Project Report

Manas Chaudhari

July 28, 2012

Contents

1 Experiments with Covariance Tracking

Normally, all the components of a color space along with gradients are used in covariance tracking. If we choose the best features, it is possible to reduce the number of features and thus, gain in speed of tracking without much loss in tracking quality. But ignoring a color component completely might lead to a major loss of information. Hence, the linear combinations of R,G and B pixel values which give best discrimination from the background are chosen:

$$F \equiv \{w_1 R + w_2 G + w_3 B \mid w_* \in [-2, -1, 0, 1, 2]\}$$

The choice of these features is done by calculating the Variance Ratio for each combination as proposed in [1]. Experiments have been performed with following different combinations of features:

- $\bar{x}$, $\bar{y}$, R, G, B, gradient
- $\bar{x}$, $\bar{y}$, H, S, V, gradient
- $\bar{x}$, $\bar{y}$, 1 combination of RGB (C_1), gradient
- $\bar{x}$, $\bar{y}$, 2 combinations of RGB (C_1, C_2), gradient

Here, $\bar{x}, \bar{y}$ denote normalized values. In this report, I have tried to show that use of linear combinations can improve tracking speed by about 80% (1.8x) and in some cases, it even improves the quality of tracking.

1.1 Calculation of Variance Ratios

First, variance ratio is calculated for each linear combination. It involves computation of:

- Normalized histograms for object and background
- Log-likelihood ratios from the object and background histograms
- Variance ratio from log-likelihood and histograms

1.1.1 Normalized Histograms for a feature

The minimum and maximum value of the feature in the object as well as the background is determined. All values of the feature are then normalized to fit in the range of 0 - (Resolution). Histograms of length equal to the resolution are created for object and background. For the experiments, resolution of 32 (or 5 bits) has been used.

1.1.2 Log-likelihood ratio

Let p, q denote the histograms corresponding to object and background respectively. The log likelihood ratio of a feature value i is given by

$$L(i) = \log \frac{max\{p(i), \delta\}}{max\{q(i), \delta\}} \tag{1}$$

where δ is a small value that prevents dividing by zero or taking the log of zero.

1.1.3 Variance Ratio

Using the equality $var(x) = Ex^2 - (Ex)^2$, we compute the variance of L(i) with respect to object class distribution p(i) as

$$\begin{aligned} var(L;p) &= E[L^2(i)] - (E[L(i)])^2 & (2) \\ &= \sum_i p(i)L^2(i) - \left[\sum_i p(i)L(i)\right]^2 & (3) \end{aligned}$$

and similarly for background class distribution q(i). The variance ratio of the log likelihood function can now be defined as

$$VarianceRatio(L;p,q) \equiv \frac{var(L;(p+q)/2)}{var(L;p) + var(L;q)}$$

Please refer to [1] for more details.

1.2 Feature Selection

After the variance ratios are calculated, the linear combinations with highest variance ratios are selected for tracking. For covariance tracking, one has to be careful to choose features such that they are linearly independent. Otherwise, the covariance matrix will become singular. Also, there is no difference in the tracking if the number of linear combinations used is equal to the number of components in the color space. The reason for this is explained later. Due to these factors, we choose at most 2 best linear combinations for the experiment.

1.2.1 Correlation criterion

When choosing more than one combinations, it might be effective to choose combinations with least correlated values. For example, consider a window in which the 'G' values of all pixels are very small. Since G is negligible, it is possible to get $R-B$ and $R-B+G$ as the combinations with highest variance ratio as they will

be almost equal. But since both of these are almost equal, it might not be a good idea to choose them for tracking. Instead another combination which has lower correlation can be chosen in spite of lower Variance Ratio. Thus, optimization needs to be done between higher Variance Ratio and lower Correlation criteria. Further testing needs to be done to verify this criterion.

1.3 Covariance Matrix computation

If F_0 is the old feature vector and the new feature vector F_1 is only a linear combination of the old feature vectors i.e. F_1 can be expressed as $W \times F_0$ where $|W| > 0$, then the new covariance matrix can be obtained from the old covariance matrix using the relation

$$C_1 = W \times C_0 \times W^T \tag{4}$$

The derivation of this relation is shown in Appendix B

1.4 Model Update

A technique described in [2] updates the model using Lie Algebra by combining information from last few models. As a result, if tracking becomes inaccurate, contribution of previous models helps in returning back. Since the features keep changing in our case, this technique cannot be used for model update directly.

A base feature vector is constructed which contains base RGB components instead of linear combinations. So feature vector $\begin{bmatrix} x & y & R-G & G+2B & I_x & I_y \end{bmatrix}$ will have base $\begin{bmatrix} x & y & R & G & B & I_x & I_y \end{bmatrix}$. Base feature vector does not change throughout the tracking. Covariance matrix is calculated using base feature vector and it is updated using the Lie Algebra technique. After the new model is obtained, new linear combinations are selected to account for changing background and the new model based on the base feature vector is transformed to get the model for actual feature vector using equation 4.

Detect background change

It is possible to detect change of background calculate the Bhattacharya distance between current background histogram and the last model's background histogram. If the distance is higher than a selected threshold, new model and linear combinations are calculated. Further testing needs to be done on this method.

2 Results

Following data was used to compare performance of tracking:

- Features used
- Tracking time for first 50 frames (T_{50})
- Mean distance between forward and reverse trajectories
- Montages of tracked window

Mean Distance between forward and reverse trajectories

After tracking is complete in forward direction, it is started in the reverse direction with the initial window as the last window in forward trajectory. Mean distance between the two trajectories is calculated using the relation:

$$D \equiv \frac{1}{N} \sum_{frame}^{N} \parallel X_f(frame) - X_r(frame) \parallel \tag{5}$$

where N is the total number of frames and X_f, X_r denote the (vector) points in forward and reverse trajectories.

In the following sections, I have shown some comparisons which show the advantage of using linear combinations.

2.1 PETS/S1

The person in this sequence is quite difficult to track as she walks in front of many cars of various colors. Another problem is the red car as the color matches with the object's color.

Backward

As you can see in Figure 2, tracking is almost 1.8 times faster compared to RGB. Also, when using RGB or HSV, tracking failed in the middle while when using linear combination(s). Thus, use of combinations improves tracking as well as makes it faster.

Forward

As you can see in Figure 3, tracking was successful using linear combinations while it failed when using normal features. Thus, use of combinations improved tracking. Also, tracking is 1.8 times faster when using combinations compared to RGB.

Figure 1: Initial Frame PETS/S1 Backward

(a) Features used: x y C_1 I_x I_y
T_{50} = 7815 ms
Tracking complete

(b) Features used: x y R G B I_x I_y
T_{50} = 13789 ms
Tracking failed at frame 85

(c) Features used: x y C_1 C_2 I_x I_y
T_{50} = 9346 ms
Tracking complete

(d) Features used: x y H S V I_x I_y
T_{50} = 13591 ms
Tracking failed at frame 100

Figure 2: PETS/S1 Backward

(a) Initial Frame

(b) Features used: x y C_1 I_x I_y
T_{50} = 6992 ms

(c) Features used: x y R G B I_x I_y
T_{50} = 12827 ms

Figure 3: PETS/S1 Forward

A Appendix: Basic Covariance Tracking

A.1 Introduction

A brief description of the tracking algorithm is as follows. At each frame, we construct a feature image (Section A.2). For a given object region, we compute the covariance matrix of the features as the model of the object (Section A.3.1). In the current frame, we find the region that has the minimum covariance distance from the model and assign it as the estimated location (Section A.3.2).

A.2 Features of an Image

We denote the observed image with I, where it might be one dimensional intensity image or three dimensional color image, or four dimensional combination of color and infrared images, or etc.
For a given rectangular window R $\subset$ F, let $\{f_k\}_{k=1..n}$ be the d-dimensional feature vectors inside R. We construct the feature vector f_k using two types of mappings; spatial attributes that are obtained from pixel co-ordinate values, and appearance attributes, i.e., color, gradient, infrared, etc. These features may be associated directly to the pixel coordinates

$$f_k = [\ x \quad y \quad I(x,y) \quad I_x(x,y) \quad ...\ ].$$

Alternatively, they can be arranged in radially symmetric relationship

$$f_k^r = [\ \|(x',y')\| \quad I(x,y) \quad I_x(x,y) \quad ...\ ]$$

where

$$\|(x',y')\| = \sqrt{(x'^2 + y'^2)}, \quad (x',y') = (x - x_0, y - y_0)$$

are the relative co-ordinates, and (x_0, y_0) are the coordinates of the window center.

Choice of Features

For gray-scale images, the following 5 features can be used:

$$[\ x \quad y \quad I(x,y) \quad I_x(x,y) \quad I_y(x,y)\ ]$$

In order to reduce computations, it is desirable to use fewer features. For color-scale images, the following 7 features can be used:

$$[\ x \quad y \quad R(x,y) \quad G(x,y) \quad B(x,y) \quad G_x(x,y) \quad G_y(x,y)\ ]$$

where G_x is the maximum of the gradient among the three colors in the x-direction and G_y in the y-direction.

HSV color space

The number of features can be further reduced by converting the RGB color space to HSV. Instead of three features for every color component, only one component C is used, where C is a combination of hue, saturation and value.

$$C = \beta H + (1 - \beta)V \tag{6}$$

where

$$\beta(S) = \begin{cases} 0, & \text{if } S < S_0 \\ (S - S_0)/(S_1 - S_0), & \text{if } S_0 < S < S_1 \\ 1, & \text{if } S > S_1 \end{cases}$$

Thus, the feature vector will contain only 5 elements:

$$[\ x \quad y \quad C \quad G_x(x,y) \quad G_y(x,y)\ ]$$

A.3 Tracking

Covariance Matrix

We represent an $M \times N$ rectangular region R with a $d \times d$ covariance matrix C_R of the feature points as

$$C_R = \frac{1}{MN} \sum_{k=1}^{MN} (f_k - \mu_R)(f_k - \mu_R)^T \tag{7}$$

where μ_R is the vector of the means of the corresponding features for the points within the region R. The covariance matrix is a symmetric matrix where its diagonal entries represent the variance of each feature and the non-diagonal entries represent their respective correlations. Covariance matrix of any region has the same size, thus it enables comparing any regions without being restricted to a constant window size. It has also an scale invariance property over the regions in different images in case the raw features such as, image gradients and orientations, are extracted according to the to scale difference. It is possible to compute covariance matrix from feature images in a very fast way using integral image representation [10]. After constructing tensors of integral images corresponding to each feature dimension and multiplication of any two feature dimensions, the covariance matrix of any arbitrary rectangular region can be computed independent of the region size.

A.3.1 Method for finding the Covariance Matrix

In the process of locating an object in a given region in an image with a given number of features, it is computationally expensive and inefficient to calculate the covariance matrix directly. Typically, for every frame we compute the covariance matrix 25 - 100 times, depending on the accuracy required. Hence, for the purpose of calculating these covariance matrices, we are using the method of sum images/integral images. This process is used to speed up the tracking process at the cost of increasing memory usage.

Integral Images

The integral image $Int(r, c)$ is defined for a gray scale input image $I(x, y)$ by

$$Int(r, c) = \sum_{x<=r} \sum_{y<=c} I(x, y), \tag{8}$$

as the sum of all pixel values inside the rectangle bounded by the upper left corner. Based on this data, intensity sums of any rectangle at any location in the image can be calculated in constant time. The entries of the covariance matrix of a region are given by

$$C_{ij} = \sum_{x<=n} \sum_{y<=m} (f_i(x, y) - \mu_i)(f_j(x, y) - \mu_j)$$

Hence, we can write C_{ij} as,

$$C_{ij} = \sum_{x<=n} \sum_{y<=m} [f_i(x, y)f_j(x, y) - \mu_i f_j(x, y) - \mu_j f_i(x, y) + \mu_i \mu_j]$$

which on simplification leads to

$$C_{ij} = \sum_{x<=n} \sum_{y<=m} f_i(x, y)f_j(x, y) - (mn)\mu_i \mu_j$$

Thus, we can obtain the covariance matrix by building $N(N + 1)/2$ integral matrices Q_{ij} for each pair of feature dimensions f_i and f_j with $i, j <= N$. These product integral matrices Q_{ij} are calculated by

$$Q_{ij}(r, c) = \sum_{x<=r} \sum_{y<=c} f_i(x, y)f_j(x, y). \tag{9}$$

Advantages of using this method

The main benefit of this method is that the covariance matrix of any region can be obtained in constant time once the integral matrices are calculated. Although a large number of calculations are required to compute the integral matrices, the overall number of computations for tracking is much less compared to when covariance matrices were obtained directly.

A.3.2 Finding best match

To obtain the most similar region to the given object, we need to compute distances between the covariance matrices corresponding to the target object window and the candidate regions. Supposing no features in the feature vector would be exactly identical, which states the covariance matrices are positive definite, it is possible apply the distance measure proposed by Frstner [7]. The distance metric uses the sum of the squared logarithms of the generalized eigenvalues to compute the dissimilarity between covariance matrices as

$$\rho(C_i, C_j) = \sqrt{\sum_{k=1}^{d} ln^2 \lambda_k(C_i, C_j)} \tag{10}$$

where $\lambda_k(C_i, C_j)$ are the generalized eigenvalues of C_i and C_j, computed from

$$\lambda_k C_i x_k - C_j x_k = 0 \quad k = 1 \dots d \tag{11}$$

and x_k are the generalized eigenvectors. The distance measure ρ satisfies the metric axioms, positivity, symmetry, triangle inequality, for positive definite symmetric matrices. At each frame we search the whole image to find the region which has the smallest distance from the current object model. The best matching region determines the location of the object in the current frame.

Calculating generalised eigenvalues of a pair of matrices

In order to find the best match, we need to compute the generalized eigenvalues for any two covariance matrices. Since covariance matrices are positive definite, we use cholesky decomposition.

$$\begin{aligned} \lambda_k C_i x_k - C_j x_k &= 0 \\ \lambda_k C_i x_k &= C_j x_k \end{aligned}$$

Now, since C_i is positive semi-definite,C_i can be decomposed as,

$$C_i = LL^T$$

where L is a lower triangular matrix.

$$\lambda_k(LL^T)x_k = C_j x_k$$
$$\left[L^{-1}C_j(L^{-1})^T\right](L^T x_k) = \lambda(L^T x_k)$$

Thus, the problem of finding generalized eigenvalues is reduced to the standard eigenvalue problem $Ax = \lambda x$ where $A = [L^{-1}C_j(L^{-1})^T]$. Since A is symmetric in this case, eigenvalues are obtained quickly.

A.3.3 Model Update

The covariance matrix is updated when the similarity measure is lesser than a threshhold. Note that smaller the similarity measure, more is the actual similarity. The threshhold is obtained from the average of similarity measures until the current frame. Once the model is updated, it is used for atleast 10 frames.
As per requirements, more sophisticated methods based on Lie Algebra [15] or Riemannian Manifolds[16] for updating models may be used.

B Appendix: Covariance Matrix Computation

Consider two linear combinations:

$$I_i = \alpha_i R + \beta_i G + \gamma_i B$$
$$I_j = \alpha_j R + \beta_j G + \gamma_j B$$

Covariance of these two features is given by

$$\begin{aligned}
\sigma^2_{I_i,I_j} &= \frac{1}{N}\sum(I_i - \mu_{I_i})(I_j - \mu_{I_j}) \\
&= \frac{1}{N}\sum I_i I_j - \mu_{I_i}\frac{1}{N}\sum I_j - \mu_{I_j}\frac{1}{N}\sum I_i + \frac{1}{N}\sum \mu_{I_i}\mu_{I_j} \\
&= \frac{1}{N}\sum(I_i I_j - \mu_{I_i}\mu_{I_j})
\end{aligned}$$

Substituting I_i and I_j, we get

$$\begin{aligned}
I_i I_j - \mu_{I_i}\mu_{I_j} &= (\alpha_i R + \beta_i G + \gamma_i B)(\alpha_j R + \beta_j G + \gamma_j B) \\
&\quad -(\alpha_i \bar{R} + \beta_i \bar{G} + \gamma_i \bar{B})(\alpha_j \bar{R} + \beta_j \bar{G} + \gamma_j \bar{B}) \\
&= \alpha_i\left(\alpha_j(R^2 - \bar{R}^2) + \beta_j(RG - \bar{R}\bar{G}) + \gamma_j(RB - \bar{R}\bar{B})\right) \\
&\quad +\beta_i\left(\alpha_j(GR - \bar{G}\bar{R}) + \beta_j(G^2 - \bar{G}^2) + \gamma_j(GB - \bar{G}\bar{B})\right) \\
&\quad +\gamma_i\left(\alpha_j(BR - \bar{B}\bar{R}) + \beta_j(BG - \bar{B}\bar{G}) + \gamma_j(B^2 - \bar{B}^2)\right)
\end{aligned}$$

Now,

$$\begin{aligned}\frac{1}{N}\sum(R^2-\bar{R}^2) &= \sigma^2_{R^2}\\ \frac{1}{N}\sum(RG-\bar{R}\bar{G}) &= \sigma^2_{RG}\\ &\vdots\end{aligned}$$

Therefore,

$$\begin{aligned}\sigma^2_{I_i,I_j} &= \alpha_i\left(\alpha_j\sigma^2_R+\beta_j\sigma^2_{RG}+\gamma_j\sigma^2_{RB}\right)\\ &\quad+\beta_i\left(\alpha_j\sigma^2_{RG}+\beta_j\sigma^2_G+\gamma_j\sigma^2_{GB}\right)\\ &\quad+\gamma_i\left(\alpha_j\sigma^2_{RB}+\beta_j\sigma^2_{GB}+\gamma_j\sigma^2_B\right)\end{aligned}$$

In matrix notation, we can write the equation as

$$\begin{bmatrix}\sigma^2_i & \sigma^2_{ij} & \dots\\ \sigma^2_{ij} & \sigma^2_j & \dots\\ \dots & \dots & \dots\end{bmatrix}=\begin{bmatrix}\alpha_i & \beta_i & \gamma_i\\ \alpha_j & \beta_j & \gamma_j\\ \dots & \dots & \dots\end{bmatrix}\begin{bmatrix}\sigma^2_R & \sigma^2_{RG} & \sigma^2_{RB}\\ \sigma^2_{RG} & \sigma^2_G & \sigma^2_{GB}\\ \sigma^2_{RB} & \sigma^2_{GB} & \sigma^2_B\end{bmatrix}\begin{bmatrix}\alpha_i & \alpha_j & \dots\\ \beta_i & \beta_j & \dots\\ \gamma_i & \gamma_j & \dots\end{bmatrix}$$

Now consider covariance of features x and I_i.

$$\begin{aligned}\sigma^2_{x,I_i} &= \frac{1}{N}\sum(x-\bar{x})(I_i-\mu_{I_i})\\ &= \frac{1}{N}\sum xI_i-\mu_{I_i}\frac{1}{N}\sum x-\bar{x}\frac{1}{N}\sum I_i+\frac{1}{N}\sum\bar{x}\mu_{I_i}\\ &= \frac{1}{N}\sum(xI_i-\bar{x}\mu_{I_i})\\ &= \frac{1}{N}\sum\left(x(\alpha_iR+\beta_iG+\gamma_iB)-\bar{x}(\alpha_i\bar{R}+\beta_i\bar{G}+\gamma_i\bar{B})\right)\\ &= \frac{1}{N}\sum\left(\alpha_i(xR-\bar{x}\bar{R})+\beta_i(xG-\bar{x}\bar{G})+\gamma_i(xB-\bar{x}\bar{B})\right)\\ &= \alpha_i\sigma^2_{x,R}+\beta_i\sigma^2_{x,G}+\gamma_i\sigma^2_{x,B}\end{aligned}$$

In matrix notation, we can write the equation as

$$\begin{bmatrix}\sigma^2_x & \sigma^2_{xy} & \sigma^2_{xi}\\ \sigma^2_{xy} & \sigma^2_y & \sigma^2_{yi}\\ \sigma^2_{xi} & \sigma^2_{yi} & \sigma^2_i\end{bmatrix}=\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0\\ 0 & 0 & \alpha_i & \beta_i & \gamma_i\end{bmatrix}\begin{bmatrix}\sigma^2_x & \sigma^2_{xy} & \sigma^2_{xR} & \sigma^2_{xG} & \sigma^2_{xB}\\ \sigma^2_{xy} & \sigma^2_y & \sigma^2_{yR} & \sigma^2_{yG} & \sigma^2_{yB}\\ \sigma^2_{xR} & \sigma^2_{yR} & \sigma^2_R & \sigma^2_{RG} & \sigma^2_{RB}\\ \sigma^2_{xG} & \sigma^2_{yG} & \sigma^2_{RG} & \sigma^2_G & \sigma^2_{GB}\\ \sigma^2_{xB} & \sigma^2_{yB} & \sigma^2_{RB} & \sigma^2_{GB} & \sigma^2_B\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & \alpha_i\\ 0 & 0 & \beta_i\\ 0 & 0 & \gamma_i\end{bmatrix}$$

Thus combining both equations, we get

$$\begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & \sigma_{xi}^2 & \sigma_{xj}^2 & \sigma_{xk}^2 \\ \sigma_{xy}^2 & \sigma_y^2 & \sigma_{yi}^2 & \sigma_{yj}^2 & \sigma_{yk}^2 \\ \sigma_{xi}^2 & \sigma_{yi}^2 & \sigma_i^2 & \sigma_{ij}^2 & \sigma_{ik}^2 \\ \sigma_{xj}^2 & \sigma_{yj}^2 & \sigma_{ij}^2 & \sigma_j^2 & \sigma_{jk}^2 \\ \sigma_{xk}^2 & \sigma_{yk}^2 & \sigma_{ik}^2 & \sigma_{jk}^2 & \sigma_k^2 \end{bmatrix} = W \times \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & \sigma_{xR}^2 & \sigma_{xG}^2 & \sigma_{xB}^2 \\ \sigma_{xy}^2 & \sigma_y^2 & \sigma_{yR}^2 & \sigma_{yG}^2 & \sigma_{yB}^2 \\ \sigma_{xR}^2 & \sigma_{yR}^2 & \sigma_R^2 & \sigma_{RG}^2 & \sigma_{RB}^2 \\ \sigma_{xG}^2 & \sigma_{yG}^2 & \sigma_{RG}^2 & \sigma_G^2 & \sigma_{GB}^2 \\ \sigma_{xB}^2 & \sigma_{yB}^2 & \sigma_{RB}^2 & \sigma_{GB}^2 & \sigma_B^2 \end{bmatrix} \times W^T$$

where

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \alpha_i & \beta_i & \gamma_i \\ 0 & 0 & \alpha_j & \beta_j & \gamma_j \\ 0 & 0 & \alpha_k & \beta_k & \gamma_k \end{bmatrix} \; and \; \begin{bmatrix} x \\ y \\ I_i \\ I_j \\ I_k \end{bmatrix} = W \times \begin{bmatrix} x \\ y \\ R \\ G \\ B \end{bmatrix}$$

Thus, the new covariance matrix can be expressed in terms of the old covariance matrix.

C Code Documentation

Before describing the acceleration, I would like to describe the code. I have used OpenCV library for handling images and Numerical Recipes 3 (nr3) for matrix operations. Covariance Tracking involves following parts:

1. Extracting data from IplImage structure (covariance.cpp)
2. Calculating integral images (covariance.cpp)
3. Calculating covariance matrix (covariance.cpp)
4. Searching for the matching window (tracking.cpp)
5. Calculating similarity between two covariance matrices (matrixops.cpp)
6. Model Update (main.cpp, matrixops.cpp)
7. Selection of best linear combinations (selectfeatures.cpp)

This section merely describes the use of different functions of the code. It does not describe the working. Please refer to Appendix A to read about the methods.

Figures 4, 5, 6 show the flowchart of the tracking algorithm. Corresponding functions in the code are also mentioned besides the blocks.

Figure 7 shows the organizational chart of the code. It shows which functions belong to which module and also displays them as per level. Higher level functions call the lower level ones.

Figure 4: Highest level algorithm of covariance tracking

Figure 5: Algorithm for searching the matching window

Figure 6: Algorithm for updating the model

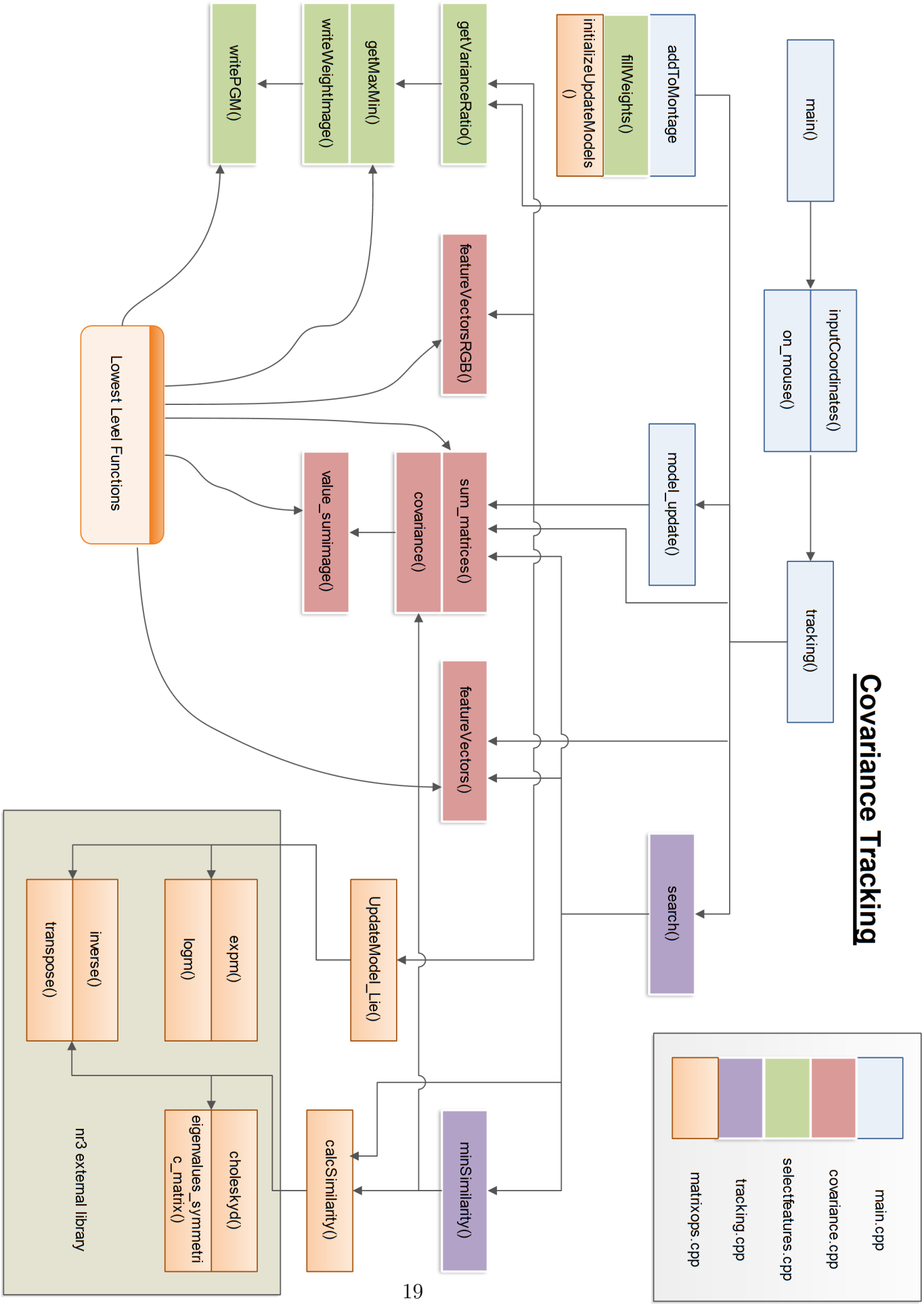

Figure 7: Organizational Chart of the code

D Appendix: Optimization of Covariance Tracking using SIMD

D.1 Code Analysis

Analysis of the code using profilers indicated that the following operations are performed for maximum time:

- Matrix multiplication
- Calculation of eigen values

The use of SIMD acceleration is most effective when dealing with large matrices. And in my case, the maximum size of the matrix is 7×7. Hence it might not be very effective to accelerate these parts. Also, since external libraries were used for these operations, it was difficult to accelerate.

Each function was accelerated separately. Hence the results are shown for individual functions instead of entire code. Results in this report have been obtained using the inbuilt profiler in Microsoft Visual Studio 2010. The original and the accelerated code was run simultaneously for many iterations and ratio of their execution times is recorded for different window sizes.

D.2 Choosing data type

32-bit single precision real vectors of size 128-bits (__m128) have been used throughout the accelerated code. IplImage contains pixel values in 8 bit unsigned char data-type. It would have been very effective as well as easier to use 8-bit unsigned int vectors instead of 32-bit float vectors. However, for further calculations, real values are required. (I tried running the code by using 32-bit integers instead of floats. But, the results were not so good.)

D.3 Optimized functions and Results

FeatureVectors

This function extracts image data from OpenCV's IplImage structure. Data in IplImage is stored in the form of array of structures. It has to be converted into 3 dimensional array of 32-bit float vectors (structure of arrays). Since IplImage contains pixel values in 8-bit unsigned int format, the data is first loaded into 8-bit integer vectors. As IplImage contains data in array of structures $[b_1 g_1 r_1 b_2 g_2 r_2 \ldots]$, data swizzling is performed.

One vector can store 16 values, therefore 48 values i.e. (pixel data of 16 points) has to be transformed into 3 vectors containing r, g and b values in one iteration.

Afterwards, each of the three 8-bit vectors is converted into four 32-bit float vectors. It is evident from the results below that this conversion requires much more time than extraction of data.

n	acceleration real	acceleration 8bit int
16	3.8	14.0
32	5.8	28.4
64	8.0	56.0
128	9.7	105.1

Sum Matrices

This function calculates two integral images described in Section A.3.1.

n	acceleration
16	31.0
32	37.3
64	46.0
128	47.0

Variance Ratio

This function calculates the variance ratios for all the linear combinations of R, G, B. Calculation of variance ratio involves computing the histograms. Hence, this function has not been accelerated completely. Parts other than histogram computation have been accelerated. Hence the acceleration is not so effective.

n	acceleration
16	1.69
32	1.60
64	1.61
128	1.90

References

[1] Collins RT, Liu Y, Leordeanu M. *2005 IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 27 Issue 10, October 2005, Pages 1631-1643*

[2] Fatih Porikli, Oncel Tuzel. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*