Trabalho Prático 1

Visualizador de Imagens HDR

Programação C

04/2021

Alcance ou intervalo dinâmico (em inglês, dynamic range) em fotografia descreve a razão entre as intensidades mínimas e máximas mensuráveis de luz (preto e branco, respectivamente). Porém, no mundo real nunca encontramos preto ou branco verdadeiros, apenas graus diversos de intensidade das fontes de luz e reflectância dos objetos. Uma imagem HDR (high dynamic range) é uma imagem capaz de representar um intervalo maior de luminosidade do que é possível com técnicas convencionais de imagens digitais ou fotografia. O

Introdução

Para demonstrar a adaptação automática do olho humano, observe a imagem abaixo (Checker shadow illusion (Edward H. Adelson): nela, os blocos A e B têm exatamente a mesma cor, mas o fato de um deles estar na sombra cria uma ilusão para os nossos olhos.

objetivo é mostrar um alcance similar ao que experimentamos com a visão humana. Esta, através de adaptação da íris e outros mecanismos, se ajusta constantemente e automaticamente de acordo com a

variação de iluminação presente nos ambientes. O cérebro humano interpreta continuamente essa informação, de forma que possamos enxergar em uma variedade enorme de condições de iluminação.



Porém, a maior parte das telas atuais de computador e TVs não têm a capacidade de exibir toda a variação luminosa em uma imagem HDR - apenas recentemente começaram a aparecer as TVs Ultra HD

(4K) com HDR, bem como alguns modelos de tablets e celulares. Portanto, também foi preciso desenvolver técnicas para exibir os resultados. Essas técnicas são chamadas, genericamente, de mapeamentos

Neste trabalho, você implementará um programa para ler uma imagem em um formato HDR e exibi-la usando um algoritmo de tone mapping. Um exemplo pode ser visto abaixo:



Se decidirmos ajustar a exposição para a parte escura, a parte clara desaparecerá (dizemos que ficará "superexposta"): Uma imagem HDR resolve esse problema, pois consegue capturar toda essa variação de intensidade luminosa. Mas como produzir imagens desse tipo? Uma técnica utilizada é combinar várias imagens obtidas com uma câmera (não HDR), mas com exposições diferentes (o que é denominado bracketing em fotografia). Por exemplo, a partir de 5 imagens do mesmo ambiente, geramos uma imagem HDR:

tonais (tone mapping). O objetivo é reduzir o contraste de uma imagem HDR, de forma a exibi-la em dispositivos com alcance dinâmico inferior.

996633

CC9900

FFCC00

FFFF00

FFFF33

FFFF66

FFFF99

FFFFCC

A imagem no formato HDF consiste primeiramente em um header, onde as informações estão organizadas da seguinte forma:

663300

993300

CC3300

FF3300

FF6600

FF6633

FF9966

FFCC99

660000

990000

CC0000

FF0000

CC3333

FF6666

FF9999

FFCCCC

003333

336666

006666

009999

00CCCC

66CCCC

66FFCC

99FFCC

003300

006600

009900

00CC00

00FF00

66FF66

99FF99

CCFFCC

003399

0033FF

0066FF

0099FF

00CCFF

00FFFF

99FFFF

CCFFFF

Funcionamento Ao ser iniciado, o programa deve carregar um arquivo de imagem. Essas imagens estão em um formato especial (sufixo .hdf). É importante entendermos a diferença entre esse formato e uma imagem convencional. Imagens são geralmente representadas por uma matriz de pontos (pixels) onde cada cor é definida por 3 componentes: vermelho (R), verde (G) e azul (B). Cada uma dessas componentes usualmente é codificada em um byte, o que produz 3 bytes por pixel (24 bits) - ou seja, 16 milhões de possíveis cores. Em outras palavras, as intensidades (R, G, B) variam de 0 a 255, onde 0 é escuro e 255 é claro. Veja abaixo como diversas cores são representadas nesse formato - cada cor está expressa pelas componentes RGB em hexadecimal. 666666 CCCC99 666699 000000 9999CC **FFFFFF** 333333 999999 CCCCCC

000099

0000CC

0000FF

3366FF

3399FF

66CCFF

99CCFF

660099

663399

9900CC

9933FF

9966FF

9999FF

CCCCFF

990066

CC0099

FF0099

FF00FF

FF66FF

FF99FF

FFCCFF

Já uma imagem HDR pode possuir valores maiores que esses limites, sendo geralmente representada em ponto flutuante (valores float, por exemplo). O formato HDF codifica as cores de forma diferente (ver seção 2.1↓), mas na prática o resultado é uma representação float. Após a leitura da imagem, o usuário poderá definir o fator de exposição desejado, a fim de "clarear" ou "escurecer" a imagem, bem como também ajustar os níveis de preto e de branco.

FF0000 Red

FFFF00

FFFFFF

0000FF Blue

A ordem do processo é a seguinte:

1. Leitura e decodificação da imagem (seção 2.1↓).

5. Criação do histograma da imagem 24 bits (seção 2.6↓).

00FF00

2. Aplicar o fator de exposição desejado (seção 2.2↓). 3. Aplicar o algoritmo de *tone mapping* (seção 2.3↓). 4. Converter o resultado para 24 bits (seção 2.5↓).

6. Ajuste dos níveis de preto e branco (seção 2.7↓). 7. Retornar ao passo 2, caso a exposição ou os níveis mínimo/máximo sejam alterados pelo teclado (o programa já faz isso). As próximas seções detalham como implementar cada parte do processo.

Após a extração das dimensões da imagem do header, você deverá chamar a função carregalmagem, que armazena, respectivamente, nas variáveis image e image8, ponteiros para a imagem original e

 $c=2^{(m-136)}$

altura

O fator de exposição é meramente um valor float (exposure) que é multiplicado por cada componente de cor (R_f,G_f,B_f) , antes de realizar o tone mapping. Na verdade, utilizamos uma potência de 2, para

 $(R_x,G_x,B_x)=2^{exposure}\cdot(R_f,G_f,B_f)$

 $R_t = rac{R_x \cdot (2,51 \cdot R_x + 0,03)}{R_x \cdot (2,43 \cdot R_x + 0,59) + 0,14}$

A correção gama se baseia na ideia que os dispositivos de exibição (telas de computador, por exemplo) não apresentam uma resposta linear a um sinal de entrada (cor). Dessa forma, podemos aplicar uma

Após a aplicação do fator de exposição, do algoritmo de tone mapping e da correção gama, o resultado ainda será uma imagem representada por floats, mas agora dentro do invervalo 0...1. Portanto, para

 $R_8 = R_c \cdot 255$

 $G_8 = G_c \cdot 255$

 $B_8 = B_c \cdot 255$

 $(R_c,G_c,B_c)=(R_t^{rac{1}{\gamma}},G_t^{rac{1}{\gamma}},B_t^{rac{1}{\gamma}})$

81 20 90 18 27 38 81 72 87 18 27 83 84 71 61 01

72 71 88 17 18 78 37 86 18 97 61 67 16 77 71 67

0000030 34 61 76 79 87 16 76 17 67 12 35 57 61 76 17 61

00 05 00 00 00 04 00 00 80 40 20 81 92

1o. pixel

rq...x7...ag.wqg

4avy..v.g.5Wav.a

imagem de saída (vazia, a ser utilizada depois) O ponteiro image aponta para o início da matriz de pixels - mas lembre-se que essa matriz é retornada como um vetor. Cada pixel é codificado em 4 bytes:

2.1

• A altura da imagem em pixels, armazenada como um inteiro sem sinal Você deverá usar a função carregaHeader, que armazena essas informações no vetor global header.

Onde c é o fator de conversão calculado a partir da mantissa:

Por exemplo, supondo o trecho da imagem abaixo:

• Os caracteres **HDF**, identificando a imagem nesse formato

Leitura e decodificação da imagem HDF

• A largura da imagem em pixels, armazenada como um inteiro sem sinal

- Componente G da $\operatorname{cor}(G_e)$
- Componente R da cor (R_e)
- Componente B da ${\sf cor}(B_e)$ • Mantissa (m)
- Esse formato é capaz de representar dados em ponto flutuante de uma forma bastante econômica. Para obter os valores RGB float (R_f, G_f, B_f) a partir dos 4 bytes do pixel, basta realizar o seguinte cálculo: $(R_f, G_f, B_f) = (R_e \cdot c, G_e \cdot c, B_e \cdot c)$

header

largura

id

0000000

0000010

0000020

- O primeiro pixel da imagem tem os valores hexa (0x80, 0x40, 0x20, 0x81) ou em decimal (128,64,32,129). O fator de conversão é então $c=2^{(129-136)}=2^{-7}=0.0078125$. Portanto, os pixels convertidos para float resultam em: (1.0, 0.5, 0.25) Observe que por ser uma imagem HDR, não há limite superior definido. Ou seja, as componentes R_f , G_f e B_f podem assumir qualquer valor.

Aplicação do fator de exposição

Esse fator deve ser inicializado com 0, o que corresponde a nenhuma alteração.

Para tanto, aplica-se a seguinte fórmula aos componentes R, G e B:

• Primeiro, reduz-se o valor da componente para 60% do original: $R_x = R_x \cdot 0, 6$

• Garanta que o valor resultante esteja no intervalo 0...1 (isto é, se for maior que 1, use 1, e vice-versa).

converter cada pixel de entrada já corrigido pelo passo anterior (R_c, G_c, B_c) para RGB 24 bits (R_8, G_8, B_8) , basta fazer:

Ou seja, multiplica-se o valor por 255, convertendo o resultado para inteiro no final.

Criação do histograma da imagem 24 bits

simular mais precisamente o comportamento de uma câmera:

Algoritmo de tone mapping

ao visualizar uma imagem.

A seguir, calcula-se o mapeamento:

Correção gama

2.5

2.6

Como já mencionado, mapeamento tonal ou tone mapping tem o objetivo de reduzir o alcance dinâmico de uma imagem, de forma que ela possa ser exibida em um dispositivo não HDR. Há duas categorias principais de algoritmos desse tipo: tone mapping global e tone mapping local. O primeiro tipo, onde está o algoritmo que será implementado, visa alterar todos os pixels da imagem seguindo um mesmo critério (ou algoritmo). Já o segundo tipo leva em consideração outras características da imagem, como bordas de objetos, etc. Esse tipo de algoritmo é muito mais sofisticado e geralmente produz resultados melhores. Neste trabalho, implementaremos um algoritmo de tone mapping global, denominado ACES (Academy Color Encoding System). Esse algoritmo visa simular o tipo de resposta que o olho humano produz

Onde γ é a correção gama desejada. Para monitores convencionais, esse valor geralmente está entre 1.8 e 2.6 (experimente usar 2.2).

Conversão para 24 bits

faixas, já que temos 256 variações por componente de cor.

Ajuste dos níveis de preto e branco

Você deverá então:

segue:

1. Inicializar o histograma de ajuste com zeros.

• A/S: reduzem e aumentam o nível de preto

• K/L: reduzem e aumentam o nível de branco

Compilação e imagens de teste

Se você estiver utilizando o Visual Studio Code, basta descompactar o zip e abrir a pasta.

Para **executar**, use F5 para usar o *debugger* ou Ctrl+F5 para executar sem o *debugger*.

• H: exibe/oculta os histogramas

Download do código base: hdrvis-base.zip

sudo apt-get install freeglut3-dev

Visual Studio Code

mingw32-make -f Makefile.mk

make -j # ou mingw32-make -j no Windows

cd build

cmake ..

Pontuação:

3. Aplicar a correção em todas as componentes do pixel:

1. Inicializar o histograma com zeros.

correção não linear nesse sinal utilizando a fórmula abaixo:

superexposta (muitas regiões claras). As imagens abaixo foram retiradas do site Cambridge in Colour - Histograms. Midtones Highlights

Em termos práticos, o histograma é essencialmente um array, onde cada posição representa uma faixa de intensidades (bins, em inglês). Para simplificar, utilizaremos um histograma com exatamente 256

2. Passar por toda a imagem, incrementando o valor armazenado na posição correspondente ao pixel no array. Observe que você deverá utilizar a intensidade (luminância) do pixel, que pode ser calculada por:

3. Ao final desse processo, é preciso normalizar o histograma, isto é, dividir cada valor armazenado pelo valor máximo contido no array. Dessa forma, o histograma armazenará valores float entre 0 e 1.

 $I=0,299\cdot R_8+0,587\cdot G_8+0,114\cdot B_8$

Um histograma de uma imagem apresenta a distribuição dos pixels de acordo com a luminosidade. Eles são muito úteis para visualizarmos se a imagem está subexposta (muitas regiões escuras) ou

O programa já define uma variável chamada histogram, que deve ser usada para isso. A razão é que o componente de visualização utiliza essa variável para exibir o histograma.

⊤ <u>C</u>hannel: RGB

Input Levels: 0

Output Levels: 0

Esse ajuste é armazenado nas variáveis minLevel e maxLevel, respectivamente. Elas recebem um valor de 0 a 255, sendo que minLevel nunca poderá ser maior que maxLevel.

A fim de facilitar a visualização, foi criado um segundo histograma (variável adjusted), que deverá ser preenchido da mesma forma que o primeiro (original) - porém, com as intensidades ajustadas como

 $I_a = min\left(1, rac{max(0, I-minLevel)}{maxLevel-minLevel}
ight) \cdot 255$

 $(R_8,G_8,B_8) = \left(rac{R_8I_a}{I},rac{G_8I_a}{I},rac{B_8I_a}{I}
ight)$

Este zip contém o projeto completo para a implementação do trabalho. Esse código já realiza a leitura de uma imagem qualquer no formato HDF, bem como a exibição da imagem 24 bits na tela (mas

Caso você queira trocar as cores dos histogramas ou dos marcadores, edite os #defines no topo do módulo opengl.c, lembrando que as componentes são todas especificadas no intervalo 0...1. Você

pode usar este site para escolher as cores, por exemplo: instantreality 1.0 - tools - color calculator. Basta selecionar a cor desejada e copiar os valores da caixa RGB Normalized decimal no código.

Alternativamente, você também pode utilizar o CMake (Cross Platform Make) para compilar pelo terminal. Para tanto, crie um diretório build embaixo do diretório do projeto e faça:

• Os trabalhos são em duplas ou individuais. A pasta do projeto deve ser compactada em um arquivo .zip e este deve ser submetido pelo Moodle até a data e hora especificadas.

O arquivo imagens-hdf.zip contém diversas imagens que podem ser utilizadas para testar o programa. Cada imagem é oferecida em duas versões: formato HDF e em JPEG, para conferência.

Essa etapa permite realizar um ajuste final no contraste da imagem, fazendo com que seja possível corrigir a variação tonal já na imagem 24 bits. A maioria dos programas de pintura digital oferece esse recurso, geralmente chamando levels.

Cancel

<u>L</u>oad...

<u>S</u>ave...

<u>A</u>uto

Options...

9 9 9

▽ Preview

O ajuste consiste em movimentar os limites inferior (nível de preto) e superior (nível de branco). A nossa aplicação já permite esse ajuste através de teclas (veja seção 3↓).

Veja na imagem abaixo a ferramenta numa versão do Adobe Photoshop (Cambridge In Colour - Levels)

Operação do programa As seguintes teclas estão disponíveis na aplicação: • Setas esquerda/direita: reduzem e aumentam o fator de exposição.

obviamente não faz a conversão). O projeto pode ser compilado no Windows, Linux ou macOS, seguindo as instruções abaixo.

Para a compilação no Windows ou no macOS, não é necessário instalar mais nada - o compilador já vem com as bibliotecas necessárias.

Caso esteja usando outro ambiente de desenvolvimento, fornecemos um Makefile para Linux e macOS, e outro para Windows (Makefile.mk).

5. Atualize também o histograma de ajuste, da mesma forma que o outro, mas com a nova intensidade $\left(I_{a}
ight)$.

4. Neste momento, os valores de (R_8, G_8, B_8) devem ser armazenados na imagem de saída, a variável image8 (é um vetor de unsigned char, isto é, bytes sem sinal).

Para a compilação no Linux, é necessário ter instalado os pacotes de desenvolvimento da biblioteca OpenGL. Para Ubuntu, Mint, Debian e derivados, instale com:

2. Passar por toda a imagem, novamente calculando a intensidade (I) de cada pixel. Calcular a nova intensidade, através da seguinte fórmula:

make Se estiver utilizando o Windows, o comando é similar:

Dessa forma, para compilar no Linux ou macOS, basta digitar:

Para compilar: use Ctrl+Shift+B (\mathbb{H}+Shift+B no macOS).

4.2 Outros ambientes ou terminal

- Avaliação
- Leia com atenção os critérios de avaliação: Extração da largura e altura da imagem: 1 ponto. Decodificação da imagem: 1,5 pontos. Aplicação do fator de exposição: 1 ponto. Algoritmo de tone mapping: 1,5 pontos. Conversão para 24 bits: 1 ponto.
- O código deve estar identado corretamente. • A cópia parcial ou completa do trabalho terá como conseqüência a atribuição de nota ZERO ao trabalho dos alunos envolvidos. A verificação de cópias é feita inclusive entre turmas. • A cópia de código ou algoritmos existentes da Internet também não é permitida. Se alguma idéia encontrada na rede for utilizada na implementação, sua descrição e referência deve constar no início

do código-fonte, como um comentário.

Não envie .rar, .7z, .tar.gz - apenas .zip.

Criação do histograma original: 1,5 pontos.

Criação do histograma ajustado: 1 ponto.

Ajuste dos níveis de preto e branco: 1,5 pontos.

Document generated by eLyXer 1.2.5 (2013-03-10) on 2021-04-09T17:59:16.945981