e a paralaxe [1].

Por seu turno, a **holografia** é a ciência envolvida no estudo da obtenção de hologramas.

Há a referir brevemente os seguintes conceitos para clarificação:

- **Difração:** Conjunto de fenómenos que ocorrem quando uma onda encontra um obstáculo, em particular a sua aparente flexão e alargamento ao atravessar orifícios [2, 3];

- **Interferência:** Semelhante à difração, mas referente à sobreposição de duas ou mais ondas;
- **Paralaxe:** Diferença na posição aparente de um objeto quando observado em locais diferentes [4, 5].

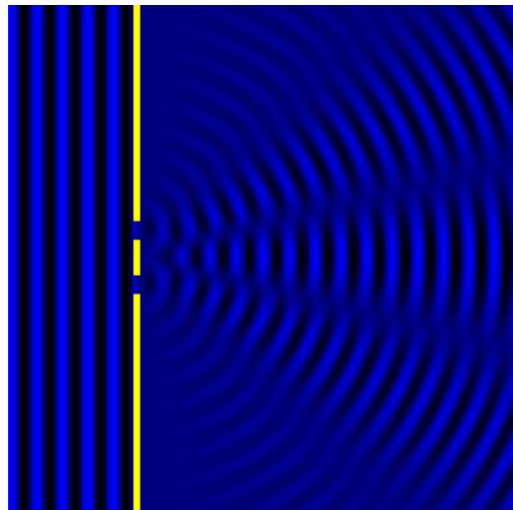


Figura 2.1: Ilustração do efeito de difração de uma onda a atravessar dois orifícios vizinhos [6].

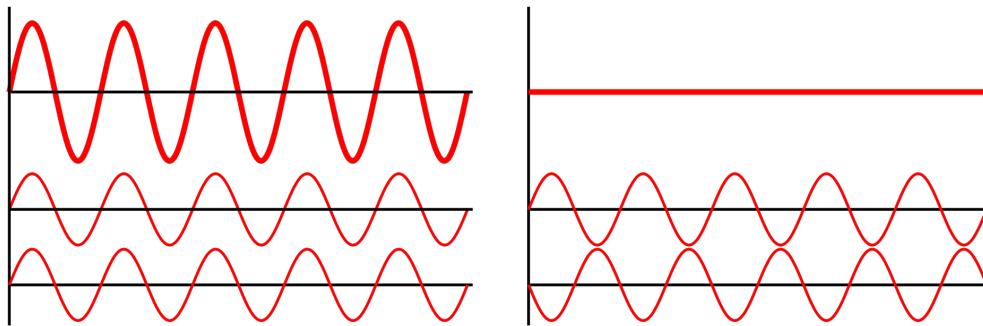


Figura 2.2: Ilustração do efeito de interferência de duas ondas [7]. Do lado esquerdo encontra-se representada uma interferência construtiva: as duas ondas sincronizadas (*i.e.* exatamente em fase) produzem uma nova onda de amplitude aumentada. Do lado direito apresenta-se um caso de interferência destrutiva: existe uma anulação das ondas por estarem exatamente fora de fase.

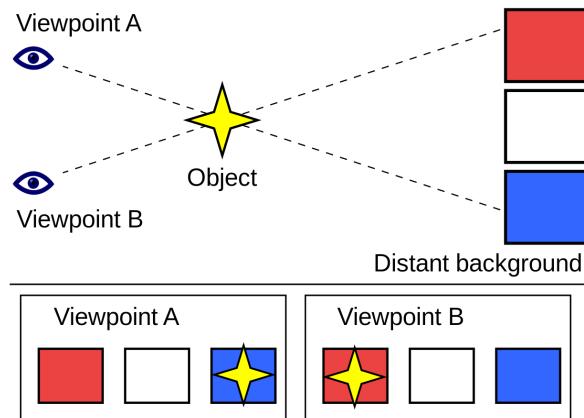


Figura 2.3: Ilustração do efeito de paralaxe de um objeto face a um padrão de fundo [8]. Devido à mudança de perspetiva do utilizador em dois pontos de vista distintos, no ponto A o observador perceciona a estrela como estando sobre o fundo azul, enquanto no ponto B esta aparenta estar sobre o fundo vermelho.

Enquanto a fotografia tradicional apenas captura a intensidade da luz refletida pelos objetos, a holografia permite capturar tanto a amplitude como a fase da onda de luz dispersa por um objeto [1, 9, 10].

Em contraste face à fotografia clássica, a qual teve as suas origens desde a descoberta da *camara obscura* por parte da China Antiga [11], a holografia como ciência conheceu o seu início com o cientista húngaro-britânico Dennis Gabor na década de 1940 enquanto este estudava avanços na tecnologia da microscopia eletrónica [12, 13]. A descoberta da holografia e, por sua vez, dos hologramas foi inesperada.

A técnica foi adaptada e é ainda hoje utilizada na microscopia eletrónica no que se chama de **holografia eletrónica**. Contudo, a **holografia ótica** só pôde avançar após a invenção do *laser* na década de 1960 [14, 15].

O ponto-chave no desenvolvimento de imagens holográficas, proposto por Dennis Gabor em 1948 numa tentativa de melhorar a microscopia eletrónica, é o uso de uma fonte de luz de referência para codificar uma onda sobreposta com outra a fim de registar os padrões de interferência [12]. Contudo, as experiências de Gabor eram limitadas pelo uso de ondas ópticas paralelas ao eixo óptico [9].

Desta forma, em 1962, Emmet Leith e Juris Upatnieks desenvolveram a técnica que viria a lançar em definitivo a holografia ótica: a **holografia ótica off-axis** [14], ou seja, o eixo óptico não é paralelo ao feixe de ondas ópticas. Esta técnica, desenvolvida no âmbito do estudo de radares, teve a sua prova prática

após a invenção do *laser* e, em 1964, os dois cientistas produziram uma série de hologramas que culminaram na atribuição do Prémio Nóbela da Física a Dennis Gabor em 1971 [14, 16].

Diferentes tipos de hologramas ópticos foram desenvolvidos nas décadas subsequentes, assim como outras áreas beneficiaram com os estudos enveredados na área da holografia [9]. Em particular, o desenvolvimento de hologramas visualizados pela transmissão de luz branca, por Stephen Benton [17], permitiu avanços na medicina, em particular na área da imagiologia, onde a holografia foi incorporada em exames como a Tomografia Axial Computurizada (TAC) e a Ressonância Magnética Nuclear (RMN) [18]. Curiosamente, esta técnica é comumente vista em cartões de crédito/débito e documentos de identificação nacional oficiais [18, 19].

2.3 O Holograma

Existem atualmente duas formas de obter e reconstruir hologramas [9]:

1. Configuração **analógica** (método original);
2. Configurações **digitais**.

De notar que existem diferentes métodos tanto para a obtenção como para a reconstrução numérica na holografia digital (Figura 2.4).

A **holografia digital** em particular é um processo de três passos [9]:

1. Geração do campo de ondas do objeto;
2. Aquisição (ou registo) do padrão de interferência;
3. Reconstrução do campo do objeto para visualização tridimensional ou em multivista bidimensional renderizada.

As configurações digitais serão as abordadas na presente Secção, sendo a analógica ignorada, uma vez que ambas apresentam princípios teóricos semelhantes e o âmbito do projeto é a holografia digital. Em particular, serão abordadas apenas as técnicas utilizadas na obtenção e reconstrução dos hologramas testados no projeto.

2.3.1 Obtenção

A obtenção de um holograma envolve a codificação da amplitude e da fase da onda de luz dispersa por um objeto. Esta varia conforme o meio onde o holograma será gravado. Uma vez que os métodos clássicos são tendencialmente

dispendiosos e morosos, a atenção maior da comunidade científica tem sido no desenvolvimento de métodos digitais [9].

Em particular, um **Holograma Gerado por Computador** (**HGC**) envolve um processo no qual o campo de ondas do objeto é computado digitalmente através de uma simulação da propagação da luz dispersa pela cena face ao plano do holograma.

De entre os métodos de **obtenção indireta** *i.e.* geração de hologramas por computador, sumariados na Figura 2.4, dois em particular serão resumidos por serem os utilizados para a obtenção dos hologramas objeto de teste no projeto [20, 21]:

1. *Point-source* [9, 22]:

É feita uma amostragem à cena tridimensional através da recolha de pontos considerados como fontes esféricas de luz. Contudo, esta técnica é de uma grande complexidade computacional (ordem $O(NM)$, onde N representa o número de pixels do holograma e M é o número total de pontos da cena).

2. *Layer-based* [9, 23] (também conhecido por *wave-field* [21]):

A cena tridimensional é dividida em camadas paralelas ao plano do holograma. Cada camada é considerada uma fonte superficial de luz cujas emissões de ondas luminosas são propagadas numericamente para o plano do holograma e somadas para obter o campo de ondas do objeto. Esta técnica apresenta uma complexidade computacional inferior face ao *point-source* (nomeadamente $O(N_z N \log(N))$, onde N representa o número de pixels do holograma e N_z é o número de camadas). Todavia, esta técnica não é aconselhável para cenas de grande profundidade.

2.3.2 Reconstrução

Esta fase do processo da holografia digital pode ser alcançada de duas formas [9]:

1. **Reconstrução numérica**, onde é feita simulação da difração da luz por parte do holograma digitalmente obtido;
2. **Reconstrução ótica**, com a propagação física de um feixe ótico com recurso a um *laser* e a um *Spatial Light Modulator* (SLM).

Apenas o primeiro tipo de reconstrução será abordado por ser o utilizado no projeto.

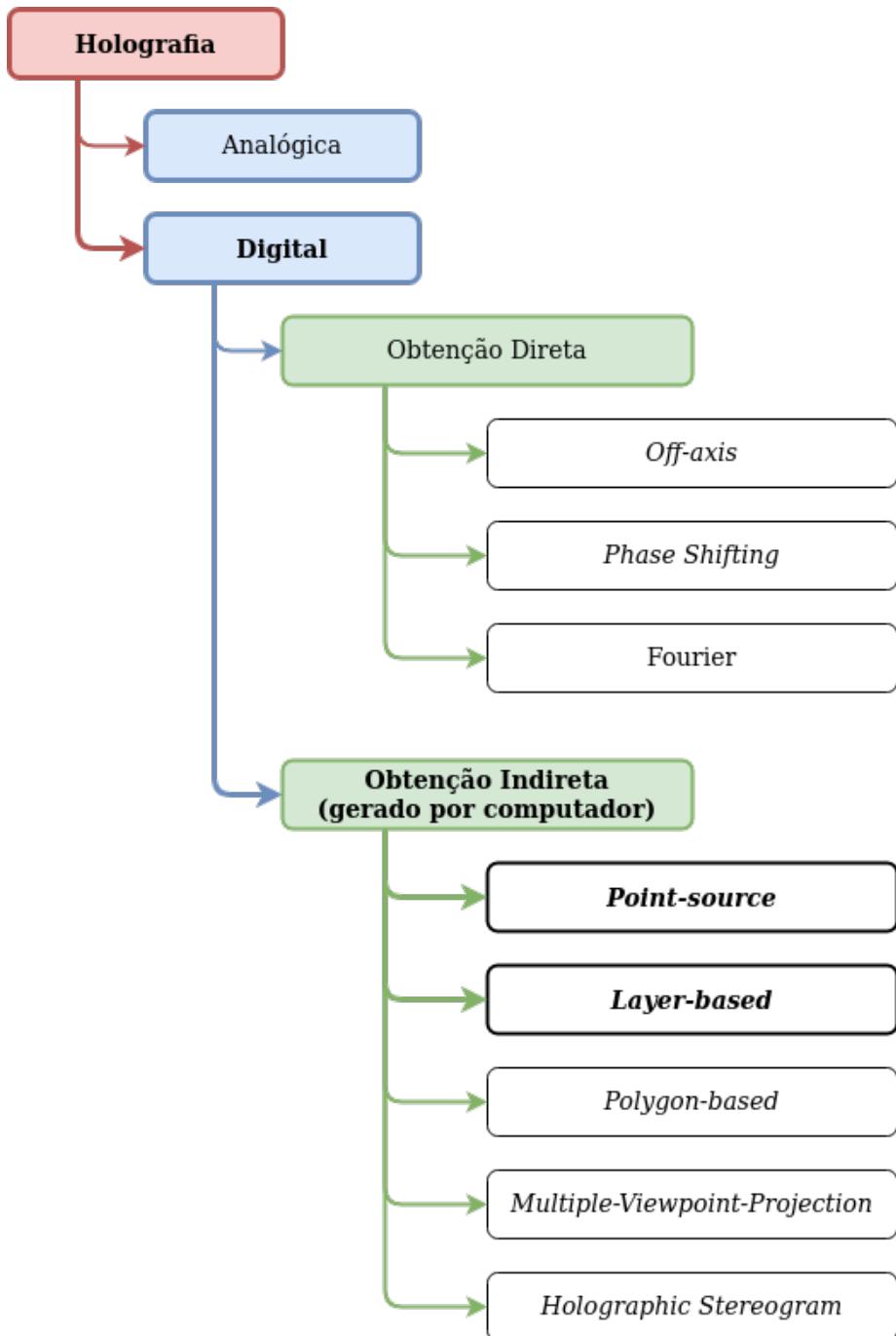


Figura 2.4: Métodos de obtenção de hologramas [9]. A negrito estão destacados os métodos utilizados na obtenção dos hologramas testados no presente projeto.

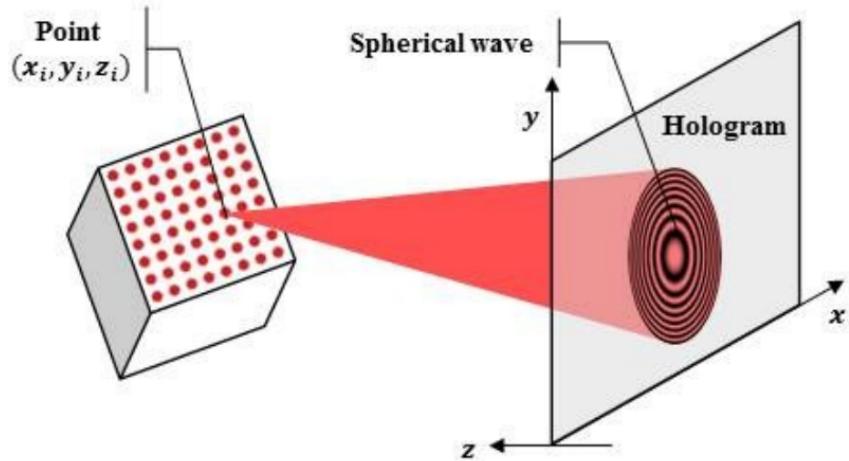


Figura 2.5: Obtenção de um HGC pela técnica *point-source* [9].

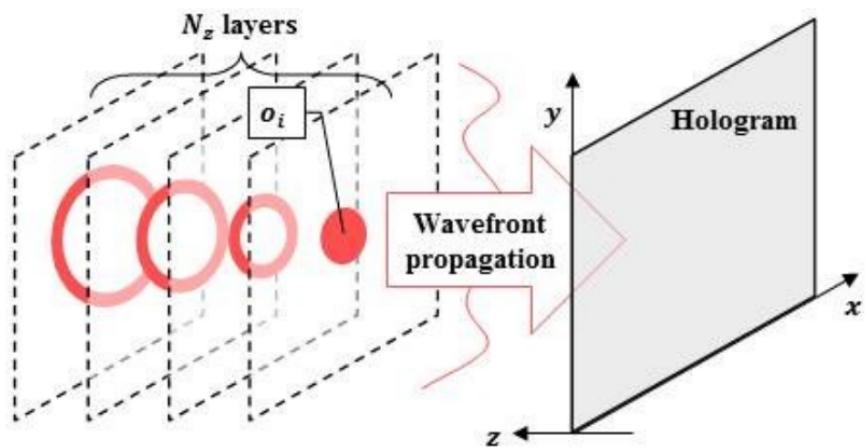


Figura 2.6: Obtenção de um HGC pela técnica *layer-based* [9].

São então definidos três métodos de reconstrução numérica consoante a distância de reconstrução [9]:

1. *Angular Spectrum Method* (ASM) (curta distância);
2. *Huygens convolution method* (HCM) (média-longa distância);
3. *Fresnel transform method* (FTM) (longa distância).

O que será utilizado no projeto é o método **ASM**. Não sendo objetivo máximo deste Capítulo explanar em detalhe os conceitos teóricos inerentes a este método, segue-se um breve resumo para esclarecimento com as principais fórmulas que serão diretamente utilizadas no desenvolvimento de *scripts* no âmbito da investigação.

A **reconstrução numérica** de um holograma consiste na propagação da amplitude complexa registada do objeto no plano da câmara digital relativa à sua posição original. Esta é feita com recurso à teoria de difração escalar (Figura 2.7) [9], a qual estabelece que um campo difratado $\psi_p(x, y; z)$ pode ser obtido a partir do campo incidente $\psi_{p_0}(x, y)$ de acordo com a equação (2.1) [24],

$$\psi_p(x, y; z) = \mathcal{F}^{-1} \left\{ \mathcal{F} \left\{ \psi_{p_0}(x, y) \right\} \times \mathcal{H}(k_x, k_y; z) \right\} \quad (2.1)$$

onde $\mathcal{H}(k_x, k_y; z)$ é a *Spatial Frequency Transfer Function* (SFTF) (equação (2.2)).

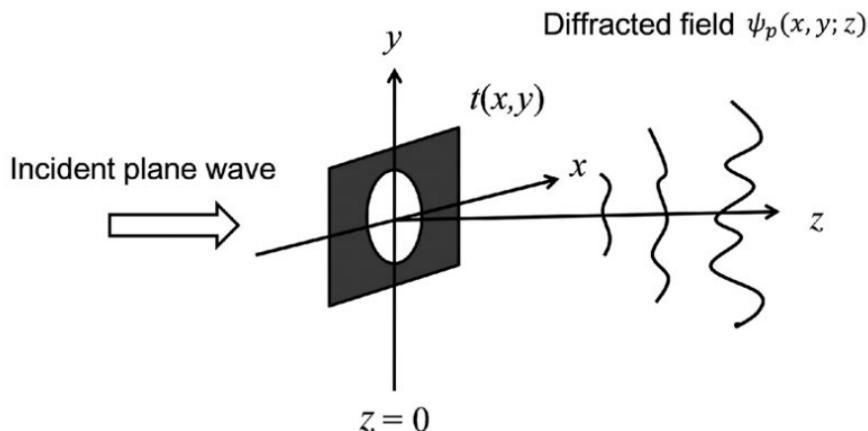


Figura 2.7: Geometria da difração, onde $t(x, y)$ é um ecrã de difração [24].

$$\mathcal{H}(k_x, k_y; z) = \exp \left[2j\pi z \sqrt{\frac{1}{\lambda^2} - k_x^2 - k_y^2} \right] \quad (2.2)$$

De notar que:

- λ é o comprimento de onda do feixe de luz;
- \mathcal{F} é a Transformada de Fourier;
- k_x e k_y são as frequências espaciais radianas.

A equação (2.1) é a fórmula de base do método ASM (também conhecido por Método da Dupla Transformada de Fourier ou Método de Convolução) [24].

2.3.3 Representação

Os dados holográficos podem ser representados de várias formas. Embora sejam todas equivalentes no sentido em que representam o mesmo objeto, algumas tornam a compressão mais eficiente.

No âmbito deste projeto, apenas é relevante a representação no campo de onda complexo, o qual pode englobar [9]:

- **Dados reais e imaginários:** utilizam um sistema de coordenadas cartesiano para representar amplitudes complexas;
- **Dados da amplitude e fase:** os valores complexos são expressos num sistema de coordenadas polares.

Os hologramas utilizados neste projeto são representados pelo formato de **amplitude-fase** (Figura 2.8).

2.4 Compressão de um Holograma

Dado que a dimensão dos ficheiros digitais pode ser avultada, tem sido do interesse da área da holografia estudar métodos eficientes de compressão que não comprometam a qualidade dos hologramas.

Neste sentido, o primeiro modelo de codificação e transmissão digital de hologramas foi proposto em 1991. Este envolvia a modulação dos dados num sinal de televisão com recurso a uma versão modificada do MPEG-2 [25]. Em 1993, o recurso aos formatos MPEG-1 e MPEG-2 foi novamente proposto para

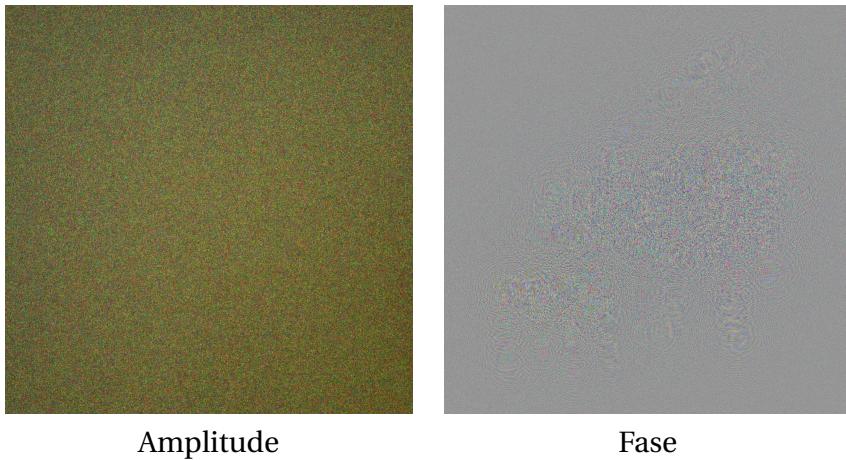


Figura 2.8: Exemplo do holograma piano4k no formato amplitude-fase.

a compressão de segmentos do holograma correspondentes a diferentes perspectivas de reconstrução [26, 27].

Contudo, apenas em 2002 foi concluído que os melhores rácios de compressão são esperados quando os hologramas digitais são representados pelas suas componentes real e imaginária de forma independente [28].

Desde então vários estudos têm sido feitos no sentido de perceber os melhores métodos de compressão, incluindo estudos baseados na quantização escalar [29] e formatos com perda (*e.g.* JPEG, JPEG2000, MPEG-4 AVC, Dirac e HEVC) [30–33].

Sendo o formato JPEG2000 o investigado no presente projeto, este será seguidamente abordado.

2.4.1 Uma Breve Introdução ao JPEG2000

O formato JPEG2000 é uma norma internacional para a compressão de imagens. Esta foi elaborada com o objetivo de mitigar os problemas e as limitações enfrentadas com o formato *Joint Photographic Experts Group* (JPEG) clássico de 1992. Uma das abordagens para alcançar este objetivo foi a substituição do uso de *Discrete Cosine Transform* (DCT) para um método baseado na transformada *wavelet* [34].

Conforme verificado pela Parte 1 da norma JPEG2000, algumas das vantagens a destacar são [34, 35]:

- *Eficiência de compressão:*
Em média 20 % superior face ao JPEG clássico.

- *Escalabilidade da qualidade:*
Possibilidade de extrair praticamente qualquer qualidade reduzida.
- *Escalabilidade da resolução:*
Possibilidade de extrair qualquer resolução que seja relacionada a uma potência de 2.
- *Acessibilidade da Region of Interest (ROI):*
Habilidade de reconstruir uma região espacial arbitrária.
- *Paralelismo:*
Capacidade de utilizar computação paralela com recurso a:
 - Múltiplos *cores* de uma *Central Processing Unit* (CPU);
 - Várias *threads* numa *Graphical Processing Unit* (GPU);
 - *Hardware* dedicado.
- *Controlo ótimo não-iterativo do rácio:*
Poder de alcançar um tamanho de compressão alvo sem recorrer a codificação iterativa.

Contudo, a norma JPEG2000 enfrenta como principal desvantagem o facto de ser computacionalmente bastante mais exigente, em especial com conteúdo de muito alta qualidade (*Ultra-High Definition* (UHD)).

A compressão nesta norma envolve várias fases, a saber [36]:

1. *Transformada de cor:*

As imagens no formato *Red, Green & Blue* (RGB) são inicialmente transformadas num novo espaço de cor, havendo duas possibilidades para tal:

- a) *Irreversible Color Transform* (ICT):

Transforma para o modelo YCbCr com recurso a números de vírgula flutuante ou de ponto fixo que geram erros de arredondamento (sendo assim chamado de “irreversível”).

- b) *Reversible Color Transform* (RCT):

Transforma para um modelo YUV modificado que não introduz erros de quantização (sendo, portanto, “reversível”). As transformações são:

$$Y = \left\lfloor \frac{R+2G+B}{4} \right\rfloor ; \quad G = Y - \left\lfloor \frac{C_B+C_R}{4} \right\rfloor ;$$

$$C_B = B - G ; \quad R = C_R + G ;$$

$$C_R = R - G ; \quad B = C_B + G .$$

2. Tiling:

A imagem é dividida em *tiles*, *i.e.* regiões retangulares que são codificadas separadamente. Cada *tile* pode ter um tamanho qualquer, podendo a imagem completa ser, no limite, uma *tile*. Todas as *tiles* têm o mesmo tamanho. Esta fase pode introduzir o mesmo efeito de blocos encontrado na norma JPEG clássica e pode ainda condicionar o valor de *Peak signal-to-noise ratio* (PSNR) da imagem comprimida.

3. Transformada wavelet:

Cada *tile* passa por uma transformada *wavelet* de tamanho arbitrário (*versus* o tamanho padrão de 8×8 do JPEG clássico). Semelhante à transformada de cor, existem dois tipos de transformada *wavelet*:

a) Irreversível:

Introduz ruído de quantização.

b) Reversível:

Recorre exclusivamente a coeficientes inteiros, pelo que não é necessário arredondamento, prevenindo assim a inserção de ruído de quantização [37–39].

4. Quantização:

Os coeficientes passam por uma quantização escalar a fim de reduzir o número de *bits* para os representar. O passo de quantização permite controlar a taxa de compressão: um maior passo de quantização aumenta a compressão, mas com a desvantagem natural de diminuir a qualidade final. Um passo de quantização unitário (*i.e.* com valor 1) leva a que este passo não seja sequer realizado.

5. Codificação:

Passo complexo que envolve um processo denominado *Embedded Block Coding with Optimal Truncation* (EBCOT). Os detalhes deste passo ultrapassam o âmbito do projeto.

Os ficheiros resultantes têm tradicionalmente a extensão *.jp2. Há ainda a referir o formato *.jpx com o qual se podem incluir animações.

Por fim, os metadados de uma imagem codificada com a norma JPEG2000 são armazenados no formato *Extensible Markup Language* (XML) [40], contrastando com o formato tradicional *Exif* utilizado pela norma JPEG clássica e outras normas de compressão e armazenamento de multimédia.

2.4.2 PSNR: uma Métrica de Relação Qualidade-Débito

Peak signal-to-noise ratio (PSNR) é uma métrica utilizada na área da engenharia que calcula o rácio entre a potência máxima possível de um sinal e a potência de sinais de ruído que afetam a fiabilidade da sua representação. Esta é expressa em *Decibel* (dB). A fórmula do PSNR é dada pela equação (2.3) [41],

$$\text{PSNR} = 20 \cdot \log(\text{MAX}_I) - 10 \cdot \log(\text{MSE}) \quad (2.3)$$

onde:

- MAX_I é o valor máximo possível do píxel da imagem (para uma representação com n bits, o valor será dado por $2^n - 1$);
- *Mean Squared Error* (MSE) (*i.e.* erro médio quadrado) é dado pela equação (2.4):

$$\text{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2 \quad (2.4)$$

Para multimédia codificada com 8 bits de profundidade em formatos com perdas, o valor de PSNR ronda tipicamente os 30 dB a 50 dB [42, 43].

2.5 Conclusões

Apesar da holografia ser uma área com décadas de história e pesquisa, esta ainda carece de normas bem estabelecidas a fim de tornar a sua manipulação, o seu estudo e o seu uso exequíveis de uma forma prática.

Assim, com base no conhecimento de interesse coletado acerca da holografia e da compressão de imagens, torna-se necessário definir quais os materiais, as ferramentas e as tecnologias necessárias para a concretização da investigação nesta área.

Capítulo

3

Tecnologias e Ferramentas Utilizadas

3.1 Introdução

O projeto apresentado só foi possível ser concluído devido à existência de tecnologias e ferramentas, assim como de materiais, os quais se revelaram essenciais.

Em particular, este Capítulo aborda:

- O *software* de reconstrução de hologramas;
- A escolha da linguagem de programação para o projeto;
- O *Software Development Kit* (SDK) de codificação de imagens no formato JPEG2000;
- O *hardware* utilizado;
- Os hologramas testados.

3.2 Tecnologias e Ferramentas

O desenvolvimento do projeto envolveu o trabalho conjunto de diversas ferramentas, nomeadamente Python 3, *Kakadu Software* e *software* escrito no âmbito do projeto JPEG Pleno.

3.2.1 Software de Reconstrução de Hologramas e sua Transcrição para Python

Do 80º Encontro do Grupo JPEG, realizado em Berlim entre 7 e 13 de julho de 2018, resultou um software desenvolvido por Antonin Gilles e Patrick Gioia, do *Institute of Research & Technology b<>com* [44]. O respetivo código, fornecido pela professora orientadora, encontra-se implementado em MATLAB.

Dada a ausência de uma licença do MATLAB para utilizar este *software* desenvolvido no âmbito do JPEG Pleno, foi necessário transcrever o código para uma nova linguagem de programação. Neste sentido, optou-se pelo Python 3.

O recurso a Python apresenta uma miríade de vantagens, entre elas:

- Linguagem *open source*;
- Utilização e distribuição gratuita da linguagem;
- Maior eficiência face ao MATLAB;
- Utilização comum no contexto de processamento multimédia e respetivos projetos;
- Vasto leque de bibliotecas, facilitando a implementação de *software* específico;
- Abundância de documentação;
- Forte comunidade *online* de suporte.

A escolha do Python foi, portanto, natural no âmbito do presente projeto.

Durante a fase de transcrição decorreu uma atualização do Python, tendo sido a última versão do código executada e testada na versão 3.8.2 em três distribuições GNU/Linux de 64 bits: Ubuntu 18.04 LTS, Fedora 31 e Linux Mint 20.

3.2.2 Kakadu Software

Uma vez que o JPEG2000 é o formato alvo deste projeto, e tendo em conta a dificuldade encontrada em projetos anteriores no contexto da holografia em utilizar a ferramenta *FFmpeg*, foi recomendado pela professora orientadora a utilização do *Kakadu Software* [45].

Este é um SDK para codificação e descodificação de imagens com recurso ao formato JPEG2000, segundo o *standard* pela JPEG.

Os comandos deste SDK de relevo para o projeto são os seguintes:

1. kdu_compress: codifica uma imagem para o formato JPEG2000.

Sintaxe:

```
kdu_compress -i input_file -o output_file -rate n
→ Cycc=<yes|no> -precise -quiet
```

onde:

- **-i input_file**: imagem de *input*;
- **-o output_file**: ficheiro de *output*;
- **-rate n**: número de *bits* por amostra (*n* pode ser um número flutuante);
- **Cycc=<yes|no>**: yes caso seja usada a codificação com transformada de cor de RGB para YCbCr, no em caso contrário;
- **-precise**: força o uso de representações de 32 *bits*;
- **-quiet**: suprime o *output* do programa.

2. kdu_expand: descodifica uma imagem no formato JPEG2000.

Sintaxe:

```
kdu_expand -i input_file -o output_file -rate n -quiet
```

onde:

- **-i input_file**: imagem de *input*;
- **-o output_file**: ficheiro de *output*;
- **-rate n**: número de *bits* por amostra (*n* pode ser um número flutuante);
- **-quiet**: suprime o *output* do programa.

3.3 Materiais Utilizados

3.3.1 Hardware

De notar que esta implementação do *software* de reconstrução de hologramas requer computadores com especificações mais generosas.

Para este projeto, dois computadores em particular executaram as várias iterações de desenvolvimento do *software* transcrito em Python:

1. *Desktop*: processador Ryzen™ 7 2700X 3.7–4.3GHz, placa gráfica NVidia® Quadro K5000 (4GB), memória RAM de 32GB e *swap* de 96GB, armazenamento SSD de 1TB;
2. Portátil: Intel® Core™ i5-10210U 1.6–4.2GHz, memória RAM de 16GB e *swap* de 8GB, armazenamento SSD de 512GB.

3.3.2 Hologramas

Os hologramas reconstruídos com o *software* desenvolvido no âmbito deste projeto são fornecidos pelo *Institute of Research & Technology b<>com* [20, 21, 46]. De entre os disponíveis, foram utilizados os seguintes hologramas com respetivas características resumidas na Tabela 3.1:

1. Dices4k (Figura 3.1);
2. DiffuseCar4k (Figura 3.2);
3. Piano4k (Figura 3.3).

Tabela 3.1: Especificações dos hologramas [20].

Holograma	size	pitch	r	g	b	z
Dices4k	4096×4096	0.4	640	532	473	0.164 – 0.328
DiffuseCar4k	4096×4096	0.4	640	532	473	0.110 – 0.250
Piano4k	4096×4096	0.4	640	532	473	0.170 – 0.313

size : Resolução	(em pixéis)
pitch : <i>Pixel pitch</i>	(em µm)
r : Comprimento de onda vermelho	(em nm)
g : Comprimento de onda verde	(em nm)
b : Comprimento de onda azul	(em nm)
z : Intervalo de localização da cena	(em cm)

3.4 Conclusões

Após a seleção das tecnologias e materiais, conforme supra-mencionados, ir-se-á proceder no Capítulo 4 ao delineamento da estratégia de investigação



Figura 3.1: Holograma Dices4k (imagem original) [20, 21, 46].



Figura 3.2: Holograma DiffuseCar4k (imagem original) [20, 21, 46].



Figura 3.3: Holograma Piano4k (imagem original) [20, 21, 46].

do projeto, a qual está intimamente ligada às escolhas apresentadas no presente Capítulo, entre elas a escolha da linguagem Python para transcrição do *software* original de reconstrução de hologramas, o SDK para realizar a codificação no formato JPEG2000 e os hologramas testados.

Capítulo

4

Etapas de Desenvolvimento e Implementação

4.1 Introdução

Para o sucesso deste projeto foi essencial delinear uma estratégia de investigação. Esta divide-se em 3 partes: reconstrução dos hologramas, compressão dos hologramas reconstruídos, e determinação da relação qualidade-débito da compressão. A cada uma destas fases atribuíram-se objetivos para a sua profícua execução:

1. *Reconstrução dos hologramas* (2º objetivo secundário do projeto):
 - a) Estudar a holografia;
 - b) Estudar as funções originais em MATLAB do Projecto JPEG Pleno;
 - c) Transcrever estas funções para a linguagem de programação Python;
 - d) Comparar o *output* entre as funções originais em MATLAB e as respetivas transcrições em Python;
 - e) Desenvolver um *script* para reconstruir cada holograma em 16 vistas distintas.
2. *Compressão dos hologramas reconstruídos* (3º objetivo secundário):
 - a) Estudar o uso do *Kakadu Software* (kdu);
 - b) Automatizar a execução dos *scripts* e *softwares* envolvidos;
 - c) Implementar um *script* para compressão (com e sem transformada de cor em diferentes *bitrates*) e descompressão dos hologramas reconstruídos.

3. *Determinação da relação qualidade-débito da compressão* (a fim de alcançar o 4º objetivo secundário, objeto de estudo do Capítulo 5):

- a) Calcular a relação qualidade-débito usando a métrica PSNR entre os hologramas originais e as imagens comprimidas;
- b) Determinar o melhor método de armazenamento dos débitos calculados;
- c) Implementar o *output* dos resultados no método selecionado;
- d) Gerar gráficos dos resultados.

As Secções subsequentes expõem o trabalho desenvolvido neste sentido e a Figura 4.1 esquematiza o processo levado a cabo para a execução do projeto.

4.2 Reconstrução dos Hologramas

O projeto iniciou-se com uma pesquisa exaustiva sobre a ciência da holografia, a qual resultou no Estado da Arte resumido no Capítulo 2.

Simultaneamente, foi efetuado um estudo das funções do *software* desenvolvido no âmbito do projeto JPEG Pleno a fim de se poder fazer a respetiva transcrição para Python. O resultado deste estudo e transcrição encontra-se exposto nas tabelas A.1, A.2 e A.3.

Estas funções foram compiladas num único *script* Python. A fim de cumprir o 2º objetivo secundário, uma nova função, `reconst_16views`, foi implementada (Tabela A.4). Esta função está, portanto, encarregue de abrir um holograma, as respetivas especificações e reconstruí-lo em exatamente 16 vistas distintas. De notar que as especificações são fornecidas por um ficheiro *JavaScript Object Notation* (JSON) cujo conteúdo inclui:

- `holo_size`: Resolução do holograma (em pixéis);
- `pitch`: Distância entre pixéis (em metros);
- `wavelengths`: Lista com os comprimentos de onda respetivos aos canais RGB (em metros);
- `z`: Distância de reconstrução (em metros);
- `pupil_size`: Lista que contém o tamanho da janela de reconstrução (em pixéis).

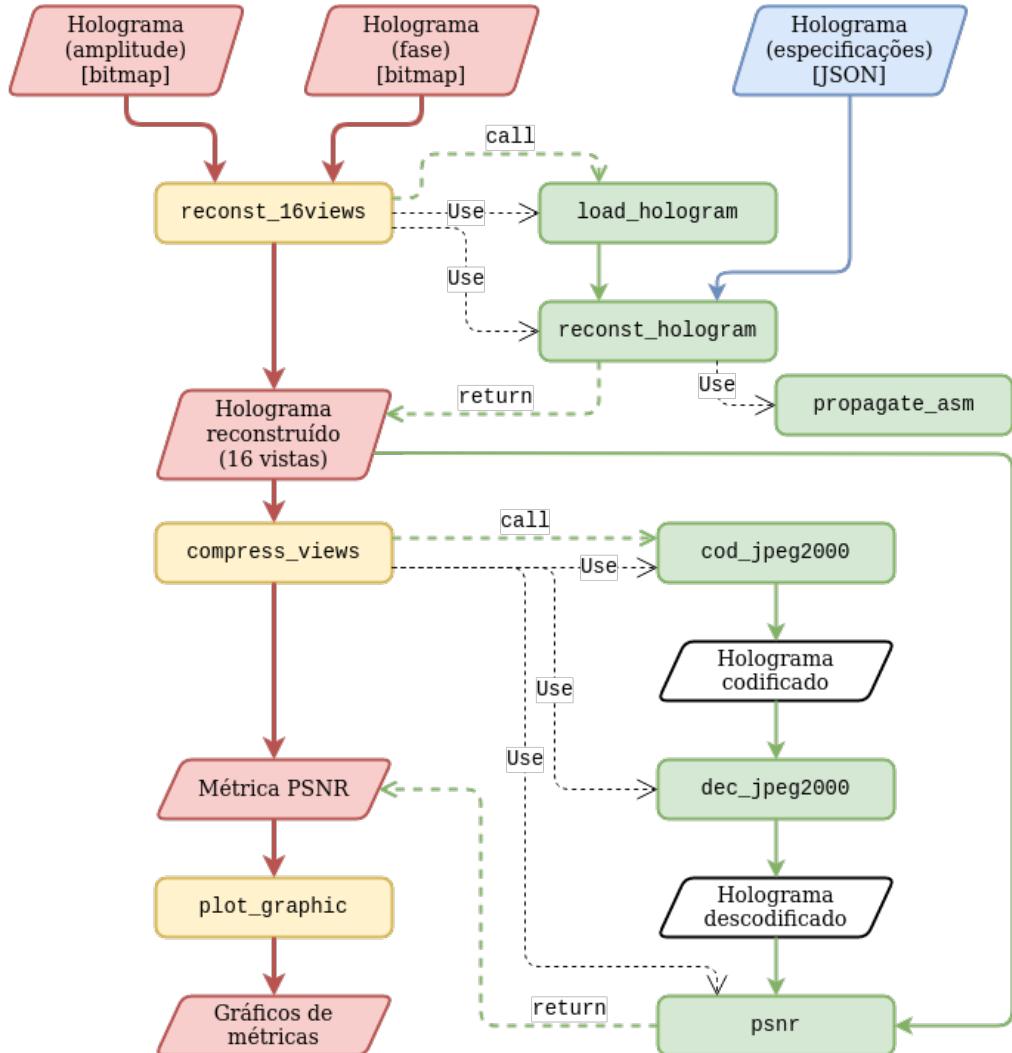


Figura 4.1: Diagrama do processo efetuado sobre um holograma para obtenção dos resultados.

O caminho vermelho/amarelo representa o trajeto principal pelo qual o holograma é processado. Os caminhos verdes detalham os processos intermédios correspondentes a um determinado processo do trajeto principal (a entrada é dada por “call” e o retorno é dado por “return”). As dependências entre as principais funções são indicadas por “Use”. A azul estão os ficheiros externos auxiliares.

Todavia, havendo uma transcrição de funções entre linguagens de programação, exigiu-se uma forma de comprovar que as funções transcritas em Python são equivalentes às originais em MATLAB. Para este fim, delineou-se a seguinte estratégia:

1. Etapa de implementação:
 - a) Transcrever as funções para Python (forme apresentado).
2. Etapa de reconstrução:
 - a) Reconstruir os hologramas numa vista com as funções originais em MATLAB;
 - b) Reconstruir os hologramas numa vista com as funções transcritas em Python.
3. Etapa de comparação e resultados:
 - a) Comparar os hologramas produzidos pelos *scripts* das duas linguagens através de uma análise visual a olho nu;
 - b) Comparar os hologramas produzidos pelos *scripts* das duas linguagens através da métrica PSNR.

Com o *script* devidamente implementado e testado, prosseguiu-se com a fase seguinte do projeto relativo à compressão dos hologramas agora reconstruídos.

4.3 Compressão dos Hologramas Reconstruídos

Após a reconstrução dos hologramas em multivista, segue-se o teste à norma JPEG2000 através de compressão e descompressão de forma a comparar com a reconstrução do holograma original (cuja métrica usada para comparação será abordada na Secção 4.4).

Conforme a estratégia delineada na Secção 4.1, esta fase do projeto envolveu o estudo da ferramenta kdu, a implementação dos *scripts* e o cálculo do PSNR.

4.3.1 Breve Estudo do *Kakadu Software*

A fase de estudo do kdu resultou no conhecimento exposto na Secção 3.2.2, onde se encontram descritos os dois comandos utilizados no âmbito do projeto.

4.3.2 Implementação e Execução do *Script* de Compressão

De forma a se poder comparar o holograma original com o holograma comprimido no formato JPEG2000, revela-se necessário realizar uma sequência de compressão e descompressão do primeiro. Tal irá dar origem a um holograma reconstruído após compressão, o qual poderá ser comparado com o original.

Conforme visto anteriormente, a ferramenta kdu é a ferramenta eleita para este processo. Contudo, não é prático executar manualmente a sequência de comandos na *shell*. Por conseguinte, um *script* escrito em Python foi implementado. Este inclui a função `compress_views` (Tabela A.5 e Figura A.2), a qual executa a seguinte sequência de operações:

1. *Compressão do holograma original, previamente reconstruído.*
É feita uma chamada ao sistema do comando `kdu_compress`.
Esta encontra-se encapsulada na função auxiliar `cod_jpeg2000` (Tabela A.6).
2. *Descompressão do holograma agora comprimido.*
É feita uma chamada ao sistema do comando `kdu_expand`.
Esta encontra-se encapsulada na função auxiliar `dec_jpeg2000` (Tabela A.7).
3. *Cálculo da relação qualidade-débito através de uma métrica de compressão.*
O holograma original e o holograma agora descomprimido são comparados com recurso à métrica PSNR (Secção 4.4).
Esta operação é realizada pela função auxiliar `psnr` (Tabela A.8).

De notar os *bitrates* testados: 0.1, 0.3, 0.6, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5 e 5.0. Há ainda a referir a realização dos testes com e sem transformada de cor.

Por fim, de forma a facilitar a extração dos resultados pretendidos segundo o objetivo primário do projeto (Secção 1.3), é gerado um ficheiro JSON por cada holograma. Este tem a seguinte estrutura:

O ficheiro JSON supra-mencionado será portanto utilizado no processo descrito na Secção 4.4.

4.3.3 Automatização dos *Scripts*

Para a otimização do processo de investigação neste projeto, foram criados dois *scripts* adicionais, os quais trabalham em conjunto a fim de automatizar todo este processo. São eles os seguintes:

```

<nome_holograma>.json
  └── no_ycbcrr
      └── <rate> ..... Número de bits por amostra
          ├── <vista>.ppm: valor ..... Posição da janela de reconstrução
          ├── avg: valor ..... PSNR médio das vistas para este rate
          └── std: valor ..... Desvio padrão
  └── ycbcr
      └── <rate>
          ├── <vista>.ppm: valor
          ├── avg: valor
          └── std: valor

```

1. *Makefile*: Introduz as 8 opções infra-apresentadas, as quais invocam o script *main.py*:
 - a) *reconst*: reconstrói o holograma fornecido por argumento em 16 vistas (Capítulo 4.2 e Tabela A.4);
 - b) *compress*: executa o *script* apresentado na Secção 4.3.2;
 - c) *plot*: calcula os gráficos com os resultados de comparação da compressão com e sem transformada de cor (Secção 4.4);
 - d) *all*: executa os 3 itens anteriores;
 - e) *test*: opção utilizada na fase de *debugging*;
 - f) *install*: instala pacotes no sistema operativo essenciais à execução dos *scripts*;
 - g) *clean*: remove os diretórios relativos a: hologramas reconstruídos, *output* do *kdu*, ficheiros JSON dos resultados do PSNR, e ficheiros de *cache* do Python;
 - h) *clean-compressed*: remove os diretórios relativos a: *output* do *kdu*, ficheiros JSON dos resultados do PSNR, e ficheiros de *cache* do Python.
2. *main.py*: *script* Python que executa as primeiras 5 opções supra-mencionadas conforme o argumento passado pelo *Makefile*. Faz uso das funções *reconst_16views* (Tabela A.4) e *compress_views* (Tabela A.5).

4.4 Determinação da Relação Qualidade Débito

Na sequência do processo de compressão e descompressão no formato JPEG2000, e conforme previamente introduzido na Secção 4.3.2, a determinação da re-

lação qualidade-débito da compressão constitui a última fase da investigação a fim destes resultados poderem ser escrutinados no Capítulo 5.

Variadas métricas estão disponíveis para se poder comparar amostras e, no caso do presente projeto, avaliar a qualidade do holograma após compressão no formato JPEG2000. De entre as métricas existentes, foi eleita a *Peak signal-to-noise ratio* (PSNR) por indicação da professora orientadora, a qual está envolvida no Projeto JPEG Pleno.

Conforme introduzido na Secção 4.3.2, foi utilizada uma função auxiliar `psnr` (Tabela A.8), invocada pela função principal `compress_views` (Tabela A.5). Este método de implementação foi escolhido para fins de eficiência de armazenamento.

Uma vez que não é mandatório armazenar os hologramas resconstruídos após compressão, sendo apenas do nosso interesse os valores de PSNR finais a si referentes, a função `dec_jpeg2000` exporta o holograma descomprimido num ficheiro temporário `temp.ppm`. Este é utilizado de seguida pela função `psnr` para cálculo do PSNR (sendo os resultados armazenados num ficheiro JSON), sendo de seguida eliminado.

4.4.1 Criação de gráficos

O ficheiro JSON gerado (descrito na Secção 4.3.2) é posteriormente carregado num novo *script* Python, especificamente implementado para gerar gráficos em si baseados para mais fácil visualização, análise e discussão (Capítulo 5). Este *script* recorre à biblioteca `matplotlib.pyplot`.

A geração de um gráfico por holograma por parte deste *script* marca o fim do processo de investigação relativo à manipulação dos hologramas e extração dos valores de PSNR.

4.5 Conclusões

Após a conclusão das 3 fases de investigação segundo a estratégia delineada inicialmente e ilustrada pela Figura 4.1, resta a exposição e análise dos resultados obtidos no Capítulo 5.

A estratégia desenhada revelou-se eficaz e permitiu a obtenção de *scripts* facilmente adaptáveis para trabalhos futuros no âmbito do Projeto JPEG Pleno, assim como os próprios resultados para análise por parte de outros interessados. De igual forma, a escolha de *standards* gratuitos e *open-source* (justificada na Secção 3.2) revelou-se acertada pelos mesmos motivos.

Capítulo

5

Testes e Resultados

5.1 Introdução

A implementação das funções e dos *scripts* do projeto e respetiva execução com os hologramas selecionados para este culmina no cumprimento do 4º e último objetivo secundário do projeto, o qual permitirá retirar conclusões para o objetivo primário identificado na Secção 1.3.

Para tal, ir-se-ão resumir os testes efetuados (Secção 5.2) e os resultados de si obtidos (Secção 5.3), assim como discuti-los na perspetiva do objetivo do projeto (Secção 5.4).

5.2 Testes

A sequência de fases de investigação apresentada no Capítulo 4 e resumida na *pipeline* ilustrada pela Figura 4.1 foi exaustivamente testada com os 3 hologramas enumerados na Secção 3.3.2.

Os parâmetros apresentados na Tabela 3.1 são fixos (a fim de garantir a reconstrução do holograma), sendo a distância de reconstrução (parâmetro *z*) em particular um intervalo. Qualquer valor dentro deste intervalo pode ser tomado. A par deste parâmetro, também o tamanho da vista (valor arbitrário menor do que a resolução do holograma) é necessário para a fase de testes, conforme as funções transcritas e implementadas ao longo do Capítulo 4. Os valores selecionados para estes dois parâmetros são apresentados na Tabela 5.1.

Os ficheiros JSON gerados pelos *scripts* previamente implementados armazenam, conforme descrito na Secção 4.3.2, os resultados doravante explorados na Secção 5.3.

Tabela 5.1: Parâmetros variáveis utilizados na reconstrução dos hologramas.

Holograma	<i>z</i>	pupil_size
Dices4k	0.0020	2048×2048
DiffuseCar4k	0.0012	2048×2048
Piano4k	0.0019	2048×2048

z : Distância de reconstrução (em metros)

pupil_size : Tamanho da vista (em pixels)

5.3 Resultados

Os resultados dos testes efetuados são os valores da métrica PSNR, em particular um por holograma, por *bitrate* (débito), por vista, e com e sem transformada de cor. No caso deste projeto, tal traduz-se em $3 \times 12 \times 16 \times 2 = 1152$ valores. Ora, esta quantidade de valores é impraticável de ser exposta e devidamente explorada.

Neste sentido, optou-se por se calcular a média dos valores de PSNR e o respetivo desvio padrão por cada *bitrate*, o que reduz para 72 valores, tornando-se alcançável a devida discussão dos resultados do projeto.

Destacam-se as tabelas e figuras que expõem os resultados da investigação. Porquanto as tabelas dispõem os valores das médias da métrica PSNR para as 16 vistas reconstruídas por cada *bitrate* e por opção de com ou sem de transformada de cor, as figuras apresentam-nos graficamente. Em particular:

- Holograma dices4k: Tabela 5.2 e Figura 5.1;
- Holograma diffuseCar4k: Tabela 5.3 e Figura 5.2;
- Holograma piano4k: Tabela 5.4 e Figura 5.3.

5.4 Discussão

Tendo em conta que quanto maior o valor do PSNR melhor será a qualidade da imagem após a compressão, uma análise visual aos gráficos das figuras 5.1, 5.2 e 5.3 permite facilmente observar a diminuição da qualidade de todos os três hologramas testados ao ser utilizada uma transformada de cor de RGB para YCbCr.

Tabela 5.2: Resultados da métrica PSNR para o holograma dices4k.

Bitrate	Sem transformada de cor		Com transformada de cor	
	avg	std	avg	std
0.1	25.432	3.852	24.768	3.684
0.3	28.513	4.772	27.071	4.662
0.6	31.221	5.427	29.311	5.160
1.0	33.723	5.919	31.176	5.591
1.5	36.210	6.160	33.000	5.870
2.0	38.212	6.270	34.743	6.369
2.5	39.942	6.337	35.930	6.410
3.0	41.418	6.116	37.445	6.658
3.5	42.824	6.114	38.966	6.676
4.0	44.097	5.957	39.966	6.521
4.5	45.310	5.704	40.962	6.645
5.0	46.209	5.411	42.084	6.780

avg : Média do PSNR (em dB)

std : Desvio padrão (em dB)

Há ainda a referir, pela observação das mesmas figuras, que a qualidade dos hologramas comprimidos aumenta com a subida do débito.

Tal é conforme o esperado uma vez que o aumento do débito implica a possibilidade de se armazenar uma maior quantidade de informação acerca da imagem original.

Por seu turno, de uma análise mais detalhada às Tabelas 5.2, 5.3 e 5.4 constata-se a ocorrência de um fenómeno igualmente antecipado nos valores do desvio padrão: este tende a diminuir para *bitrates* mais elevados.

Do exposto, 3 questões colocam-se e que são meritórias de discussão no âmbito da investigação:

1. *Qual o motivo para a transformada de cor levar à significativa redução de qualidade após compressão?*

A transformada de cor de RGB para YUV 4:2:0 implica perda de informação nos canais G (*Green*) e B (*Blue*) a fim de tirar partido das características da visão humana. Por si só, esta transformada de cor torna a perda de informação imperceptível ao olho humano, sendo esta de uma aparente qualidade equivalente.

Todavia, a compressão no formato JPEG2000 implica uma nova perda

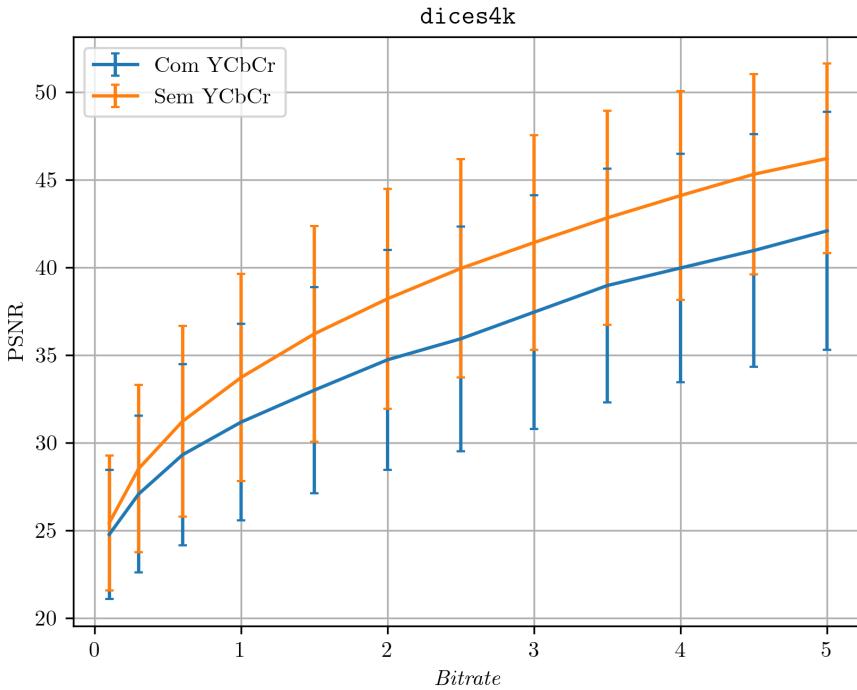


Figura 5.1: Variação da média da métrica PSNR relativo ao holograma dices4k. *Valor maior significa melhor qualidade.*

de informação uma vez que este é um formato de compressão com perdas.

Porquanto a transformada de cor de RGB para YUV 4:2:0 não se traduz numa redução significativa em imagens clássicas, tal não pode ser dito para a holografia. A quantidade de *speckle* (*i.e.* de ruído) nos hologramas leva a que haja uma diminuição da correlação entre píxeis vizinhos nos canais U e V, levando a que a redução destes coeficientes não seja mais negligenciável no que diz respeito à qualidade final. Tal acaba por se traduzir numa diminuição mais significativa do valor de PSNR uma vez que existem perdas que se acentuam nos passos seguintes da compressão.

É, portanto, teorizado que, no caso dos hologramas, a aplicação consecutiva de dois processos que implicam perdas de informação levem a uma significativa redução da qualidade final do holograma comprimido, resultando em valores de PSNR claramente inferiores.

Tabela 5.3: Resultados da métrica PSNR para o holograma diffuseCar4k.

Bitrate	Sem transformada de cor		Com transformada de cor	
	avg	std	avg	std
0.1	29.441	4.763	27.438	3.795
0.3	33.938	6.170	29.762	5.089
0.6	38.299	6.820	32.336	6.019
1.0	42.364	6.864	35.398	7.046
1.5	46.053	6.540	38.747	7.682
2.0	48.611	5.938	41.496	7.635
2.5	50.706	5.339	44.037	7.577
3.0	52.038	4.599	45.643	6.937
3.5	53.174	3.899	47.263	6.262
4.0	54.096	3.318	48.878	5.400
4.5	54.819	2.812	50.110	4.567
5.0	55.280	2.310	50.851	3.706

avg : Média do PSNR (em dB)

std : Desvio padrão (em dB)

2. *Por que razão o desvio padrão se revela menor em bitrates maiores?*

O aumento da qualidade dos hologramas comprimidos com *bitrates* maiores (conforme observado anteriormente) dever-se-á explicar pela maior quantidade de informação armazenada. Tal leva à existência de menos ruído nas imagens comprimidas, o que implica uma maior aproximação face à imagem original.

A par do aumento do PSNR, é de denotar que o aumento da informação armazenada deverá igualmente uma maior consistência na qualidade dos hologramas comprimidos, independentemente da vista tomada.

Ora, estatisticamente, uma maior consistência dos dados traduz-se na diminuição do desvio padrão.

3. *Por que razão o desvio padrão se revela menor em bitrates menores?*

Paralelamente ao ponto anterior, a diminuição do débito introduz um considerável aumento do ruído nos hologramas comprimidos, o que leva a um decaimento rápido do valor de PSNR à medida que se diminui o *bitrate* utilizado.

É, pois, expectável que não existam grandes variações no PSNR inde-

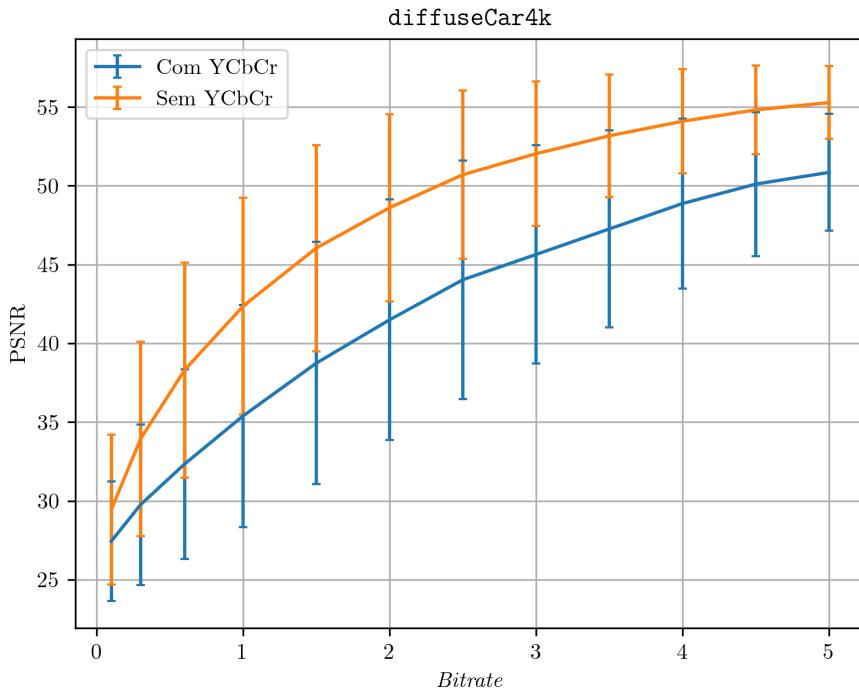


Figura 5.2: Variação da média da métrica PSNR relativo ao holograma *diffuseCar4k*. *Valor maior significa melhor qualidade.*

pendentemente da vista tomada na reconstrução do holograma. Por outras palavras, é de esperar consistência na falta de qualidade das imagens (*e.g.* é altamente improvável obter um PSNR de 50 dB quando se obtém consistentemente valores a rondar os 20 dB para um *bitrate* de 0.1).

Tal deverá justificar o fenómeno da notória diminuição do desvio padrão para *bitrates* mais baixos.

De referir que a reconstrução em multivista permite verificar que a qualidade de compressão pode igualmente depender da vista a partir da qual é reconstruída.

5.5 Conclusões

A análise de três grandes variáveis foi levada em consideração durante a investigação e que neste Capítulo foram abordadas:

Tabela 5.4: Resultados da métrica PSNR para o holograma piano4k.

Bitrate	Sem transformada de cor		Com transformada de cor	
	avg	std	avg	std
0.1	27.352	3.958	26.156	3.579
0.3	31.336	4.660	29.166	4.538
0.6	34.930	4.847	31.711	4.558
1.0	38.291	4.827	33.782	4.631
1.5	41.393	4.697	36.231	4.775
2.0	43.914	4.291	38.247	4.858
2.5	45.947	4.019	40.340	4.911
3.0	47.654	3.632	41.937	4.582
3.5	48.987	3.236	43.430	4.722
4.0	50.300	2.873	45.279	4.550
4.5	51.290	2.715	46.741	4.316
5.0	52.210	2.471	47.638	3.642

avg : Média do PSNR (em dB)

std : Desvio padrão (em dB)

1. Reconstrução em 16 vistas distintas;
2. Uso de uma transformada de cor de RGB para YCbCr;
3. Uso de 12 *bitrates* distintos.

Tendo sido assim cumpridos os objetivos secundários propostos para o projeto e após a discussão dos resultados obtidos sob a perspectiva de três questões, a retirada de conclusões para responder ao objetivo primário torna-se, portanto, exequível.

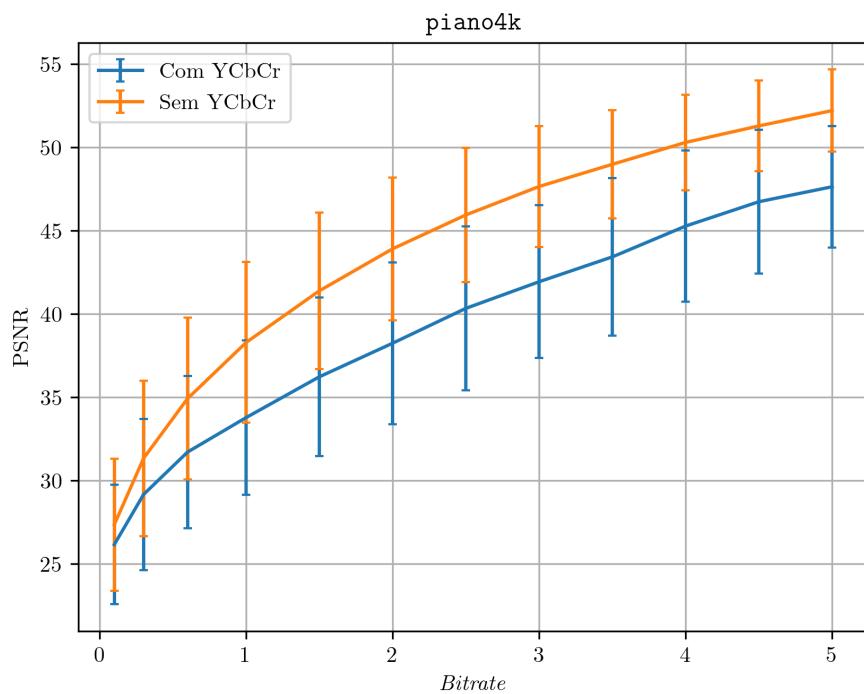


Figura 5.3: Variação da média da métrica PSNR relativo ao holograma piano4k. *Valor maior significa melhor qualidade.*

Capítulo

6

Conclusões e Trabalho Futuro

6.1 Conclusões Principais

Com o término da investigação apresentada, foram alcançadas as seguintes etapas:

- Implementação de um conjunto de *scripts* para a reconstrução, compressão e descompressão de hologramas com cor em multivista;
- Efetiva reconstrução de 3 hologramas em 16 vistas distintas;
- Utilização do codificador JPEG2000 para a compressão dos hologramas reconstruídos;
- Estudo dos hologramas comprimidos com a métrica PSNR.

A frutífera conclusão destas etapas permitiu o alcance dos objetivos inicialmente traçados para este projeto.

Neste sentido, e após os resultados expostos na Secção 5.3 e a sua respetiva discussão na Secção 5.4, podem-se enumerar as seguintes conclusões acerca do uso do formato JPEG2000 para a compressão de hologramas no plano do objeto:

1. O recurso à transformada de cor não é desejável;
2. O uso de maiores *bitrates* produz hologramas comprimidos de melhor e mais consistente qualidade para diferentes vistas.
3. Considerando que um PSNR entre 30 dB e 50 dB é assumido como um intervalo que indica qualidade aceitável (*i.e.* perdas visualmente imperceptíveis) para multimédia comprimida em formatos com perda [42, 43],

débitos tão baixos quanto 0.3 revelaram-se suficientes para uma compressão satisfatória.

Por consequência, e a fim de responder ao objetivo primário do presente projeto, conclui-se que o formato JPEG2000, no âmbito do Projeto JPEG Pleno, é adequado para a compressão de hologramas com cor em multivista.

6.2 Trabalho Futuro

No âmbito do projeto inicialmente proposto, foram cumpridos todos os objetivos delineados.

Contudo, durante a investigação, foi notória a ineficiência do uso de uma única *thread* conforme implementado nos *scripts* desenvolvidos. Seria, portanto, interessante o uso de *multithreading* a fim de alcançar um menor tempo de execução e uma vasta melhoria no uso dos recursos disponibilizados pelos *hardwares* utilizados (mencionados na Secção 3.3.1). Outra alternativa seria o uso de computação paralela com recurso a placas gráficas, a par da técnica utilizada pelos investigadores do *Institute of Research & Technology b>com* (e.g. com recurso à tecnologia *Compute Unified Device Architecture* (CUDA) [21]).

A par do concluído na Secção 6.1, propõe-se o estudo alargado a mais hologramas para se poder determinar se existe um *bitrate* mínimo comum aceitável ou se este deve sempre depender do holograma original.

No mesmo sentido, caso o débito dependa do holograma, é plausível a pertinência de um estudo sobre métodos para determinação ótima do débito aquando da compressão.

Ademais, não tendo sido objetivo do presente projeto, é de alto interesse o estudo da redução do *bitrate* utilizado na compressão a fim de alcançar hologramas comprimidos sem perda de qualidade percetível e um tamanho de ficheiro o mais reduzido possível.

Por fim, o estudo da determinação das vistas com PSNR ótimo é uma natural extensão do presente projeto. Tendo em conta que o uso de multivista não levou a uma melhoria na utilização da transformada de cor, a redução do *speckle* (ruído) pode ser estudada a fim de verificar se existe melhoria dos resultados com o seu uso.

Apêndice

A

Documentação de Funções

A.1 Função load_hologram

Tabela A.1: Documentação da função `load_hologram` [44].

Nome da função
<code>load_hologram</code>
Protótipo original em MATLAB
<code>function [hologram] = load_hologram(ampli_path, phase_path)</code>
Protótipo transscrito em Python
<code>def load_hologram(ampli_path, phase_path)</code>
Descrição
Esta função carrega um holograma da base de dados b>com a partir dos seus ficheiros de amplitude e fase.
Inputs
ampli_path: Diretório do ficheiro da imagem da amplitude (caminho relativo ou absoluto). phase_path: Diretório do ficheiro da imagem da fase (caminho relativo ou absoluto).
Output
Modulação complexa do holograma (3 canais: RGB).
Efeitos colaterais
Não aplicável.
Dependências
Não aplicável.

A.2 Função propagate_asm

Tabela A.2: Documentação da função propagate_asm [44].

Nome da função
propagate_asm
Protótipo original em MATLAB
<pre>function [v] = propagate_asm(u, pitch, wavelength, z)</pre>
Protótipo transscrito em Python
<pre>def propagate_asm(u, pitch, wavelength, z)</pre>
Descrição
Esta função simula a propagação no plano complexo u sobre a distância z utilizando o ASM.
Inputs
u: Campo de onda de luz do plano de <i>input</i> (um canal). pitch: Distância entre pixeis (em metros). wavelength: Comprimento de onda do canal de cor a propagar (em metros). z: Distância de propagação ao longo do eixo ótico (em metros).
Output
Campo de onda de luz no plano de destino (um canal).
Efeitos colaterais
Não aplicável.
Dependências
Não aplicável.

A.3 Função reconst_hologram

Tabela A.3: Documentação da função `reconst_hologram` [44].

Nome da função
<code>reconst_hologram</code>
Protótipo original em MATLAB
<pre>function [recons] = reconsHologram(hologram, pitch, wavelengths, z, pupilPos, pupilSize)</pre>
Protótipo transrito em Python
<pre>def reconst_hologram(hologram, pitch, wavelengths, z, pupil_pos, pupil_size)</pre>
Descrição
<p>Esta função reconstrói o holograma a uma distância z, utilizando o ASM. Permite o uso de uma janela para obter reconstruções de diferentes pontos de vista.</p>
Inputs
<p><code>hologram</code>: Holograma de modulação complexa (3 canais: RGB).</p> <p><code>pitch</code>: Distância entre pixeis (em metros).</p> <p><code>wavelengths</code>: Comprimentos de onda de luz (em metros, 3 canais: RGB).</p> <p><code>z</code>: Distância de reconstrução (em metros).</p> <p><code>pupilPos</code>: Posição da janela (em pixeis, canto superior direito). <i>Valor por defeito.</i> [0, 0].</p> <p><code>pupilSize</code>: Tamanho da janela (em pixeis, altura \times largura). <i>Valor por defeito.</i> None.</p>
Output
Reconstrução numérica do holograma (3 canais: RGB).
Efeitos colaterais
Não aplicável.
Dependências
Função <code>propagate_asm</code> (Tabela A.2).

A.4 Função `reconst_16views`

Tabela A.4: Documentação da função `reconst_16views`.

Nome da função	<code>reconst_16views</code>
Protótipo em Python	<code>def reconst_16views(hologram)</code>
Descrição	<p>Reconstrói o holograma fornecido por argumento em 16 vistas.</p> <p><i>Processo.</i> Carrega o holograma (dois ficheiros <i>bitmap</i>: amplitude e fase) e o respetivo ficheiro de especificações (formato JSON). Calcula as posições das 16 vistas tendo em conta o tamanho do holograma.</p>
Inputs	<p><code>hologram</code>: Nome do holograma a ser reconstruído.</p> <p><i>Valor por defeito.</i> “dices4k”.</p>
Output	Não aplicável.
Efeitos colaterais	Produz, dentro da pasta <code>./reconst/</code> , uma nova pasta com o nome do holograma. O respetivo conteúdo incluirá as 16 vistas (ficheiros <code>*.ppm</code>) correspondentes às reconstruções produzidas pela função.
Dependências	<p>Função <code>load_hologram</code> (Tabela A.1);</p> <p>Função <code>reconst_hologram</code> (Tabela A.3).</p> <p><i>Ver Figura A.1.</i></p>

A.5 Função compress_views

Tabela A.5: Documentação da função compress_views.

Nome da função	compress_views
Protótipo em Python	<pre>def compress_views(hologram, ycbcr, rate)</pre>
Descrição	<p>Comprimir os hologramas e calcular o PSNR.</p> <p><i>Processo.</i> Para cada vista, o holograma é codificado, descodificado e analisado em termos do PSNR. São criados 16 ficheiros *.jp2 por pasta <code>rate_n</code>, correspondentes às 16 vistas reconstruídas, assim como um ficheiro JSON por holograma na pasta <code>kduOutput</code> com as métricas da relação qualidade-débito da compressão (PSNR).</p>
Inputs	<p><code>hologram</code>: Nome do holograma a comprimir. <i>Valor por defeito.</i> “dices4k”.</p> <p><code>ycbcr</code>: <i>Flag</i> que indica ao kdu se deve ser efetuada uma transformada de cor. <i>Valor por defeito.</i> False.</p> <p><code>rate</code>: número de <i>bits</i> por amostra. <i>Valor por defeito.</i> 1.0.</p>
Output	Não aplicável.
Efeitos colaterais	Armazena as imagens no formato *.jp2 nas pastas <code>rate_n</code> , assim como os ficheiros JSON na pasta <code>kduOutput</code> , segundo a seguinte árvore de diretórios:
	<pre>./kduOutput └── <nome_holograma> ├── no_ycbcr └── rate_n..... n: número de bits └── ycbcr</pre>
Dependências	<p>Função <code>cod_jpeg2000</code> (Tabela A.6); Função <code>dec_jpeg2000</code> (Tabela A.7); Função <code>psnr</code> (Tabela A.8).</p> <p><i>Ver Figura A.2.</i></p>

A.6 Função cod_jpeg2000

Tabela A.6: Documentação da função cod_jpeg2000.

Nome da função	
	cod_jpeg2000
Protótipo em Python	
	<code>def cod_jpeg2000(in_path, out_path, ycbcr, rate)</code>
Descrição	
	Invoca ao sistema a execução do comando kdu_compress, conforme descrito na Secção 3.2.2.
Inputs	
	<p><code>in_path</code>: caminho do ficheiro de <i>input</i>. <code>out_path</code>: caminho do ficheiro de <i>output</i>. <i>Valor por defeito</i>. “out.jp2”.</p> <p><code>ycbcr</code>: <i>flag</i> que indica ao kdu se deve ser efetuada uma transformada de cor. <i>Valor por defeito</i>. False.</p> <p><code>rate</code>: número de <i>bits</i> por amostra. <i>Valor por defeito</i>. 1.0.</p>
Output	
	Não aplicável.
Efeitos colaterais	
	Gera uma imagem no formato *.jp2 no diretório definido por <code>out_path</code> .
Dependências	
	Não aplicável.

A.7 Função dec_jpeg2000

Tabela A.7: Documentação da função dec_jpeg2000.

Nome da função
dec_jpeg2000
Protótipo em Python
<code>def dec_jpeg2000(in_path, out_path, ycbcr, rate)</code>
Descrição
Invoca ao sistema a execução do comando kdu_expand, conforme descrito na Secção 3.2.2.
Inputs
in_path: caminho do ficheiro de <i>input</i> . out_path: caminho do ficheiro de <i>output</i> . <i>Valor por defeito</i> . “out.jp2”. ycbcr: <i>flag</i> que indica ao kdu se deve ser efetuada uma transformada de cor. <i>Valor por defeito</i> . False. rate: número de <i>bits</i> por amostra. <i>Valor por defeito</i> . 1.0.
Output
Não aplicável.
Efeitos colaterais
Gera uma imagem no formato *.tmp no diretório definido por out_path.
Dependências
Não aplicável.

A.8 Função psnr

Tabela A.8: Documentação da função psnr.

Nome da função
psnr
Protótipo em Python
<code>def psnr(p1, p2)</code>
Descrição
Calcula a relação qualidade-débito da compressão com a métrica PSNR entre duas imagens.
Inputs
p1: caminho para a primeira imagem. p2: caminho para a segunda imagem.
Output
Valor calculado do PSNR.
Efeitos colaterais
Não aplicável.
Dependências
Não aplicável.

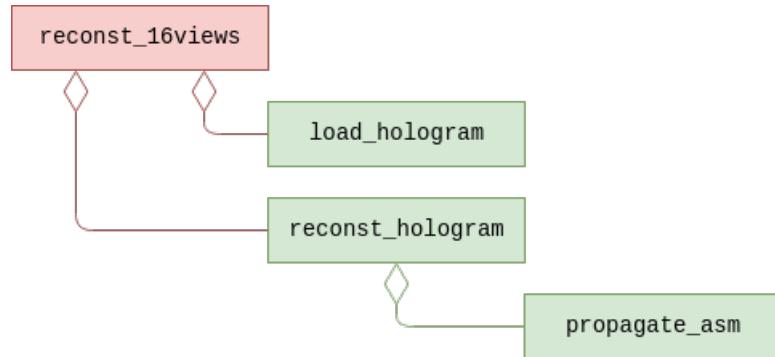


Figura A.1: Diagrama de dependências da função `reconst_16views`.

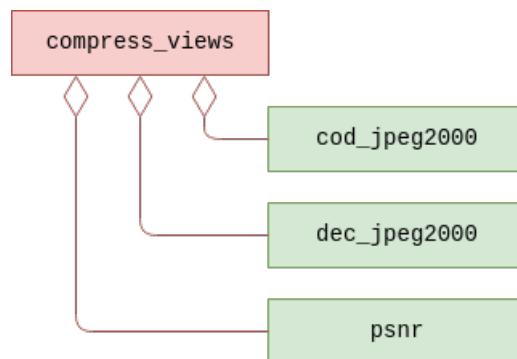


Figura A.2: Diagrama de dependências da função `compress_views`.

Bibliografia

- [1] HoloCenter — Center for the Holographic Arts, “What is a Hologram? — holocenter,” 2020, Último acesso a 5 de setembro de 2020. [Online]. Available: <http://holocenter.org/what-is-holography/what-is-a-hologram>
- [2] F. M. Grimaldi, *Physico-Mathesis De Lumine, Coloribus, Et Iride, Aliisque Adnexis Libri Duo* (1665). Kessinger Legacy Reprints, 2010.
- [3] F. Cajori, *A history of physics in its elementary branches, including the evolution of physical laboratories*. Nova Iorque: The Macmillan Company, 1929.
- [4] A. Stevenson, *Shorter Oxford English dictionary on historical principles*. Oxford New York: Oxford University Press, 2007.
- [5] M. Waite, *Pocket Oxford English dictionary*. Oxford New York: Oxford University Press, 2013.
- [6] F. Hwang, Fu-Kwun; Esquembre, “Generation of an interference pattern from two-slit diffraction,” 2011, Último acesso a 5 de setembro de 2020. [Online]. Available: <https://commons.wikimedia.org/w/index.php?curid=17003343>
- [7] Haade, “Interference of two waves,” 2010, Último acesso a 5 de setembro de 2020. [Online]. Available: <https://commons.wikimedia.org/w/index.php?curid=10073387>
- [8] Booyabazooka, “An image showing an example of parallax,” 2006, Último acesso a 5 de setembro de 2020. [Online]. Available: <https://commons.wikimedia.org/w/index.php?curid=1335592>
- [9] J. B.-l. Image, E. Group, J. Photographic, and E. Group, “Overview of Holography Version 3.0,” no. October, 2018.
- [10] D. A. Spencer, *The focal dictionary of photographic technologies*. London Englewood Cliffs, N.J: Prentice-Hall, 1973.

- [11] R. Krebs, *Groundbreaking scientific experiments, inventions, and discoveries of the Middle Ages and the Renaissance.* Greenwood Press, 2004.
- [12] D. Gabor, “A New Microscopic Principle,” *Nature*, vol. 161, no. 4098, pp. 777–778, may 1948.
- [13] Gabor, D., “Microscopy by reconstructed wave-fronts,” *Proc. R. Soc. London. Ser. A. Math. Phys. Sci.*, vol. 197, no. 1051, pp. 454–487, jul 1949.
- [14] E. N. Leith and J. Upatnieks, “Reconstructed Wavefronts and Communication Theory,” *J. Opt. Soc. Am.*, vol. 52, no. 10, p. 1123, oct 1962.
- [15] Leith, Emmett N. and Upatnieks, Juris, “Wavefront Reconstruction with Diffused Illumination and Three-Dimensional Objects,” *J. Opt. Soc. Am.*, vol. 54, no. 11, p. 1295, nov 1964.
- [16] “The Nobel Prize in Physics 1971 — NobelPrize.org,” 2020, Último acesso a 5 de setembro de 2020. [Online]. Available: <https://www.nobelprize.org/prizes/physics/1971/summary/>
- [17] *Applications of Holography and Optical Data Processing.* Elsevier, 1977.
- [18] W. Saxon, “Stephen A. Benton, 61, an Expert Researcher in Holography,” nov 2003. [Online]. Available: <https://www.nytimes.com/2003/11/14/us/stephen-a-benton-61-an-expert-researcher-in-holography.html>
- [19] V. Toal, *Introduction to holography.* Boca Raton: CRC Press, 2012.
- [20] A. Gilles, P. Gioia, R. Cozot, and L. Morin, “Hologram Repository,” 2018, Último acesso a 26 de março de 2020. [Online]. Available: <https://hologram-repository.labs.b-com.com/{#/}hologram-repository>
- [21] Gilles, Antonin and Gioia, Patrick and Cozot, Rémi and Morin, Luce, “Hybrid approach for fast occlusion processing in computer-generated hologram calculation,” *Appl. Opt.*, vol. 55, no. 20, pp. 5459–5470, jul 2016.
- [22] B. R. Brown and A. W. Lohmann, “Complex Spatial Filtering with Binary Masks,” *Appl. Opt.*, vol. 5, no. 6, p. 967, jun 1966.
- [23] A. W. Lohmann, “Three-dimensional properties of wave-fields,” *Optik (Stuttg.)*, vol. 51, pp. 105–107, 1978.
- [24] T.-C. Poon, *Introduction to modern digital holography : with MATLAB.* Cambridge England, United Kingdom: Cambridge University Press, 2014.

- [25] K. Sato, K. Higuchi, and H. Katsuma, "Holographic television by liquid crystal device," in *1991 Third Int. Conf. Hologr. Syst. Components Appl.*, 1991, pp. 20–23.
- [26] YOSHIKAWA and H., "Digital Holographic Signal Processing," *TAO First Int. Symp. Dec. 1993*, 1993.
- [27] H. Yoshikawa and J. Tamai, "Holographic image compression by motion picture coding," S. A. Benton, Ed., mar 1996, pp. 2–9.
- [28] T. J. Naughton, Y. Frauel, B. Javidi, and E. Tajahuerce, "Compression of digital holograms for three-dimensional object reconstruction and recognition," *Appl. Opt.*, vol. 41, no. 20, p. 4124, jul 2002.
- [29] A. Arrifano, M. Antonini, and M. Pereira, "Multiple Description Coding of Digital Holograms Using Maximum-A-Posteriori," Tech. Rep., 2013.
- [30] Y. Xing, B. Pesquet-Popescu, and F. Dufaux, "Comparative study of scalar and vector quantization on different phase-shifting digital holographic data representations," in *3DTV-Conference*. IEEE Computer Society, 2014.
- [31] E. Darakis, M. Kowiel, R. Näsänen, and T. J. Naughton, "Visually lossless compression of digital hologram sequences," in *Image Qual. Syst. Perform. VII*, vol. 7529. SPIE, jan 2010, p. 752912.
- [32] J. Peixeiro, C. Brites, J. Ascenso, and F. Pereira, "Digital holography: Benchmarking coding standards and representation formats," in *Proc. - IEEE Int. Conf. Multimed. Expo*, vol. 2016-Augus. IEEE Computer Society, aug 2016.
- [33] A. M. G. Pinheiro, M. Pereira, and M. Bernardo, "Benchmarking coding standards for digital holography represented on the object plane," in *Opt. Photonics, Digit. Technol. Imaging Appl. V*, P. Schelkens, T. Ebrahimi, and G. Cristóbal, Eds., vol. 10679. SPIE, may 2018, p. 18.
- [34] D. S. Taubman and M. W. Marcellin, "JPEG2000: Standard for Interactive Imaging," 2002.
- [35] P.-A. Lemieux, R. Mathew, A. Naman, S. Ogawa, M. Smith, D. Taubman, and O. Watanabe, "High Throughput JPEG 2000 (HTJ2K) and the JPH file format: a primer," Tech. Rep.
- [36] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer US, 2002.

- [37] A. Bovik, *The essential guide to video processing*. Amsterdam Boston: Academic Press/Elsevier, 2009.
- [38] D. Le Gall and A. Tabatabai, “Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques,” in *ICASSP-88, Int. Conf. Acoust. Speech, Signal Process.* IEEE, pp. 761–764.
- [39] M. Unser and T. Blu, “Mathematical properties of the JPEG2000 wavelet filters,” *IEEE Trans. Image Process.*, vol. 12, no. 9, pp. 1080–1090, sep 2003.
- [40] “JPEG - JPEG 2000,” Último acesso a 5 de setembro de 2020. [Online]. Available: <https://jpeg.org/jpeg2000/>
- [41] D. Salomon, *Data compression: the complete reference*. London: Springer, 2007.
- [42] S. Welstead, *Fractal and wavelet image compression techniques*. Bellingham, Wash: SPIE Optical Engineering Press, 1999.
- [43] M. Barni, *Document and image compression*. Boca Raton, FL: CRC/Taylor & Francis, 2006.
- [44] A. Gilles and P. Gioia, “JPEG Pleno Holography — Numerical reconstruction software,” no. July, pp. 1–3, 2018.
- [45] “Kakadu Software,” Último acesso a 30 de março de 2020. [Online]. Available: <https://kakadusoftware.com/>
- [46] A. Gilles, P. Gioia, R. Cozot, and L. Morin, “Computer generated hologram from Multiview-plus-Depth data considering specular reflections,” in *2016 IEEE Int. Conf. Multimed. Expo Work.* IEEE, jul 2016, pp. 1–6.