Registro de Imagens

Alexandre Xavier Falcão

Instituto de Computação - UNICAMP

afalcao@ic.unicamp.br

Introdução

- Sejam $\hat{I} = (D_I, \vec{I})$ e $\hat{J} = (D_J, \vec{J})$ duas imagens que precisam ser registradas em um mesmo sistema de coordenadas.

- Sejam $\hat{I} = (D_I, \vec{I})$ e $\hat{J} = (D_J, \vec{J})$ duas imagens que precisam ser registradas em um mesmo sistema de coordenadas.
- Podemos considerar $\hat{I}$, por exemplo, como imagem **fixa** (referência) e colocar $\hat{J}$ (imagem **móvel**) no sistema de coordenadas de $\hat{I}$.

- Sejam $\hat{I} = (D_I, \vec{I})$ e $\hat{J} = (D_J, \vec{J})$ duas imagens que precisam ser registradas em um mesmo sistema de coordenadas.
- Podemos considerar $\hat{I}$, por exemplo, como imagem **fixa** (referência) e colocar $\hat{J}$ (imagem **móvel**) no sistema de coordenadas de $\hat{I}$.
- O registro consiste em encontrar uma ou mais transformações geométricas, dependendo da região $\mathcal{R} \subset D_J$, tais que para todo $q \in D_J$, exista $\phi_{\mathcal{R}}(q) = p \in D_I$, de modo a obtermos a imagem registrada $\hat{R} = (D_I, \vec{J})$.

- O registro é dito **rígido** se ϕ independe de $\mathcal{R}$ e envolve apenas transformações de translação e rotação.

Introdução

- O registro é dito **rígido** se ϕ independe de $\mathcal{R}$ e envolve apenas transformações de translação e rotação.
- O registro é dito **deformável global** quando ϕ independe de $\mathcal{R}$ e envolve transformações de translação, rotação, e escala.

- O registro é dito **rígido** se ϕ independe de $\mathcal{R}$ e envolve apenas transformações de translação e rotação.
- O registro é dito **deformável global** quando ϕ independe de $\mathcal{R}$ e envolve transformações de translação, rotação, e escala.
- O registro é dito **deformável local** nos casos em que $\phi_{\mathcal{R}}$ varia com a região $\mathcal{R} \subset D_J$.

Introdução

- O registro é dito **rígido** se ϕ independe de $\mathcal{R}$ e envolve apenas transformações de translação e rotação.
- O registro é dito **deformável global** quando ϕ independe de $\mathcal{R}$ e envolve transformações de translação, rotação, e escala.
- O registro é dito **deformável local** nos casos em que $\phi_{\mathcal{R}}$ varia com a região $\mathcal{R} \subset D_J$.
- O caso deformável local é normalmente resolvido com uma deformação global, seguida da deformação local de pontos de controle de uma malha de *patches* formada por *splines* ao longo das direções x, y, e z, e finalizada com a interpolação dos valores de $\vec{J}$ dentro de cada *patch* (i.e., região $\mathcal{R}$).

Esta aula trata o problema de encontrar uma transformação geométrica ϕ que possa ser aplicada a todos os pontos do domínio D_J (registro deformável global). Por exemplo:

$$\phi = \mathbf{R}_z(\theta_z)\mathbf{R}_y(\theta_y)\mathbf{R}_x(\theta_x)\mathbf{T}(t_x, t_y, t_z)\mathbf{S}(s_x, s_y, s_z)$$

é uma transformação com 9 parâmetros, que podem ser encontrados por **otimização** de uma **função critério** (objetivo).

A transformação ótima ϕ^* pode ser encontrada por

- cálculo direto via correspondência entre 4 pontos dados em cada domínio, D_I e D_J, no caso 3D (ou entre 3 pontos no caso 2D);

A transformação ótima ϕ^* pode ser encontrada por

- cálculo direto via correspondência entre 4 pontos dados em cada domínio, D_I e D_J, no caso 3D (ou entre 3 pontos no caso 2D);
- busca no espaço de parâmetros de ϕ via correspondências entre subconjuntos $\mathcal{S}_I \subset D_I$ e $\mathcal{S}_J \subset D_J$ de pontos extraídos de ambos domínios (e.g., usando SIFT - *Scale-Invariant Feature Transform*), ou entre características das imagens $\hat{I}$ e $\hat{J}$ associadas a esses pontos, ou ainda entre características de $\hat{I}$ (e.g., gradiente) e pontos $\mathcal{S}_J \subset D_J$ (e.g., pontos de borda).

Introdução

A transformação ótima ϕ^* pode ser encontrada por

- cálculo direto via correspondência entre 4 pontos dados em cada domínio, D_I e D_J, no caso 3D (ou entre 3 pontos no caso 2D);
- busca no espaço de parâmetros de ϕ via correspondências entre subconjuntos $\mathcal{S}_I \subset D_I$ e $\mathcal{S}_J \subset D_J$ de pontos extraídos de ambos domínios (e.g., usando SIFT - *Scale-Invariant Feature Transform*), ou entre características das imagens $\hat{I}$ e $\hat{J}$ associadas a esses pontos, ou ainda entre características de $\hat{I}$ (e.g., gradiente) e pontos $\mathcal{S}_J \subset D_J$ (e.g., pontos de borda).
- Em qualquer caso, ϕ^* é extrapolada aos demais spels de D_J para gerar $\hat{R} = (D_I, \vec{J})$.

Por cálculo direto

Uma estratégia simples, mas inexata, é encontrar os parâmetros de

$$\phi = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

usando a **regra de Cramer** e a correspondência entre 4 pontos $p_i = (x_i, y_i, z_i, 1)$ de D_I e $q_i = (x'_i, y'_i, z'_i, 1)$ de D_J, respectivamente, para $i = 1, 2, 3, 4$.

Por cálculo direto

Neste caso, a primeira linha de ϕ é obtida por

$$
\begin{aligned}
a_{11} &= \begin{vmatrix} x'_1 & y_1 & z_1 & 1 \\ x'_2 & y_2 & z_2 & 1 \\ x'_3 & y_3 & z_3 & 1 \\ x'_4 & y_4 & z_4 & 1 \end{vmatrix} \div \begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix} \\
a_{12} &= \begin{vmatrix} x_1 & x'_1 & z_1 & 1 \\ x_2 & x'_2 & z_2 & 1 \\ x_3 & x'_3 & z_3 & 1 \\ x_4 & x'_4 & z_4 & 1 \end{vmatrix} \div \begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix}
\end{aligned}
$$

Por cálculo direto

$$
\begin{aligned}
a_{13} &= \begin{vmatrix} x_1 & y_1 & x'_1 & 1 \\ x_2 & y_2 & x'_2 & 1 \\ x_3 & y_3 & x'_3 & 1 \\ x_4 & y_4 & x'_4 & 1 \end{vmatrix} \div \begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix} \\
a_{14} &= \begin{vmatrix} x_1 & y_1 & z_1 & x'_1 \\ x_2 & y_2 & z_2 & x'_2 \\ x_3 & y_3 & z_3 & x'_3 \\ x_4 & y_4 & z_4 & x'_4 \end{vmatrix} \div \begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix}
\end{aligned}
$$

e as demais linhas seriam obtidas de modo similar, variando as colunas com y'_i, $i = 1, 2, 3, 4$, para a linha 2 e com z'_i, $i = 1, 2, 3, 4$, para a linha 3.

Por otimização paramétrica

A busca pela transformação ótima ϕ^* pode envolver a minimização (maximização) de uma **função critério** (objetivo) entre vários candidatos ϕ. Por exemplo, para $\mathcal{S}_J \subset D_J$ e $\mathcal{S}_I \subset D_I$ dados, considere $p \in \mathcal{S}_I$ como o pixel mais próximo de $\phi(q)$. Podemos **minimizar**

$$\sum_{\forall q \in \mathcal{S}_J, p \in \mathcal{S}_I} w_1(p,q) \qquad \text{para}$$

$$w_1(p,q) = \begin{cases} \|q-p\|^2 & \text{se } \|p-\phi(q)\| < \delta \\ w_{\max} & \text{no caso contrário} \end{cases}$$

$$\sum_{\forall q \in \mathcal{S}_J, p \in \mathcal{S}_I} w_2(p,q) \qquad \text{para}$$

$$w_2(p,q) = \begin{cases} \|\vec{I}(q)-\vec{I}(p)\|^2 & \text{se } \|p-\phi(q)\| < \delta \\ w_{\max} & \text{no caso contrário} \end{cases}$$

onde δ é baixo e $w_{\max}$ é o maior peso possível em cada caso.

Por otimização paramétrica

- Para $\mathcal{S}_J \subset D_J$ (pontos de borda) e $\hat{G} = (D_I, G)$ (gradiente de $\hat{I}$), podemos **maximizar** $\sum_{\forall q \in \mathcal{S}_J, p = \phi(q)} G(p)$.

Por otimização paramétrica

- Para $\mathcal{S}_J \subset D_J$ (pontos de borda) e $\hat{G} = (D_I, G)$ (gradiente de $\hat{I}$), podemos **maximizar** $\sum_{\forall q \in \mathcal{S}_J, p=\phi(q)} G(p)$.
- Se $\vec{I} = I$ e $\vec{J} = J$, podemos ainda esperar alta dependência estatística entre as subimagens $(\mathcal{S}_I, I)$ e $(\mathcal{S}_J, J)$, onde $\mathcal{S}_I$ é formado pelos pixels $p = \phi(q)$ para todo $q \in \mathcal{S}_J$.

Por otimização paramétrica

- Para $\mathcal{S}_J \subset D_J$ (pontos de borda) e $\hat{G} = (D_I, G)$ (gradiente de $\hat{I}$), podemos **maximizar** $\sum_{\forall q \in \mathcal{S}_J, p=\phi(q)} G(p)$.
- Se $\vec{I} = I$ e $\vec{J} = J$, podemos ainda esperar alta dependência estatística entre as subimagens $(\mathcal{S}_I, I)$ e $(\mathcal{S}_J, J)$, onde $\mathcal{S}_I$ é formado pelos pixels $p = \phi(q)$ para todo $q \in \mathcal{S}_J$.
- Sejam $L_1 = \max_{\forall p \in \mathcal{S}_I}\{I(p)\}$ e $L_2 = \max_{\forall q \in \mathcal{S}_J}\{J(q)\}$, $h_1(l_1)$ o histograma da subimagem $(\mathcal{S}_I, I)$, $h_2(l_2)$ o histograma da subimagem $(\mathcal{S}_J, J)$, e $h_{12}(l_1, l_2)$ o histograma conjunto das subimagens $(\mathcal{S}_I, I)$ e $(\mathcal{S}_J, J)$. Podemos **maximizar** a informação mútua $\sum_{\forall l_1 \in [0, L_1]} \sum_{\forall l_2 \in [0, L_2]} h_{12}(l_1, l_2) \log \frac{h_{12}(l_1, l_2)}{h_1(l_1) h_2(l_2)}$.

Por otimização paramétrica

- Em qualquer caso, estamos interessados em maximizar (minimizar) uma função critério $F(\vec{\theta})$, onde $\vec{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$ é o vetor de parâmetros da transformação ϕ.

Por otimização paramétrica

- Em qualquer caso, estamos interessados em maximizar (minimizar) uma função critério $F(\vec{\theta})$, onde $\vec{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$ é o vetor de parâmetros da transformação ϕ.
- Encontrar ϕ^* é encontrar $\vec{\theta}^*$ ótimo.

Por otimização paramétrica

- Em qualquer caso, estamos interessados em maximizar (minimizar) uma função critério $F(\vec{\theta})$, onde $\vec{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$ é o vetor de parâmetros da transformação ϕ.
- Encontrar ϕ^* é encontrar $\vec{\theta}^*$ ótimo.
- A busca exaustiva é inviável e a busca por gradiente descendente (ascendente) de $F(\vec{\theta})$ é sensível a mínimos (máximos) locais.

Por otimização paramétrica

- Em qualquer caso, estamos interessados em maximizar (minimizar) uma função critério $F(\vec{\theta})$, onde $\vec{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$ é o vetor de parâmetros da transformação ϕ.
- Encontrar ϕ^* é encontrar $\vec{\theta}^*$ ótimo.
- A busca exaustiva é inviável e a busca por gradiente descendente (ascendente) de $F(\vec{\theta})$ é sensível a mínimos (máximos) locais.
- Uma abordagem popular é o algoritmo RANSAC, que avalia famílias aleatórias de $\vec{\theta}$ em várias iterações.

- Em qualquer caso, estamos interessados em maximizar (minimizar) uma função critério $F(\vec{\theta})$, onde $\vec{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$ é o vetor de parâmetros da transformação ϕ.
- Encontrar ϕ^* é encontrar $\vec{\theta}^*$ ótimo.
- A busca exaustiva é inviável e a busca por gradiente descendente (ascendente) de $F(\vec{\theta})$ é sensível a mínimos (máximos) locais.
- Uma abordagem popular é o algoritmo RANSAC, que avalia famílias aleatórias de $\vec{\theta}$ em várias iterações.
- Vamos estudar uma nova abordagem de busca multi-escala no espaço de parâmetros de ϕ (MSPS - *Multi-Scale Parameter Search*).

Método de otimização MSPS

- Para um dado valor inicial $\vec{\theta}_0$, o método MSPS propõe a avaliação da função critério $F(\vec{\theta}_0 + \vec{\Delta})$ em torno de $\vec{\theta}_0$ para deslocamentos $\vec{\Delta}$ ao longo do eixo de cada parâmetro θ_i, $i = 1, 2, \ldots, n$, em várias escalas $j = 1, 2, \ldots, m$, e nas direções que combinam os melhores deslocamentos de cada eixo e escala.

Método de otimização MSPS

- Para um dado valor inicial $\vec{\theta}_0$, o método MSPS propõe a avaliação da função critério $F(\vec{\theta}_0 + \vec{\Delta})$ em torno de $\vec{\theta}_0$ para deslocamentos $\vec{\Delta}$ ao longo do eixo de cada parâmetro θ_i, $i = 1, 2, \ldots, n$, em várias escalas $j = 1, 2, \ldots, m$, e nas direções que combinam os melhores deslocamentos de cada eixo e escala.
- Encontrado o melhor deslocamento $\vec{\Delta}^*$, a nova posição de busca é atualizada para $\vec{\theta}_{t+1} \leftarrow \vec{\theta}_t + \vec{\Delta}^*$.

Método de otimização MSPS

- Para um dado valor inicial $\vec{\theta}_0$, o método MSPS propõe a avaliação da função critério $F(\vec{\theta}_0 + \vec{\Delta})$ em torno de $\vec{\theta}_0$ para deslocamentos $\vec{\Delta}$ ao longo do eixo de cada parâmetro θ_i, $i = 1, 2, \ldots, n$, em várias escalas $j = 1, 2, \ldots, m$, e nas direções que combinam os melhores deslocamentos de cada eixo e escala.
- Encontrado o melhor deslocamento $\vec{\Delta}^*$, a nova posição de busca é atualizada para $\vec{\theta}_{t+1} \leftarrow \vec{\theta}_t + \vec{\Delta}^*$.
- O processo se repete até não ser possível melhorar $F(\vec{\theta}_t)$.

Mais formalmente, seja $\Delta_{i,j} > 0$ um deslocamento ao longo do eixo θ_i para a escala j:

$$\Delta_{i,j} = \frac{j^d}{2m^d}(u_i - l_i)$$

onde $u_i - l_i > 0$ é a diferença entre os maiores e menores valores de cada parâmetro θ_i, $i = 1, 2, \ldots, n$, e d é uma constante que descreve o grau de aumento das escalas.

O método leva em conta os seguintes deslocamentos:

- O melhor deslocamento ao longo de cada eixo θ_i e escala j

$$\vec{\Delta}^*_{i,j} = (0, \ldots, \Delta^*_{i,j}, \ldots, 0),$$

onde $\Delta^*_{i,j} \in \{\Delta_{i,j}, 0, -\Delta_{i,j}\}$ tal que

$$F(\vec{\theta}_t + \vec{\Delta}^*_{i,j}) = \min \left\{ \begin{array}{l} F(\vec{\theta}_t + \vec{\Delta}_{i,j}), \\ F(\vec{\theta}_t), \\ F(\vec{\theta}_t - \vec{\Delta}_{i,j}) \end{array} \right\}.$$

Similarmente para a maximização.

Método de otimização MSPS

O método leva em conta os seguintes deslocamentos:

- O melhor deslocamento ao longo de cada eixo θ_i e escala j

$$\vec{\Delta}^*_{i,j} = (0, \ldots, \Delta^*_{i,j}, \ldots, 0),$$

onde $\Delta^*_{i,j} \in \{\Delta_{i,j}, 0, -\Delta_{i,j}\}$ tal que

$$F(\vec{\theta}_t + \vec{\Delta}^*_{i,j}) = \min \left\{ \begin{array}{l} F(\vec{\theta}_t + \vec{\Delta}_{i,j}), \\ F(\vec{\theta}_t), \\ F(\vec{\theta}_t - \vec{\Delta}_{i,j}) \end{array} \right\}.$$

Similarmente para a maximização.

- Os deslocamentos resultantes $\vec{\Delta} s_j = \sum_{i=1}^{n} \vec{\Delta}^*_{i,j}$, dos melhores deslocamentos para cada escala $j = 1, 2, \ldots, m$.

- Os deslocamentos resultantes $\vec{\Delta}p = \sum_{i=1}^{n} \vec{\Delta}p_i$, onde

$$\vec{\Delta}p_i = (0, \ldots, \Delta p_i, \ldots, 0)$$

é o melhor deslocamento ao longo das escalas para o parâmetro θ_i

$$F(\vec{\theta}_t + \vec{\Delta}p_i) \quad = \quad \min_{j=1,2,\ldots,m} \left\{ F(\vec{\theta}_t + \vec{\Delta}^*_{i,j}) \right\}.$$

Método de otimização MSPS

A escolha de $\vec{\Delta}^*$ é finalmente dada por

$$F(\vec{\theta}_t + \vec{\Delta}^*) \quad = \quad \min \left\{ \begin{array}{ll} F(\vec{\theta}_t + \vec{\Delta}p). & \\ F(\vec{\theta}_t + \vec{\Delta}p_i) & \text{para } i = 1, 2, \ldots, n. \\ F(\vec{\theta}_t + \vec{\Delta}s_j) & \text{para } j = 1, 2, \ldots, m. \end{array} \right\}.$$

Quando não é possível melhorar $\vec{\theta}_t$, modificamos

$$\Delta_{i,j} \leftarrow \frac{\Delta_{i,j}}{2^\alpha}$$

para fins de refinar a busca, onde $\alpha > 0$. $*$ Ver algoritmo em `http://www.ic.unicamp.br/~reltech/2011/11-15.pdf`.