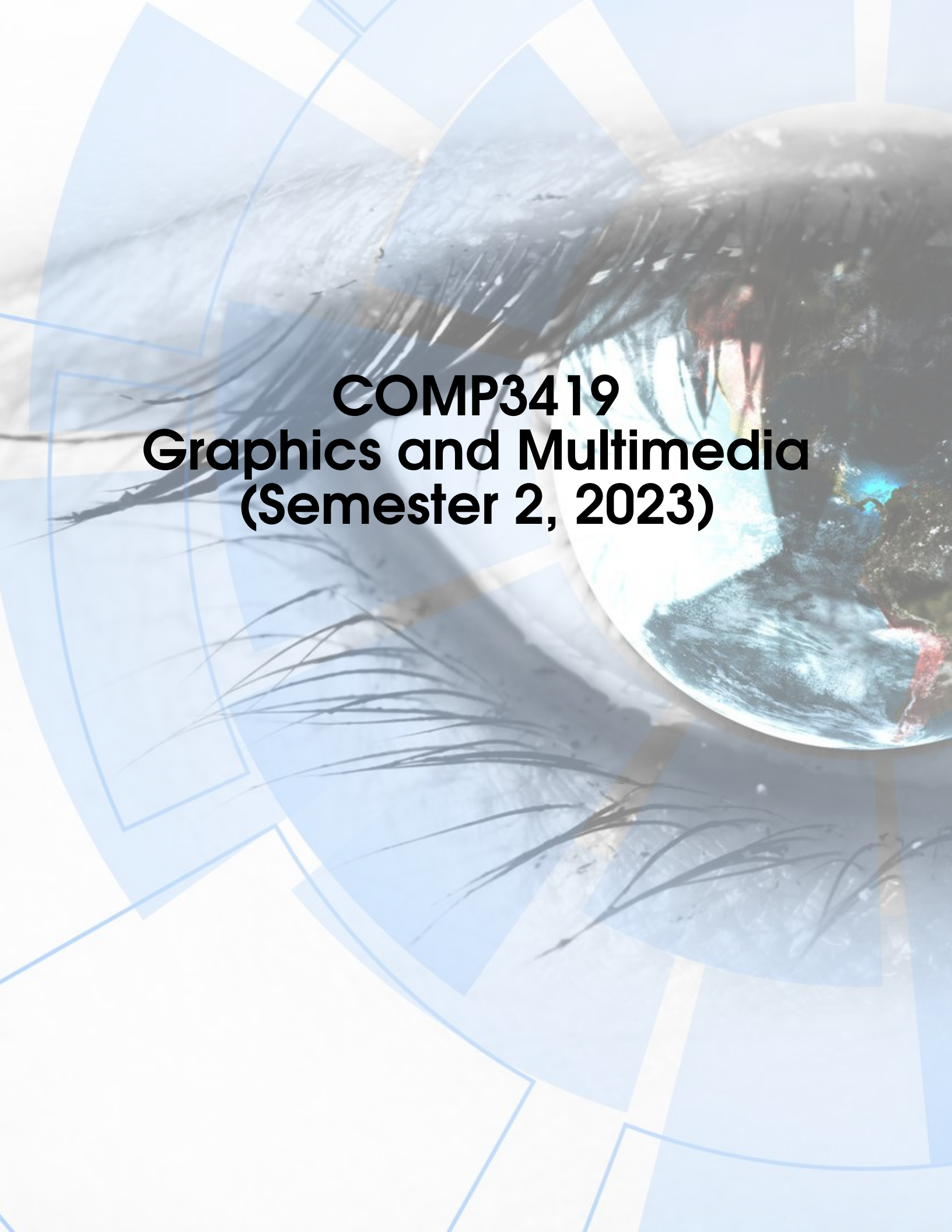


COMP3419
Graphics and Multimedia
(Semester 2, 2023)

Assignment 1-a Specification

1 Key Information

- The mark of **"COMP3419 Assignment 1-a Motion Estimation and Visualization"** will be marked based on canvas submission. Due Time: before 23:59, Sunday of Week 8 (24-Sep-2023).
- This individual assignment is worth **8%** of your final assessment.
- **Submission Deliverable**: Students are asked to create a **zip** file of all deliverable, including all source code and a demo video. A **README** txt file (to describe the steps/instructions regarding how to get their source code running to derive the expected outputs) should be included within the zip file. Please be aware of the following submission **restrictions** that (1) this zip file should be named as "SIDxxx_Ass1a.zip" where xxx denotes the student ID (e.g., "SID450003419_Ass1a"); (2) the format of source code is restricted to **py** or **ipynb**; (3) the demo video should be named as "SIDxxx_Ass1a" and the format of the demo video is restricted to **mp4**.
- Student's assignment **will only be marked if** all deliverable can be **accessed** from the Canvas System, and they can be **runnable** following instructions provided in README txt file. Failing to follow these restrictions or missing of demo video would cause a deduction of **4 marks**. Once plagiarism detected by the Canvas system, the student will receive no mark immediately, as well as other related penalties from university.

2 Demo Video

- Students are required to use the video provided as the input source for this assignment **(monkey.avi)**.
- Students are asked to record a demo video including their output video and demonstration of their implementation/code.
- The length of the demo video recording should not exceed **1 minute** and should be sufficient enough to show the motion changes.

3 General Marking Policy

Late Submission & Demonstration Policy
For the late submission cases, penalties will be assigned according to the university wide late penalties for assignment Clause 7A of the Assessment Procedures.
Special Consideration and Arrangements
While you are studying, there may be circumstances or essential commitments that impact your academic performance. Our special consideration and special arrangements process is there to support you in these situations. More information on how to lodge the special consideration application, can be found from this webpage.

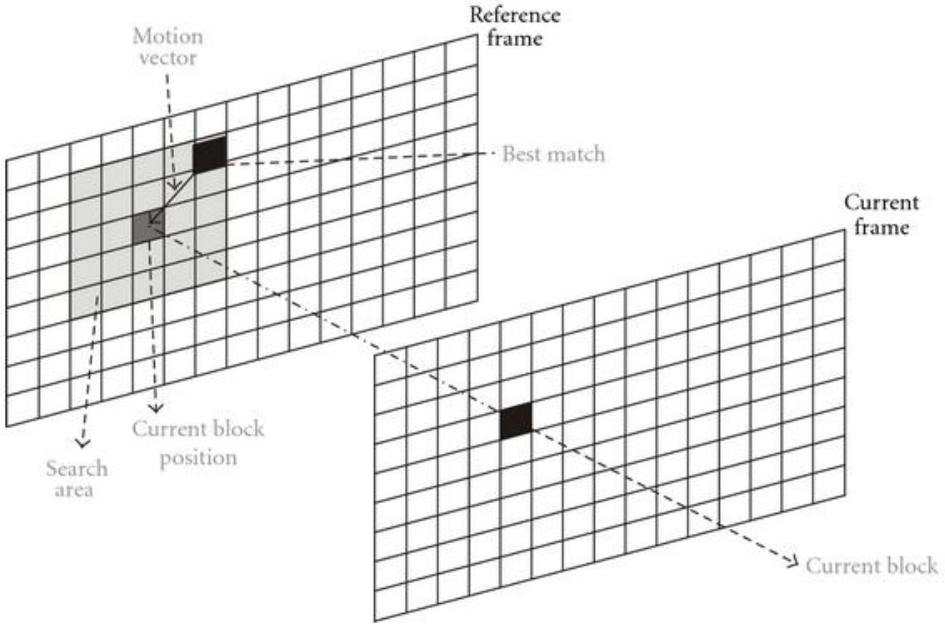
4 Motion Estimation and Visualization (Assignment 1-a)

This assignment is to perform the motion estimation with macroblock matching on the video provided. The basic premise of motion estimation is that in most cases, consecutive video frames will be similar except for changes induced by objects moving within the frames. The basic idea of motion estimation is to define grids of block regions on two adjacent frames and find the displacement vector (a 2D Cartesian vector in 2D videos) between the matched blocks. To describe the meta-algorithm step by step:

1. Iterate the video frames F_i of size (f_x, f_y); Define a grid block of size ($2k+1$, $2k+1$), whose size should be odd for convenience in determining the coordinates of centroid point (or, origin point) for each grid block. Each frame F_i results in $\frac{f_x f_y}{(2k+1)^2}$ grid blocks overall.
2. For **each grid block** $B_i^{(x,y)}$ (whose centroid point) at (x,y) in current frame F_i, search for all candidate blocks in next reference frame F_{i+1}, and then find its matched grid block $B_{i+1}^{(x',y')}$ at (x',y') in F_{i+1}.
   - Note: $B_i^{(x,y)}$ is also dubbed as the source block, while $B_{i+1}^{(x',y')}$ is also called as the destination or target block. F_i represents the current frame, while F_{i+1} denotes the next or reference frame. Please see Fig. 1(a) for visual illustration.
   - Among all candidates blocks within the search area of $B_i^{(x,y)}$ in F_{i+1}, the matched grid block $B_{i+1}^{(x',y')}$ should produce the minimum **s**um-of-**s**quared-**d**istances (*SSD*) to $B_i^{(x,y)}$.
   - Simplifying $B_i^{(x,y)}$ and $B_{i+1}^{(x',y')}$ as B_i and B'_{i+1}, respectively, their **sq**uare **r**oot of *SSD* can be computed as

$$sqrt(SSD(B_i, B'_{i+1})) = \sqrt{\sum_{bx=-k}^{k}\sum_{by=-k}^{k}\sum_{bc=0}^{2}[B_i(bx,by,bc) - B'_{i+1}(bx,by,bc)]^2}, \quad (1)$$

     where bx, by, and bc denote the inside-**b**lock index of pixel location in **x**-direction, **y**-direction, and **c**olor channels (i.e., RGB), respectively. For example, when $bx = by = -k$ and $bc = 0$, then $B_i(bx,by,bc)$ denotes the *R-channel* value of pixel at *top-left corner* of B_i.
3. The (centroid-point) displacement vector from source block (B_i) to target block (B'_{i+1}) can be represented as a 2-D vector, i.e., $(x'-x, y'-y)$. Next, save the displacement vectors for **all source blocks** B_i in frame F_i as a 3D matrix of shape ($\frac{f_x}{2k+1}$, $\frac{f_y}{2k+1}$, 2).
4. For better visualization purpose, filter out the displacement vectors computed in *Step 3* to remove unexpected noises, whose $sqrt(SSD)$ are outside of a self-defined thresholding range (T_{min}, T_{max}).



(a) Illustration of the block matching

(b) Example arrow visualization of extracted motion vectors.

Figure 1: The illustration of block matching algorithm and the extracted optical flows.

- In other words, neglect the noisy displacement vectors whose $sqrt(SSD) \notin (T_{min}, T_{max})$.
- The optimal values of T_{min} and T_{max} vary on different videos or implementation details, which should be determined by experiments.

5. Draw arrows to visualise these selected (centroid-point) displacement fields on frame F_i.
6. Repeat *Step 1-5* for all frames.

Tip: To speed up for *Step 2*, an assumption could be reached that there are high chances that these matched blocks might appear in positions close to the source block B_i. Therefore, we

could search for its neighbouring blocks only within a certain radius R, instead of taking the entire reference frame F_{i+1} as its search area.

Tip: The number of the extracted frames are large and the processing time may be long if you want to do all the frames. We strongly recommend that you should:

1. Extract a few frames which the motions can be clearly observed.
2. Implement macroblock matching on these frames.
3. Make sure your implementation works and motion can be captured.
4. Apply to all the frames to generate the output.

Tip: A ***helper_function.py*** is provided to help students draw the arrows in *Step 5*. Please download it from Canvas, and use it as your starting point. Students are also welcome to draw arrows in their preferred shapes, such as the one in Fig. 1b.

For more details of in-depth understanding of the video estimation algorithm, you may refer to this link.

For a faster solution of estimating optical flow with Lucas-Kanade method please refer to this link. Please note for the assignment submission you are **NOT** required to use the Lucas-Kanade method.