3
Redução de ruído

3.1.
Algoritmo *NLM*

Como a maioria dos algoritmos para redução de ruído, o algoritmo *NLM* usa o cálculo de médias como forma de eliminar ruído. A diferença está em que, enquanto a maioria dos algoritmos usa o fato de que as características que estão próximas entre si na imagem tendem a ter valores semelhantes e portanto podem ser usadas para calcular a média, o algoritmo *NLM* parte de outra suposição: imagens naturais tem características que se repetem e que podem ser detectadas globalmente, e não localmente. Para remover o ruído de um pixel *p*, o algoritmo procura características semelhantes a aquelas no entorno de *p* por toda a imagem, e atribui um peso a cada pixel de acordo com a "semelhança" da sua vizinhança com a vizinhança de *p*. A filtragem de *p* é portanto efetuada através de uma média ponderada de todos os pixels da imagem. A questão de semelhança pode ser explicada usando a Figura 4:

Figura 4 semelhança

Vizinhanças de pixels como para p_1, p_2 e p_3 levam os valores altos para a média $w(p_1, p_2)$ e $w(p_1, p_3)$ enquanto que vizinhanças muito diferentes como para *p1* e *p4* produzem valores baixos para a média $w(p_1, p_4)$. Os valores dos pixels p_2 e p_3 terão pesos muito mais altos na média computada para p_1 do que p_4.

A semelhança é computada usando a ***distância Euclidiana ponderada*** da vizinhança de pixel. Esta métrica foi provada adequada em [2], e consistente e um ambiente com ruído pois ela aumenta a distância entre dois pixels originalmente idênticos por uma constante. Isto será demonstrado depois das definições a seguir.

Dada uma imagem com ruído $v = \{v(i) \mid i \in I\}$, o valor estimado $NL(v)(i)$ é computado como a média ponderada de todos os pixels da imagem:

Equação 1. $$NL(v)(i) = \sum_{j \in I} w(i,j)v(j)$$

onde o peso pertence ao intervalo [0,1] e a soma de todos os pesos é 1

$0 \le w(i,j) \le 1$ e $\sum_{j \in I} w(i,j) = 1$.

Um sistema de vizinhança para *I* é uma família $N = \{N_i\}_{i \in I}$ de subconjuntos de *I* tal que, para todo $i \in I$:

1. $i \in N_i$
2. $j \in N_i \Rightarrow i \in N_j$

O subconjunto N_i é chamado de ***vizinhança*** ou janela de semelhança para i. As janelas de semelhança podem ter tamanhos e formatos distintos, mas por simplicidade uma janela retangular será utilizada. A semelhança entre os pixels i e j depende da semelhança entre a intensidade dos níveis de cinza, dada pelos vetores $v(N_i)$ e $v(N_j)$. Na Figura 4, a vizinhança dos pixels p_1, p_2 e p_3 é dada pela coleção dos níveis de cinza de todos os pixels nos quadrados que os circundam. Veja a Figura 5 para um exemplo.

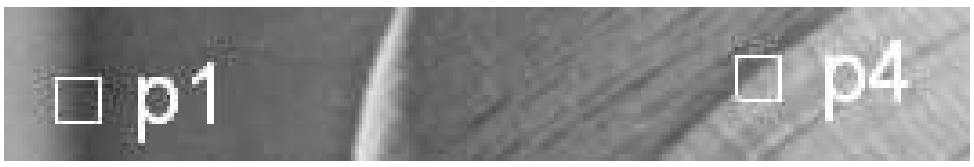

Figura 5 Vizinhança de similaridade

Pixels com vizinhanças de cinza similar a $v(N_i)$ terão pesos maiores no cálculo da média do pixel i. A semelhança é computada usando uma versão com pesos da distância Euclidiana. A ***distância Euclidiana quadrática*** (L^2) entre dois vetores

$$x = (x_1, x_2, ..., x_k),\ y = (y_1, y_2, ..., y_k)$$

é definida por:

Equação 2. $$d^2(x,y) = \sum_{j=1}^{k} (x_j - y_j)^2$$

e a ***distância Euclidiana quadrática ponderada*** (L^2 *ponderada*) é definida como:

Equação 3. $$d_a^2(x,y) = \sum_{j=1}^{k} a_j (x_j - y_j)^2 \quad e \quad a_j \geq 0$$

onde a contribuição da distância em cada eixo j à distância total é ponderada pelo coeficiente a_j.

No algoritmo *NLM*, os pesos $a_j, j = 1,...,k$ são definidos usando um kernel Gaussiano bidimensional. A função de kernel é a versão bidimensional da distribuição normal, dada pela equação:

$$G(i,j) = \frac{1}{4\pi\sigma^2} e^{\frac{i^2+j^2}{-4\sigma^2}}$$

A forma geral do kernel Gaussiano bidimensional está exibida no lado direito da Figura 6, e um kernel 5x5 está desenhado no lado esquerdo. Quando o kernel é aplicado à vizinhança centrada no pixel *p*, o pixel central (*p*) recebe o maior peso e os outros pixels da vizinhança recebem peso exponencialmente inverso à sua distância a *p*.

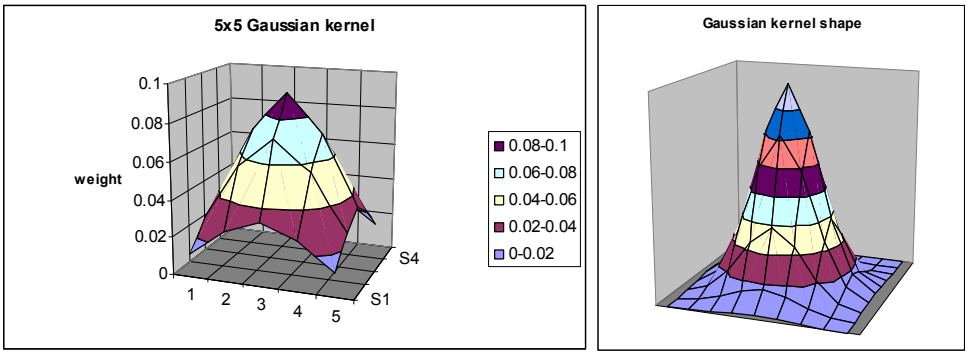

Figura 6 Kernel Gaussiano

O exemplo acima define uma vizinhança de 5x5 pixels e um kernel Gaussiano estão definidos; d_a^2 é computada entre 2 vetores de tamanho 25. Os coeficientes do kernel são computados a partir função Gaussiana bidimensional com desvio padrão a: $G_a(i,j) = \frac{1}{4\pi a^2} e^{\frac{i^2+j^2}{-4a^2}}$ para valores de $i, j = -2,-1,0,1,2;$ os coeficientes são normalizados de tal forma que $\sum_{j=1}^{k} a_j = 1$ por

$$a_{i*5+j} = \frac{G_a(i,j)}{\sum_{i,j=(-2,\ldots,2)} G_a(i,j)}$$

O vetor do kernel R^{25} neste caso está descrito Figura 7:

$$\frac{1}{577} \text{ x } \begin{array}{|c|c|c|c|c|} \hline 2 & 7 & 12 & 7 & 2 \\ \hline 7 & 31 & 52 & 31 & 7 \\ \hline 12 & 52 & 127 & 52 & 12 \\ \hline 7 & 31 & 52 & 31 & 7 \\ \hline 2 & 7 & 12 & 7 & 2 \\ \hline \end{array}$$

Figura 7 5x5 Kernel Gaussiano

Note que a soma de todos os 25 componentes do vetor (depois de normalizados por $\frac{1}{577}$) é 1, e o peso de cada pixel no kernel é inversamente proporcional à sua distância ao centro.

Depois de discutir a política de atribuição de pesos, podemos voltar a definição da distância. *Ni* e *Nj* são janelas de semelhança centradas no pixels *i* e *j* que se correspondem e portanto

$$v(Ni)=(v(Ni_1),v(Ni_2),...,v(Ni_k)) \text{ e } v(Nj)=(v(Nj_1),v(Nj_2),...,v(Nj_k))$$

são ambos vetores de intensidade de cinza com a mesma cardinalidade do kernel Gaussiano. A distância L^2 ponderada, d_a^2, pode ser escrita como:

$$|| v(Ni)-v(Nj) ||_{2,a}^2 = \sum_{l=1}^{k} a_l (v(Ni_l)-v(Nj_l))^2$$

Após definir a distância, podemos definir como pesos são atribuídos a cada pixel de acordo com a Equação 1. Os pesos associados à distância são definidos por:

Equação 4. $$w(i,j)=\frac{1}{Z(i)}e^{(-1)\frac{||v(Ni)-v(Nj)||_{2,a}^2}{h^2}},$$

onde Z(i) é o fator de normalização:

Equação 5. $$Z(i)=\sum_{j\in I} e^{(-1)\frac{||v(Ni)-v(Nj)||_{2,a}^2}{h^2}},$$

e o *fator de decaimento h* controla o decaimento da função exponencial e portanto o decaimento dos pesos em função da distância Euclidiana. Note que *h* é um "fator de filtragem" no sentido de que é responsável por definir os pesos adequados de acordo com a distância computada. Para valores pequenos de *h*, uma distância muito pequena deve ser computada para ter alguma contribuição, e para valores altos de *h*, mesmo distâncias consideráveis podem influenciar o valor do pixel. Isto significa que se escolhermos um valor de *h* muito alto, podemos distorcer a imagem e, se escolhermos um valor de *h* muito baixo, podemos não remover suficientemente o ruído. A Figura 8 mostra o decaimento rápido da função de ponderação para valores baixos de h como uma função da distância. A curva no alto da figura, $h^2 = 20$, decai vagarosamente com a distância, enquanto que a curva na parte de baixo da figura, $h^2 = 2$, decai abruptamente e resulta em pesos bastante pequenos para distâncias acima de $d_a^2 = 5$.

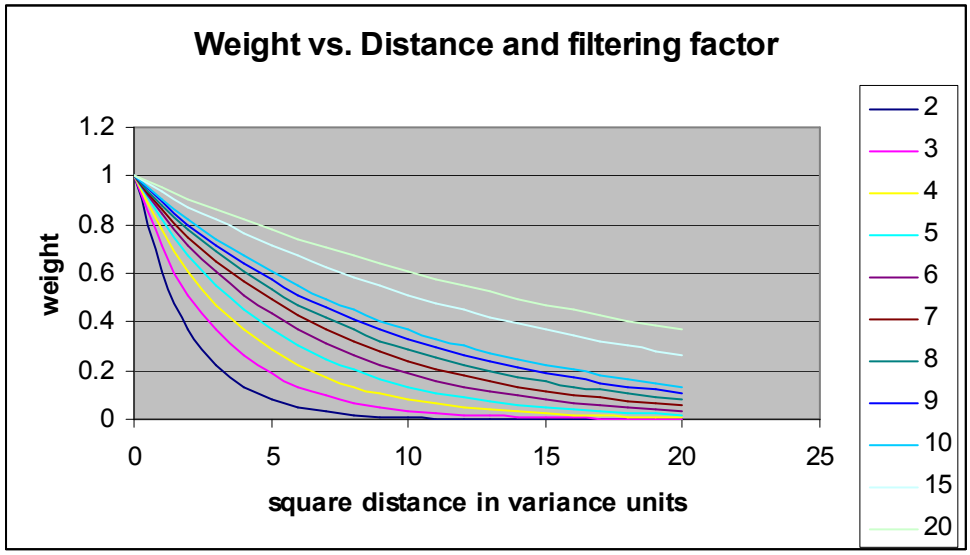

Figura 8 Peso como uma função da distância e do fator de decaimento

A Figura 9 demonstra que o efeito do fator de decaimento na imagem filtrada. No alto à esquerda, a imagem com ruído para σ_n=20. As 3 outras imagens são filtradas pelo *NLM* com fatores de decaimento diferentes. As imagens no topo à direita possuem $h^2 = 200$; na parte de baixo à esquerda, $h^2 = 100$; e na parte de baixo à direita, $h^2 = 2000$.

Figura 9 Efeito do fator de decaimento na imagem filtrada

A imagem na parte de baixo à esquerda está "pouco filtrada" e mostra um considerável nível de ruído. A imagem na parte de baixo à direita está "muito filtrada" e muitos detalhes finos desapareceram junto com o ruído. A imagem no topo à direita, filtrada de acordo com a recomendação original de $h^2 = 10\sigma_n$, apresenta um bom equilíbrio entre perda de detalhes e ruído residual.

Note que para "ruído branco" com desvio padrão σ_n e média zero, o valor esperado da distância L^2 ponderada entre duas vizinhanças é dado por:

$$E\left\|v(N_i) - v(N_j)\right\|_{2,a}^2 = \left\|u(N_i) - u(N_j)\right\|_{2,a}^2 + 2\sigma_n^2 .$$

A distância L^2 ponderada das duas vizinhanças da imagem com ruído possui uma distância fixa de $2\sigma_n^2$ da distância das mesmas vizinhanças na imagem original antes da adição de ruído branco. Se as duas vizinhanças originalmente

similares forem comparadas $\left(\left\|u(N_i)-u(N_j)\right\|=0\right)$, as vizinhanças com ruído deverão ter uma distância L^2 ponderada de $2\sigma_n^2$. Isto mostra que a distância L^2 ponderada é de fato consistente entre a imagem original e a imagem com ruído, adicionando duas vezes a variância do ruído à distância computada.

3.2. Comentários sobre *NLM* para vídeo

Depois de apresentar os princípios do *NLM*, podemos comentar que o *NLM* não é específico para imagens estáticas. *NLM* pode operar de forma semelhante nos quadros de um filme, usando para computar a média para os pixels de outros quadros. Isto é lógico pois a probabilidade de uma série de quadros consecutivos compartilhar muitos pixels similares é muito alta. Métodos anteriores que tentaram usar médias tomadas entre quadros usualmente enfrentam o problema de "estimativa de movimento": a necessidade de estimar o movimento relativo entre dois quadros de tal forma que o pixel filtrado possa ser identificado em quadros diferentes. O *NLM*, no entanto, não precisa levar em consideração este problema, pois a pesquisa por vizinhanças semelhantes pode ser feita sobre todos os pixels em quadros subseqüentes ou usando uma janela de busca que limite a pesquisa em três dimensões em lugar de duas. Este algoritmo tem a mesma complexidade (de tempo) que o algoritmo original multiplicada pelo número de quadros que participar do cálculo da média. A alta complexidade deste algoritmo impede que seja utilizado em aplicações reais de processamento de vídeo.

3.3. Complexidade de tempo do *NLM*

A complexidade do tempo do *NLM* pode ser escrita como $O\left(n^2 w\right)$ onde *n* é o número de pixels na imagem e *w* é o tamanho da janela de vizinhança. Quando se considera o uso do *NLM* para imagens coloridas de 3Mpixel, com três planos de cores e uma janela de vizinhança pequena de tamanho 25, o número de operações atinge $3*\left(3*2^{20}\right)^2*25=7.42*10^{14}$. Isto é claramente impraticável em tempo razoável. Por esta razão, *NLM* define uma *janela de busca* dentro da qual a busca por vizinhanças semelhanças será efetuada. Isto depende da suposição de

que em uma imagem natural características próximas tendem a ser semelhantes. Isto nos permite reduzir a complexidade do tempo para $O(nsw)$, quando s é o tamanho da janela de busca. Repetindo o exemplo acima com uma janela de busca de tamanho 400, temos $3*(3*2^{20})*400*25 = 9.44*10^{10}$ operações. Isto ainda é da ordem de 100Giga operações.

Note que usando uma janela de busca em lugar de pesquisar toda a imagem, o algoritmo modificado abandona a idéia de filtrar características semelhantes na imagem que estão a uma distância maior do que a janela de busca. Na Figura 10, ao filtrar p_1, podemos usar o valor de p_2 na filtragem mas não o valor de p_3 pois ele está fora da janela de busca (indicada pelo quadrado branco em volta de p_1).

Para concluir, mesmo sacrificando características do algoritmo original, o algoritmo modificado não reduz o tempo de processamento a níveis razoáveis.

Outra forma imediata de reduzir a complexidade seria reduzir o tamanho da janela de semelhança. Isto pode causar distorções na imagem por falta de informação sobre semelhança. Os autores do *NLM* [1] recomendam uma janela de semelhança de 9x9 ou 7x7 para imagens em níveis de cinza e uma janela de semelhança de 5x5 ou mesmo 3x3 para imagens coloridas com pouco ruído. Podemos então concluir que não há muita margem para acelerar o algoritmo sem sacrificar esta característica importante.

Figura 10 Janela de semelhança