Computer vision

# *Assignment 3 – Stereo Vision*

*Names*


Mostafa Ahmed Abd El Hameed

18011774

Omar Ashraf

18011111

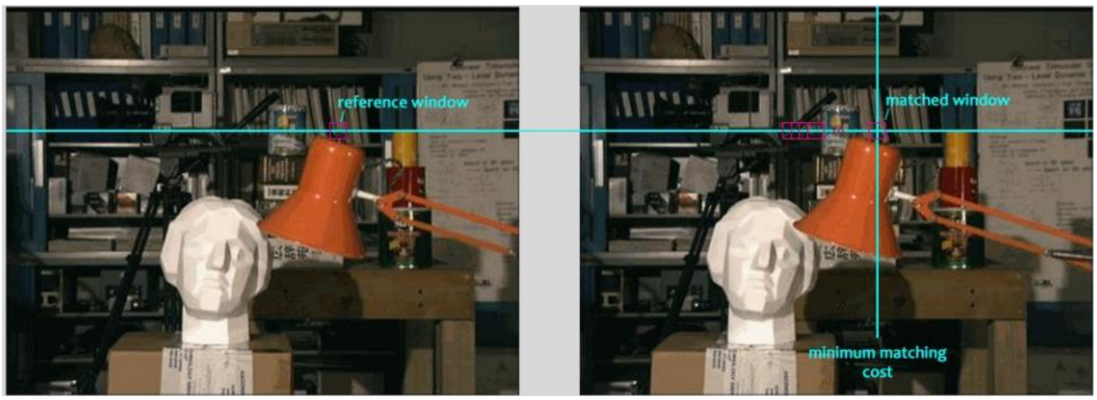
Mohamed Gomma

18011425

# *First part : block matching*

In this method we find the disparity in the images by matching every pixel in the left image to a certain pixel in the right image.

- First, we define a reference window in the left image, then we slide a similar window across the epipolar line in the right image and compare the contents with the reference window using a certain metric. This process is repeated for every pixel in each scanline.




- We compare the results of window sizes 1 (pixel to pixel comparison), 5, and 9.

```python
def block_matching(img1, img2, window_size, metric):

  height, width = img1.shape
  disparity_map = np.zeros((height, width), np.uint8)
  half_window = window_size // 2

  # Loop through every pixel in the left image
  for y in range(half_window, height - half_window):
    for x in range(half_window, width - half_window):

      # Get the current window
      window = img1[y - half_window: y + half_window + 1, x - half_window: x + half_window + 1]
      # Define the minimum SSD and best disparity
      min_metric = 255 * window_size * window_size
      # min_metric = 1000000000000
      best_disparity = 0

      # compare with each window from the right image
      for xRight in range(half_window, width - half_window):
        # Get the current window from the right image
        current_window = img2[y - half_window: y + half_window + 1, xRight - half_window: xRight + half_window + 1]
        # Calculate the SSD / SAD
        m = metric(window, current_window)
        # if the SSD / SAD is smaller than the minimum metric update the minimum mertric and the best disparity
        if m < min_metric:
          min_metric = m
          best_disparity = np.abs(x-xRight)

      # Update the disparity map
      disparity_map[y, x] = best_disparity

  return disparity_map
```

- We use 2 metrics to estimate matching cost:
  - Sum of Absolute Differences (SAD) Computed by calculating the absolute difference of the reference window and matching window's pixels element by element then adding them up.
  - Sum of Squared Differences (SSD) Computed by calculating the squared difference of the reference window and matching window's pixels element by element then adding them up.

*Cons of using this method*

1. This method is taking so long as it goes O(ROWS*COLUMNS) * O(COLUMS)
2. We can reduce the complexity by having a limit on the disparity so we can limit the number of comparisons for each window.

## *Second part : Dynamic Programming*

- This method uses a dynamic programming algorithm to calculate the minimum cost for matching a whole row in the images.

For every pixel in the image there are 3 possible conditions:

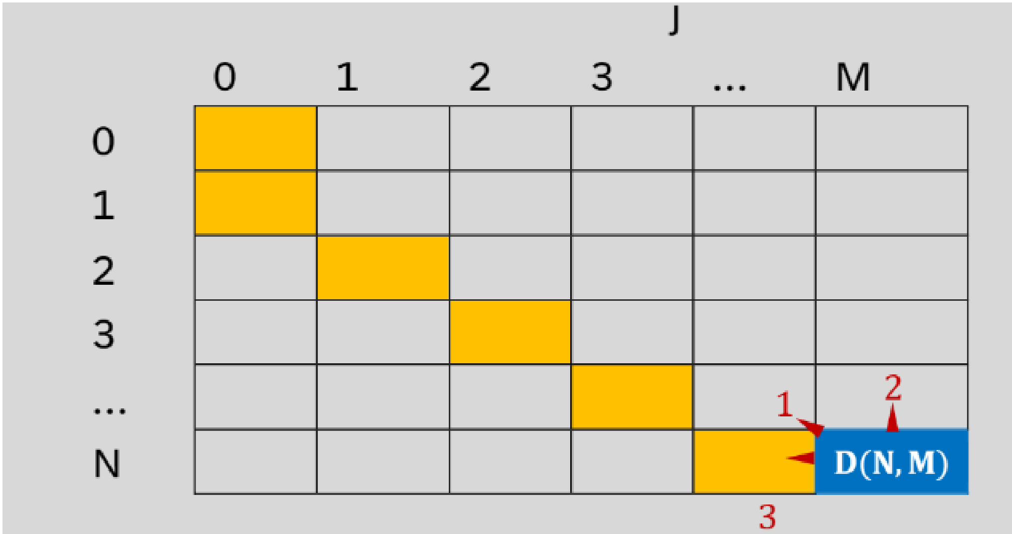
- The pixel is visible in the left and right images (it is matched).
- The pixel is visible in the left image but occluded in the right image.
- The pixel is visible in the right image but occluded in the left image.

$$d_{ij} = \frac{(I_l(i) - I_r(j))^2}{\sigma^2}$$

where $\sigma$ is some measure of pixel noise. The cost of skipping a pixel (in either scanline) is given by a constant $c_0$ . For the experiments here we will use $\sigma = 2$ and $c_0 = 1$. Given these costs, we can compute the optimal (minimal cost) alignment of two scanlines recursively as follows:

1. $D(1,1) = d_{11}$
2. $D(i,j) = min(D(i-1,j-1) + d_{ij}, D(i-1,j) + c_0, D(i,j-1) + c_0)$

- After D is fully computed, the total cost of matching two scanlines can be found in the element $D(\mathrm{N},\mathrm{M})$.
- We now perform a backwards pass from $D(N, M)$ along the optimal path (minimum cost) to get the optimal alignment for the 2 images and calculate the disparity of each pixel in the corresponding pixel maps.




Starting at (i, j) = (N,N)

*we choose the minimum value of D from (i-1, j-1), (i-1, j), (i, j- 1).*

- Selecting (i-1, j)corresponds to skipping a pixel in Il, so the left disparity map of i is zero.
- Selecting (i, j − 1)corresponds to skipping a pixel in Ir, and the right disparity map of j is zero.
- Selecting (i-1, j-1) matches pixels (i, j), and therefore both disparity maps at this position are set to the absolute difference between i and j.

Sample output