CCD code transcript

Shulei ZHU

31 January 2011

Contents

1 function1: main

1.1 variables

1.1.1 constant parameters

- **int** resolution: the amount of points on the contour, equidistant distributed

- **int** t: degree of B-Spline Curve

- **int** h: search radius in the normal direction

- **int** delta_h: distance step in the normal direction

- **double** sigma_hat:
$$\hat{\sigma} = \frac{h}{2.5}$$

- **double** $\text{gamma}_1 = 0.5$, $\text{gamma}_2 = 4$, $\text{gamma}_3 = 4$

$$\gamma_1 = 0.5, \gamma_2 = 4, \gamma_3 = 4$$

- **double** $\text{sigma} = \text{sigma}_{\text{hat}}/\text{gamma}_3$

$$\sigma = \frac{\hat{\sigma}}{\gamma_3}$$

- **double** kappa = 0.5

$$\kappa = 0.5$$

$$\Sigma_{v,s} = M_s^2(d^{=})/\omega_(d_v^{=}) - m_{v,s} * (m_{v,s})^t + \kappa * I$$

here, I is a identy matrix, to avoid singular

- **double** c:parameter used to update model covariance matrix

$$\Sigma_\Phi^{new} = c * \Sigma_\Phi + (1 - c)(H_\Phi E)^{-1}$$

- **int** $\text{normal\_points}_{\text{number}}$

$$\text{normal\_points\_number} = floor(\frac{h}{\delta_h})$$

- **double** tol = 0

- **double** iter = 0

1.1.2 matrix and points

- **CvPoint2D64f** pts_tmp: used to store all resulting control points

- **cv::Mat** Phi: model parameters, it is a 6x1 matrix

$$\Phi = \begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \end{bmatrix}$$

- **cv::Mat** Sigma_phi: covariance matrix, it is a 6x6 matrix

$$\Sigma_\Phi = \begin{bmatrix} \sigma_{00} & \sigma_{01} & \sigma_{02} & \sigma_{03} & \sigma_{04} & \sigma_{05} \\ \sigma_{10} & \sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} & \sigma_{15} \\ \sigma_{20} & \sigma_{21} & \sigma_{22} & \sigma_{23} & \sigma_{24} & \sigma_{25} \\ \sigma_{30} & \sigma_{31} & \sigma_{32} & \sigma_{33} & \sigma_{34} & \sigma_{35} \\ \sigma_{40} & \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_{44} & \sigma_{45} \\ \sigma_{50} & \sigma_{51} & \sigma_{52} & \sigma_{53} & \sigma_{54} & \sigma_{55} \end{bmatrix}$$

- **cv::Mat** mean_vic: mean value of points located in the vicinity of the contour, resolutionx6

- **cv::Mat** cov_vic: covariance matrix of points located in the vicinity of the contour, resolutionx18

- **cv::Mat** nabla_E:

$$\nabla_E = \nabla_{E_1} + \nabla_{E_2} = 2 * (\Sigma_\Phi^*)^{-1} * \Phi - \sum_{k,l} J_{a_1} \hat{\Sigma}_{k,l}^{-1} (I_{k,l} - \hat{I}_{k,l})$$

- **cv::Mat** hessian_E:

$$\text{hessian}E = \text{hessian}E_1 + \text{hessian}E_2 = (\Sigma_\Phi)^{-1} + \sum_{k,l} J_{a_1} \hat{\Sigma}_{k,l}^{-1} J_{a_1}$$

- **cv::Mat** tmp_cov:

$$\text{tmp\_cov} = \hat{\Sigma}_{k,l}$$

- **cv::Mat** tmp_cov_inv:

$$\text{tmp\_cov\_inv} = \hat{\Sigma}_{k,l}^{-1}$$

- **cv::Mat** tmp_jacobian:

$$\text{tmp\_jacobian} = \begin{bmatrix} \frac{\partial p_x}{\partial x_0} & \frac{\partial p_x}{\partial x_1} & \frac{\partial p_x}{\partial x_2} & \frac{\partial p_x}{\partial x_3} & \frac{\partial p_x}{\partial x_4} & \frac{\partial p_x}{\partial x_5} \\ \frac{\partial p_y}{\partial x_0} & \frac{\partial p_y}{\partial x_1} & \frac{\partial p_y}{\partial x_2} & \frac{\partial p_y}{\partial x_3} & \frac{\partial p_y}{\partial x_4} & \frac{\partial p_y}{\partial x_5} \end{bmatrix}$$

- **cv::Mat** tmp_pixel_diff:

$$\text{tmp\_pixel\_diff} = I_{k,l} - \hat{I}_{k,l}$$

- **cv::Mat** nv: normal vector(both directions)

$$\text{nv}[0] = \frac{-bs'.y}{\sqrt{(bs'.x)^2 + (bs'.y)^2}} \tag{1}$$

$$\text{nv}[1] = \frac{-bs'.x}{\sqrt{(bs'.x)^2 + (bs'.y)^2}} \tag{2}$$

- **CvPoint** tmp1, tmp2:
temporary points used to store those points in the normal direction as well as negative normal direction

$$tmp1.x = round(bs.x + \delta_h * nv[1]) \tag{3}$$

$$tmp1.y = round(bs.y + \delta_h * nv[2]) \tag{4}$$

- **CvPoint2D64f** tmp_dis1, tmp_dis2:
store the distance from a point in normal(negative norml) direction to the point on the curve

$$tmp\_dis1.x = (tmp1.x - bs.x) * nv[1] + (tmp1.y - bs.y) * nv[2] \tag{5}$$

$$tmp\_dis1.y = (tmp1.x - bs.x) * nv[2] + (tmp1.y - bs.y) * nv[1] \tag{6}$$

$$tmp\_dis2.x = (tmp2.x - bs.x) * nv[1] + (tmp2.y - bs.y) * nv[2] \tag{7}$$

$$tmp\_dis2.y = (tmp2.x - bs.x) * nv[2] + (tmp2.y - bs.y) * nv[1] \tag{8}$$

- **cv::Mat** vic
  - col_1, col_2: coordinates of x and y
  - col_3, col_4: the distance between a normal points and the point on the curve d_v(x), d_v(y)
  - col_5: the probability

$$P_{v,1}(x, m_\phi, \hat{\sigma}) = 0.5 * erf(\frac{d'_v(x)}{(\sqrt(2) * \hat{\sigma})})$$

  - col_6: the probability of pixel p to belong to the desired side s.

$$W_s(p) = max(0, [a - \gamma_1)/(1 - \gamma_1)]^4)$$

- col_7, col_8 : evaluates the proximity of pixel p to the curve

$$W_p(d_p, \sigma_p) = c * max[0, exp(-d_p^2/2 * \sigma_p'^2) - exp(-\gamma_2))]$$

$$\sigma_p' = \gamma_3 * \sigma_p + \gamma_4$$

$$W_{sp} = W_s * W_p$$

- col_9: access the distance $|d_v{}^{=} - d_p{}^{=}|$ between pixel p and pixel v along the curve

$$W' = 0.5 * exp(-|d_v{}^{=} - d_p{}^{=}|/\alpha)/\alpha$$

- col_{10}: the derivative of col_5:

$$\frac{1}{\sqrt{2 * \pi * \sigma}} * exp\left\{\frac{-d_{k,l}^2}{(2 * \hat{\sigma} * \hat{\sigma})}\right\}$$

so last $\omega_p s = W_s * W'$

- **cv::Mat** normalized_param
to save the normalized parameters of $vic[3]$, dimension: resolution x 2, the first column save the normalized coefficient outside the curve, the second column store the one inside the curve

- **cv::Mat** bs_old:
dimension is 4, the structure is similar as bs

1.2 while loop

1.2.1 generate new control points

$$Q = W\Phi + Q_0$$

here, we use pts_tmp to represent Q, use pts to represent Q_0

1.2.2 set nv, mean_vic, cov_{vic}, nabla_E hessian_E as zero matrices

1.2.3 create a new B-Spline contour use new control points(degree is 3, it is a Uniform quadratic B-spline)

$$\mathbf{bs}_(t) = \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{i-1} \\ \mathbf{Q}_i \\ \mathbf{Q}_{i+1} \end{bmatrix}$$

for $t \in [0, 1], i = 1, 2 \ldots m - 2$

1.2.4 initialize the covariance matrix

$$\Sigma_\Phi = \frac{\dim_\Phi}{h^2}\mathcal{H} = \frac{\dim_\Phi}{h^2} W^T \mathcal{U} W$$

1.2.5 computing the tolerance

$$\text{tol} = \sum_{i=0}^{\text{resolution}} (\vec{\text{bs\_new}} - \vec{\text{bs\_old}}) \times \vec{n}$$

1.2.6 compute the normal vector and save old axis components and normal vector into bs_old

$$\vec{t} = \frac{\partial \vec{\text{bs}}}{\partial \vec{l}}$$

where $\vec{l}$ along the curve, therefore

$$\vec{nv} = [-t_y, t_x]$$

1.2.7 calculate all values of elements in matrix vic

$$
\begin{aligned}
vic[1] &= y && (9)\\
vic[2] &= x && (10)\\
vic[4] &= distance.x && (11)\\
vic[5] &= distance.y && (12)\\
vic[6] &= \frac{1}{2}erf(\frac{distance.x}{\sqrt{2}*\sigma}) && (13)\\
vic[7] &= \left(\frac{vic[4]-\gamma_1}{1-\gamma_1}\right)^4 && (14)\\
vic[8] &= -64\left(0.75-vic[4]\right)^4+0.25 && (15)\\
vic[3] &= \max\left(\exp\{\frac{-(distance.x)^2}{2*\hat{\sigma}^2}\}-\exp(-\gamma_2), 0.0\right) && (16)\\
vic[9] &= \frac{\exp\left\{\frac{-|distance.y|}{\alpha}\right\}}{2\alpha} && (17)\\
vic[10] &= \frac{\exp\left\{\frac{-(distance.x)^2}{2\sigma^2}\right\}}{\sqrt{2\pi}\sigma} && (18)
\end{aligned}
$$

in addition, we have to computing the normalization parameter fro $vic[3]$

1.2.8 computing weights for both sides

$$
\begin{aligned}
w_1 &= \sum \mathrm{w}_{\mathrm{p},1} = \frac{1}{c_1}\sum vic[7]*vic[3]*vic[9] && (19)\\
w_2 &= \sum \mathrm{w}_{\mathrm{p},2} = \frac{1}{c_2}\sum vic[8]*vic[3]*vic[9] && (20)
\end{aligned}
$$

1.2.9 calculate the average values of all pixels located in each side of the vicinity respectively

$$
m_{k,s} = \hat{I}_{k,s} = \frac{M_s(d_k)}{\omega_s(d_k)}
$$

1.2.10 calculate the local covariance matrix

$$\Sigma_{k,s} = \frac{M_s^2(d_k)}{\omega_s(d_k)} - m_{k,s} * m_{k,s}^2 + \kappa I$$

where

$$\begin{aligned}
\omega_s(d_k) &= \sum_{p \in \mathcal{V}} \omega_{p,s}(d_k) &(21)\\
M_s(d_k) &= \sum_{p \in \mathcal{V}} \omega_{p,s}(d_k) I_p &(22)\\
M_s^2(d_k) &= \sum_{p \in \mathcal{V}} \omega_{p,s}(d_k) I_p * I_p^T &(23)
\end{aligned}$$

1.2.11 calculate ∇_E and H_E

$$\begin{aligned}
E &= E_1 + E_2 &(24)\\
E_1 &= \frac{1}{2}(\Phi - m_\Phi^*)^{-1}(\Sigma_\Phi^*)^{-1}(\Phi - m_\Phi^*) &(25)\\
E_2 &= \frac{1}{2}\sum_k \sum_l (I_{k,l} - \hat{I}_{k,l}(\Phi))^T \hat{\Sigma}_{k,l}^{-1}(I_{k,l} - \hat{I}_{k,l}(\Phi)) &(26)\\
\nabla E_1 &= 2(\Sigma_\Phi^*)^{-1}\Phi &(27)\\
\nabla E_2 &= -\sum_{k,l} J_{a_1}^T \hat{\Sigma}_{k,l}^{-1}(I_{k,l} - \hat{I}_{k,l}) &(28)\\
\mathbf{H}E_1 &= (\Sigma_\Phi^*)^{-1} &(29)\\
\mathbf{H}E_2 &= \sum_{k,l} J_{a_1}^T \hat{\Sigma}_{k,l}^{-1} J_{a_1} &(30)\\
\nabla E &= \nabla E_1 + \nabla E_2 &(31)\\
\mathbf{H}E &= \mathbf{H}E_1 + \mathbf{H}E_2 &(32)
\end{aligned}$$