Anleitung zur optimalen Implementierung für die normierte Kreuzkorrelation

Dorian Schneider
Lehrstuhl für Bildverarbeitung, RWTH Aachen

February 28, 2012

1 Normiere Kreuzkorrelation

$$f(u,v) = \frac{z(u,v)}{n(h,v)} = \frac{\sum_{x,y}\left[f(x,y) - \bar{f}_{u,v}\right]\left[t(x-u,y-v) - \bar{t}\right]}{\sqrt{\sum_{x,y}\left[f(x,y) - \bar{f}_{u,v}\right]^2 \sum_{x,y}\left[t(x-u,y-v) - \bar{t}\right]^2}} \tag{1}$$

$mit:$

$t(x,y) \Rightarrow$ Template $\in \mathbb{R}^{x \times y}$

$\bar{t} \Rightarrow \mu$ des Templates

$f(x,y) \Rightarrow$ Eingangsbild

$\bar{f}_{u,v} \Rightarrow \mu$ der Bildregion unter dem Template

$f(u,v) \Rightarrow$ korreliertes Bild

$\sum\limits_{x,y} \Rightarrow$ Summe unter dem Fenster des Templates

2 Der Zähler

Der Term $[t(x-u,y-v)-\bar{t}]$ im Zähler $z(u,v)$ der Gleichung (1)kann vorberechnet werden, der Zähler lässt sich demnach umschreiben.

Es gilt:
$t'(x-u,y-v) = [t(x-u,y-v)-\bar{t}]$

$$\begin{aligned}
z(u,v) &= \sum_{x,y} \left[f(x,y) - \bar{f}_{u,v}\right] t'(x-u,y-v) \\
&= \sum_{x,y} \left[f(x,y)t'(x-u,y-v) - \bar{f}_{u,v}t'(x-u,y-v)\right] \\
&= \sum_{x,y} f(x,y)t'(x-u,y-v) - \sum_{x,y} \bar{f}_{u,v}t'(x-u,y-v) \\
&= \sum_{x,y} f(x,y)t'(x-u,y-v) - \bar{f}_{u,v} \sum_{x,y} t'(x-u,y-v)
\end{aligned}$$

$$\text{da } t' \text{ Mittelwert frei ist gilt:} \sum_{x,y} t'(x-u,y-v) = 0$$

$$\Rightarrow z(u,v) = \sum_{x,y} f(x,y)t'(x-u,y-v)$$

$$\Rightarrow z(u,v) = \mathcal{F}^{-1}\left[\mathcal{F}(f(x,y)) \cdot \mathcal{F}^{*}(t'(x,y))\right] \tag{2}$$

3 Der Nenner

$$n(u,v) = \sqrt{\sum_{x,y} \left[f(x,y) - \bar{f}_{u,v}\right]^2 \sum_{x,y} \left[t(x-u,y-v) - \bar{t}\right]^2}$$

$$\Rightarrow \sum_{x,y} \left[f(x,y) - \bar{f}_{u,v}\right]^2 = \sum_{x,y} \left[f^2(x,y) - 2\bar{f}_{u,v} f(x,y) + \bar{f}^2_{u,v}\right]$$

$$\Rightarrow \sum_{x,y} 2\bar{f}_{u,v} f(x,y) = 2\bar{f}_{u,v} \sum_{x,y} f(x,y) = 2A\bar{f}^2_{u,v}, A = (x \cdot y)$$

$$\Rightarrow \sum_{x,y} \left[f^2(x,y) - 2\bar{f}_{u,v} f(x,y) + \bar{f}^2_{u,v}\right] = \sum_{x,y} f^2(x,y) - 2A\bar{f}^2_{u,v} + \sum_{x,y} \bar{f}^2_{u,v}$$

$$= \sum_{x,y} f^2(x,y) - 2A\bar{f}^2_{u,v} + A \cdot \bar{f}^2_{u,v} = \sum_{x,y} f^2(x,y) - A \cdot \bar{f}^2_{u,v}$$

$$= \sum_{x,y} f^2(x,y) - \frac{1}{A} \left[\sum_{x,y} f(x,y)\right]^2$$

$$\Rightarrow n(u,v) = \sqrt{\left[\sum_{x,y} f^2(x,y) - A \cdot \bar{f}^2_{u,v}\right] \sum_{x,y} t'^2}$$

mit t_σ der Standardabweichung von $t(x,y)$:

$$n(u,v) = \sqrt{\sum_{x,y} f^2(x,y) - \frac{1}{A} \left[\sum_{x,y} f(x,y)\right]^2} \cdot \sqrt{A} \cdot t_\sigma \tag{3}$$

4 Gesamtgleichung

Aus Gleichung (1), (2) und (3) ergibt sich:

$$f(u,v) = \frac{\mathcal{F}^{-1}\left[\mathcal{F}(f(x,y)) \cdot \mathcal{F}^*(t'(x,y))\right]}{\sqrt{\sum_{x,y} f^2(x,y) - \frac{1}{A}\left[\sum_{x,y} f(x,y)\right]^2} \cdot \sqrt{A} \cdot t_\sigma} \tag{4}$$

5 Implementierung

1. Offline: Standardabweichung des Templates mal $\sqrt{A}$ berechnen $= t_\sigma$, im GPU speicher halten.

2. Offline: konjugiert komplexe FFT des Mittelwert befreiten Templates berechnen, FFT im Speicher halten. FFT berechnen mit CUFFT.

3. Online: 2 Integralbilder (laufende Summen, nicht Zeilenweise) des Bildes berechnen: 1) Integralbild I_{f^2} des quadrierten Bildes und 2) Quadriertes Integralbild I_f^2 des Bildes. Es empfiehlt sich eine CUDA Variante[1] zu nutzen. Beide Bilder im GPU Speicher halten.

4. Die Summen der Integralbilder auf Templategröße begrenzen. Für jedes Pixel der Integralbilder auf der GPU folgende Berechnung machen:

$$
\begin{aligned}
I'_{f^2}(x,y) = &+ I_{f^2}(x-1, y-1) \\
&+ I_{f^2}(x+M+-1, y+N-1) \\
&- I_{f^2}(x+M+-1, y-1) \\
&- I_{f^2}(x-1, y+N-1)
\end{aligned} \tag{5}
$$

$$
\begin{aligned}
I'^2_f(x,y) = &+ I^2_f(x-1, y-1) \\
&+ I^2_f(x+M+-1, y+N-1) \\
&- I^2_f(x+M+-1, y-1) \\
&- I^2_f(x-1, y+N-1)
\end{aligned} \tag{6}
$$

mit M, N Höhe und Breite des Templates

5. FFT des Bildes berechnen, mit Template FFT multiplizieren, inverse FFT anwenden. Das ergibt den Zähler in Gleichung (4). Zählerbild z(x,y) im GPU Speicher halten.

6. Über CUDA Kernel, pixelweise das Nennerbild bestimmen:

$$
n(x,y) = \sqrt{I'_{f^2}(x,y) - \frac{I'^2_f(x,y)}{A}} \cdot t_\sigma \tag{7}
$$

7. Gesamtergebnise bestimmen über pixelweise Division:

$$
f_x corr(x,y) = \frac{z(x,y)}{n(x,y)} \tag{8}
$$

[1] Bilgic, Horn, Masaki:"Efficient Integral Image Computation on the GPU", 2010

6 Matlab Implementierung

```
function fCorr = fastXcorr(t, f)
    t = double(t);
    f = double(f);
    [m n] = size(t);
    A = m*n;

    % Schritt 1: Standardabweichung Template
    t_sigma = sqrt(A)*std(t(:))

    % Schritt 2: FFT des Mittelwert freien Templates
    t = t -mean(t(:));

    f_size = size(f);
    outsize = f_size + [m n] - 1;

    fft_t = fft2(rot90(t,2),outsize(1),outsize(2));

    % Schritt 3 und 4: Integralbilder, mit gegrenzten Summen
    If2_1 = intImage(f.^2,m,n);
    I2f_1 = intImage(f,m,n).^2;

    % Schritt 5: Zaehler bestimmen
    fft_f   = fft2(f,outsize(1),outsize(2));
    zaehler = real(ifft2(fft_f .* fft_t));

    % Schritt 6: Nenner bestimmen
    nenner = sqrt( If2_1 -  I2f_1./A).*t_sigma;

    % Schritt 7: Gesamtergebnis bestimmen
    fCorr = zaehler./nenner;


% Integralbild berechnen.
% Das muss man in C++ anders angehen , siehe Text
function I = intImage(A,m,n)
    B = padarray(A,[m n]);
    s = cumsum(B,1);
    c = s(1+m:end-1,:)-s(1:end-m-1,:);
    s = cumsum(c,2);
    I = s(:,1+n:end-1)-s(:,1:end-n-1);
```