Universidade da Beira Interior
Departamento de Informática

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

Elaborado por:

Raquel Sofia Brás Guerra

Orientador:

Professora Doutora Maria Manuela Areias da Costa Pereira de Sousa

27 de Agosto de 2020

Agradecimentos

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

ASM *Angular Spectrum Method*

JPEG *Joint Photographic Experts Group*

SDK *Software Development Kit*

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 padrões *standardizados* para a codificação e compressão de **hologramas** em formato digital.

1.2 Motivação

Dada a referida ausência de *standards* 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 padrões 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 hologramas reconstruidos 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

Capítulo

2

Estado da Arte

2.1 Introdução

Cada capítulo intermédio deve começar com uma breve introdução onde é explicado com um pouco mais de detalhe qual é o tema deste capítulo, e como é que se encontra organizado (i.e., o que é que cada secção seguinte discute).

2.2 Objetivos

2.3 Perspetiva Histórica

2.4 Conceitos Base

2.4.1 Holografia

Quando um objeto é iluminado, a luz é dispersa pela superfície desse objeto, criando uma onde. Esta onda contém toda a informação sobre a luz: a amplitude define o brilho e a fase representa a forma do objeto. Enquanto as fotografias clássicas gravam apenas a intensidade da luz, um holograma preserva a fase do objeto através das características de interferência e difração da luz, guardando assim toda a informação necessária à reconstrução 3D do objeto original.

2.4.1.1 Princípios de Holografia

O principio de holografia foi descoberto em 1948 pelo físico Dennis Gabor enquanto investigava microscopia de eletrões.

Ao contrário da fotografia convencional, que permite a captura da intensidade da luz, holografia permite guardar a amplitude e a fase da onde de luz dispersa por um objeto. Com a iluminação correta, o holograma produz a onda de luz original, permitindo ao utilizador observar o objeto tal como se estivesse fisicamente presente.

2.4.1.2 Representação de Dados Holográficos

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.

- Dados reais e imaginários — Utiliza 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.

2.4.1.3 Reconstrução de Holograma

2.4.2 Compressão

2.4.3 JPEG2000

2.4.4 Cor

2.4.4.1 RGB

2.4.4.2 YCBCR

2.5 Estado da Arte

Primeira proposta para codificação digital de hologramas data 1991, Sato et al. captura franjas holográficos usando uma câmara que foram por sua vez modulados em sinal TV e transmitidos para um recetor [1]. (captured the holographic fringes using a camera, which was then modulated into a TV signal and transmitted to the receiver.);

Em 1993, Yoshikawa notou que não era prática a aplicação da compressão de imagem 2d diretamente no holograma. Propôs a compressão do holograma em segmentos que correspondem a diferentes perspetivas de reconstrução. Segmentos foram comprimidos com MPEG-1 e MPEG-2 [2,3]. (ver resultados)

Em 2002, Naughton et al. estudou a compressibilidade da holografia digital de mudança de fase usando vários algoritmos de compressão sem perdas [4]. Concluiram que são esperadas melhores taxas de compressão quando o holograma digital é codificado em componentes reais e imaginarias independentemente.

Em [4] foram também estudadas outras técnicas de compressão com perdas tais como subamostragem e quantificação, sendo a última muito eficaz. A eficácia da quantização tanto na simulação numérica como na ótica foi confirmada por Mills e Yamaguchi [5].

A quantização no domínio da reconstrução (não sei o que isto quer dizer) de hologramas de mudança de fase de foram analisados por Darakis and Soraghan [6].

Naughton et al. em 2003 e Darakis et al. em 2006 demonstraram que a aplicação direta de wavelets standard em hologramas não é muito eficiente, visto que as wavelets standard são tipicamente usadas no processamento de sinais com poucas variações (smooth signals). Propuseram a utilização de uma outra familia de wavelets Fresnelets. Fresnelets foram também aplicadas em 2003 por Liveling et al. [8]

Em 2006, Seo et al. propôs comprimir segmentos do holograma usando multi-vistas e temporal prediction dentro de MPEG-2 modificado.

Em 2010 Darkis et al. Determinaram a taxa de compressão mais elevada que pode ser obtida em hologramas mantendo uma qualidade de reconstrução visualmente sem perdas. Nos seus ensaios foram usados MPEG-4 e Dirac. Na informação amplitude-fase foi aplicado um método multiple description coding utilizando máximo à posterior. Mostrou-se um mecanismo poderoso para mitigar erros no canais em hologramas digitais.

Em 2013 Blinder investigou a decomposição alternativa em hologramas off-axis. Em 2014 Still, Xing e Dufaux estudaram codificação sem perdas baseada em quantização vetorial.

Recentemente Peixeiro et al. [9] realizou um benchmark dos codificadores standard de imagens aplicados em hologramas digitais, em conjunto com os formatos de representação principais. Foram comparados os seguintes codificadores de imagem padrão JPEG; JPEG 2000; H.264/AVC intra; HEVC intra. Os autores concluiram que os melhores formatos de representação são phase-shiffted e real-imaginário

Em 2016, Dufaux review o estado da arte da compressão de hologramas digitais

2.6 Conclusões

Cada capítulo intermédio deve referir o que demais importante se conclui desta parte do trabalho, de modo a fornecer a motivação para o capítulo ou passos seguintes.

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*. O respetivo código, fornecido pela professora orientadora, encontra-se implementado em MATLAB.

Dada a ausência de uma licença do MATLAB para utilizar este *sofware* 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*.

Este é um SDK para codificação e descodificação de imagens com recurso ao formato JPEG2000, segundo o *standard* pela *Joint Photographic Experts Group* (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 reconstruidos com o *software* desenvolvido no âmbito deste projeto são fornecidos pelo *Institute of Research & Technology b<>com*. De entre os disponíveis, foram utilizados os seguintes hologramas com as respetivas características:

1. `Dices4k` (Figura 3.1):
    - Resolução: 4096×4096;
    - *Pixel pitch*: 0.4 µm;
    - Comprimento de onda vermelho: 640 nm;
    - Comprimento de onda verde: 532 nm;
    - Comprimento de onda azul: 473 nm;
    - Localização da cena: entre 0.164 e 0.328 cm.
2. `DiffuseCar4k` (Figura 3.2):
    - Resolução: 4096×4096
    - *Pixel pitch*: 0.4 µm;
    - Comprimento de onda vermelho: 640 nm;
    - Comprimento de onda verde: 532 nm;
    - Comprimento de onda azul: 473 nm;
    - Localização da cena: entre 0.11 e 0.25 cm.
3. `Piano4k` (Figura 3.3):
    - Resolução: 4096×4096
    - *Pixel pitch*: 0.4 µm;
    - Comprimento de onda vermelho: 640 nm;
    - Comprimento de onda verde: 532 nm;
    - Comprimento de onda azul: 473 nm;
    - Localização da cena: entre 0.17 and 0.313 cm.

Figura 3.1: Holograma `Dices4k` (imagem original).

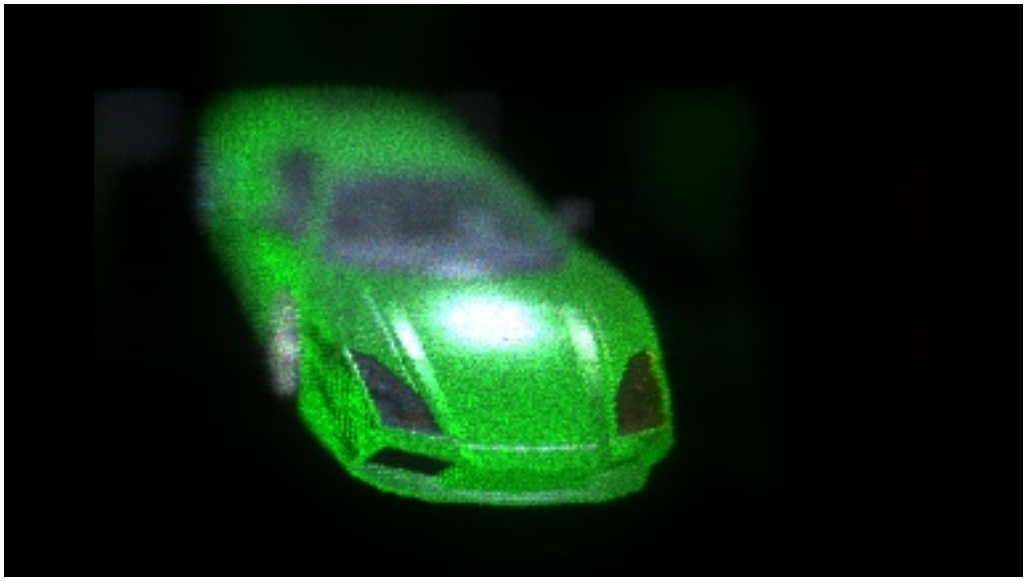

Figura 3.2: Holograma `DiffuseCar4k` (imagem original).

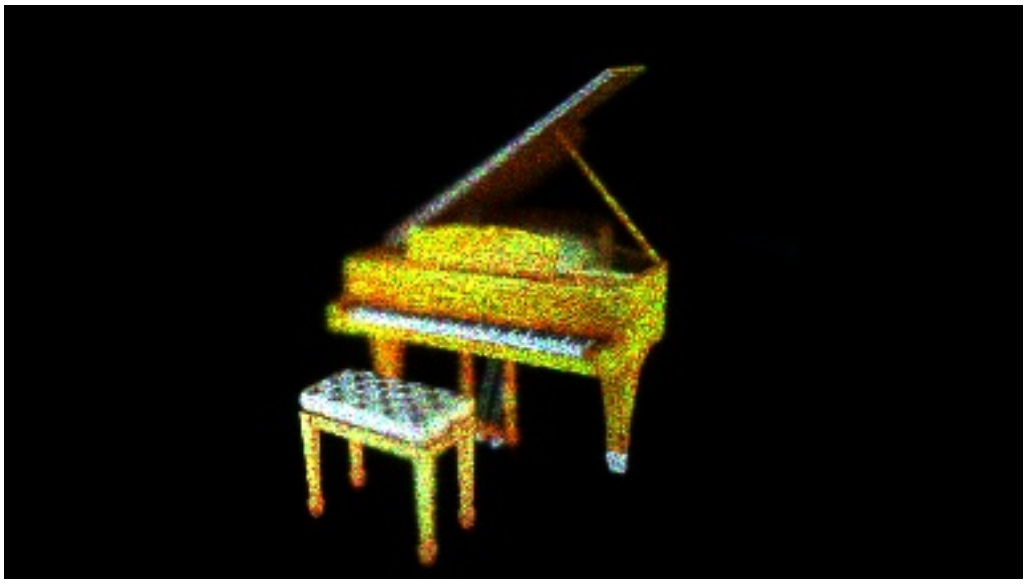

Figura 3.3: Holograma `Piano4k` (imagem original).

3.4 Conclusões

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

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.

4.3 Compressão dos Hologramas Reconstruídos

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignis-

Tabela 4.1: Resumo da documentação da função `load_hologram`.

Nome da função `load_hologram`
Protótipo original em MATLAB `function [hologram] = load_hologram(ampli_path, phase_path)`
Protótipo transcrito em Python `def load_hologram(ampli_path, phase_path)`
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: R-G-B).

sim rutrum.

4.4 Determinação das Métricas de Compressão

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

4.5 Conclusões

Tabela 4.2: Resumo da documentação da função `propagate_asm`.

Nome da função
`propagate_asm`
Protótipo original em MATLAB
`function [v] = propagate_asm(u, pitch, wavelength, z)`
Protótipo transcrito em Python
`def propagate_asm(u, pitch, wavelength, z)`
Descrição
Esta função simula a propagação no plano complexo `u` sobre a distância `z` utilizando o *Angular Spectrum Method* (ASM).
Inputs
`u`: Campo de onda de luz do plano de *input* (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).

Tabela 4.3: Resumo da documentação da função `reconst_hologram`.

Nome da função	`reconst_hologram`
Protótipo original em MATLAB	`function [recons] = reconsHologram(hologram, pitch, wavelengths, z, pupilPos, pupilSize)`
Protótipo transcrito em Python	`def reconst_hologram(hologram, pitch, wavelengths, z, pupil_pos=[0,0], pupil_size=None)`
Descrição	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.
Inputs	`hologram`: Holograma de modulação complexa (3 canais: R-G-B). `pitch`: Distância entre pixeis (em metros). `wavelengths`: Comprimentos de onda de luz (em metros, 3 canais: R-G-B). `z`: Distância de reconstrução (em metros). `pupilPos`: Posição da janela (em pixeis, canto superior direito). `pupilSize`: Tamanho da janela (em pixeis, altura × largura).
Output	Reconstrução numérica do holograma (3 canais: R-G-B).

Capítulo

5

Testes e Resultados

5.1 Introdução

Cada capítulo intermédio deve começar com uma breve introdução onde é explicado com um pouco mais de detalhe qual é o tema deste capítulo, e como é que se encontra organizado (i.e., o que é que cada secção seguinte discute).

5.2 Resultados

5.2.1

5.3

5.4 Conclusões

Cada capítulo intermédio deve referir o que demais importante se conclui desta parte do trabalho, de modo a fornecer a motivação para o capítulo ou passos seguintes.

Capítulo

6

Conclusões e Trabalho Futuro

6.1 Conclusões Principais

Esta secção contém a resposta à questão:
Quais foram as conclusões princípais a que o(a) aluno(a) chegou no fim deste trabalho?

6.2 Trabalho Futuro

Esta secção responde a questões como:
O que é que ficou por fazer, e porque?
O que é que seria interessante fazer, mas não foi feito por não ser exatamente o objetivo deste trabalho?
Em que outros casos ou situações ou cenários – que não foram estudados no contexto deste projeto por não ser seu objetivo – é que o trabalho aqui descrito pode ter aplicações interessantes e porque?