Tracking

2022-03-07 00:00:00 -0600 CST

Introduction

Tracking features and objects is required in many applications ranging from autonomous driving to security. Vision tracking systems are often used for live sports broadcasts to keep track of players, the ball, and other visual queues related to the game.

Figure 1: Source: https://azbigmedia.com/lifestyle/ball-tracking-technology-changes-way-fans-consume-practice-sport-of-golf/

Naive tracking will detect an object per frame without any regard for prior information. More sophisticated trackers will consider the previous frame as a starting point to their search space. However, even these trackers many need to initialize after a certain amount of time if their estimate drifts too far away from the object's actual location.

Example of the importance of reliable tracking for driving assistance: https://youtu.be/NSDTZQdo6H8?t=898

Tracking with Optical Flow

Image motion can be described as complex changes in image intensity from one time to another, displaced by δ.

The displacement can be modeled as an *affine motion field* where the point is warped and translated by $\boldsymbol{d}$:

$$\delta = D\mathbf{x} + \mathbf{x},$$

where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix}.$$

When comparing features between an image at t and $t-1$, the feature centered at $\mathbf{x}$ is transformed by

$$\mathbf{I}_t(A\mathbf{x} + \mathbf{d}) = \mathbf{I}_{t-1}(\mathbf{x}).$$

Here, $A = I_2 + D$, where I_2 is the 2×2 identity matrix. This addition will be explained later.

Smaller variations between frames are less reliable for parameter estimation, so a pure translational model is better in these cases. That is

$$\delta = \mathbf{d}.$$

Computing Image Motion

Computing image motion then becomes a minimization problem.

$$\epsilon = \int\int_W \left[\mathbf{I}_t(A\mathbf{x} + \mathbf{d}) - \mathbf{I}_{t-1}(\mathbf{x})\right]^2 w(\mathbf{x}) d\mathbf{x}$$

This is used in a minimization problem which will minimize the dissimilarity between the tracked features between frames. The point is weighted by a function w over a window W.

Minimization of this error involves taking the derivative of ϵ with respect to the unknowns in D and displacement vector $\mathbf{d}$:

$$\frac{1}{2}\frac{\partial \epsilon}{\partial D} = \int\int_W \Big[\mathbf{I}_t(A\mathbf{x}+\mathbf{d}) - \mathbf{I}_{t-1}(\mathbf{x})\Big]\mathbf{g}\mathbf{x}^T w d\mathbf{x} = 0$$
$$\frac{1}{2}\frac{\partial \epsilon}{\partial \mathbf{d}} = \int\int_W \Big[\mathbf{I}_t(A\mathbf{x}+\mathbf{d}) - \mathbf{I}_{t-1}(\mathbf{x})\Big]\mathbf{g} w d\mathbf{x} = 0$$

where

$$\mathbf{g} = \left(\frac{\partial \mathbf{I}_t}{\partial x}, \frac{\partial \mathbf{I}_t}{\partial y}\right)^T$$

is this spatial gradient of the image intensity. We have computed these image gradients before!

To linearlize $\mathbf{I}_t$, a Taylor series expansion of it can be used, taking just the linear term:

$$\mathbf{I}_t(A\mathbf{x}+\mathbf{d}) = \mathbf{I}_t(\mathbf{x}) + \mathbf{g}^T(\mathbf{u}),$$

where

$$\mathbf{u} = D\mathbf{x} + \mathbf{d}.$$

The authors argue that this approximation is reasonable assuming the motion in the images is small.

Plugging this back into the derivatives above yields

$$\int\int_W \mathbf{g}\mathbf{x}^T(\mathbf{g}^T\mathbf{u}) w d\mathbf{x} = \int\int_W \big[\mathbf{I}_{t-1}(\mathbf{x}) - \mathbf{I}_t\big]\mathbf{g}\mathbf{x}^T w d\mathbf{x}$$
$$\int\int_W \mathbf{g}(\mathbf{g}^T\mathbf{u}) w d\mathbf{x} = \int\int_W \big[\mathbf{I}_{t-1}(\mathbf{x}) - \mathbf{I}_t\big]\mathbf{g} w d\mathbf{x}$$

This is solved iteratively, following the Newton method, starting with the following values at $t = 0$:

$$D_0 = I$$
$$\mathbf{d}_0 = \mathbf{0}$$
$$\mathbf{I}_0 = \mathbf{I}(\mathbf{x}).$$

At step i the values are updated to

$$D_i$$
$$\mathbf{d}_i$$
$$\mathbf{I}_i = \mathbf{I}_{i-1}(A_i\mathbf{x} + \mathbf{d}_i).$$

A More Compact Representation

At this point, the authors convert this representation into a more compact form in which the unknowns in D and the values of $\mathbf{d}$ are separated from the function. To achieve this, we start with our current system of equations.

$$\int\int_W \mathbf{g}\mathbf{x}^T(\mathbf{g}^T\mathbf{u}) w d\mathbf{x} = \int\int_W \big[\mathbf{I}_{t-1}(\mathbf{x}) - \mathbf{I}_t(\mathbf{x})\big]\mathbf{g}\mathbf{x}^T w d\mathbf{x}$$
$$\int\int_W \mathbf{g}(\mathbf{g}^T\mathbf{u}) w d\mathbf{x} = \int\int_W \big[\mathbf{I}_{t-1}(\mathbf{x}) - \mathbf{I}_t(\mathbf{x})\big]\mathbf{g} w d\mathbf{x}$$

To achieve this, the Kronecker product is used. This is a generalization of the outer product from vectors to matrices. For two matrices $A \in \mathbb{R}^{p\times q}$ and $B \in \mathbf{R}^{m\times n}$, $A \otimes B$ is a $p \times q$ block matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1q} \\ \vdots & \ddots & \vdots \\ a_{p1}B & \cdots & a_{pq}B \end{bmatrix}.$$

It has two particularly useful properties for this problem:

1. $A^T \otimes B^T = (A \otimes B)^T$
2. $v(AXB) = (B^T \otimes A)v(X)$, where v is the vectorization operator.

Using this product and the properties just listed, we can extract the unknowns D and $\mathbf{d}$ from the equations above.

First, note that $\mathbf{g}^T\mathbf{u}$ appear in both of the equations. These can then be rewritten as follows.

$$\begin{aligned} \mathbf{g}^T\mathbf{u} &= \mathbf{g}^T(D\mathbf{x}+\mathbf{d}) \\ &= \mathbf{g}^T D\mathbf{x} + \mathbf{g}^T\mathbf{d} \\ &= v(\mathbf{g}^T D\mathbf{x}) + \mathbf{g}^T\mathbf{d} && \text{vectorization} \\ &= (\mathbf{x}^T \otimes \mathbf{g}^T)v(D) + \mathbf{g}^T\mathbf{d} && \text{property 2} \\ &= (\mathbf{x} \otimes \mathbf{g})^T v(D) + \mathbf{g}^T\mathbf{d} && \text{property 1} \end{aligned}$$

Likewise, the term $\mathbf{g}\mathbf{x}^T$ that appears on the right side of the first equation in our set to solve can be written as

$$\begin{aligned} v(\mathbf{g}\mathbf{x}^T) &= v(\mathbf{g}\mathbf{1}\mathbf{x}^T) \\ &= \mathbf{x} \otimes \mathbf{g}. \end{aligned}$$

Plugging this into the first equation to solve from above yields

$$\int\int_W (\mathbf{x}\otimes\mathbf{g})((\mathbf{x}\otimes\mathbf{g})^T v(D) + \mathbf{g}^T\mathbf{d}) w d\mathbf{x} = \int\int_W \big[\mathbf{I}_{t-1}(\mathbf{x}) - \mathbf{I}_t(\mathbf{x})\big](\mathbf{x}\otimes\mathbf{g}) w d\mathbf{x}.$$

Expanding these terms out produces

$$\left(\int\int_W (\mathbf{x}\otimes\mathbf{g})(\mathbf{x}\otimes\mathbf{g})^T w d\mathbf{x}\right) v(D) + \left(\int\int_W (\mathbf{x}\otimes\mathbf{g})\mathbf{g}^T w d\mathbf{x}\right)\mathbf{d} = \int\int_W \left[\mathbf{I}_{t-1}(\mathbf{x}) - \mathbf{I}_t(\mathbf{x})\right](\mathbf{x}\otimes\mathbf{g}) w d\mathbf{x}.$$

The authors further simplify this equation using the following variables:

$$\begin{aligned} U(\mathbf{x}) &= (\mathbf{x}\otimes\mathbf{g})(\mathbf{x}\otimes\mathbf{g})^T \\ V(\mathbf{x}) &= (\mathbf{x}\otimes\mathbf{g})\mathbf{g}^T \\ \mathbf{b}(\mathbf{x}) &= \left[\mathbf{I}_{t-1}(\mathbf{x}) - \mathbf{I}_t(\mathbf{x})\right] v(\mathbf{g}\mathbf{x}^T). \end{aligned}$$

Then the above equation can be written as

$$\left(\int\int_W U(\mathbf{x}) w d\mathbf{x}\right) v(D) + \left(\int\int_W V(\mathbf{x}) w d\mathbf{x}\right)\mathbf{d} = \int\int_W \mathbf{b}(\mathbf{x}) w d\mathbf{x}.$$

To write the second equation in a similar way, the authors introduce two additional variables

$$\begin{aligned} Z(\mathbf{x}) &= \mathbf{g}\mathbf{g}^T \\ \mathbf{x}(\mathbf{x}) &= \left[\mathbf{I}_{t-1}(\mathbf{x}) - \mathbf{I}_t(\mathbf{x})\right]\mathbf{g}. \end{aligned}$$

Then,

$$\left(\int\int_W V^T(\mathbf{x}) w d\mathbf{x}\right) v(D) + \left(\int\int_W Z(\mathbf{x}) w d\mathbf{x}\right)\mathbf{d} = \int\int_W \mathbf{c}(\mathbf{x}) w d\mathbf{x}.$$

These equations can be written in a simple form: $A\mathbf{x} + B\mathbf{y} = \mathbf{z}$. A symmetric block matrix $T \in \mathbb{R}^{6\times 6}$ is then introduced:

$$\begin{aligned} T &= \int\int_W \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} w d\mathbf{x} \\ &= \int\int_W \begin{bmatrix} x^2g_x^2 & x^2g_xg_y & xyg_x^2 & xyg_xg_y & xg_x^2 & xg_xg_y \\ x^2g_xg_y & x^2g_y^2 & xyg_xg_y & xyg_y^2 & xg_xg_y & xg_y^2 \\ xyg_x^2 & xyg_xg_y & y^2g_x^2 & y^2g_xg_y & yg_x^2 & yg_xg_y \\ xyg_xg_y & xyg_y^2 & y^2g_xg_y & y^2g_y^2 & yg_xg_y & yg_y^2 \\ xg_x^2 & xg_xg_y & yg_x^2 & yg_xg_y & g_x^2 & g_xg_y \\ xg_xg_y & xg_y^2 & yg_xg_y & yg_y^2 & g_xg_y & g_y^2 \end{bmatrix} w d\mathbf{x}. \end{aligned}$$

Look very closely at Z. Does that remind you of anything? Harris corner detection!

The unknowns are vectorized as

$$\mathbf{z} = \begin{bmatrix} v(D) \\ \mathbf{d} \end{bmatrix}.$$

The product vector is defined as

$$\begin{aligned} \mathbf{a} &= \int\int_W \begin{bmatrix} \mathbf{b} \\ \mathbf{c} \end{bmatrix} w d\mathbf{x} \\ &= \int\int_W \left[\mathbf{I}_{t-1}(\mathbf{x}) - \mathbf{I}_t(\mathbf{x})\right] \begin{bmatrix} xg_x \\ xg_y \\ yg_x \\ yg_y \\ g_x \\ g_y \end{bmatrix} w d\mathbf{x}. \end{aligned}$$

Thus, the iterative solution requires solving the 6×6 linear system

$$T\mathbf{z} = \mathbf{a}.$$

Back to Computing Image Motion

With this more compact representation, the iterative solution is easier to achieve. The authors conveniently note that the deformation of the feature window between frames will be relatively small, so D could be set to 0 for tracking. This leads to the solution of a much smaller system for each time step:

$$Z\mathbf{d} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}.$$

Note that this is only true for small steps between frames. It is also important to measure the dissimilarity between the feature at the initial frame as it changes over time with the iterative estimates. If it changes too much, the dissimilarity will be high, indicating that it is no longer a reliable feature to track.

Picking the Best Feature

Shi and Tomasi posit that the best feature is one that can be tracked well.

"We can track a window from frame to frame if this system represents good measurements, and if it can be solved reliably."

They analyze the basic equation $Z\mathbf{d} = \mathbf{e}$, which is solved during tracking. If both eigenvalues of Z are large and do not differ by several orders of magnitude, the feature can be tracked reliably. That is, they accept a window if the eigenvalues of Z satisfy

$$\min(\lambda_1, \lambda_2) > \lambda.$$

In practice, λ is determined by selecting a lower bound based on a region of uniform brightness in the image as well as an upper bound based on features such as corners. The selected value of λ is somewhere in between.

Measuring dissimilarity

To determine if a feature is still reliable over a longer time period, a measure of dissimilarity is used to measure the original feature versus its warped version at the current frame. Consider the sequence below over 21 frames.

Figure 5.1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

Figure 2: Source: Shi and Tomasi.

Their method successfully tracks the speed limit sign, as seen below.

Figure 5.2: A window that tracks the traffic sign visible in the sequence of figure 5.1. Frames 1,6,11,16,21 are shown here.

Figure 3: Source: Shi and Tomasi.

Figure 5.4: The same windows as in figure 5.2, warped by the computed deformation matrices.

Figure 4: Source: Shi and Tomasi.

They note the importance of using an affine deformation to track reliable features. The figure below plots the dissimilarity using translation versus the deformation matrix over time.

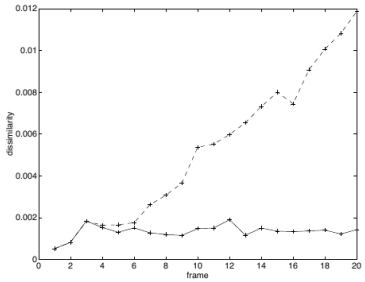

Figure 5: Dissimilarity over time using translation (dashed) versus affine (solid) (Shi and Tomasi).

They also present a case when the feature is lost to occlusion, thus the dissimilarity of both approaches increases greatly over time.

Figure 5.6: The bright window from figure 5.5, visible as the bright rectangular spot in the first frame (a), is occluded by the traffic sign (c).

Figure 6: Source: Shi and Tomasi.

Figure 5.8: The same windows as in figure 5.6, warped by the computed deformation matrices.

Figure 7: Source: Shi and Tomasi.

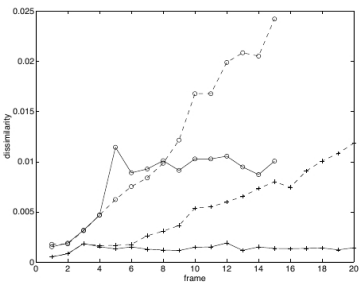

Figure 8: Sign tracking (plusses) versus window tracking (circles) (Shi and Tomasi).

Kalman Filters

Kalman Filters

A **Kalman filter** is a linear dynamic model that models conditional probabilities following normal distributions. It is an simple and effective model for tracking motion even in the presence of noise from measurements. Kalman filters keep an estimate of the state and can update their estimates based on the given observations.

$\mathbf{X}_i$ - State of object at step i

$\mathbf{Y}_i$ - Measurement at step i

There are two primary tasks to deal with, the real-time tracking task and the offline smoothing task. We are more interested in the tracking task, so we will focus on that.

Tracking task: $P(X_k|Y_0, \ldots, Y_k)$

Smoothing task: $P(X_k|X_0, \ldots, Y_n)$

Assumptions:

- $P(Y_k|X_0, \ldots, X_N, Y_0, \ldots, Y_N) = P(Y_k|X_k)$
- $P(X_k|X_0, \ldots, X_{k-1}) = P(X_k|X_{k-1})$

Prediction Task

When tracking an object, we use the model to first predict a state and then update its current parameters given some measurement. We want to predict the state given measurements: $P(\mathbf{X}_i|\mathbf{Y}_0 = \mathbf{y}_0, \ldots, \mathbf{Y}_{k-1} = \mathbf{y}_{k-1})$.

Given the previous observations up to $k-1$, what is our model's estimate of the current state? This can help establish a search location if we were looking to narrow our detection algorithm.

Correction

$P(\mathbf{X}_i|\mathbf{Y}_0 = \mathbf{y}_0, \ldots, \mathbf{Y}_i = \mathbf{y}_i)$ is the cur rent distribution. This is the estimate given the actual observation at i. Note that this observation could be given from a noisy measurement.

Linear Dynamics

Assuming linear models, the problem becomes much simpler. We can model the observations and state using normal distributions.

$$\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$$

The measurements themselves can be modeled as

$$\mathbf{y}_k \sim \mathcal{N}(\mathcal{B}_k\mathbf{x}_k, \Sigma_k),$$

where k is the current step.

For the future, the model will be represented as

$$\mathbf{x}_i \sim \mathcal{N}(\mathcal{D}_i\mathbf{x}_{i-1}; \Sigma_{d_i})$$
$$\mathbf{y}_i \sim \mathcal{N}(\mathcal{M}_i\mathbf{x}_i; \Sigma_{m_i}).$$

For covering the algorithm, we will use notation following Forsyth and Ponce. $\bar{\mathbf{x}}_i^-$ is the mean of $P(\mathbf{x}_i|y_0, \ldots, y_{i-1})$ and $\bar{\mathbf{x}}_i^+$ is the mean of $P(\mathbf{x}_i|y_0, \ldots, y_i)$. Σ_i^- and Σ_i^+ are the covariances of those distributions.

However, by making a convenient assumption about the observation model, we will mainly need to focus on the state model. That is, the matrix $\mathcal{M}_i$ is defined so that the mean is simply the state position at step i.

What does our state represent?

In a simple model, $\mathbf{x} = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$. That is the 2D position and velocity of the object being tracked. The corresponding covariance matrix is $\mathbf{x}\mathbf{x}^T$.

How do we predict the position and velocity of the next time step?

$$\mathbf{p}_k = \mathbf{p}_{k-1} + \Delta t\mathbf{v}_{k-1}$$
$$\mathbf{v}_k = \mathbf{v}_{k-1}$$

This is making a simple, yet surprisingly effective, assumption that the velocity is constant. We can write this as a matrix vector product:

$$\mathbf{x}_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{k-1}$$

Compactly, the prediction for step t is $\bar{\mathbf{x}}_k^- = D_i\bar{\mathbf{x}}_{k-1}^+$.

Since we updated every point $\mathbf{x}_{k-1}$, we also need to make a prediction about the covariance matrix. This is also achieved by multiplying every point by D_i

$$\Sigma_i^- = D_i\Sigma_{i-1}^+D_i^T$$

What if we want to add additional knowledge like acceleration?

Our position prediction then follows

$$\begin{aligned}\mathbf{p}_i &= \mathbf{p}_{i-1} + \Delta t \mathbf{v}_{i-1} + \frac{1}{2}\Delta t^2 \mathbf{a}_{i-1} \\ \mathbf{v}_i &= \mathbf{v}_{i-1} + \Delta t \mathbf{a}_{i-1}\end{aligned}$$

We can simplify this and assume constant acceleration, then $\mathbf{a}_i = \mathbf{a}_{i-1}$. The resulting update equation for $\mathbf{x}_k$ becomes

$$\mathbf{x}_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \mathbf{a} & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \mathbf{a} \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{k-1}$$

However, if something like acceleration is a known **control** factor in our system, then it does not need to be part of the state. In this case, we could separate the update vector into

$$\mathbf{x}_k = \mathcal{D}_i \mathbf{x}_{k-1} + B_k \mathbf{u}_k,$$

where B_k is the **control matrix** and $\mathbf{u}_k$ is the **control vector**.

One last consideration is that of uncertainty due to factors outside of our system. This can also be modeled using a Gaussian with zero mean and covariance Σ_d:

$$\xi_k \sim \mathcal{N}(\mathbf{0}, \Sigma_d).$$

With this added noise, the prediction step becomes

$$\begin{aligned}\bar{\mathbf{x}}_k^- &= \mathcal{D}_k \bar{\mathbf{x}}_{k-1}^+ + B_k \mathbf{u}_k \\ \Sigma_k^- &= \mathcal{D}_k \Sigma_{k-1}^+ \mathcal{D}_k^T + \xi_k.\end{aligned}$$

In words, $\bar{\mathbf{x}}_k^-$ is the prediction of our current state based on the previous best estimate with an added correction term based on known factors (acceleration). Σ_k^- is the updated uncertainty based on the old uncertainty with added Gaussian noise to reflect unknown factors.

Making Corrections

As we are tracking an object, we may have some way of getting measurements. This could be through an object detector or other physical sensor. The new measurement can refine our current set of parameters. Kalman filters work well even if the measurement is noisy.

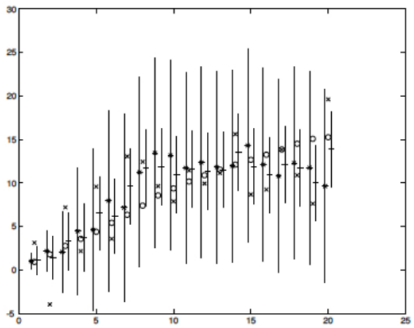

Figure 9: 1D Kalman Filter. Source: Forsyth and Ponce

In the figure above, the initial prediction has a large amount of uncertainty. After updating with the current measurement, the uncertainty is reduced.

The goal here is to reconcile the uncertainty of our predicted state with the uncertainty of the measurement. That is, we want to know the distribution over the union of these two distributions. This is achieved by multiplying the Gaussians together.

$$\mathcal{N}(\bar{\mathbf{x}}_i^+, \Sigma_i^+) = \mathcal{N}(\bar{\mathbf{x}}_i^-, \Sigma_i^-) * \mathcal{N}(\mathbf{y}_i, \Sigma_{m_i})$$

Solving for $\bar{\mathbf{x}}_i^+$ and Σ_i^+ yields

$$\begin{aligned}\bar{\mathbf{x}}_i^+ &= \bar{\mathbf{x}}_i^- + \mathcal{K}_i(\mathbf{y}_i - \bar{\mathbf{x}}_i^-) \\ \Sigma_i^+ &= \Sigma_i^- - \mathcal{K}_i \Sigma_i^-,\end{aligned}$$

where

$$\mathcal{K}_i = \Sigma_i^- (\Sigma_i^- + \Sigma_{m_i})^{-1}.$$

$\mathcal{K}_i$ is called the **Kalman gain**.

Let's combine the prediction and correction steps

We have a state distribution with mean and variance

$$\begin{aligned}\bar{\mathbf{x}}_k^- &= \mathcal{D}_k \bar{\mathbf{x}}_{k-1}^+ + B_k \mathbf{u}_k \\ \Sigma_k^- &= \mathcal{D}_k \Sigma_{k-1}^+ \mathcal{D}_k^T + \xi_k\end{aligned}$$

as well as an observation distribution with mean and variance $\mathbf{y}_k$ and Σ_{m_k}.

Plugging these into the update equations yield

$$\bar{\mathbf{x}}_i^+ = \mathcal{D}_k \bar{\mathbf{x}}_{k-1}^+ + \mathcal{K}_i(\mathbf{y}_i - \mathcal{D}_k \bar{\mathbf{x}}_{k-1}^+)$$
$$\Sigma_i^+ = \mathcal{D}_k \Sigma_{k-1}^+ \mathcal{D}_k^T - \mathcal{K}_i \mathcal{D}_k \Sigma_{k-1}^+ \mathcal{D}_k^T,$$

where

$$\mathcal{K}_i = \mathcal{D}_k \Sigma_{k-1}^+ \mathcal{D}_k^T (\mathcal{D}_k \Sigma_{k-1}^+ \mathcal{D}_k^T + \Sigma_{m_i})^{-1}.$$