Lista de Exercícios 1

Vinicius Graciano Santos - vgs@dcc.ufmg.br

13 de abril de 2010

1 Exercícios Teóricos

1. O principal problema tratado pela Visão Computacional é encontrar uma maneira de computar as propriedades do mundo 3D a partir de uma ou mais imagens. As propriedades mais interessantes são a forma, posição, orientação e a velocidade de objetos nessas imagens. As áreas de Visão Computacional, Processamento Digital de Imagens e Computação Gráfica estão intrinsecamente relacionadas, mas cada uma tenta responder a uma pergunta diferente, sendo elas:

   - **Visão Computacional:** Dada uma imagem, como é possível descobrir qual o modelo matématico utilizado para gerá-la?
   - **Processamento Digital de Imagens:** Dada uma imagem, qual transformação deve ser aplicada para gerarmos uma outra imagem com certas características e propriedades?
   - **Computação Gráfica:** Dado um modelo matématico de uma cena, como renderizar de forma eficiente e realista uma imagem dessa cena?

2. Um sistema de aquisição de imagens consiste basicamente de três componentes: uma câmera, um *frame grabber* e um computador onde é feito o processamento da imagem. A câmera utiliza uma composição de lentes para tentar diminuir ao máximo o espalhamento dos raios de luz de entrada. Esses raios atingem os sensores contidos no plano da imagem, que é conhecido como CCD, um *grid* de tamanho $n \times m$ de foto-sensores. O objetivo de cada um é transformar a quantidade de luz (fótons) incidida em um sinal elétrico contínuo, denominado *sinal de vídeo*. Esse sinal é então enviado para o *frame grabber* que se responsabiliza por digitalizá-lo em uma matriz inteira $N \times M$ e a armazena em um *buffer* de memória. Finalmente, a imagem pode ser transferida para um computador onde pode ser visualizada e editada.

   Cada pixel da imagem, na maioria dos casos, é armazenado utilizando-se um único byte para representações em tons de cinza, ou em uma tupla de três bytes que representam as cores vermelho, verde e amarelo.

   Infelizmente, esse processo pode gerar várias distorções na imagem. Por exemplo, a razão entre as dimensões do *frame buffer* pode não ser a mesma do *CCD*, o que irá gerar um efeito de escala entre as duas representações. Além disso, a distância entre os elementos do *CCD* não é nula e os sensores podem ser considerados pequenos retângulos discretos. Essas características tornam o processo de aquisição de imagens uma amostragem discreta do sinal que entra na câmera. Dessa forma, o efeito de *aliasing* é esperado. Um outro ponto negativo é que o conjunto de lentes também não formam um sistema óptico perfeito, ou seja, os raios de luz sofrem várias distorções ao passar por elas. Por exemplo, se um emissor de luz pontual estiver diretamente apontado para a câmera, a imagem resultante que esperamos obter é a de exatamente um ponto, mas isso não é o que ocorre. Na verade, o ponto é espalhado na imagem final e podemos ver claramente um efeito de embaçamento. Se dois

pontos luminosos estiverem muito próximos, o espalhamento de ambos pode levar a aparição de um único borrão na imagem, o que irá dificultar a identificação dos dois pontos, já que eles serão representados por um único borrão.

Conhecer a função que descreve esse espalhamento é essencial para vários algoritmos de Visão Computacional, essa função é denominada *point-spread function (PSR)*.

3. Pelo *Teorema de Nyquist*, se um sinal é amostrado em uma frequência uniforme f_S então para o reconstruirmos totalmente é necessário que o seu espectro não possua frequências maiores ou iguais a $\frac{f_S}{2}$. Ou seja, o intervalo de amostragem deve ser maior do que $\frac{1}{2B}$, sendo B a maior frequência do espectro do sinal. Caso essa restrição não seja atendida, a função resultante da interpolação das amostras não irá convergir para a função que representa o sinal original. Isso deve-se ao fato de que durante o processo de reconstrução pode existir a sobreposição dos espectros originais que são transladados e somados de acordo com a frequência de amostragem e podem perder parte da informação original ao serem sobreposto, o que irá acarretar em uma perda dos dados originais. Esse fenômeno é conhecido como *aliasing*.

4. Definiçao dos conceitos:

   - Radiância de uma cena: é a energia da luz, por unidade de área, emitida idealmente por cada ponto de uma superfície no espaço 3D em uma dada direção.
   - Irradiância de uma imagem: é a energia da luz por unidade de área em cada ponto do plano da imagem.

   A diferença básica entre os dois conceitos é que irradiância se refere á interação da câmera com os raios de luz refletidos pelo ambiente enquanto que radiãncia se refere à interação da luz com as superfícies dos objetos que compõe a cena. De outra maneira, a irradiância é registrada pelos sensores da câmera enquanto que a radiância, ao ser composta para toda a cena, irá definir como a irradiância será registrada.

5. O modelo Lambertiano considera que cada ponto de uma superfície irá refletir a mesma quantidade de energia em todas as direções. Esse modelo consegue representar de forma aceitável superfícies granuladas, como papéis e alguns tipos de rochas. O principal problema desse modelo é que alguns materiais possuem uma característica especular, onde a maior parte da luz refletida tende a seguir em uma direção específica. Dessa maneira, um exemplo de um possível problema da utilização do modelo Lambertiano é a localização de um objeto a partir de imagens distintas. Se esse objeto apresentar um modelo de reflexão especular, é bem provável que ao utilizarmos o modelo Lambertiano o algoritmo não irá se comportar de forma adequada caso uma das imagens esteja na direção de concentração dos raios refletidos e a outra em um lugar distinto.

6. (a) Um ponto é projetado como um ponto, mas vários pontos podem possuir a mesma projeção. Considere $\mathbf{P} = [X, Y, Z]^T$, sua projeção é $\mathbf{p} = f_d \left[\frac{X}{Z}, \frac{Y}{Z}, 1\right]^T$, sendo f_d a distância focal. Agora considere uma reta r que passa pelo foco, definido na origem:

$$r \begin{cases} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = t \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \end{cases}$$

   Qualquer ponto $\mathbf{P} \in r$, será projetado para $p = f[1, 1, 1]^T$.

(b) Duas retas paralelas que também são paralelas ao plano de projeção serão projetadas conservando-se essas características. Duas retas paralelas que também são paralelas ao eixo óptico se intercedem em um ponto de fuga contido na própria imagem. Para todos os outros casos, as retas podem ser paralelas ou não, deixando de preservar a distância e o ângulo entre elas.

Demonstração: Sejam duas retas r_1 e r_2 que passam pelos pontos $[a,b,c]^T$ e $[d,e,f]^T$, respectivamente; com o vetor diretor $\mathbf{d} = [u,v,w]^T$. Os pontos $\mathbf{A_1} = [a,b,c]^T$, $\mathbf{B_1} = [a+u, b+v, c+w]^T \in r_1$ e as suas projeções são dadas por:

$$\begin{aligned} \mathbf{a_1} &= f_d \left[\frac{a}{c}, \frac{b}{c}, 1\right]^T \\ \mathbf{b_1} &= f_d \left[\frac{a+u}{c+w}, \frac{b+v}{c+w}, 1\right]^T \end{aligned}$$

Dessa maneira, podemos encontrar o vetor diretor $\overrightarrow{a_1b_1}$ da projeção de r_1 e repetir o mesmo procedimento para r_2. Logo:

$$\begin{aligned} \overrightarrow{a_1b_1} &= f_d \left[\frac{cu-aw}{c^2+cw}, \frac{cv-bw}{c^2+cw}, 0\right]^T \\ \overrightarrow{a_2b_2} &= f_d \left[\frac{fu-dw}{f^2+fw}, \frac{fv-ew}{f^2+fw}, 0\right]^T \end{aligned}$$

Para que as retas continuem paralelas é necessário que $||\overrightarrow{a_1b_1} \times \overrightarrow{a_2b_2}|| = 0$, o que, desconsiderando os pontos de divergência, nos leva a:

$$(ae-bd)w^2 + ((bf-ce)u + (cd-af)v)w = 0 \tag{1}$$

Que possui as soluções:

$$w = \begin{cases} 0 \\ \frac{(ce-bf)u+(af-cd)v}{ae-bd} \end{cases}$$

Com $w = 0$ temos que as retas são paralelas ao plano de projeção e continuarão paralelas ao serem projetadas. Também para os casos em que a segunda solução for nula, o mesmo irá ocorrer. Com relação aos pontos de descontinuidade, se o vetor diretor $\mathbf{d} = [0,0,1]^T$, por (1) temos que $ae - bd = 0$, o que irá caracterizar um ponto de fuga contido dentro da própria imagem projetada, onde as retas se encontram. Por inspeção, é possível verificar que para vários casos as retas perdem o paralelismo ao serem projetadas.

(c) Se o plano do círculo for paralelo ao plano da imagem, sua projeção será um círculo, caso contrário, sua projeção será uma elipse. *Demonstração:* Sejam os vetores $\mathbf{n} = [u,v,w]^T$ e $\mathbf{m} = [a,b,c]^T$, linearmente independentes. Os pontos $\mathbf{P}$ que definem um círculo de raio R centrado em $\mathbf{C}$ e com vetor normal $\mathbf{n}$ são dados por:

$$\begin{cases} \mathbf{n}.\mathbf{m} = 0 \\ \mathbf{P} = R\cos(t)\mathbf{m} + R\sin(t)(\mathbf{n} \times \mathbf{m}) + \mathbf{C},\ t \in [0, 2\pi] \end{cases}$$

Logo, as coordenadas (x,y) da projeção desses pontos serão:

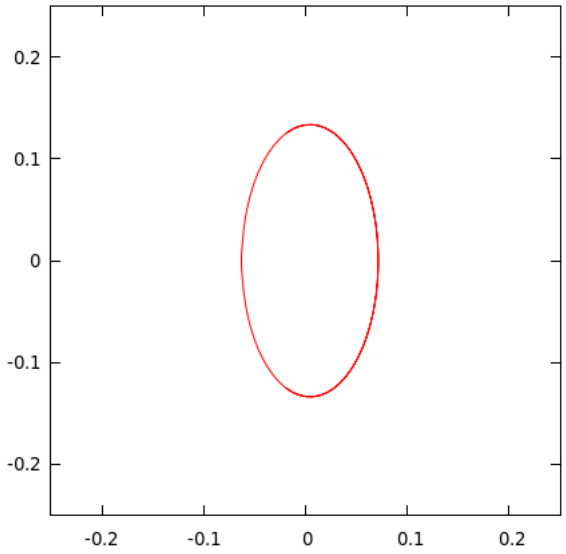
$$\begin{aligned} x &= f_d \frac{aR\cos(t) + (vc-wb)R\sin(t) + C_x}{cR\cos(t) + (ub-va)R\sin(t) + C_z} \\ y &= f_d \frac{bR\cos(t) + (wa-uc)R\sin(t) + C_y}{cR\cos(t) + (ub-va)R\sin(t) + C_z} \end{aligned}$$

Definindo-se $\mathbf{n} = [0, 0, 1]^T$ e $\mathbf{m} = [1, 0, 0]^T$ temos um círculo com a normal paralela à normal do plano da imagem e portanto, sua projeção é um círculo dado por:

$$[x, y]^T = f_d \left[\frac{R\cos(t) + C_x}{C_z}, \frac{R\sin(t) + C_y}{C_z} \right]^T$$

Por inspeção, podemos ver que elipses são formadas para $\mathbf{n} \neq \alpha[0, 0, 1]^T$ com $\alpha \in \mathbb{R}$. Por exemplo, tomando-se $\mathbf{n} = [1, 0, 1]^T$, $\mathbf{m} = [1, 0, -1]^T$, $\mathbf{C} = [0, 0, 10]^T$ e $R = 1$, temos:

$$[x, y]^T = -f_d \left[\frac{\cos(t)}{\cos(t) + 10}, \frac{2\sin(t)}{\cos(t) + 10} \right]^T$$



2 Exercícios Práticos

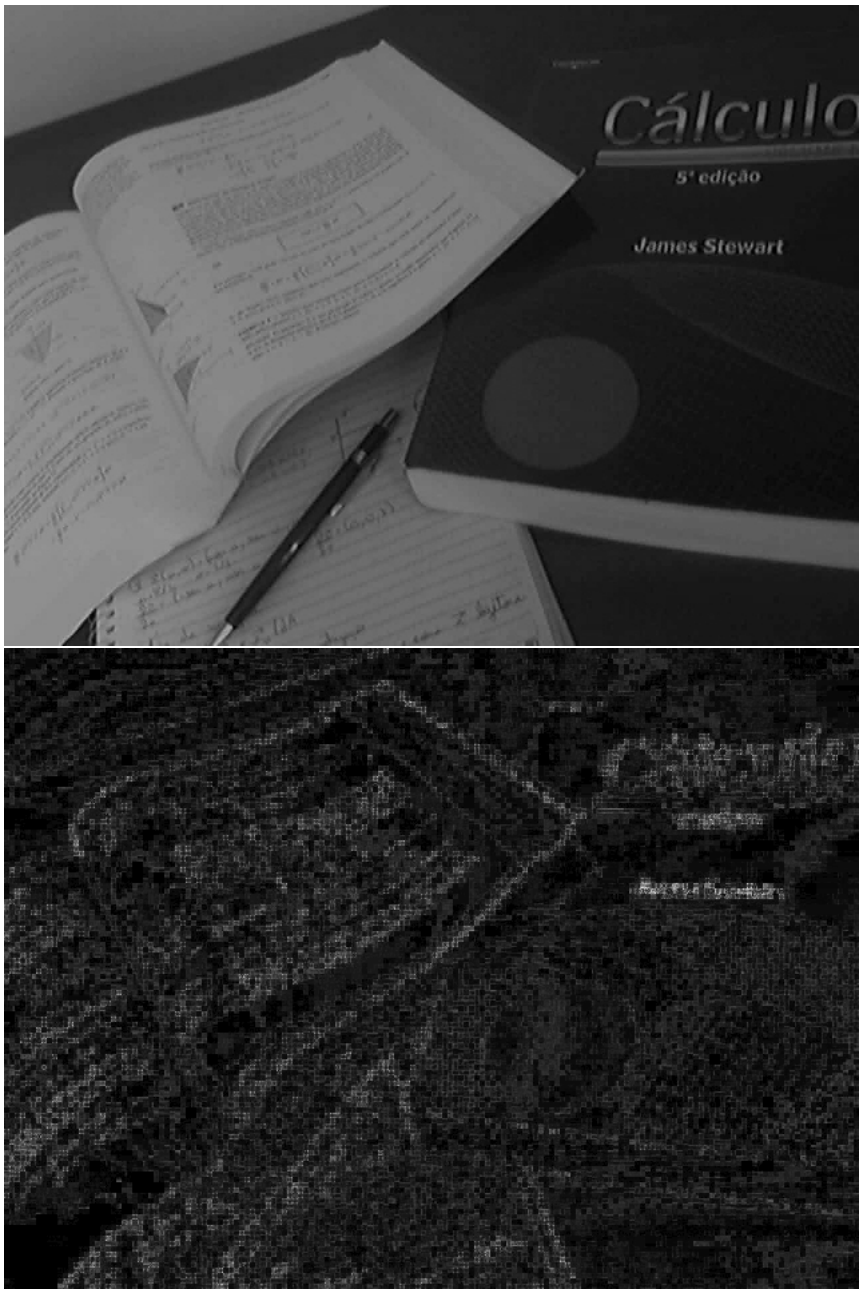
1. Gerar imagens de intensidade a partir de imagens de distância. Feito no Scilab com o auxílio da biblioteca sivp para salvar a imagem.

```
input_name  = "curvblock-1.txt";
output_name = "image2.jpg";

fp = mopen(input_name, "r");
height = mfscanf(fp, "%d rows")
width  = mfscanf(fp, "%d collumns\n")
mgetl(fp, 2);

//Reads the mask.
mask = zeros(height, width);
for i = 1:height
  for j = 1:width
    mask(i, j) = mfscanf(fp, "%d");
  end
end

//Reads the X, Y and Z matrices.
//X and Y are discarded, we only need the Z matrix!
Z = zeros(height, width);
for i = 1:3
  for j = 1:height
    for k = 1:width, Z(j, k) = mfscanf(fp, "%f"); end
  end
end
mclose(fp);

//Makes Z(i,j) belongs to [0, 1] for all i, j.
min_dist = min(Z);
ratio = 1/(max(Z) - min_dist);
for i = 1:height
  for j = 1:width
    if mask(i, j) == 1 then, Z(i, j) = (Z(i, j) - min_dist)*ratio; end
```

```
33    end
34  end
35
36  //sivp function to save the Z matrix.
37  //Basically, it will just multiply it by 255 and save it as a jpg.
38  imwrite(Z, output_name);
```

2. (a) O sistema de captura foi composto por uma webcam *QuickCam®Zoom*^{TM} com 1.3 Megapixels da *Logitech*. As imagens possuem uma resolução de 640×480 e foram convertidas em uma escala de cinza. Informações sobre a distância focal máxima e mínima não foram encontradas. O controle desse parâmetro é feito manualmente através de uma interface contida na câmera.

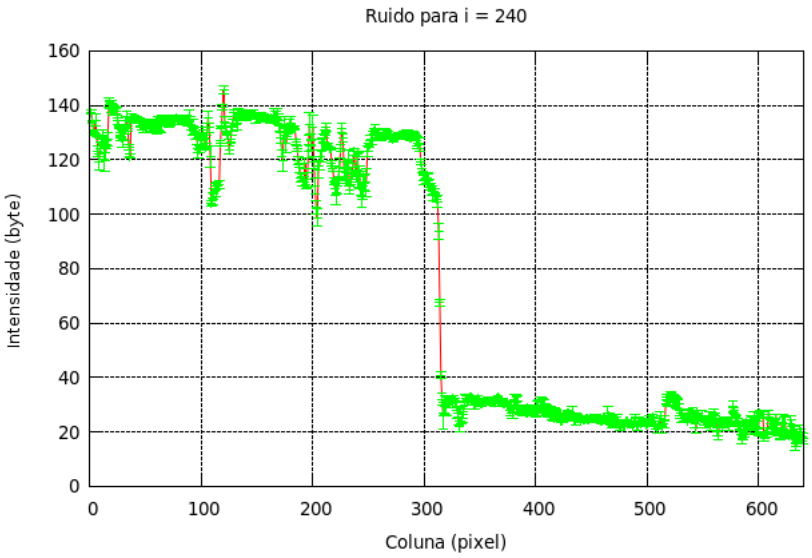
   (b) Segue uma amostra das imagens utilizadas e o resultado do mapa de ruído:



Claramente o mapa de ruído possui uma relação com a imagem. Pelos resultados é possível concluir que o ruído médio foi baixo, devido à coloração escura dominante, embora várias regiões apresentem um alto indice de ruído, principalmente nas bordas dos objetos; o que é esperado por serem regiões de alta frequência.

O resultado mostra que o modelo aditivo não representa com acurácia o ruído presente em uma imagem, já que ele deveria ser uma função sem uma relação direta com a mesma.

(c) Utilizando-se a linha 240:



(d) A média de $\sigma(i,j)$ é aproximadamente 1.15. Esse valor é uma medida que expressa o erro médio de cada pixel.

(e) O pior caso de aquisição de rúido é dado por $max(\sigma(i,j))\ \forall\ i,\ j$. Esse valor é aproximadamente 10.5.

(f) Utilizando-se o modelo aditivo, que expressa uma imagem $I(i,j)$ como sendo uma composição da imagem real $I_R(i,j)$ com uma função-ruído $R(i,j)$, ou seja, $I(i,j) = I_R(i,j) + R(i,j)$ e dados o poder médio do sinal P_s e o ruído médio P_R, podemos expressar o SNR pela razão $SNR = \frac{P_s}{P_R}$. Com base nessas informações, temos:

i. A soma de duas cenas semelhantes e com as mesmas condições tende a não alterar significativamente o SNR, já que ambas as intensidades e os ruídos serão somados. Logo:

$$\begin{aligned} I_1 &\approx I_2 \\ I_1 + I_2 &\approx 2(I_{R1} + R_1) \\ SNR &\approx \frac{2P_s}{2P_R} = \frac{P_s}{P_R} \end{aligned}$$

ii. A soma de uma cena mais intensa do que a outra deve aumentar o SNR, já que o poder médio do sinal irá aumentar e a alteração no ruído deve ser pequena.

3. Subtração de imagens utilizando OpenCV.

```
#include <cv.h>
#include <highgui.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
        IplImage *img1 = 0, *img2 = 0, *output;

        if (argc < 4){
                perror("Usage: subimages [input1] [input2] [output]\n");
                return EXIT_FAILURE;
        }

        //Unix parameters order!
        img1 = cvLoadImage(argv[1]);
        img2 = cvLoadImage(argv[2]);

        if (!img1 || !img2){
                perror("Couldn't load the image files!\n");
                return EXIT_FAILURE;
        }
        else if (img1->height != img2->height || img1->width != img2->width){
                perror("Input images must have the same size!\n");
                return EXIT_FAILURE;
        }
        else if (img1->depth != img2->depth){
                perror("Input images must have the same pixel depth!\n");
                return EXIT_FAILURE;
        }
        else if (img1->nChannels != img2->nChannels){
                perror("input images must have the same number of channels!");
                return EXIT_FAILURE;
        }

        output = cvCreateImage(cvSize(img1->width, img1->height), img1->depth, img1->nChannels);
        cvSub(img1, img2, output, NULL);
        cvSaveImage(argv[3], output);
        cvReleaseImage(&img1);
        cvReleaseImage(&img2);
        cvReleaseImage(&output);
        return EXIT_SUCCESS;
}
```

3 Exercícios de Pesquisa

A formação de imagens em abientes participativos, como a água, parece ser complexa devido a características do meio como a absorção e o espalhamento da luz. Em um oceano, apenas através de um corpo simples e relativamente pequeno formado por água a transmissão da luz pode ser analisada de forma exata, utilizando-se parâmetros oceanográficos simples, como o coeficiente volumétrico de atenuação, o coeficiente de espalhamento e a função de espalhamento volumétrica do ambiente. [1]

Desde os anos 70, vários pesquisadores sugeriram que esse modelo pode ser descrito através de um sistema linear, podendo empregar-se a *point-spread function* (*PSF*) para descrever certas propriedades da imagem e do processo de espalhamento. [2]

Diante de todos esses problemas, o uso dos algoritmos de Visão Computacional a partir das imagens sem tratamento traria resultados catastróficos e imprevisíveis. A maior parte do trabalho dentro de ambientes participativos está no tratamento das imagens, com o objetivo de simplificá-las em modelos já conhecidos dos quais vários algoritmos de Visão podem ser aplicados.

Referências

[1] L. Zhishen, D. Tianfu, W. Gang. ROV Based Underwater Blurred Image Restoration. Journal of Ocean University of China (English Edition), Volume 2, Number. April, 2003.

[2]] Jaffe, J.S. Computer modeling and the design of optical underwater imaging system. IEEE. J. Oceanic Engineering, 15(2):101-111. 1990.