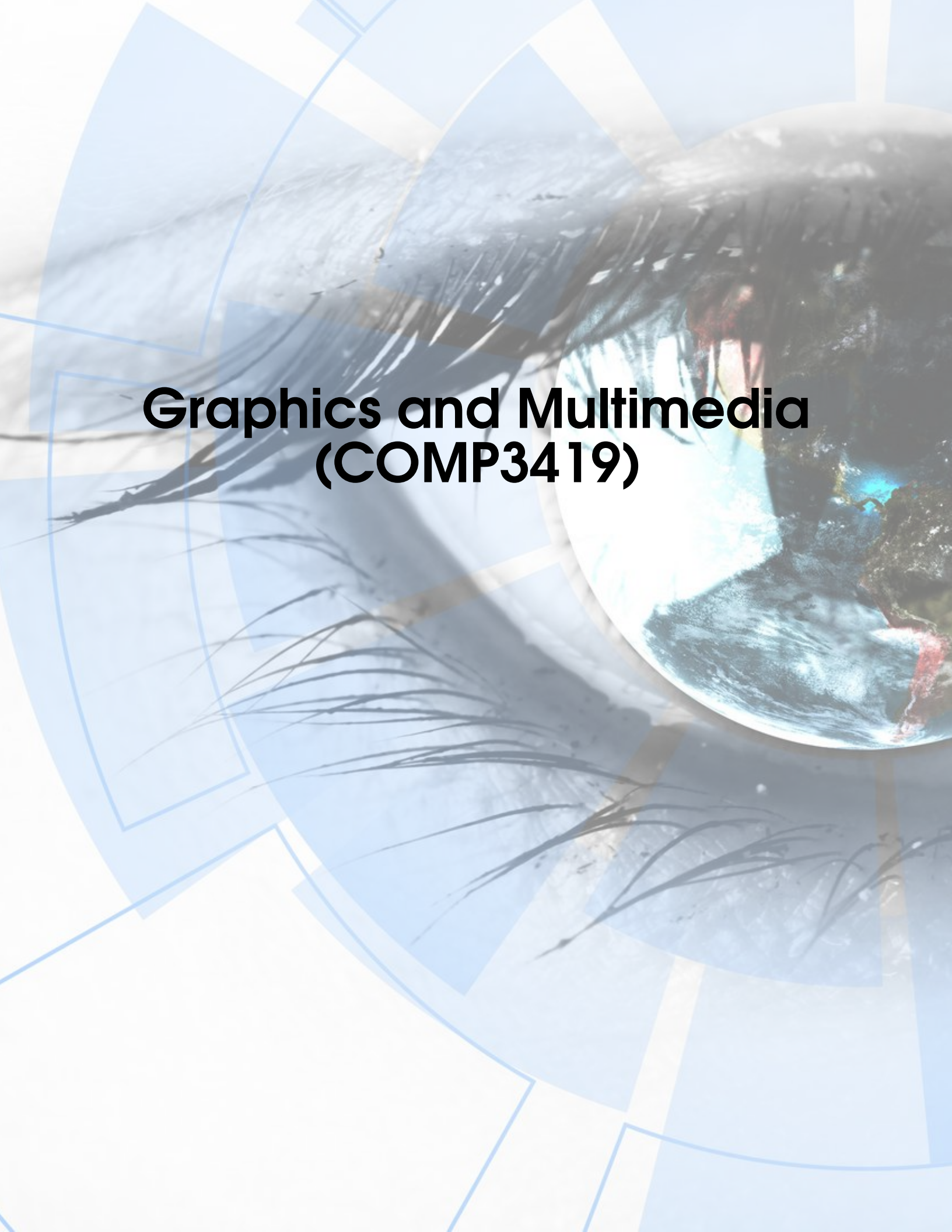
Graphics and Multimedia (COMP3419)



4. Image / Video Processing II

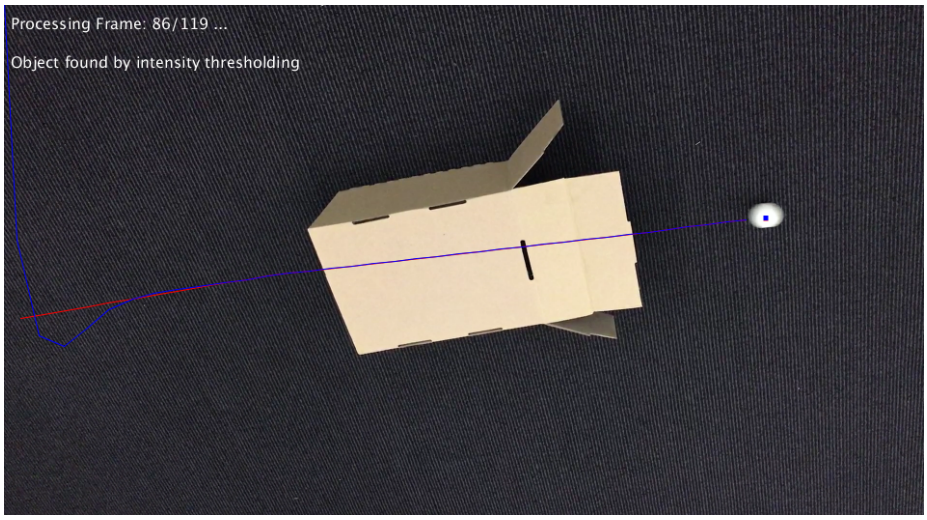


Figure 4.1: The example tracking of *pingpang.mov*: The red line is the tracking based on pixel intensity; the blue line is the tracking based on Kalman filter. They overlap after a few frames of the video. The Kalman filter tracking does not stop when the pingpang ball is covered by the box.

4.1 Intensity based Object Tracking

This task is to track the X-Y coordinates of a pingpang ball which rolls on the ground in *pingpang.mov*. At first, you need to draw a red circle at the centre of the pingpang ball in each frame. Then connect the detected locations with a red line to see the track. Please note that you can skip the frames at the beginning and the end of this video when the ball is out of the frame. In the middle of this video, since ball is covered by a box, the tracking is expected to be lost. To briefly walk through this pipeline:

1. Load the video '*pingpang.mov*' and extract frames from it.
2. Convert each frame into grey scaled.
3. Use a pixel intensity threshold to segment the ball (try large values like 250). There would be some noise points in your segmentation results due to the lighting. However the noises would not seriously affect your tracking if the segmentation is reasonable.
4. For all the segmented foreground pixels (most of them belong to the ball), calculate their mean coordinates (centre of mass) as

$$\bar{x} = \frac{\sum_{x\in\Omega} x}{N}, \bar{y} = \frac{\sum_{y\in\Phi} y}{N} \tag{4.1}$$

   where $\Omega = \{x_1, x_2, \ldots, x_i, \ldots, x_N\}$ is the set of X coordinates of N foreground pixels; $\Phi = \{y_1, y_2, \ldots, y_i, \ldots, y_N\}$ is the set of Y coordinates of N foreground pixels.
5. Draw a red circle at the coordinate $(\bar{x}, \bar{y})$
6. Draw line segments between all detected $(\bar{x}, \bar{y})$ of all previous frames to show the detected track. It means you need to store all the previously tracked coordinates and draw a segment between each adjacent pair of stored coordinates.

4.2 Motion field based Object Tracking (Adv.)

In videos that objects cannot be easily thresholded, the motion based tracking would be preferable. To track the coordinates of the car in *quadrangle.mov*, please try to use the macroblock matching framework (or optical flow) you implemented to extract the motion field vectors. Then threshold the magnitudes of the motion fields to track the car. The coordinates of most pedestrians could be filtered out in this method, because the car moves faster than the rest objects.

The following equation defines the optical flow:

$$\begin{aligned}
H(x,y) &= I(x+u, y+v) \\
I(x+u, y+v) &= I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + higher\ order\ terms \\
I(x+u, y+v) &\approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \quad (Seitz) \\
I_x &= \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y} \\
0 &= I(x+u, y+v) - H(x,y) \\
0 &\approx I(x,y) + I_x u + I_y v - H(x,y) \\
0 &\approx (I(x,y) - H(x,y)) + I_x u + I_y v \\
0 &\approx I_t + I_x u + I_y v \\
0 &\approx I_t + \nabla I[u, v]
\end{aligned} \tag{4.2}$$

where I is the original image; H(x,y) represents the small motion; (u, v) represents the displacement of (x, y).

4.3 Kalman Filter based Object Tracking (Adv.)

In *pingpang.mov*, the tracking of the pingpang ball is lost when it is covered by a box. Such conditions would cause tracking errors in practice. The Kalman filter tracking is useful for keeping

the tracking going reasonably even when the object is absent. The Kalman filter estimates the next location of the object with its previous speed and acceleration, thus it is able to output the coordinates of the object even when the current coordinates of the object is missing. This method could deal with the tracking tasks with multiple objects and object overlapping. If you are interested, please refer to video tutorials at `https://www.youtube.com/watch?v=CaCcOwJPytQ&list=PLX2gX-ftPVXU3oUFNATxGXY90AULiqnWT&index=1`.

Kalman filter is an iterative process to estimate the value when the measurement of the value contains uncertainty or random error. Each iteration is divided into two steps.

The first step is prediction:

$$X_t = AX_{t-1} + Bu_t + w_t \tag{4.3}$$

$$P'_t = AP_{t-1}A^T + Q \tag{4.4}$$

The first equation 4.3 is the state equation. We estimate the state based on previous estimation. A and B are parameter matrices related to δt which is the sampling time rate. u_t is control variable matrix. w_t represents noise in the process.

The second equation 4.4 represents the estimation of state covariance which is the error in the estimate X_t. Q is the process noise covariance matrix. It prevents the estimated state covariance from getting too small or becoming 0.

We predict the current value based on previous estimation and calculate the estimation error during this step. Initial estimated value and estimation error are given in equation 4.8.

The second step is correction:

$$K_t = \frac{P_t C^T}{CP_t C^T + R} \tag{4.5}$$

$$X_t = X_t + K_t(M_t - C * X_t) \tag{4.6}$$

$$P_t = (1 - K_t C)P'_t \tag{4.7}$$

K_t in equation 4.5 is the Kalman Gain. C is a parameter matrix. R is measurement covariance matrix which is the error in the measurement.

Equation 4.6 is the correction of our estimation. M_t is the real measurement of the value. As you can see here, Kalman Gain represents the portion of the real measurement (M_t) we will use when we update our estimation. The larger the measurement error is, the less we will use measurement and the more stable our estimation is.

Equation 4.7 recalculates the estimates error and P_t is the error in the estimate.

In this step, we take measurement to update our estimation. Kalman Gain here represents the weight of measurement versus estimation.

In our pingpang example, we set parameters as follow:

$$
\begin{cases}
A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
B = \begin{bmatrix} \frac{dt^2}{2} \\ \frac{dt^2}{2} \\ dt \\ dt \end{bmatrix} \\
C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\
Q = \begin{bmatrix} \frac{dt^4}{4} & 0 & \frac{dt^3}{2} & 0 \\ 0 & \frac{dt^4}{4} & 0 & \frac{dt^3}{2} \\ \frac{dt^3}{2} & 0 & dt^2 & 0 \\ 0 & \frac{dt^3}{2} & 0 & dt^2 \end{bmatrix} \\
R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
X_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
P_0 = Q \\
u_t = 0.005 \\
w_t = 0
\end{cases}
\tag{4.8}
$$

where dt is the sampling time rate which is set as 1.