#1

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = W(p) \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3}$$

where $W(p)$ is a 3×3 matrix such that

$$W(p) = \begin{bmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$x = (u, v)^T$$

$$P = \left(p_1, \dots, p_6\right)^T$$

$$\frac{\partial w}{\partial_P} = \begin{pmatrix} \frac{\partial w_u}{\partial p} \\ \frac{\partial w_v}{\partial p} \end{pmatrix} = \begin{pmatrix} \frac{\partial w_u}{\partial p_1} & \cdots & \frac{\partial w_u}{\partial p_6} \\ \frac{\partial w_v}{\partial P_1} & \cdots & \frac{\partial w_v}{\partial p_6} \end{pmatrix}$$

$$F\left(p_1, p_2, \dots, p_6\right) = \left(f_U(p_1, \dots, p_6), f_V(p_1, \dots, p_6)\right)$$

$$J_F(p_1, \cdots p_6) = \frac{\partial\left(f_U, f_V\right)}{\partial\left(p_1, \dots, p_2\right)} = \begin{pmatrix} \frac{\partial f_U}{\partial p_1} & \cdots & \frac{\partial f}{\partial p_6} \\ \frac{\partial f_V}{\partial p_1} & \cdots & \frac{\partial f_V}{\partial P_6} \end{pmatrix}$$

$$F(P + \Delta p) \approx f(P) + J_F(p)\Delta p$$

F = = W

$$w(x, p) = \begin{bmatrix} (1+p_1)\,u + p_3 v + p_5 \\ p_2 u + (1+p_4) v + p_6 \\ 1 \end{bmatrix}$$

$$\frac{\partial w}{\partial p} = \begin{pmatrix} u & 0 & v & 0 & 1 & 0 \\ 0 & u & 0 & v & 0 & 1 \end{pmatrix}$$

#2

$$G = \nabla T(W(x;\Delta p)) = \nabla T \ (\partial W \ / \partial p \ )$$

$$\nabla T = \left(\frac{\partial T}{\partial u}, \frac{\partial T}{\partial v}\right)$$

$$G = \left(\frac{\partial T}{\partial u}\frac{\partial w_u}{\partial p}, \frac{\partial T}{\partial v}\frac{\partial w_v}{\partial p}\right) = \left(\frac{\partial T}{\partial u}, \frac{\partial T}{\partial v}\right)\begin{pmatrix} u & 0 & v & 0 & 1 & 0 \\ 0 & u & 0 & v & 0 & 1 \end{pmatrix}$$

3

$$\Delta p = H^{-1} G^T [I - T]$$

The computational cost for computing W(x;p) depends on p so there is O(p) per pixels so that for G we are reapeting the process for n times so then finally it takes O(pn) and for computing H it takes time O(n p^2) as we are repeating whole the process p times.

4

The Inverse Compositional Algorithm

Pre-compute:

(3) Evaluate the gradient ∇T of the template $T(\mathbf{x})$
(4) Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x}; \mathbf{0})$
(5) Compute the steepest descent images $\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
(6) Compute the Hessian matrix using Equation (38)

Iterate:

(1) Warp I with $\mathbf{W}(\mathbf{x};\mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
(2) Compute the error image $I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})$
(7) Compute $\sum_{\mathbf{x}}[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^{\mathrm{T}}[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]$
(8) Compute $\Delta\mathbf{p}$ using Equation (37)
(9) Update the warp $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$

until $\|\Delta\mathbf{p}\| \leq \epsilon$

1. O(pn)
2. O(n)

Step 3,4 ,5 ,6 per composition and others are done per iteration

3. O(n)
4. O(pn)
5. O(pn)
6. O(p^2 n)

In total O(p^2n) for precomposition

7. O(Pn)
8. O(p^3)
9. O(p^2)

In total per iteration (step 1,2, 7,8,9) has O(pn+p^3)

O(np+p3) in total

#5
Image derivatives in practical way can be computed by using small convolution filters of size 2 x 2 or 3 x 3, such as the Laplacian, Sobel, Roberts and Prewitt operators.

#6
we should choose a window with corner for key points. If the window is 5*5 then we have 25 equations for every pixel

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \;\; v]$$

And lets assume all neighbors have same (u,v)
Then:

$$\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}$$

$$\underset{25\times2}{A} \; \underset{2\times1}{d} = \underset{25\times1}{b}$$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

- λ_1 and λ_2 are the eigen values of of A^TA
- when λ_1 and λ_2 are small, the region is flat.
- when $\lambda_1 >> \lambda_2$ or vice versa, the region is edge.

- when λ_1 and λ_2 are large and $\lambda_1 \sim \lambda_2$, the region is a corner.