HW4

Orr Zwebner 203253422
Gideon Bonwit 307920942

1. Assume that you apply the optical flow algorithm (OF) on a pair of images with a given set of parameters. Let p be a pixel for which the algorithm fails to compute the OF. Moreover, changing a single parameter results in computing the OF of p. Write what the parameter may be, why the algorithm fails in the first case and why it succeeds in the second case.

Answer:
The parameter that can cause the difference between success and failure of the optical flow algorithm is window size which refers to the size of the patch around a pixel representing the neighborhood size of the pixel. We will observe 2 options:

1. **Too Small Window Size**: If the window size is too small, it may not capture enough local information which is represented by the matrix C, for a reliable estimation of the optical flow. Specifically, the gradients computed over a smaller patch may not be representative of the actual motion pattern, leading to an underdetermined system where the rank of the matrix C is less than 2. This results in insufficient constraints to uniquely solve for the motion vectors $u\ and\ v$ uniquely, causing the algorithm to fail. In the second case, by increasing the window size, we encompass more pixels in the neighborhood, which can lead to a more robust estimate of the local gradients and improve the reliability of the gradients (and their sums) represented by the C matrix.
2. **Too Large Window Size**: If the window size is too big, the neighborhood of a pixel might include multiple types of movements. Pixels far from the central pixel p_0 might exhibit different or no motion, leading to a miscalculation in the local constant flow, because the optical flow equation doesn't hold for all pixels within the window ($\exists p \in w(p_0)\ s.t\ u(p) \neq \frac{dx}{dt}\ and\ \backslash\ or\ v(p) \neq \frac{dy}{dt}$). In the second case, by decreasing the size of the window, we can more likely capture a single motion pattern, which aligns with the assumption of local smooth flow and enables the correct computation of the optical flow for pixel p.

2. Consider the optical flow algorithm which we learned in class. On which camera motion it is expected to fail? Give a short explanation to your answer.

Answer:

A camera motion that is expected to fail is a motion that is not a translation, such as rotation.

The optical flow equation relies on several assumptions, and camera rotation challenges these assumptions in ways that are not easily accommodated by simple optical flow models.

Basic Optical Flow Equation

The optical flow equation is derived from the brightness constancy assumption, which posits that the appearance (brightness) of any point in the scene remains constant over time, despite motion. Mathematically, this is expressed as:

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

Where $I(x, y, t)$ is the brightness of a pixel at position (x, y) at time t, and dx, and dy represent small changes in position and time.

By applying a Taylor expansion and neglecting higher-order terms, we get the optical flow constraint equation:

$$I_x u + I_y v = -I_t$$

Where:
- I_x and $I_y$are the spatial derivatives of the image brightness in the x and y directions, respectively.
- I_t is the temporal derivative of brightness.
- u and v are the optical flow components in the x and y directions, respectively.

Failure During Camera Rotation

1. **Appearance Changes**: Rotation changes the relative orientation of surfaces to the camera, potentially altering the apparent brightness $I(x, y, t)$ of objects due to perspective effects, shading changes, or both. Since the optical flow equation relies on brightness constancy, these appearance changes can lead to incorrect flow estimates. Mathematically, this means that

$$I(x, y, t) \neq I(x + dx, y + dy, t + dt)$$

for rotating objects, contradicting the fundamental assumption.

2. **Global Motion vs. Local Gradient**: The optical flow equation uses local image gradients I_x and I_y to infer motion. During rotation, especially around the camera's axis, the motion of points in the image does not correspond linearly to these local gradients. This discrepancy arises because rotational motion in three dimensions (3D) causes complex changes in the two-dimensional (2D) image plane that are not captured by simple linear models.

Additionally, various other factors might be influenced by rotational motion, but it's important to recognize that these can also result from any type of movement.

1. **Violation of Small Motion Assumption:** The optical flow constraint equation assumes that the motion between consecutive frames is small (dx, dy, dt). However, rotation can cause significant changes in the position of objects relative to the camera, especially for objects at different depths. This leads to large values of dx and \ dy, which can violate the assumptions used in deriving the optical flow constraint equation.

2. **Occlusions and Disocclusions**: Rotation can bring new objects into the field of view or hide others, introducing discontinuities in optical flow that are not accounted for in the linear model. These occlusions and disocclusions violate the assumption that every point (x, y) at time t has a corresponding point $x + dx, y + dy$ at time $t + dt$.

In essence, camera rotation introduces complexities—large displacements, appearance changes due to perspective shifts, and occlusions—that exceed the capabilities of the basic optical flow model, leading to its mathematical inadequacy in accurately capturing the motion under these conditions.

3. Assume two pixels have the same optical flow and the camera is static. Does it necessarily implies that they are projections of two 3D points that move at the same 3D direction? If so explain why, and if not give a specific counter example.

Answer:
No, it does not necessarily imply that they are projections of two 3D points that move at the same 3D direction. Optical flow measures the apparent motion of image brightness patterns between two frames. If the camera is static, optical flow corresponds to the projection of the actual 3D motion in the scene onto the 2D image plane. Two pixels having the same optical flow could mean that they are part of the same moving object or that they are parts of different objects that happen to move in such a way that their motion projects to the same 2D motion in the image plane.

It is noticble that in the Optical Flow equation there is no mention to the Z axis (depth):

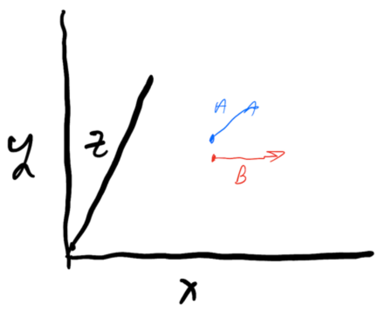
$$I_x u + I_y v = -I_t$$

Additionally, due to the aperture problem, the observed optical flow vector could be consistent with many different actual motion vectors in 3D space. The ambiguity arises because the motion could include a component parallel to the edge direction that is not detectable through the aperture.

An example:
Two cars traveling at the same speed on a road. One car is moving away from you and right (in depth), while the other car is crossing your field of vision from left to right.

In the 2D image plane (like a photo or a frame of video), both cars might appear to move the same distance across the image from one moment to the next, thus having the same optical flow in the image plane. However, in reality:

- **Car Moving in Depth (A):** The first car is actually moving away from you.
- **Car Moving Horizontally (B):** The second car is moving perpendicular to your line of sight, so its motion is easily observed as movement across the image from one side to the other.



4. Consider two images of the same static scene that were taken from two cameras. Assume that the COP of the cameras are identical (no translation only rotation and maybe different internal parameters).
Prove formally: the two images are related by an homography transformation.
Hint: You can prove it using the assumption that the world coordinate system is the same as the coordinate system of one of the cameras, and the rotation between the cameras, as well as the internal parameters of each camera are known. If you use these assumptions, you have to explain why it is ok to use them.

Answer:
Let I_1, I_2 be two images as described above. In order to prove that I_1, I_2 are related by an homography transformation, we need to show that $\exists H \in \mathbb{R}^{3\times3}$ such that for any 2 corresponding point p, q where $p \in I_1$ and $q \in I_2 \rightarrow \tilde{p} = H\tilde{q}$. Let $p \in I_1$ be a pixel in the static scene, such that $\exists q \in I_2$ where q correspond to p.
Assumption:

- The COP of both images is the world coordinate system, thus the Euclidean coordinates in 3D of the COP (of both images) Is $(0,0,0)$. Such an assumption is legal, because it is given that they are identical, and if the world coordinates is different, we can adjust the proof by translating the COP to it's true coordinates.

Let $s_{x1}, s_{y1}, o_{x1}, o_{y1}, f_1$ be the intrinsic parameters of I_1 and $s_{x2}, s_{y2}, o_{x2}, o_{y2}, f_2$ be the intrinsic parameters of I_2.

Let $R_1, R_2 \in \mathbb{R}^{3\times3}$ be the rotation matrices respectively. Because of assumption 1, the translation vectors $T_1, T_2 = 0$. Let P be the object point that is projected, s.t $\tilde{p} = M_1\tilde{P}$, $\tilde{q} = M_2\tilde{P}$ and M_1, M_2 are the projection matrices of the images. Denote:

$$M_{int1} = \begin{pmatrix} s_{x1}f_1 & 0 & o_{x1} & 0 \\ 0 & s_{y1}f_1 & o_{y1} & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, R_1 = \begin{pmatrix} R_{11}^1 & R_{12}^1 & R_{13}^1 \\ R_{21}^1 & R_{22}^1 & R_{23}^1 \\ R_{31}^1 & R_{32}^1 & R_{33}^1 \end{pmatrix}$$

$$M_1 = M_{int1}\begin{bmatrix} R_1 & -R_1T \\ 0 & 1 \end{bmatrix} = M_{int1}\begin{bmatrix} R_1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{pmatrix} s_{x1}f_1R_{11}^1 + o_{x1}R_{31}^1 & s_{x1}f_1R_{12}^1 + o_{x1}R_{32}^1 & s_{x1}f_1R_{13}^1 + o_{x1}R_{33}^1 & 0 \\ s_{y1}f_1R_{21}^1 + o_{y1}R_{31}^1 & s_{y1}f_1R_{22}^1 + o_{y1}R_{32}^1 & s_{y1}f_1R_{23}^1 + o_{y1}R_{33}^1 & 0 \\ R_{31}^2 & R_{32}^2 & R_{33}^2 & 0 \end{pmatrix}$$

$$M_{int2} = \begin{pmatrix} s_{x2}f_2 & 0 & o_{x2} & 0 \\ 0 & s_{y2}f_2 & o_{y2} & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, R_2 = \begin{pmatrix} R_{11}^2 & R_{12}^2 & R_{13}^2 \\ R_{21}^2 & R_{22}^2 & R_{23}^2 \\ R_{31}^2 & R_{32}^2 & R_{33}^2 \end{pmatrix}$$

$$M_2 = M_{int2}\begin{bmatrix} R_2 & -R_2T \\ 0 & 1 \end{bmatrix} = M_{int2}\begin{bmatrix} R_2 & 0 \\ 0 & 1 \end{bmatrix}$$
$$= \begin{pmatrix} s_{x2}f_2R_{11}^2 + o_{x2}R_{31}^2 & s_{x2}f_2R_{12}^2 + o_{x2}R_{32}^2 & s_{x2}f_2R_{13}^2 + o_{x2}R_{33}^2 & 0 \\ s_{y2}f_2R_{21}^2 + o_{y2}R_{31}^2 & s_{y2}f_2R_{22}^2 + o_{y2}R_{32}^2 & s_{y2}f_2R_{23}^2 + o_{y2}R_{33}^2 & 0 \\ R_{31}^2 & R_{32}^2 & R_{33}^2 & 0 \end{pmatrix}$$
$$\tilde{P} = \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

$$\tilde{p} = M_1\tilde{P} = \begin{pmatrix} s_{x1}f_1R_{11} + o_{x1}R_{31} & s_{x1}f_1R_{12} + o_{x1}R_{32} & s_{x1}f_1R_{13} + o_{x1}R_{33} & 0 \\ s_{y1}f_1R_{21} + o_{x2}R_{31} & s_{y1}f_1R_{22} + o_{x2}R_{32} & s_{y1}f_1R_{23} + o_{x2}R_{33} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \end{pmatrix}\begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix} =^{*}$$
$$\begin{pmatrix} s_{x1}f_1R_{11} + o_{x1}R_{31} & s_{x1}f_1R_{12} + o_{x1}R_{32} & s_{x1}f_1R_{13} + o_{x1}R_{33} \\ s_{y1}f_1R_{21} + o_{x2}R_{31} & s_{y1}f_1R_{22} + o_{x2}R_{32} & s_{y1}f_1R_{23} + o_{x2}R_{33} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}\begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix}$$

*Multiplication of the zero column in M_1

Thus we will mark $A = \begin{pmatrix} s_{x1}f_1R_{11} + o_{x1}R_{31} & s_{x1}f_1R_{12} + o_{x1}R_{32} & s_{x1}f_1R_{13} + o_{x1}R_{33} \\ s_{y1}f_1R_{21} + o_{x2}R_{31} & s_{y1}f_1R_{22} + o_{x2}R_{32} & s_{y1}f_1R_{23} + o_{x2}R_{33} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}$.

Denote that $A \in \mathbb{R}^{3\times 3}$ and A is invertible (W.L.O.G, we will assume that that intrinsic M_1 and R_1 are not sparse matrices).

Therefore $\tilde{p} = A\begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix}$.

In the same way we will mark $B \in \mathbb{R}^{3\times 3}$, an invertible matrix, to "replace" the projection matrix M_2 for I_2, where $\tilde{q}$ is the projection of $\tilde{P}$ (by definition – it is corresponding to $\tilde{p}$).

Thus $\tilde{q} = M_2\tilde{P} = B\begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} \Longrightarrow \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} = B^{-1}\tilde{q} \Longrightarrow \tilde{p} = A\begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} = AB^{-1}\tilde{q}$.

Note that $AB^{-1} \in \mathbb{R}^{3\times 3}$. Mark $H = AB^{-1} \Longrightarrow \tilde{p} = H\tilde{q} \Longrightarrow I_1, I_2$ are related by an homography transformation.

5. Consider a video captured by a camera that is attached to the side of a car that moves on a straight road. Assume you have a perfect optical flow algorithm, which is applied to the video.
   What is the expected optical flow (OF) of the projection of the buildings that are parallel to the road with the same distance to the road? Describe its orientation and size, and whether it is fixed for all pixels that are projection of these buildings. Give a short explanation for your answer.

Answer:

The expected optical flow of the projection of the buildings, which are parallel to the road and at the same distance from it, is represented by two vectors for each pixep_0:

$u(p_0) = \frac{\partial x}{\partial t}$ – Represents the horizontal motion of p_0.

$v(p_0) = \frac{\partial y}{\partial t}$ Represents the vertical motion of p_0.

The selection of $p_0{}'s$ depends on the window (patch) size and other considerations we will not discussed here, but we will assume that there are sufficient pixels for each building and the window size is appropriate (for example – the window around a general p_0 is a neighborhood of pixels that are projection of the building only and not beyond its shape).

<u>Orientation:</u>

The motion of the car causes a relative movement of the buildings in the opposite direction from the car's trajectory. As the car moves on a straight road parallel to the buildings, the relative motion of the buildings is purely horizontal; therefore, there is only a change in the x-direction ($\frac{\partial x}{\partial t} \neq 0$) with no vertical movement ($\frac{\partial y}{\partial t} = 0$).

The optical flow of the projection of buildings is expected to be a horizontal for all pixels. $\forall p_0$ that are part of a building, $u(p_0)$ will be determined by the car's movement. The magnitude of this vector, which will be described in the size section, is in the opposite direction of the car's travel. Since the movement is horizontal only (with no vertical component), $v(p_0) = 0\ \forall p_0$ that are part of a building.

<u>Size:</u>

The optical flow the projection of objects depends on the object's distance from the camera. For two objects with the same 3D motion but at different distances from the camera, the optical flow of their projection will differ. Assuming all the building are at the same distance from the road, and thus in the same distance from the car and the camera, the magnitude of $u(p_0)$, $\forall p_0$ that are part of a building, is solely a function of the car's velocity. If the car's velocity is constant - $u(p_0)$ is constant $\forall p_0$. Nevertheless, As previously stated, $v(p_0) = 0\ \forall p_0$ as there is no vertical movement.

6. Assume the OF we learned in class is applied once to a pair of successive frames and once to frames that are 20 frames apart using the same set of parameters. You may assume that the camera motion is constant, and that the scene is static.

   (a) On which regions the computation of the OF is expected to fail in both cases? Give a short explanation to your answer, including algebraic justification.

   (b) Where the OF is expected to fail only for the 20 frames apart case? Explain your answer and suggest a method to overcome this failure.

Answer:

a. The optical flow algorithm is based on intensity changes of pixels relatively to their surrounding pixels. In the given problem, the scene is static and the camera is moving. In such a case, regions that might cause the algorithm to fail, are regions where there is not enough intensity change among pixels in the region. 2 examples:

- "Blank wall problem" - If the region doesn't have enough texture, meaning the color of the region has a permanent color among all it's pixels, $\forall p_0$ and $\forall p_i \in w(p_0)$, the gradient are close to zero $\rightarrow I_x(p_i) \approx 0, I_y(p_i) \approx 0 \rightarrow$ the C matrix will be sparse:

$$C = A^T A = \begin{pmatrix} \Sigma I_x^2(p_i) & \Sigma I_x(p_i) I_y(p_i) \\ \Sigma I_x(p_i) I_y(p_i) & \Sigma I_y^2(p_i) \end{pmatrix} \approx \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \rightarrow rank(C) = 0$$

- "Aperture problem" – a bit more complexed issue than the "blank wall problem", if there is a region which includes texture of a line (not necessarily horizontal), one of the following can happen:

   i. Horizontal or Vertical texture: one of the gradients will be close to zero $I_x(p_i) \approx 0$ or $I_y(p_i) \approx 0$, thus W.L.O.G:

$$C = A^T A = \begin{pmatrix} \Sigma I_x^2(p_i) & \Sigma I_x(p_i) I_y(p_i) \\ \Sigma I_x(p_i) I_y(p_i) & \Sigma I_y^2(p_i) \end{pmatrix} \approx^* \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \rightarrow rank(C) = 1$$

   ii. Diagonal texture: the gradients will be dependent on each other i.e. $I_x(p_i) \approx c \cdot I_y(p_i)$ for some constant c, thus W.L.O.G:

$$C = A^T A = \begin{pmatrix} \Sigma I_x^2(p_i) & \Sigma I_x(p_i) I_y(p_i) \\ \Sigma I_x(p_i) I_y(p_i) & \Sigma I_y^2(p_i) \end{pmatrix} \approx \begin{pmatrix} c^2 \cdot \Sigma I_y^2(p_i) & \text{c} \cdot \Sigma I_y^2(p_i) \\ \text{c} \cdot \Sigma I_y^2(p_i) & \Sigma I_y^2(p_i) \end{pmatrix} \approx^* \begin{pmatrix} 0 & 0 \\ c & 1 \end{pmatrix}$$

$\rightarrow rank(C) = 1$

*Including ranking the matrices

In both cases, C is not invertible, thus $A^+ = C^{-1} A^T$ is not computable, therefore $\begin{pmatrix} u(p_0) \\ v(p_0) \end{pmatrix} = A^+ b$ is not computable $\forall p_0$ in this region.

b. The optical flow algorithm relies on the assumption that the motion of an object is small (relatively). In our case, where one pair are two successive frames and the other pair are frames that are 20 frames apart, an object can move in a significant motion, such that for two successive frames the algorithm "capture" the motion (it is still small enough), but in the 20 frames pair motion is 20 times bigger, causing the algorithm fails. We will prove how a large motion can cause the algorithm to fail:
According to Taylor Series for a pixel $p = (p_x, p_y)$:
$I(p_x + dx, p_y + dy, t + dt) = I(p_x, p_y, t) + \frac{\partial I}{\partial x}(p)dx + \frac{\partial I}{\partial y}(p)dy + \frac{\partial I}{\partial t}(p)dt + \frac{\partial^2 I}{\partial x^2}(p)dx + \frac{\partial^2 I}{\partial y^2}(p)dy + \frac{\partial^2 I}{\partial t^2}(p)dt + \cdots$

The Bright constancy assumption assumes 2 assumptions:
1. $I(p_x + dx, p_y + dy, t + dt) = I(p_x, p_y, t)$
2. $I(p_x + dx, p_y + dy, t + dt)$ can be calculated directly by the first order of the Taylor series, because the rest of the orders are negligible:

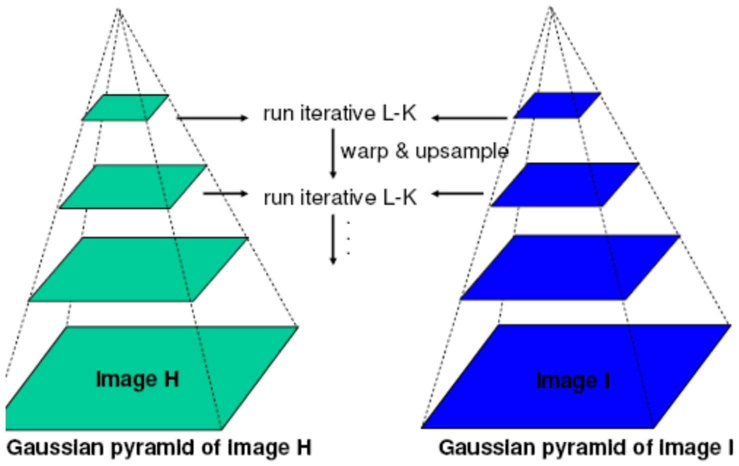
$I(p_x, p_y, t) = I(p_x + dx, p_y + dy, t + dt) = I(p_x, p_y, t) + \frac{\partial I}{\partial x}(p)dx + \frac{\partial I}{\partial y}(p)dy + \frac{\partial I}{\partial t}(p)dt$
$\rightarrow \frac{\partial I}{\partial x}(p)dx + \frac{\partial I}{\partial y}(p)dy + \frac{\partial I}{\partial t}(p)dt = 0 \rightarrow I_x(p)dx + I_y(p)dy = -I_t(p)dt$
In the case of a significant motion, at the successive pair of frames the rest of the order of the Taylor Serie might be negligible, but no so in the 20 frames pair. Thus:

$$\frac{\partial^2 I}{\partial x^2}(p)dx + \frac{\partial^2 I}{\partial y^2}(p)dy + \frac{\partial^2 I}{\partial t^2}(p)dt + \cdots > 0 \rightarrow$$

$$I(p_x, p_y, t) = I(p_x + dx, p_y + dy, t + dt)$$
$$= I(p_x, p_y, t) + \frac{\partial I}{\partial x}(p)dx + \frac{\partial I}{\partial y}(p)dy + \frac{\partial I}{\partial t}(p)dt + \frac{\partial^2 I}{\partial x^2}(p)dx + \frac{\partial^2 I}{\partial y^2}(p)dy + \frac{\partial^2 I}{\partial t^2}(p)dt + \cdots$$

$\rightarrow \frac{\partial I}{\partial x}(p)dx + \frac{\partial I}{\partial y}(p)dy + \frac{\partial I}{\partial t}(p)dt + \frac{\partial^2 I}{\partial x^2}(p)dx + \frac{\partial^2 I}{\partial y^2}(p)dy + \frac{\partial^2 I}{\partial t^2}(p)dt + \cdots = 0 \rightarrow$
$I_x(p)dx + I_y(p)dy \neq -I_t(p)dt$

In order to solve the above problem, we can use a pyramid of optical flow:
Resize the image to a smaller size that is small enough to compute OF. Calculate the new "resized" motion vectors using OF and resize back to the original size. During the iterations of the resizing, "inject" the original image in each layer of iteration and run OF again in the current resized image. Every iteration after computing OF, use warping to extend the image to "next" size until the image reaches the original size:


run iterative L-K
warp & upsample
run iterative L-K
Image H
Image I
Gaussian pyramid of image H
Gaussian pyramid of image I

7. The basic change detection algorithm we learned in class, computes a single image as a background model using a median image. The algorithm has several parameters.

   (a) List the set of parameters.

   (b) Assume that the intensity values of the pixel p at frame i is given by $p[i]$, and $p = [10, 10, 100, 10, 202, 10, 30, 205, 201, 200, 201]$. Using one set of parameters, the value $p(11) = 201$ was detected as background while using another set of parameters, the value $p(11) = 201$ was detected as foreground. Give a single parameter of the algorithm such that changing its value can explain this difference. Give a short explanation for your answer including the list of all parameters in the two cases.

Answer:

a. List of parameters of the basic change detection algorithm learned in class:
   a. n – sequence of n frames (images).
   b. k – Taking the median of the last k frames
   c. S – update every S frames
   d. Th – threshold that determent if a pixel is a foreground or background: $d_k(p) = |I_B(p) - I_k(p)|, |d_k(p)| > th$ so p is moving

b. The parameter that could explain the difference in detection is the **Threshold (Th)**. This value determines how different a pixel's intensity must be from the background model to be considered foreground. If the threshold is set low, slight changes from the median can be considered foreground, whereas a higher threshold might require more significant changes to classify a pixel as foreground.
   a. First case:
      i. intensity values of the pixel p[i] at frame i is given by p[i] (as in the question) = [10,10,100,10,202,10,30,205,201,200,201]
      ii. $k = 11$
      iii. $median = 100$
         $sorted\ array - [10, 10, 10, 10, 30, \mathbf{100}, 200, 201, 201, 202, 205]$
      iv. $th = 50$
      v. in this case $p\,[11] = 201$ is consider foreground because:
         1. $d_k(p) = |I_B(p) - I_k(p)| = |100 - 201| = 101$
         2. $101 > 50 = th$
      vi. For the same case but $th = 190$ the same pixel p[11] = 201 where consider as background because –
         1. $d_k(p) = |I_B(p) - I_k(p)| = |100 - 201| = 101$
         2. $101 < 190 = th$

8. Suggest a change detection algorithm where the background value of each pixel is based on a patch around the pixel rather than the intensity values of the pixel. List the set of parameters of your algorithm. Discuss the pros and cons of using a patch rather than a pixel to model the pixel background value.

Answer:

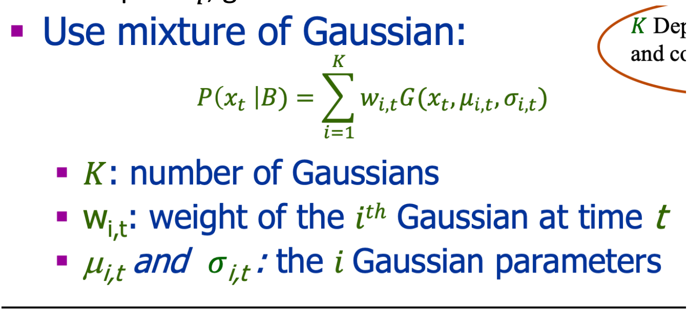
In class we have learned an algorithm of change detection based on the intensity values of each pixel. We saw a few methods, mostly statistics methods, of how to determine if a current pixel is a background pixel, based on its intensity value changing across a series of frames over time. The problem of relying solely on the pixel's intensity, is that the algorithm is exposed to noise and can assign a pixel as foreground because of noise in a single/few frames (for example shadows), assuming that a change in a static frame will be in more than a single pixel. We considered two ways to implement an algorithm of change detection based on a patch around a pixel:

1. "Smooth" every frame using a convolution mask (e.g. Gaussian mask).
2. In the parametric model, label a pixel as background/foreground based on a mask of PMF's (and not only the pixel's PMF's).

The first option seems to be the "obvious" solution, so we decided to show the second. We will implement with a GMM modeling of the background:

<u>Algorithm:</u>

1. For each pixel q, generate a GMM model as shown in class:



- Use mixture of Gaussian:

$$P(x_t \mid B) = \sum_{i=1}^{K} w_{i,t} G(x_t, \mu_{i,t}, \sigma_{i,t})$$

  - K: number of Gaussians
  - $w_{i,t}$: weight of the i^{th} Gaussian at time t
  - $\mu_{i,t}$ *and* $\sigma_{i,t}$: the i Gaussian parameters

K Dep
and co

2. Update the weights and the Gaussians as shown in class (class 9 slides 16,17).
3. For each pixel q_i in the patch around q:
   a. $x_{ti} = I(q_i, t)$ for a frame t.
   b. $\forall G_{ji} \in B$, where B is the set of dominant gaussians according to section 2:
      i. $M(x_{ti}, G_{ji}) = \begin{cases} 1 & |x_{ti} - \mu_{jti}| < 2.5\sigma_{jti} \\ 0 & |x_{ti} - \mu_{jti}| \geq 2.5\sigma_{jti} \end{cases}$
   c. $b_{ti} = \begin{cases} 1 & \exists G_{ji}\ s,t\ M(x_{ti}, G_{ji}) = 1 \\ 0 & else \end{cases}$ \\ a Boolean variable indicating whether q_i is assigned as background pixel
4. $b_q = Mask(b_{ti}\ values)$ (b_{ti} are the Boolean variables of the patch around q).
5. If $b_q > Th$ - assign q as background pixel
6. Else, q is foreground.

Explanation:
For any pixel q, the algorithm assigns it as a background pixel only if neighborhood "supports" it. The decision is based on the values obtained after applying the mask to q's neighborhood. For example:

$$Mask = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}, patch\ b_{ti}\ values = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, Th = 10$$

$$Mask * patch = \begin{pmatrix} 9 & 9 & 3 \\ 11 & 11 & 5 \\ 7 & 7 & 5 \end{pmatrix}$$

We are interested in the value in the middle (represents the masked value of q): $11 > 10 \rightarrow q$ is a background pixel.

In addition, we considered whether to mask the b_{ti} values for a pixel q only if $b_{tq} = 1$ (meaning – only if q is determined as a background pixel in the pixel version of the algorithm as shown in class), but we decided that the noise can be opposite as well – a background pixel might not be determined as a background while all its neighborhood pixels are (for example a shadow over a "green" background as shown in class.

Parameters:
K – number of Gaussians in each model
$w_{it}, \mu_{it}, \sigma_{it}$ – the initialized parameters of the gaussian for each pixel in each frame
Mask – a convolution mask. Includes the size and weights (defaulted to a gaussian mask).
Th – threshold for a background pixel (according to the mask values)
α – learning rate for the GMM.

Pros:
1. **Robustness**: as the same for the other algorithms we learned in class, when determine a pixel as a background/foreground based only on its own value we are exposed to outliers because of noise or other issues. Generally, a change in a statics image will be of an object that is bigger than a pixel, thus a change of a pixel will be together with its neighborhood.
2. **Flexibility**: By fine-tuning the mask size and threshold parameters, we can tailor the algorithm's sensitivity to regions/frames characterized by different levels of activity. In quieter areas where movement is minimal, a smaller mask and lower threshold can conserve computational resources while maintaining accuracy. Conversely, in regions bustling with motion, employing a larger mask and a higher threshold can ensure the detection of subtle yet significant changes, enhancing the algorithm's precision amidst the complexity of the dynamic background.

Cons:

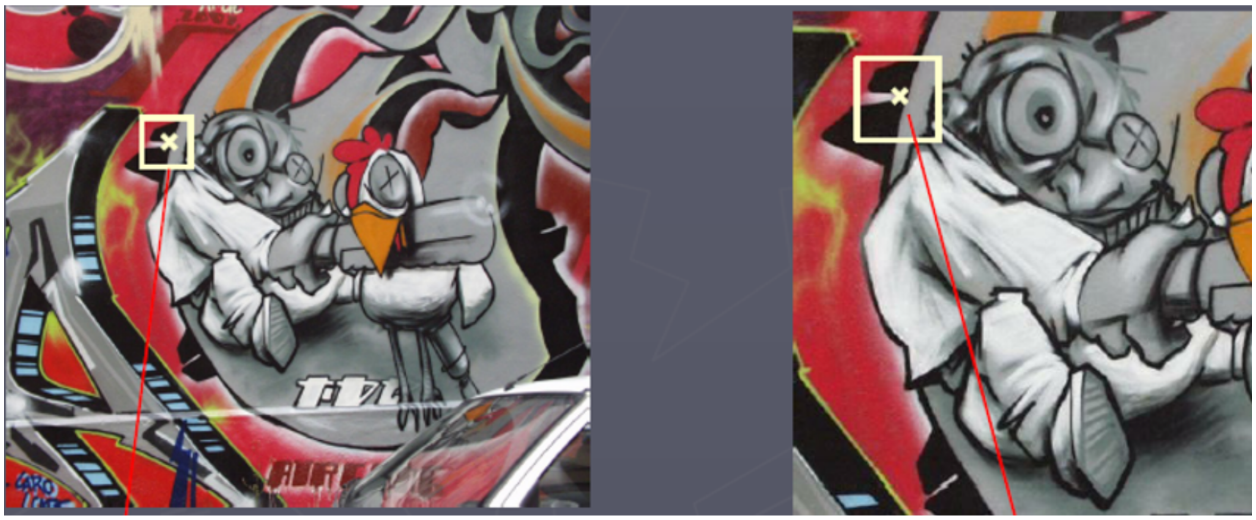
1. **Computational complexity**: even though we can compute for each pixel its GMM as same as the single pixel algorithm, and then use it when necessary as part of a patch, we still have additional computations: the convolutions for each pixel, the Booleans variables. Nevertheless, the next version of the algorithm can be to determine "areas" of background according to the convolutions. Meaning, instead of computing the convolution for each pixel, we can determine a patch as foreground/background according to the number of zeros/ones in the $patch\ b_{ti}\ values$ matrix. In the second next version, we can increase the size of the patch and decide whether an area is background according to the rank of the $patch\ b_{ti}\ values$ matrix (sparse matrix → low rank → foreground area).
2. **Additional parameters**: we added few additional parameters to the single pixel algorithm: Mask and Th. Both of them are depended on the given images (resolution and intensity values). Wrong values will cause false output. In addition, like most of the parameters, they should be optimized by parameters tuning, which consumes additional Computational complexity and time.

9. Assume that two images of the same object is captured by two cameras with the same internal parameters, but from a different distance. Assume that in order to match feature points, the Harris corner detector is applied to the two images. Which parameter of the corner detector should be modified in order to obtain corresponding corners? Give a short explanation to your answer.

Answer:
When capturing images of the same object from different distances, the scale of the features within the images changes. The Harris corner detector, by default, does not account for scale changes, as it uses a fixed window size to detect corners. To obtain corresponding corners in both images, the **main** parameter that should be modified is the **window size**.

Window Size parameter determines the size of the region around each pixel used to calculate the corner response. If the object appears larger in one image (the image taken from a closer distance), a larger window size should be used to ensure that the corners detected in this image are comparable to those detected in the image taken from further away, which contains smaller-looking features. Conversely, if the object appears smaller, a smaller window size might be necessary to detect finer details.

By modifying the window size, you effectively adjust the detector's sensitivity to the scale of the features in the image. A larger window size can encompass and detect larger corners that result from being closer to the object, while a smaller window size can detect smaller corners resulting from being further away. This adjustment helps in identifying corners that correspond to the same physical points on the object, despite the difference in distance from which the images were taken.



Another parameter that can be changed is **Gaussian Filter Scale σ_w:**

- **A larger σ_w results** in greater smoothing, which can help in reducing noise and making the algorithm less sensitive to small, irrelevant details. This can be useful when the object is closer and appears larger, as it can help to filter out fine-scale corners and focus on more significant features that correspond better across different scales.
- **A smaller σ_w** results in less smoothing, retaining more of the image's fine details. This would be beneficial when the object is further away and appears smaller in the image, as it allows the detection of smaller-scale corners.

Adjusting the Gaussian filter scale σ_w can be seen as a way to make the Harris corner detector more adaptive to scale changes, similar to changing the window size. It can help in detecting corners that are consistent across images taken from different distances by matching the scale of smoothing to the scale of the features in the image.

10. Assume that you are given the projections of the 3D points $\{P_i\}_{i=1}^k$ onto two images that were captured from different locations. Let $\{p_i\}_{i=1}^k$ and $\{q_i\}_{i=1}^k$ be the sets of these projections. Moreover, you are given that p_i and q_i are corresponding points and $k > 50$. Suggest a method to test whether $\{P_i\}_{i=1}^k$ are located on a single plane. If your method requires parameters, list them and explain how they affect the results.

Answer:

To verify if the given set of 3D points $\{P_i\}_{i=1}^k$ are located on a single plane, we propose a two-step method:

1. Homography transformation:
   a. Compute the homography matrix $H \in \mathbb{R}^{3\times3}$ s.t $\tilde{q}_i = H\tilde{p}_i$ using RANSAC with at least four different corresponding points $p_i \in \{p_i\}_{i=1}^k, q_i \in \{q_i\}_{i=1}^k$.
   b. Within the RANSAC framework, confirm the presence of a homography transformation across additional corresponding points:
      i. For each p_i calculate $\tilde{q}'_i = H\tilde{p}_i$.
      ii. For all q_i, check that $MSE(q_i, q_i') < \epsilon$, where ϵ is a predefined threshold.

2. Plane equation validation:
   If all pairs of corresponding points maintain the homography transformation, this indicates that either all 3D points lie on a planar surface or the two images were captured by cameras with identical Centers of Projection (COP) (solely involving pure rotation without translation).
   Given that true pure rotation (no translation) is unrealistic, it is unlikely for a homography transformation to hold with a large number of corresponding points (k > 50) if there is any translation since even minor translations would lead to discrepancies that violate the homography.
   Nevertheless, in the 'single plane' scenario, even if the planar surface is not 'pure,' the bounded error would still allow the entire set of corresponding points to conform to the homography transformation.
   Even so, we will verify that all 3d points ($\forall P_i$) are coplanar:
   a. Compute coefficients A, B, C, D using RANSAC with at least four 3D points, s.t for each chosen 3D point $P_i = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix}$ the following equation is satisfied:
   $$AP_x + BP_y + CP_z + D = 0$$
   b. For all the additional 3D points $P_j = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix}$, check that:
   $|AP_x + BP_y + CP_z + D| < \delta$, where δ is an input threshold.

RANSAC parameters: Number of iterations, threshold, probability, distance metric (sample size, model and consensus are given by the method).

Additional Parameters:
Assuming the planar surface is not 'pure', which means it is not perfectly 2D but has some depth (such as boards, screens, and other planar objects), we have introduced two parameters to bound the error for the allowable 'depth' of the planar surface:
$\boldsymbol{\epsilon}$: A threshold for the homography transformation. Given a computed corresponding point $q_i{}'$ $(\tilde{q}'_i = H\tilde{p}_i)$, which is assumed to be on the same plane as p_i, ensure that the distance from the actual corresponding point q_i is bounded by ϵ distance (using MSE).
$\boldsymbol{\delta}$: A threshold for the distance of 3D points from the plane computed by RANSAC. Given that may not perfectly conform the plane equation, this bounds the permissible deviation by δ.
Both parameters can be identical to the RANSAC threshold parameter (although not obliged to).

This method, while allowing for minor deviations from planarity due to real-world imperfections, should effectively distinguish between a set of points on a plane and those that are not.