OpenTracker Implement Notes

rockking.jy@gmail.com

July 14, 2018

Contents

1 Math Foundation

1.1 Basic Notations

Complex-value functions $g, h : \mathbb{R} \to \mathbb{C}$ are periodic with period $T > 0$.

Space $L^2(T)$: Hilbert space equipped with an inner product $\langle \cdot, \cdot \rangle$ of periodic functions with period $T > 0$.

For $g, h \in L^2(T)$,

$$\langle g, h \rangle = \frac{1}{T} \int_0^T g(t)\overline{h(t)}dt \tag{1.1}$$

where bar means complex conjugation.

Complex exponential functions:

$$e_k(t) = e^{i2\pi kt/T} \tag{1.2}$$

$\{e_k(t)\}_{-\infty}^{\infty}$ forms an orthonormal basis for $L^2(T)$.

1.2 Matrix Derivatives [1]

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^{\boldsymbol{H}}\boldsymbol{a}) = \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{a}^{\boldsymbol{H}}\boldsymbol{x}) = \boldsymbol{a} \tag{1.3}$$

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{A}\boldsymbol{B}) = \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{x}}\boldsymbol{B} + \boldsymbol{A}\frac{\partial \boldsymbol{B}}{\partial \boldsymbol{x}} \tag{1.4}$$

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{A}^{-\mathbf{1}}) = \boldsymbol{A}^{-1}\frac{\partial \boldsymbol{A}}{\partial \boldsymbol{x}}\boldsymbol{A}^{-1} \tag{1.5}$$

$$\frac{\partial}{\partial \boldsymbol{x}}(\|\boldsymbol{A}\boldsymbol{x}\|^2) = \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^{\boldsymbol{H}}\boldsymbol{A}^{\boldsymbol{H}}\boldsymbol{A}\boldsymbol{x}) = \boldsymbol{A}^{\boldsymbol{H}}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}^{\boldsymbol{H}}\boldsymbol{A}\boldsymbol{x} = 2\boldsymbol{A}^{\boldsymbol{H}}\boldsymbol{A}\boldsymbol{x} \tag{1.6}$$

1.3 Kronecker product

If $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{B}$ is a $p \times q$ matrix, then the Kronecker product $A \otimes B$ is the $mp \times nq$ block matrix:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}_{mp \times nq} \tag{1.7}$$

1.4 Forbenius norm

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{\mathrm{trace}(A^{\dagger}A)} = \sqrt{\sum_{i=1}^{\min\{m,n\}}\sigma_i^2(A)} \tag{1.8}$$

1.5 Fourier transformation

Fourier transformation:

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\ e^{-2\pi i x\xi}\, dx \tag{1.9}$$

Inverse Fourier transformation:

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi)\ e^{2\pi i x\xi}\, d\xi \tag{1.10}$$

For discrete points: $\{\mathbf{x_n}\} := x_0, x_1, \ldots, x_{N-1}$ and $\{\mathbf{X_k}\} := X_0, X_1, \ldots, X_{N-1}$, we have:

Discrete Fourier transformation:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} \tag{1.11}$$

Inverse discrete Fourier transformation

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i2\pi kn/N} \tag{1.12}$$

Fourier transformation of $g \in L^2(T)$:

$$\hat{g}[k] = \langle g, e_k \rangle = \frac{1}{T} \int_0^T g(t) e^{-i2\pi kt/T} dt \tag{1.13}$$

Any $g \in L^2(T)$ can be expressed as: $g(t) = \sum_{-\infty}^{\infty} \hat{g}[k] e_k$.

Shift property in $L^2(T)$:

For any real number L, if $h(t) = g(t - L)$, then

$$\hat{h}[k] = e^{-2\pi ikL/T} \hat{g}[k] \tag{1.14}$$

Parseval's formula:

$$\|g\|_{L^2}^2 = \|\hat{g}\|_{l^2}^2 \tag{1.15}$$

where $\|g\|^2 = \langle g, g \rangle$ and $\|\hat{g}\|_{l^2}^2 = \sum_{-\infty}^{\infty} \left|\hat{g}[k]\right|^2$.

Poisson summation formula:

$$\sum_{n=-\infty}^{\infty} f(n) = \sum_{k=-\infty}^{\infty} \hat{f}\,(k) \tag{1.16}$$

$$\sum_{n=-\infty}^{\infty} s(t + nT) = \sum_{k=-\infty}^{\infty} \frac{1}{T} \cdot \hat{s}\left(\frac{k}{T}\right) e^{i2\pi \frac{k}{T} t} \tag{1.17}$$

Symmetry of discrete Fourier transformation

From equation (1.11), we can have:

$$X_{N-k} = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi(N-k)n/N} = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi n} e^{+i2\pi kn/N} = \sum_{n=0}^{N-1} x_n \cdot e^{+i2\pi kn/N} = \overline{X}_k \tag{1.18}$$

So the Fourier coefficient is symmetry to the axis $(N + 1)/2$, with a complex conjugation, this property could be used to save the memory and computing resources.

1.6 FFT

FFT is short Fast Fourier Transformation, which fully use the symmetry property of Fourier transformation.

1.7 Correlation

Correlation definition:

$$(f \star g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f^*(\tau) g(t+\tau)\, d\tau \tag{1.19}$$

Discrete correlation:

$$(f \star g)[k] = \sum_{l=-\infty}^{\infty} f^*[l] g[k+l] \tag{1.20}$$

Correlation properties:

$$\widehat{f \star g} = \hat{f}^* \hat{g} = \hat{f} \hat{g}^* \tag{1.21}$$

where * means complex conjugate.

1.8 Convolution

Convolution definition:

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau \tag{1.22}$$

$$= \int_{-\infty}^{\infty} f(t-\tau) g(\tau)\, d\tau \tag{1.23}$$

Discrete convolution:

$$(f * g)[k] = \sum_{l=-\infty}^{\infty} f[l] g[k-l] \tag{1.24}$$

$$= \sum_{l=-\infty}^{\infty} f[k-l] g[l] \tag{1.25}$$

Circular convolution operation(with normalization) $* : L^2(T) \times L^2(T) \to L^2(T)$ here is defined by:

$$(g * h)(t) = \frac{1}{T} \int_0^T g(t-s) h(s) ds \tag{1.26}$$

Convolution properties:

$$\widehat{g * h} = \hat{g} \hat{h}, \qquad \widehat{gh} = \hat{g} * \hat{h} \tag{1.27}$$

To calculate discrete convolution, one of the inputs is converted into a **Toeplitz matrix**:

$$y = h * x = \begin{bmatrix} h_1 & 0 & \dots & 0 & 0 \\ h_2 & h_1 & \dots & \vdots & \vdots \\ h_3 & h_2 & \dots & 0 & 0 \\ \vdots & h_3 & \dots & h_1 & 0 \\ h_{m-1} & \vdots & \dots & h_2 & h_1 \\ h_m & h_{m-1} & \vdots & \vdots & h_2 \\ 0 & h_m & \dots & h_{m-2} & \vdots \\ 0 & 0 & \dots & h_{m-1} & h_{m-2} \\ \vdots & \vdots & \vdots & h_m & h_{m-1} \\ 0 & 0 & 0 & \dots & h_m \end{bmatrix}_{(m+n-1)\times n} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \tag{1.28}$$

or

$$\begin{bmatrix} h_1 & h_2 & h_3 & \dots & h_{m-1} & h_m \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_n & 0 & 0 & 0 & \dots & 0 \\ 0 & x_1 & x_2 & x_3 & \dots & x_n & 0 & 0 & \dots & 0 \\ 0 & 0 & x_1 & x_2 & x_3 & \dots & x_n & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \dots & 0 \\ 0 & \dots & 0 & 0 & x_1 & \dots & x_{n-2} & x_{n-1} & x_n & \vdots \\ 0 & \dots & 0 & 0 & 0 & x_1 & \dots & x_{n-2} & x_{n-1} & x_n \end{bmatrix}_{m\times(m+n-1)} \tag{1.29}$$

1.9 Newton Method

1.10 Gauss-Newton Method

1.11 Conjugate Gradient [2]

2 HOG feature [3] [4]

2.1 Original HOG feature [3]

Basic idea: local object appearance and shape can often be characterized rather well by the distribution of local intensity gradient or edge directions, even without precise knowledge of the corresponding gradient or edge position.

Input - detection window with size: 64×128.

1. Divide the detection window into 16×16 blocks of 50% overlap.
   - $7 \times 15 = 105$ blocks in total $((64 \div (16 \div 2) - 1) \times (128 \div (16 \div 2) - 1)$.
2. Each block consists of 2×2 cells with size 8 pixels $\times$ 8 pixels.
3. Compute gradients for each pixel.
   - Using $[-1, 0, 1]$ gradient filter.
4. Quantize the gradient orientation into 9 orientations bins in $0° - 180°$ (with $20°$ interval).

- The vote is the gradient magnitude.
  For color images, calculate separate gradients for each color channel, and take the one with the largest norm at the pixel's gradient vector.
- Interpolate votes bi-linearly between neighboring bin center.
  For example, suppose a pixel have orientation 80°, next to it has 70° and 90° bins, then: $85-70=15$, $90-85=5$, so, for bin 70° it contributes: $15/(15+5)=3/4\times$ gradient magnitude, and for bin 90°: $5/(15+5)=1/4\times$ gradient magnitude.
- The vote can also be weighted by Gaussian to down-weight near the edge.

5. For each cell accumulating the 1-D histogram of gradient directions or edge orientations over the pixels of the cell.
6. Normalization for blocks, for better invariance to illumination, shadowing etc..
   - L2-Hys (Lowe-style clipped L2 norm) block normalization. That is, L2-norm followed by clipping (limiting the maximum values of v to 0.2) and renormalizing.
7. Concatenate histograms.
   - Feature dimension: $36(=4\times 9)$ for each block (in this example, 105 blocks in total, so 36×105 features in total for this input detection window).

2.2 Improved HOG feature [4]

Basic idea: using 13-dimensional feature instead of previous 36-dimensional with no significant effect, and add contrast sensitive and contrast insensitive features to a 31-dimensional feature vector to improve the performance.

Input - image size: $w\times h$.

1. Pixel-Level Feature Maps.
   - For each pixel (x, y), let $\theta(x,y)$ be the orientation and $r(x,y)$ be the magnitude of the intensity gradient using $[-1,0,1]$ as gradient filter.
   - The gradient calculated at the pixel is discretized into one of p values by contrast sensitive (B_1) or contrast insensitive (B_2):
   $$B_1(x,y)=round(\frac{p\theta(x,y)}{2\pi}) \mod p \tag{2.1}$$
   $$B_2(x,y)=round(\frac{p\theta(x,y)}{\pi}) \mod p \tag{2.2}$$
   - The feature vector $F(x,y)_b$ at pixel (x, y) is:
   $$F(x,y)_b=\begin{cases} r(x,y) & \text{if } b=B(x,y) \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$
   where $b\in\{0,\cdots,p-1\}$.
   - Now we have a pixel-level feature map for the image.
2. Spatial Aggregation.
   - Let $k>0$ be the cell size.
   - Aggregate pixel-level feature vectors to obtain a cell-based feature map C(i,j).

- In C(i,j), where $0 \leq i \leq \lfloor (w-1)/k \rfloor$, $0 \leq j \leq \lfloor (h-1)/k \rfloor$.
- Pixel (x, y) maps to $(\lfloor x/k \rfloor, \lfloor y/k \rfloor)$.
- The feature vector at a cell is the sum (or average) of the pixel-level features in that cell.
- Each pixel contributes to the feature vectors in the four cells around it using bilinear interpolation.
- This aggregation provides some invariance to small deformations and reduces the size of a feature map.

3. Normalization and Truncation.

- Normalization provides the invariance to gain, while gradients provides invariant to bias.
- Let $T_\alpha(v)$ denote the truncation of a vector v by α.
- The feature map is obtained by concatenating the result of normalizing the map C with respect to each normalization factor followed by truncation:

$$H(i,j) = \begin{bmatrix} T_\alpha(C(i,j)/N_{-1,-1}(i,j)) \\ T_\alpha(C(i,j)/N_{+1,-1}(i,j)) \\ T_\alpha(C(i,j)/N_{+1,+1}(i,j)) \\ T_\alpha(C(i,j)/N_{-1,+1}(i,j)) \end{bmatrix} \tag{2.4}$$

- We use four normalization factors for C(i,j): $N_{\delta,\gamma}(i,j)$ with $\delta, \gamma \in \{-1, 1\}$,

$$N_{\delta,\gamma}(i,j) = (\left\|C(i,j)\right\|^2 + \left\|C(i+\delta,j)\right\|^2 + \left\|C(i,j+\gamma)\right\|^2 + \left\|C(i+\delta,j+\gamma)\right\|^2)^{1/2} \tag{2.5}$$

  Each factor measures the "gradient energy" in a square block of four cells containing (i,j).
- Commonly, HOG features using $p = 9$, discretized with B_2, cell size $k = 8$ and truncation $\alpha = 0.2$, so we have 36-dimensional feature vector (9 orientations $\times$4 normalizaions).

4. PCA and Analytic Dimensionality Reduction.

- 9 contrast insensitive orientations, 18 contrast sensitive orientations, 4 normalization factors, so we have $4 \times (9 + 18) = 108$ dimensional feature vectors.
- Use an analytic projection of 108 dimensional features to:
  (a) 27 (=9+18) sums over 4 different normalizations.
  (b) 4 dimensional features sums over 9 contrast insensitive orientations.
- In total, the final feature map has 31-dimensional vectors (27+4).

3 CN feature [5]

Color names (CN) are linguistic color labels assigned by humans to represent colors in the world. In English, it contains eleven basic color terms: black, blue, brown, green, orange, pink, purple, red, white and yellow.

Color naming is an operation that associates RGB observations with linguistic color labels.

In [5], it provides the mapping automatically learned from images retrieved with Google-image search. It maps the RGB values to a probabilistic 11 dimensional color representation which sums up to 1.

4 Deep feature [6]

Equation to calculate output size from input for convolution layer:

$$|Ouput| = \lfloor \frac{|Input| - size + pad1 + pad2}{stride} \rfloor + 1 \tag{4.1}$$

Layer	Name	size	stride	rFieldStride	pad	input	output
0	Input	[224,224,3,10]		1		224 x 224	224 x 224
1	conv1	[7,7,3,96]	[2, 2]	1	[0,0,0,0]	224 x 224	109 x 109
2	relu1			2		109 x 109	109 x 109
3	**norm1**			2		109 x 109	109 x 109
4	pool1	[3, 3]	[2, 2]	2	[0,1,0,1]	109 x 109	54 x 54
5	conv2	[5,5,96,256]	[2, 2]	4	[1,1,1,1]	54 x 54	26 x 26
6	relu2			8		26 x 26	26 x 26
7	norm2			8		26 x 26	26 x 26
8	pool2	[3, 3]	[2, 2]	8	[0,1,0,1]	26 x 26	13 x 13
9	conv3	[3,3,256,512]	[1, 1]	16	[1,1,1,1]	13 x 13	13 x 13
10	relu3			16		13 x 13	13 x 13
11	conv4	[3,3,512,512]	[1, 1]	16	[1,1,1,1]	13 x 13	13 x 13
12	relu4			16		13 x 13	13 x 13
13	conv5	[3,3,512,512]	[1, 1]	16	[1,1,1,1]	13 x 13	13 x 13
14	**relu5**			16		13 x 13	13 x 13

5 MOSSE [7]

x_i: training images ; f: filter; g_i: actual target; y_i: desired target, here we choose 2D Gaussian centered on the target in training image.

The correlation filter performs on the training image, gave the target that we want to tracking:

$$x_i \star f = g_i \tag{5.1}$$

The maximum point of g_i is the centre of the target.

With the property of correlation (1.21), we have:

$$\widehat{x_i \star f} = \hat{x}_i \odot \hat{f}^* \tag{5.2}$$

$\odot$ means element-wise multiplication. And this could speed up the localization procedure.

5.1 Localization

1. Do Fourier transform of the input image: $\hat{x}_i = \mathcal{F}(x_i)$, and of the filter: $\hat{f} = \mathcal{F}(f)$.
2. Do the element-wise multiplication: $\hat{g}_i = \hat{x}_i \odot \hat{f}^*$.
3. Transform back to the spatial domain using the inverse FFT to $g_i = \mathcal{F}^{-1}(\hat{g}_i)$.
4. The maximum point is the location of the target.

5.2 Training

To train a filter, MOSSE minimizes the error between actual output and the desired output:

$$\min_{\hat{f}} \sum_{i=1}^{m} |\hat{x}_i \odot \hat{f}^* - \hat{y}_i|^2 \tag{5.3}$$

Because correlation in the Fourier domain is an element- wise multiplication, each element of the filter can be optimized independently.

$$\min_{\hat{f}_{\omega\nu}} \sum_{i=1}^{m} |\hat{x}_{i\omega\nu}\hat{f}^*_{\omega\nu} - \hat{y}_{i\omega\nu}|^2 \tag{5.4}$$

For each element of $\hat{f}$, we do the partial, and equals to zero to get the minimum point:

$$0 = \frac{\partial}{\partial f^*_{\omega\nu}} \sum_{i=1}^{m} |\hat{x}_{i\omega\nu} \odot \hat{f}^*_{\omega\nu} - \hat{y}_{i\omega\nu}|^2 \tag{5.5}$$

$$\hat{f} = \frac{\sum_{i=1}^{m} \hat{x}_i \odot \hat{y}^*_i}{\sum_{i=1}^{m} \hat{x}_i \odot \hat{x}^*_i} \tag{5.6}$$

5.3 Update method

For each new sample i:

$$\hat{f}_i = \frac{A_i}{B_i} \tag{5.7}$$

$$A_i = \eta\hat{x}_i \odot \hat{y}^*_i + (1-\eta)A_{i-1} \tag{5.8}$$

$$B_i = \eta\hat{x}_i \odot \hat{x}^*_i + (1-\eta)B_{i-1} \tag{5.9}$$

where η is the learning rate.

5.4 Failure Detection

Peak to Sidelobe Ratio (PSR): g is split into the peak which is the maximum value and the sidelobe which is the rest of the pixels excluding an 11×11 window around the peak. The PSR is then defined as:

$$PSR = \frac{g_{max} - \mu_{s1}}{\theta_{s1}} \tag{5.10}$$

where g_{max} is the peak values and μ_{s1} and θ_{s1} are the mean and standard deviation of the sidelobe.

6 CSK [8]

7 KCF / DCF [9]

8 C-COT [10]

8.1 Continuous Learning Formulation

x_j: Training Samples.

$d \in \{1, \cdots, D\}$: Feature channel of training samples (ex. if choose deep layer 1, 5 as features, then $D = 2$).

N_d: Number of spatial sample points in feature channel d (ex. for HOG feature $N_{HOG} = 31 \times 105$ subsection 2.1).

$\chi = \mathbb{R}^{N_1} \times \cdots \times \mathbb{R}^{N_D}$.

$\{x^d_j[n]\}, n \in \{0, \cdots, N_d - 1\}$: Feature points in channel d of training sample x_j.

$[0,T) \subset \mathbb{R}$: Spatial support of the feature map, where T is the size of the support region.

J_d: Interpolation operator of training sample x in feature channel d, defined as:

$$J_d\{x^d\}(t) = \sum_{n=0}^{N_d-1} x^d[n] b_d(t - \frac{T}{N_d} n) \tag{8.1}$$

$f^d \in L^2(T)$ is the continuous filter for feature channel d.

$S_f : \chi \to L^2(T)$: maps a sample $x \in \chi$ to a confidence score function defined on the interval $[0,T)$, defined as:

$$S_f\{x\} = \sum_{d=1}^{D} f^d * J_d\{x^d\},\ x \in \chi. \tag{8.2}$$

the convolution here is the circular convolution in continuous domain as defined in (1.22).

The target is localized by maximizing the confidence score in an image region.

The filter f is trained by minimizing the functional:

$$E(f) = \sum_{j=1}^{m} \alpha_j \left\| S_f\{x_j\} - y_j \right\|_{L^2}^2 + \sum_{d=1}^{D} \left\| w f^d \right\|_{L^2}^2 \tag{8.3}$$

8.2 Training the filter f in Fourier domain

To train the filter f, we minimize the function (8.3) in the Fourier domain.

By (1.14) and (1.11), we have:

$$\begin{aligned}
\widehat{J_d\{x^d\}}[k] &= \sum_{n=0}^{N_d-1} x^d[n] \widehat{b_d(t - \frac{T}{N_d} n)}[k] \\
&= \sum_{n=0}^{N_d-1} x^d[n] e^{-i\frac{2\pi}{N_d}kn} \hat{b}_d[k] \\
&= \hat{b}_d[k] \sum_{n=0}^{N_d-1} x^d[n] e^{-i\frac{2\pi}{N_d}kn} \\
&= X^d[k] \hat{b}_d[k]
\end{aligned} \tag{8.4}$$

With the property of (1.27), we have:

$$\widehat{S_f\{x\}}[k] = \sum_{d=1}^{D} \hat{f}^d[k] X^d[k] \hat{b}_d[k] \tag{8.5}$$

By using Parseval's formula (1.15):

$$E(f) = \sum_{j=1}^{m} \alpha_j \left\| \sum_{d=1}^{D} \hat{f}^d X_j^b \hat{b}_d - \hat{y}_j \right\|_{l^2}^2 + \sum_{d=1}^{D} \left\| \hat{w} * \hat{f}^d \right\|_{l^2}^2 \tag{8.6}$$

The functional $E(f)$ can be minimized with respect of $\hat{f}^d[k]$.

8.3 Training the filter $\hat{f}$ as finite matrix

In practice the filter f needs to be represented by a **finite** set. So we obtain a **finite** representation by minimizing (8.6) over the finite-dimensional subspace $V = span\{e_k\}_{-K_1}^{K_1} \times \cdots \times span\{e_k\}_{-K_D}^{K_D} \subset L^2(T)^D$, by assuming $\hat{f}^d[k] = 0$ for $|k| > K_d$. Here we set $K_d = \lfloor \frac{N_d}{2} \rfloor$.

Define:

$$\hat{\boldsymbol{f}}^d = \begin{bmatrix} \hat{f}^d[-K_d] \\ \vdots \\ \hat{f}^d[K_d] \end{bmatrix}_{(2K_d+1)\times 1} \in \mathbb{C}^{2K_d+1},\ \hat{\boldsymbol{f}} = \begin{bmatrix} \hat{\boldsymbol{f}}^1 \\ \hat{\boldsymbol{f}}^2 \\ \vdots \\ \hat{\boldsymbol{f}}^D \end{bmatrix}_{\sum_{d=1}^D (2K_d+1)\times 1},\ \hat{\boldsymbol{y}}_j = \begin{bmatrix} \hat{y}_j[-K] \\ \vdots \\ \hat{y}_j[K] \end{bmatrix}_{(2K+1)\times 1} \tag{8.7}$$

where $K := \max_d K_d$.

And define:

$$A_j^d = \begin{bmatrix} X_j^d[-K_d]\hat{b}_d[-K_d] & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & X_j^d[K_d]\hat{b}_d[K_d] \end{bmatrix}_{(2K_d+1)\times(2K_d+1)} \tag{8.8}$$

$$A_j = \begin{bmatrix} 0_{(K-K_1)\times(2K_1+1)} & \cdots & 0_{(K-K_D)\times(2K_D+1)} \\ A_j^1 & \cdots & A_j^D \\ 0_{(K-K_1)\times(2K_1+1)} & \cdots & 0_{(K-K_D)\times(2K_D+1)} \end{bmatrix}_{(2K+1)\times\sum_{d=1}^D(2K_d+1)} \tag{8.9}$$

Let L be the number of non-zero coefficients $\hat{w}[k]$, such that $\hat{w}[k] = 0$ for all $|k| > L$.

Define W_d to be the $(2K_d + 2L + 1) \times (2K_d + 1)$ Toeplitz matrix corresponding to the convolution operator $W_d\hat{\boldsymbol{f}}^d = vec\ \hat{w} * \hat{f}^d$ (1.28):

$$W_d\hat{\boldsymbol{f}}^d = \begin{bmatrix} \hat{w}[-K_d] & 0 & \dots & 0 & 0 \\ \hat{w}[-K_d+1] & \hat{w}[-K_d] & \dots & \vdots & \vdots \\ \hat{w}[-K_d+2] & \hat{w}[-K_d+1] & \dots & 0 & 0 \\ \vdots & \hat{w}[-K_d+2] & \dots & \hat{w}[-K_d] & 0 \\ \hat{w}[K_d-1] & \vdots & \dots & \hat{w}[-K_d+1] & \hat{w}[-K_d] \\ \hat{w}[K_d] & \hat{w}[K_d-1] & \vdots & \vdots & \hat{w}[-K_d+1] \\ 0 & \hat{w}[K_d] & \dots & \hat{w}[K_d-2] & \vdots \\ 0 & 0 & \dots & \hat{w}[K_d-1] & \hat{w}[K_d-2] \\ \vdots & \vdots & \vdots & \hat{w}[K_d] & \hat{w}[K_d-1] \\ 0 & 0 & 0 & \dots & \hat{w}[K_d] \end{bmatrix}_{(2K_d+2L+1)\times(2K_d+1)} \begin{bmatrix} \hat{f}^d[-K_d] \\ \vdots \\ \hat{f}^d[K_d] \end{bmatrix}_{(2K_d+1)\times 1} \tag{8.10}$$

Define $W = W_1 \oplus \cdots \oplus W_D$.

Then the matrix version of (8.6) becomes:

$$E_V(f) = \sum_{j=1}^m \alpha_j \left\| A_j\hat{\mathbf{f}} - \hat{\mathbf{y}}_j \right\|_2^2 + \left\| W\hat{\mathbf{f}} \right\|_2^2 \tag{8.11}$$

8.4 The normal equations of $\hat{\boldsymbol{f}}$

Define the matrix with m samples:

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{bmatrix}_{(2K+1)m \times \sum_{d=1}^{D}(2K_d+1)}, \hat{\boldsymbol{y}} = \begin{bmatrix} \hat{\boldsymbol{y}}_1 \\ \hat{\boldsymbol{y}}_2 \\ \vdots \\ \hat{\boldsymbol{y}}_m \end{bmatrix}_{(2K+1)m \times 1}, \Gamma = \alpha_1 I \oplus \cdots \oplus \alpha_m I \tag{8.12}$$

The equation (8.11) becomes:

$$\begin{aligned} E_V(f) &= (A\hat{\mathbf{f}} - \hat{\mathbf{y}})^H \Gamma (A\hat{\mathbf{f}} - \hat{\mathbf{y}}) + \hat{f}^H W^H W \hat{f} \\ &= (\hat{\mathbf{f}}^H A^H - \hat{\mathbf{y}}^H)\Gamma(A\hat{\mathbf{f}} - \hat{\mathbf{y}}) + \hat{f}^H W^H W \hat{f} \\ &= \hat{\mathbf{f}}^H A^H \Gamma A \hat{\mathbf{f}} - \hat{\mathbf{f}}^H A^H \Gamma \hat{\mathbf{y}} - \hat{\mathbf{y}}^H \Gamma A \hat{\mathbf{f}} + \hat{\mathbf{y}}^H \Gamma \hat{\mathbf{y}} + \hat{f}^H W^H W \hat{f} \end{aligned} \tag{8.13}$$

Do the derivative to $\hat{\boldsymbol{f}}$ for the equation(8.13) by using the properties (1.3) and (1.6):

$$A^H \Gamma (A\hat{\boldsymbol{f}} - \hat{\boldsymbol{y}}) + W^H W \hat{\boldsymbol{f}} = 0 \tag{8.14}$$

So the minimizer of (8.11) is found by solving the equations:

$$(A^H \Gamma A + W^H W)\hat{\boldsymbol{f}} = A^H \Gamma \hat{\boldsymbol{y}} \tag{8.15}$$

8.5 The desired output y_j

$g_T(t)$: T-periodic repetition of function g, defined as:

$$g_T(t) = \sum_{n=-\infty}^{\infty} g(t - nT) \tag{8.16}$$

$\hat{g}_T[k]$: Fourier transformation of g_T, defined as:

$$\begin{aligned} \hat{g}_T[k] &\overset{(1.13)}{=} \langle g_T, e_k \rangle \\ &\overset{(1.1)}{=} \frac{1}{T}\int_0^T g_T(t) e^{-i2\pi kt/T} dt \\ &\overset{(8.16)}{=} \frac{1}{T}\int_0^T \sum_{n=-\infty}^{\infty} g(t - nT) e^{-i2\pi kt/T} dt \\ &\overset{(1.17)}{=} \frac{1}{T}\int_0^T \frac{1}{T}\sum_{k'=-\infty}^{\infty} \hat{g}(\frac{k'}{T}) e^{i2\pi k't/T} e^{-i2\pi kt/T} dt \\ &= \frac{1}{T}\int_0^T \sum_{k' \neq k} \hat{g}(\frac{k'}{T}) e^{i2\pi(k'-k)t/T} d(\frac{t}{T}) + \frac{1}{T}\hat{g}(\frac{k}{T}) \\ &= Const \times e^{i2\pi(k'-k)t/T}|_0^T + \frac{1}{T}\hat{g}(\frac{k}{T}) \\ &= \frac{1}{T}\hat{g}(\frac{k}{T}) \end{aligned} \tag{8.17}$$

$y_j(t) = \sum_{n=-\infty}^{\infty} z_j(t - nT)$ is the periodic repetition of the Gaussian function $z_j(t) = e^{-\frac{1}{2\sigma^2}(t-u_j)^2}$, here $u_j \in [0, T)$ is the estimated location of the target in the corresponding sample x_j.

The Fourier transformation of $z_j(t)$ is:

$$\begin{aligned}\hat{z}_j[k] &\overset{(1.9)}{=} \int_{-\infty}^{\infty} z_j(t)e^{-i2\pi kt}dt \\ &= \int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}(t-u_j)^2}e^{-i2\pi kt}dt \\ &= \int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}\tau^2}e^{-i2\pi k(\tau+u_j)}d\tau \\ &= e^{-i2\pi ku_j}\int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}\tau^2}e^{-i2\pi k\tau}d\tau \\ &= e^{-i2\pi ku_j}e^{-2(\sigma\pi k)^2}\int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}(\tau+i2\pi\sigma^2 k)^2}d\tau \\ &= \sqrt{2\pi\sigma^2}e^{-i2\pi ku_j}e^{-2(\sigma\pi k)^2}\end{aligned} \tag{8.18}$$

So the Fourier transformation of $y_j(t)$ is:

$$\begin{aligned}\hat{y}_j[k] &\overset{(8.17)}{=} \frac{1}{T}\hat{z}_j(\frac{k}{T}) \\ &\overset{(8.18)}{=} \frac{\sqrt{2\pi\sigma^2}}{T}e^{-i2\pi\frac{k}{T}u_j-2(\sigma\pi\frac{k}{T})^2}\end{aligned} \tag{8.19}$$

$$\hat{y}_j = \begin{bmatrix}\hat{y}_j[-K] \\ \vdots \\ \hat{y}_j[K]\end{bmatrix}_{(2K+1)\times 1} \tag{8.20}$$

where $K = max_d K_d$.

8.6 Interpolation function b_d

Standard cubic spline kernel [13] (3.79):

$$b(t) = \begin{cases}(a+2)|t|^3 - (a+3)t^2 + 1 & \text{if } |t| < 1 \\ a|t|^3 - 5at^2 + 8a|t| - 4a & \text{if } 1 \le |t| < 2 \\ 0 & \text{otherwise,}\end{cases} \tag{8.21}$$

First rescaling b to the sample interval T/N_d, then shifted half an interval $T/(2N_d)$ to align the origin of the continuous coordinate system with the sampling intervals of the feature map:

$$c_d(t) = b(\frac{N_d}{T}(t - \frac{T}{2N_d})) \tag{8.22}$$

The Fourier transformation of it is:

$$\begin{aligned}\hat{c}_d(k) &= \int_{-\infty}^{\infty} c_d(t)e^{-i2\pi kt}dt \\ &= \int_{-\infty}^{\infty} b(\frac{N_d}{T}(t - \frac{T}{2N_d}))e^{-i2\pi kt}dt \\ &= \int_{-\infty}^{\infty} b(\frac{N_d}{T}\tau)e^{-i2\pi k(\tau+\frac{T}{2N_d})}d\tau \\ &= e^{-i\pi\frac{T}{N_d}k}\frac{T}{N_d}\int_{-\infty}^{\infty} b(\tau')e^{-i2\pi k\frac{T}{N_d}\tau'}d\tau' \\ &= \frac{T}{N_d}e^{-i\pi\frac{T}{N_d}k}\hat{b}(\frac{T}{N_d}k)\end{aligned} \tag{8.23}$$

$b_d(t) = \sum_{n=-\infty}^{\infty} c_d(t - nT)$ is the periodic repetition of $c_d(t)$, its Fourier transformation is:

$$\begin{aligned} \hat{b}_d[k] &\overset{(8.17)}{=} \frac{1}{T}\hat{c}_d(\frac{k}{T}) \\ &\overset{(8.23)}{=} \frac{1}{N_d}e^{-i\frac{\pi}{N_d}k}\hat{b}(\frac{k}{N_d}) \end{aligned} \tag{8.24}$$

where the Fourier transformation of $b(t)$ is:

$$\hat{b}[k] = \frac{6(1 - cos2\pi k) + 3a(1 - cos4\pi k) - (6 + 8a)\pi k \sin 2\pi k - 2a\pi k \sin 4\pi k}{4k^4\pi^4} \tag{8.25}$$

8.7 Spatially Regularization W [11]

To resolve the problem of unwanted boundary effects of discriminatively learned correlation filters, a spatially regularization is introduced:

$$w(m, n) = \mu + \eta(m/P)^2 + \eta(n/Q)^2 \tag{8.26}$$

where $P \times Q$ denotes the target size.

8.8 Tracking Frameworks

8.8.1 Localization

1. Extract the feature maps $\{x^d[n]\} : n \in \{0, \cdots, N_d - 1\}, d \in \{1, \cdots, D\}$ from the region of interest in an image.
2. Multiply it by the cosine window.
3. Do DFT on the extracted feature maps:

$$X^d[k] = \mathcal{F}(x^d[n]) \tag{8.27}$$

4. Calculate $\widehat{S_f\{x\}}[k]$ according to (8.5):

$$\widehat{S_f\{x\}}[k] = \sum_{d=1}^{D} \hat{f}^d[k]X^d[k]\hat{b}_d[k] \tag{8.28}$$

5. Calculate confidence score $s = S_f\{x\}$ by inverse DFT:

$$s(t) = \mathcal{F}^{-1}(\widehat{S_f\{x\}}[k]) \tag{8.29}$$

6. To maximizing the score $s(t) : t \in [0, T)$,
   (a) Using grid search on $s(T\frac{n}{2K+1})$ for $n = 0, \cdots, 2K$ to find a rough initial estimate $s(t)$.
   (b) Do Fourier series expansion $s(t) = \sum_{-K}^{K} \hat{s}[k]e_k(t)$, and use the result above as the initial state, using Newtons method (1.9) to do iterative optimization of it.

8.8.2 Training

Using Conjugate Gradient (1.11) to iteratively solving the equation (8.15).

8.8.3 Sampling method

For each frame j, add one training sample x_j. The weights for each sample set to decay exponentially $\alpha_j \sim (1-\gamma)^{M-j}$. If the number of samples has reached a maximum number M_{max}, the sample with the smallest weight is replaced.

8.8.4 Parameters

Check file params.h in the code https://github.com/rockkingjy/OpenTrackers.

9 ECO [12]

9.1 Factorized Convolution Operator

9.1.1 Factorized convolution

$\hat{f}_i$ and P_i with the under index i means the value of the ith iteration.

Instead of learning one separate filter of each feature channel d, we use a smaller set of basis filters $f^1, \cdots, f^C$, where $C < D$. Then the filter for any feature layer d could be constructed as: $f^d = \sum_{c=1}^{C} p_{d,c} f^c$.

Then the filter could be written as: $(Pf)_{D\times 1}$, so we have the factorized convolution operator from (8.2), with the linearity property of convolution,

$$S_{Pf}\{x\} = Pf * J\{x\} = \sum_{c=1}^{C}\sum_{d=1}^{D} p_{d,c} f^c * J_d\{x^d\} = f * P^T J\{x\} \tag{9.1}$$

where

$$f = \begin{bmatrix} f^1 \\ \vdots \\ f^C \end{bmatrix}, P = \begin{bmatrix} P_{1,1} & \cdots & P_{1,C} \\ \vdots & \cdots & \vdots \\ P_{D,1} & \cdots & P_{D,C} \end{bmatrix}_{D\times C}, J\{x\} = \begin{bmatrix} J\{x\}^1 \\ \vdots \\ J\{x\}^D \end{bmatrix} \tag{9.2}$$

Here P^T resembles a linear dimensionality reduction operator.

We now learning the filter f and P^T jointly.

Add a extra regularization using the Forbenius norm (ch. 1.4) of P, and now using just a **single** sample, the loss of (8.3) becomes:

$$E(f,P) = \left\| S_{Pf}\{x\} - y \right\|_{L^2}^2 + \sum_{c=1}^{C} \| w f^c \|_{L^2}^2 + \lambda \|P\|_F^2 \tag{9.3}$$

Do the Fourier transformation,

$$\widehat{S_{Pf}\{x\}} = P\widehat{f * J\{x\}} = \sum_{d=1}^{D}\sum_{c=1}^{C} p_{d,c} \hat{f}^c[k] X^d[k] \hat{b}_d[k] = (\hat{z}^T P \hat{f})[k] \tag{9.4}$$

where $\hat{z}^d[k] = \widehat{J_d\{x^d\}}[k] = X^d[k]\hat{b}_d[k]$.

By applying the Parseval's formula (1.15),

$$E(f,P) = \left\| \hat{z}^T P \hat{f} - \hat{y} \right\|_{l^2}^2 + \sum_{c=1}^{C} \left\| \hat{w} * \hat{f}^c \right\|_{l^2}^2 + \lambda \|P\|_F^2 \tag{9.5}$$

We use Gauss-Newton (1.10) and Conjugate Gradient (1.11) methods to optimize the quadratic subproblems. The Gauss-Newton method is derived by linearizing the residuals in (9.5) using a first order Taylor series expansion,

$$\begin{aligned}\hat{z}^T(P_i+\Delta P)(\hat{f}_i+\Delta\hat{f}) &\approx \hat{z}^T P_i\hat{f}_{i,\Delta}+\hat{z}^T\Delta P\hat{f}_i \\ &= \hat{z}^T P_i\hat{f}_{i,\Delta}+(\hat{f}_i\otimes\hat{z})^T vec(\Delta P)\end{aligned} \tag{9.6}$$

where $\hat{f}_{i,\Delta}=\hat{f}_i+\Delta\hat{f}$, and $\otimes$ is Kronecker product (ch. 1.3).

Substitute (9.6) into (9.5):

$$\tilde{E}(\hat{f}_{i,\Delta},P+\Delta P)=\left\|\hat{z}^T P_i\hat{f}_{i,\Delta}+(\hat{f}_i\otimes\hat{z})^T vec(\Delta P)-\hat{y}_j\right\|_{l^2}^2+\sum_{c=1}^{C}\left\|\hat{w}*\hat{f}^c\right\|_{l^2}^2+\lambda\|P_i+\Delta P\|_F^2 \tag{9.7}$$

which is a linear least squares problem.

Then use Conjugate Gradient to optimize each Gauss-Newton subproblem to obtain the new filter $\hat{f}^*_{i,\Delta}$ and ΔP^*, and the filter and matrix is updated as: $\hat{f}_{i+1}=\hat{f}^*_{i,\Delta}$ and $P_{i+1}=P_i+\Delta P^*$.

9.1.2 Matrix version of Factorized Convolution Operator

To make things clear, I present all the matrix here:

$$\hat{\boldsymbol{z}}^d=\begin{bmatrix}\hat{z}^d[-K_d]\\ \vdots\\ \hat{z}^d[K_d]\end{bmatrix}_{(2K_d+1)\times 1},\ \hat{\boldsymbol{z}}=\begin{bmatrix}\hat{\boldsymbol{z}}^1\\ \hat{\boldsymbol{z}}^2\\ \vdots\\ \hat{\boldsymbol{z}}^D\end{bmatrix}_{\sum_{d=1}^{D}(2K_d+1)\times 1} \tag{9.8}$$

$$\hat{\boldsymbol{f}}_i^c=\begin{bmatrix}\hat{f}_i^c[-K_c]\\ \vdots\\ \hat{f}_i^c[K_c]\end{bmatrix}_{(2K_c+1)\times 1},\ \hat{\boldsymbol{f}}_i=\begin{bmatrix}\hat{\boldsymbol{f}}_i^1\\ \hat{\boldsymbol{f}}_i^2\\ \vdots\\ \hat{\boldsymbol{f}}_i^C\end{bmatrix}_{\sum_{d=1}^{C}(2K_c+1)\times 1} \tag{9.9}$$

$$\hat{\boldsymbol{f}}_{i,\Delta}^c=\begin{bmatrix}\hat{f}_{i,\Delta}^c[-K_c]\\ \vdots\\ \hat{f}_{i,\Delta}^c[K_c]\end{bmatrix}_{(2K_c+1)\times 1},\ \hat{\boldsymbol{f}}_{i,\Delta}=\begin{bmatrix}\hat{\boldsymbol{f}}_{i,\Delta}^1\\ \hat{\boldsymbol{f}}_{i,\Delta}^2\\ \vdots\\ \hat{\boldsymbol{f}}_{i,\Delta}^D\end{bmatrix}_{\sum_{c=1}^{C}(2K_c+1)\times 1} \tag{9.10}$$

$$\hat{\boldsymbol{y}}_j=\begin{bmatrix}\hat{y}_j[-K]\\ \vdots\\ \hat{y}_j[K]\end{bmatrix}_{(2K+1)\times 1} \tag{9.11}$$

$$\boldsymbol{P}=\left[p_{d,c}\right]_{\sum_{d=1}^{D}(2K_d+1)\times\sum_{c=1}^{C}(2K_c+1)},\ \boldsymbol{P}\hat{\boldsymbol{f}}_{\boldsymbol{i}}=\left[p_{d,c}\hat{\boldsymbol{f}}_i^c\right]_{\sum_{d=1}^{D}(2K_d+1)\times 1},\ \hat{\boldsymbol{z}}^{\boldsymbol{T}}\boldsymbol{P}\hat{\boldsymbol{f}}_{\boldsymbol{i},\boldsymbol{\Delta}}\text{ is a scala.} \tag{9.12}$$

$$\hat{\boldsymbol{f}}_i\otimes\hat{\boldsymbol{z}}=\left[\hat{\boldsymbol{f}}_{\boldsymbol{i}}^{\boldsymbol{c}}\hat{\boldsymbol{z}}^{\boldsymbol{d}}\right]_{\sum_{c=1}^{C}(2K_c+1)\sum_{d=1}^{D}(2K_d+1)\times 1},\ B_f=\begin{bmatrix}(\hat{\boldsymbol{f}}_i\otimes\hat{\boldsymbol{z}})[-K]^T\\ \vdots\\ (\hat{\boldsymbol{f}}_i\otimes\hat{\boldsymbol{z}})[K]^T\end{bmatrix}_{(2K+1)\times\sum_{c=1}^{C}(2K_c+1)\sum_{d=1}^{D}(2K_d+1)} \tag{9.13}$$

where $K := \max_d K_c$.

$$\boldsymbol{p} = \Big[vec(p_{d,c})\Big]_{\sum_{c=1}^{C}(2K_c+1)\sum_{d=1}^{D}(2K_d+1)\times 1}, \ \boldsymbol{\Delta p} = \Big[vec(\Delta p_{d,c})\Big]_{\sum_{c=1}^{C}(2K_c+1)\sum_{d=1}^{D}(2K_d+1)\times 1} \tag{9.14}$$

$$A_j^c = \begin{bmatrix} X_j^c[-K_c]\hat{b}_c[-K_c]\boldsymbol{p}_c & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & X_j^c[K_c]\hat{b}_c[K_c]\boldsymbol{p}_c \end{bmatrix}_{(2K_c+1)\times(2K_c+1)} \tag{9.15}$$

$$A_P = \begin{bmatrix} 0_{(K-K_1)\times(2K_1+1)} & \cdots & 0_{(K-K_C)\times(2K_C+1)} \\ A_j^1 & \cdots & A_j^C \\ 0_{(K-K_1)\times(2K_1+1)} & \cdots & 0_{(K-K_C)\times(2K_C+1)} \end{bmatrix}_{(2K+1)\times\sum_{c=1}^{C}(2K_c+1)} \tag{9.16}$$

The the equation (9.7) becomes:

$$\tilde{E}(\hat{\boldsymbol{f}}_{i,\Delta}, \boldsymbol{p}+\boldsymbol{\Delta p}) = \Big\| A_P\hat{\boldsymbol{f}}_{i,\Delta} + B_f\boldsymbol{\Delta p} - \hat{\boldsymbol{y}}\Big\|_2^2 + \Big\|W\hat{\boldsymbol{f}}_{i,\Delta}\Big\|_2^2 + \lambda\|\boldsymbol{p}+\boldsymbol{\Delta p}\|_2^2 \tag{9.17}$$

To expand it:

$$\begin{aligned}
\tilde{E}(\hat{\boldsymbol{f}}_{i,\Delta}, \boldsymbol{p}+\boldsymbol{\Delta p}) =& (A_P\hat{\boldsymbol{f}}_{i,\Delta} + B_f\boldsymbol{\Delta p} - \hat{\boldsymbol{y}})^H(A_P\hat{\boldsymbol{f}}_{i,\Delta} + B_f\boldsymbol{\Delta p} - \hat{\boldsymbol{y}}) \\
&+ (W\hat{\boldsymbol{f}}_{i,\Delta})^H(W\hat{\boldsymbol{f}}_{i,\Delta}) + \lambda(\boldsymbol{p}+\boldsymbol{\Delta p})^H(\boldsymbol{p}+\boldsymbol{\Delta p}) \\
=& (\hat{\boldsymbol{f}}_{i,\Delta}^H A_P^H + \boldsymbol{\Delta p}^H B_f^H - \hat{\boldsymbol{y}}^H)(A_P\hat{\boldsymbol{f}}_{i,\Delta} + B_f\boldsymbol{\Delta p} - \hat{\boldsymbol{y}}) \\
&+ (\hat{\boldsymbol{f}}_{i,\Delta}^H W^H)(W\hat{\boldsymbol{f}}_{i,\Delta}) + \lambda(\boldsymbol{p}^H + \boldsymbol{\Delta p}^H)(\boldsymbol{p}+\boldsymbol{\Delta p}) \\
=& (f^H A A f + f^H A^H B\Delta p - f^H A^H y) + (\Delta p^H B^H A f + \Delta p^H B^H B\Delta p - \Delta p^H B^H y) \\
&+ (-y^H A f - y^H B\Delta p + y^H y) + f^H W^H W f + \lambda(p^H p + p^H\Delta p + \Delta p^H p + \Delta p^H\Delta p)
\end{aligned} \tag{9.18}$$

,the last formula removed the hats and index to make it clean.

Do the derivative to $\hat{\boldsymbol{f}}_{i,\Delta}$ and $\boldsymbol{\Delta p}$ for the equation by using the properties (1.3) and (1.6), and equals them to zero, then rearrange the equations, we have:

$$\begin{bmatrix} A_P^H A_P + W^H W & A_P^H B_f \\ B_f^H A_P & B_f^H B_f + \lambda I \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{f}}_{\boldsymbol{i},\boldsymbol{\Delta}} \\ \boldsymbol{\Delta p} \end{bmatrix} = \begin{bmatrix} A_P^H\hat{\boldsymbol{y}} \\ B_f^H\hat{\boldsymbol{y}} - \lambda\boldsymbol{p} \end{bmatrix} \tag{9.19}$$

Then we use Conjugate Gradient to iteratively solve this equation.

9.2 Generative Sample Space Model

The sampling method of C-COT is depicted in (ch. 8.8.3), it has a issue of storing a large set of recent training samples. So a probabilistic generative model is proposed.

The approach is based on the joint probability distribution of $p(x, y)$. So, replacing (8.3) by:

$$E(f) = \mathbb{E}\{\big\|S_f\{x_j\} - y\big\|_{L^2}^2\} + \sum_{d=1}^{D}\Big\|wf^d\Big\|_{L^2}^2 \tag{9.20}$$

here $\mathbb{E}$ is evaluated over the distribution $p(x, y)$. If we suppose $p(x, y) = \sum_{j=1}^{m}\alpha_j\delta_{x_j,y_j}(x, y)$, then we get the original equation (8.3).

Notice that, the y_j in (8.3) only differ by a translation, it equals the shifting of the feature map. Hence, we could assume that the target is centered in the image region and all $y = y_0$ are identical. Then the distribution

can be factorized as $p(x,y) = p(x)\delta_{y_0}(y)$. Thus, we only need to estimate $p(x)$, employ Gaussian Mixture Model (GMM):

$$p(x) = \sum_{l=1}^{L} \pi_l \mathcal{N}(x; \mu_l; I) \tag{9.21}$$

where L is the number of components, π_l is the prior weight of component l, μ_l is its mean. I is covariance matrix.

To update the GMM, if the number of components exceeds the L, we discard the component if its π_l is below a threshold. Else, we merge the two closest components by:

$$\pi_n = \pi_k + \pi_l, \mu_n = \frac{\pi_k \mu_k + \pi_l \mu_l}{\pi_k + \pi_l} \tag{9.22}$$

The distance of two components are calculated by $||\mu_k - \mu_l||$. Then the final loss is expressed as:

$$E(f) = \sum_{l=1}^{L} \pi_l \left\| S_f\{\mu_l\} - y_0 \right\|_{L^2}^2 + \sum_{d=1}^{D} \left\| w f^d \right\|_{L^2}^2 \tag{9.23}$$

Compare to (8.3), it just do the following replacements:

$$m \to L,\ \alpha_j \to \pi_l,\ x_j \to \mu_l \tag{9.24}$$

so using the same method as C-COT to train it.

9.3 Update strategy

Update the filter by starting the optimization process in every N_sth frame instead of every time.

9.4 Tracking Frameworks

9.4.1 Initialization for frame 0 with bounding box

1. Initialize all the parameters.
2. Extract features from the first frame (image):$\{x^d[n] : n \in \{0, \cdots, N_d - 1\}, d \in \{1, \cdots, D\}\}$.
3. Multiply the features by the cosine window.
4. Do DFT on the features:
$$X^d[k] = \mathcal{F}(x^d[n]) \tag{9.25}$$
5. Interpolate the features to the continuos domain:
$$\widehat{J_d\{x^d\}}[k] = X^d[k]\hat{b}_d[k] \tag{9.26}$$
6. Initialize projection matrix P, by using PCA to the interpolated features.
7. Do the feature reduction for each feature channel $\hat{J}_d$:
$$\hat{J}_c = P^T \hat{J}_d \tag{9.27}$$
8. Initialize and update sample space.
9. Calculate sample energy and projection map energy.

10. Initialize filter $\hat{\boldsymbol{f}}_0$ and it's derivative $\Delta\hat{\boldsymbol{f}}_0$.
11. Train the tracker.
12. Update the projection matrix P.
13. Re-project the sample and update the sample space.
14. Update distance matrix of sample space.
15. Update filter $\hat{\boldsymbol{f}}_0$.

9.4.2 Localization for frame i

1. Extract the features $\{x^d[n] : n \in \{0, \cdots, N_d - 1\}, d \in \{1, \cdots, D\}\}$ from the region of interest in the frame i for different scales.
2. Do the feature reduction for each feature channel x^d:
$$x^c = P^T x^d \tag{9.28}$$
3. Multiply the feature by cosine window.
4. Do DFT on the extracted features:
$$X^c[k] = \mathcal{F}(x^c[n]) \tag{9.29}$$
5. Interpolate the features to the continuos domain:
$$\widehat{J_c\{x^c\}}[k] = X^c[k]\hat{b}_c[k] \tag{9.30}$$
6. Calculate score in Fourier domain $\widehat{S_f\{x\}}[k]$ according to (8.5):
$$\widehat{S_f\{x\}}[k] = \sum_{c=1}^{C} \hat{f}^c[k]\widehat{J_c\{x^c\}}[k] \tag{9.31}$$
7. Calculate confidence score $s = S_f\{x\}$ by inverse DFT:
$$s(t) = \mathcal{F}^{-1}(\widehat{S_f\{x\}}[k]) \tag{9.32}$$
8. To maximizing the score $s(t) : t \in [0, T)$,
   (a) Using grid search on $s(T\frac{n}{2K+1})$ for $n = 0, \cdots, 2K$ to find a rough initial estimate $s(t)$.
   (b) Do Fourier series expansion $s(t) = \sum_{-K}^{K} \hat{s}[k]e_k(t)$, and use the result above as the initial state, using Newtons method (1.9) to do iterative optimization of it.
9. Update position and scale.

9.4.3 Training for frame i

1. Get the sample calculated in localization.
2. Shift the sample so that the target is centered.
3. Update sample space.
4. Train the tracker every N_sth frame.
5. Update the projection matrix P.
6. Update filter $\hat{\boldsymbol{f}}_i$.

10 Some tips of transfer from matlab to $c++$

- Debug tricks, to show the line number, file number and functions:

```
#define debug(a, args...) printf("%s(%s:%d) " a "\n",
        __func__, __FILE__, __LINE__, ##args)
#define ddebug(a, args...) printf("%s(%s:%d) " a "\n",
        __func__, __FILE__, __LINE__, ##args)
```

- Debug tricks, to show the property of cv::mat:

```
void imgInfo(cv::Mat mat)
{
        int type = mat.type();
        string r;

        uchar depth = type & CV_MAT_DEPTH_MASK;
        uchar chans = 1 + (type >> CV_CN_SHIFT);

        switch (depth)
        {
        case CV_8U:
                r = "8U";
                break;
        case CV_8S:
                r = "8S";
                break;
        case CV_16U:
                r = "16U";
                break;
        case CV_16S:
                r = "16S";
                break;
        case CV_32S:
                r = "32S";
                break;
        case CV_32F:
                r = "32F";
                break;
        case CV_64F:
                r = "64F";
                break;
        default:
                r = "User";
                break;
        }

        r += "C";
        r += (chans + '0');

        debug("%s %d x %d", r.c_str(), mat.rows, mat.cols);
        //return r;
}
```

- Std functions better to name out to prevent default link to other functions. For example:

```
void absTest()
{
        std::vector<float> v{0.1, 0.2};
        float abs = abs(1.23f);
        debug("False result:%f", abs);

        abs = std::abs(1.23f);
        debug("True result:%f", abs);
}
```

  For the False result, it sometimes link to other library and give answer 1.

- Take care of the parameter types, better to write it out clearly.

```
void accumulateTest()
{
        std::vector<float> v{0.1, 0.2};
        float sum = std::accumulate(v.begin(), v.end(), 0);
        debug("False result:%f", sum);

        sum = std::accumulate(v.begin(), v.end(), 0.0f);
        debug("True result:%f", sum);
}
```

  For the False result, it gave answer 0, while the True result is 0.3.

- In opencv, color order: BGR, in matlab: RGB.

- In opencv mat, the data store in this order: $channel -> x -> y$. Test example:

```
void opencvTest()
{
        float *newdata = (float *)malloc(sizeof(float) * (2 * 3 * 4));
        for (int i = 0; i < 2 * 3 * 4; i++)
        {
                newdata[i] = i;
        }

        cv::Mat mat = cv::Mat(cv::Size(3, 4), CV_32FC(2), newdata);

        printf("\nInfo of original mat:");
        imgInfo(mat);
        for (int i = 0; i < 2 * 3 * 4; i++)
        {
                printf("%f ", mat.at<float>(0, i));
        }
        printf("\n");

        std::vector<cv::Mat> splitmat;
        cv::split(mat, splitmat);

        printf("\nInfo of splited mat:");
        imgInfo(splitmat[0]);
```

```
        printf("channel 0:\n");
        for (int j = 0; j < mat.rows; j++)
        {
                for (int i = 0; i < mat.cols; i++)
                {
                        printf("%f ", splitmat[0].at<float>(j, i));
                }
                printf("\n");
        }
        printf("\n");
        printf("channel 1:\n");
        for (int j = 0; j < mat.rows; j++)
        {
                for (int i = 0; i < mat.cols; i++)
                {
                        printf("%f ", splitmat[1].at<float>(j, i));
                }
                printf("\n");
        }
}
```

- When using libcaffe.so, using the BLVC caffe version. I used the Nvidia version and GPU not working, wasted my days.

Reference

[1] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of electronic imaging*, vol. 16, no. 4, p. 049901, 2007.

[2] J. R. Shewchuk *et al.*, "An introduction to the conjugate gradient method without the agonizing pain," 1994.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[5] J. Van De Weijer, C. Schmid, and J. Verbeek, "Learning color names from real-world images," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007.

[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[7] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2544–2550, IEEE, 2010.

[8] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *European conference on computer vision*, pp. 702–715, Springer, 2012.

[9] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[10] M. Danelljan, A. Robinson, F. Shahbaz Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *ECCV*, 2016.

[11] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4310–4318, 2015.

[12] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *CVPR*, 2017.

[13] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.