COMP37212 Lab3 Report

Rui Xu

Abstract

This lab investigates various aspects of computer vision, with a specific focus on stereo imagery processing. It encompasses techniques such as focal length calculation, disparity map generation, and depth estimation within a scene. Additionally, the concept of selective focus is explored, highlighting its significance in image analysis and manipulation.

1.1 Focal Length Calculation

In this part, A supplied image pair of photography umbrellas is used. To calculate the focal length of the cameras in millimeters, we need to first convert the focal length from pixels to millimeters using the physical sensor size and the image resolution. Then, we can use the formula:

$$focallength(mm) = focallength(pixels) \times \frac{sensorwidth(mm)}{imagewidth(pixels)}$$

According to the formula and the description of the coursework introduction, the focal length of the camera is 41.744 mm.

1.2 Disparity Map

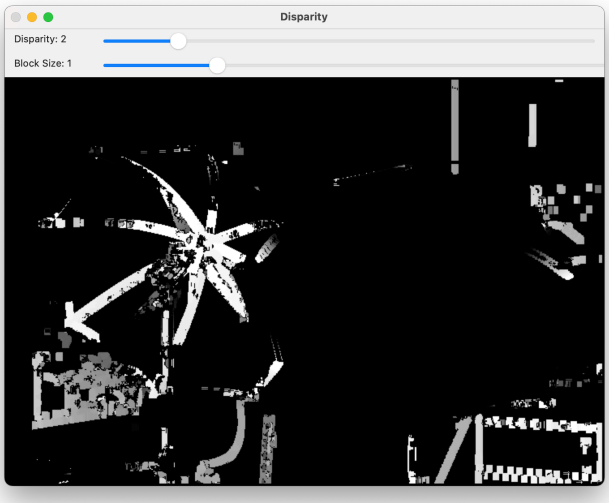
The `getDisparityMap` function is used to compute the disparity map of a scene given two images: a left image and a right image. The disparity map is a representation of the depth information of the scene, which is obtained by comparing the two images. The function uses “StereoBM_create” method from the OpenCV library to create a StereoBM (Block Matching) object with the specified number of disparity and block size. The disparity map is then computed by calling the `compute` method on the StereoBM object, passing in the left and right images. The function returns the disparity map as a floating-point image, and this returned disparity map can be used to analyze the depth information of the scene.

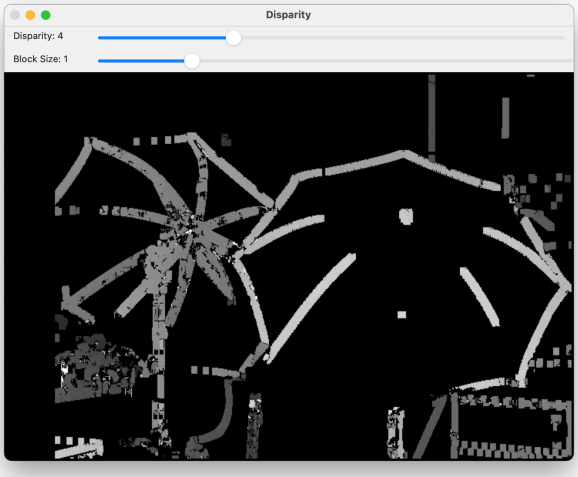
A trackbar is added so that we can move left or right to select a value within a specified range. In this context, the trackbars are used to adjust the parameters of the disparity map computation in real-time. The number of Disparity parameter controls the number of disparities that the block matching algorithm should consider, and the Block Size parameter controls the size of the blocks that the algorithm compares between the two images. In order to make better use of the window space, I scaled and organized the trackbar, so the actual value of disparity is 16 multiplied “Disparity” shown on the trackbar window. And the actual value of “Block Size” is 5 multiplied by the disparity shown on the trackbar window and then plus 5 (To make sure SADWindowSize must be odd, be within 5 to 255). In other words, If the disparity value on the trackbar is 4, the real number of disparity is 4 * 16 = 64, if the block size value on the trackbar is 20, the actual value should be 20 * 2 + 5 = 45.

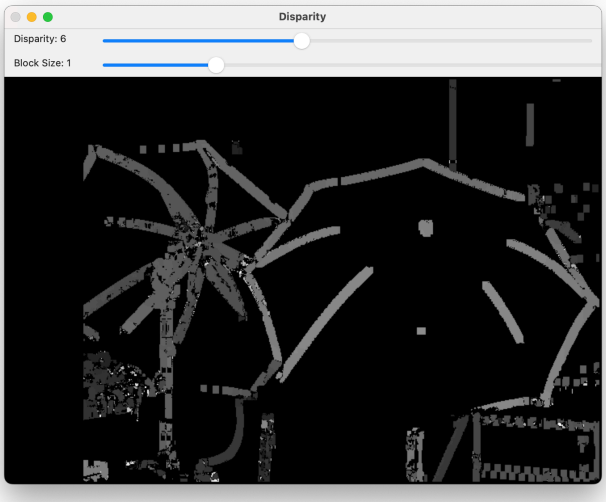
Conclusion and Discussion

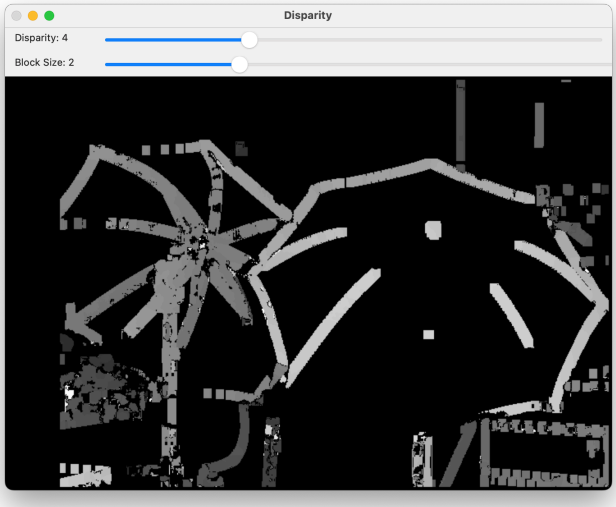
From the images below, we can observe the it’s necessary balance between Disparity and block size to achieve optimal image effects. An intriguing phenomenon is that higher disparity may lead to a decrease in image brightness, while larger block sizes can cause lines in the image to thicken.

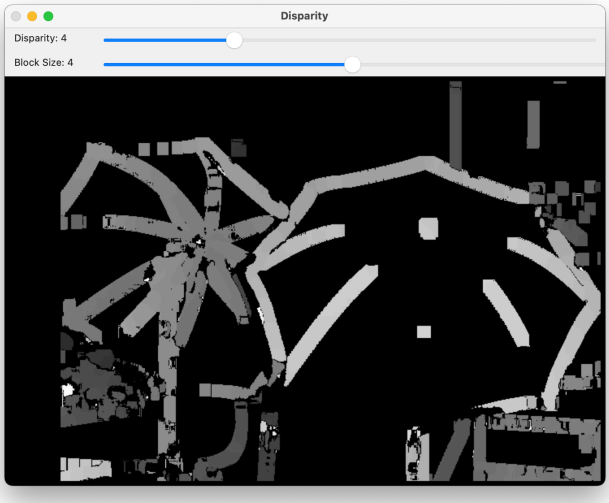
Images of the whole window with the trackbars in different positions

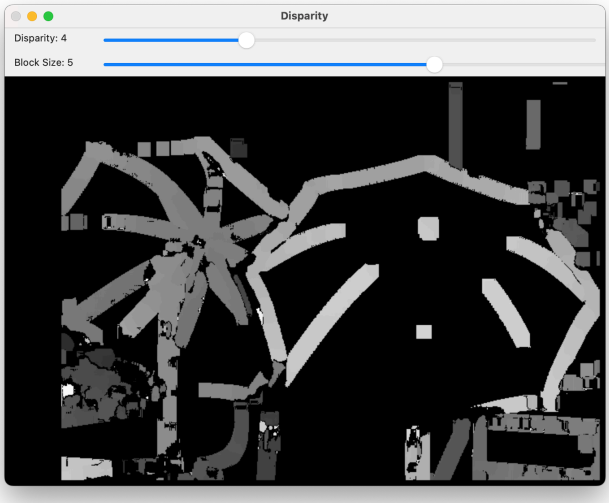












1.3 Views of the Scene

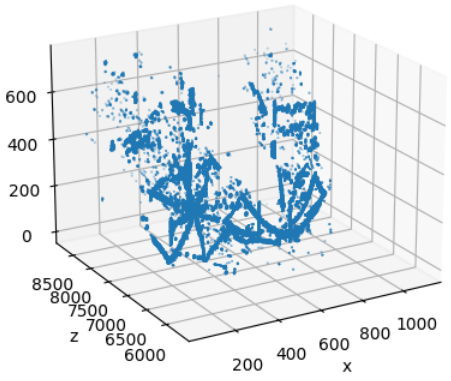
The “calcDepthMap” function calculates the depth map of a scene given its disparity map. The depth map represents the distance from the camera to each pixel in the scene.

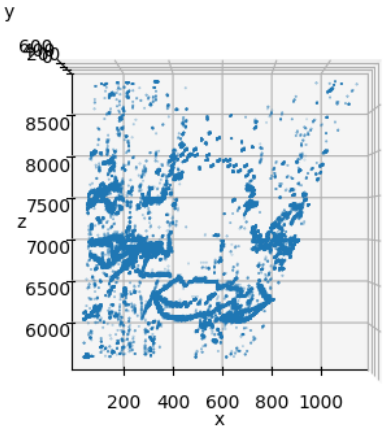
$$Z = \text{baseline} \times \frac{f}{d + \text{doffs}}$$

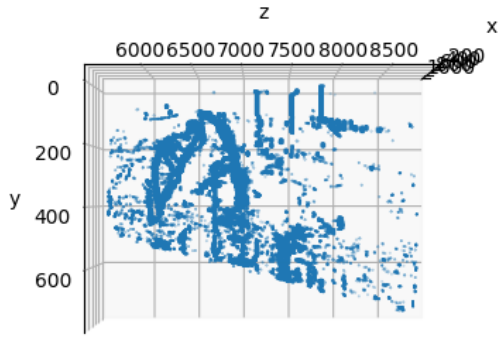
The formula is derived from the principle of triangulation, which is the basis for stereo vision. The depth of a point in the scene is inversely proportional to its disparity: points with high disparity are close to the camera, while points with low disparity are far from the camera.

The plot function visualizes the depth map of the scene in 3D. It takes as input the disparity map calculated by the “calcDepthMap” function. Three plots are shown below.

3D, top and side view of umbrellas scene







Conclusion and Discussion

In general, these three images all depict the approximate shape of the umbrella, whether from the top, side, or in a 3D rendering, effectively facilitating the 3D reconstruction of the scene.

2 Selective Focus

In the realm of selective focus, depth estimation plays an important role in delineating focused and blurred regions within a scene. Depth can be derived from the disparity map, representing positional discrepancies between left and right camera views. The depth approximation formula utilizes the disparity map to estimate the depth of objects. It inversely relates the disparity to the depth, where smaller disparities correspond to larger depths and vice versa. The parameter k serves as a modulator influencing the depth calculation.

$$\text{depth} = \frac{1}{\text{disparity} + k}$$

Moreover, incorporating a trackbar allows for real-time adjustment of K and Foreground thresholds, facilitating interactive exploration of focus manipulation. Similar to Section 1.1, we also scaled and reorganized the trackbar to make better use of the window space.

A background blur function in this section applies a blur effect to the background of the image based on the depth map. This effect blurs out the background while keeping the foreground in focus, similar to what a camera does when it focuses on a specific object. The depth threshold is used to distinguish between the

foreground and the background. The function first calculates a normalized depth map where the values range from 0 to 1. Then, it applies a Gaussian blur to the input image. The blurred image and the original image are then combined using the normalized depth map as a weight: areas with a high depth value (corresponding to the background) are taken from the blurred image, while areas with a low depth value (corresponding to the foreground) are taken from the original image. The result is a greyscale image where the background is heavily blurred, creating a depth-of-field effect.

Conclusion and Discussion

By adjusting the appropriate K value and foreground value, we can almost perfectly filter out the model's body shape. In addition, the current sample image does not seem to fluctuate significantly due to changes in the foreground threshold, and requires large numerical changes to affect its effect. At the same time, Gaussian blur can effectively filter out background noise.

Disparity 2
Block Size 10
K value 3
Gaussian Size 0
Foreground 20

Disparity 2
Block Size 10
K value 3
Gaussian Size 10
Foreground 75

Disparity 2
Block Size 10
K value 3
Gaussian Size 10
Foreground 20

Appendix

Disparity map function

```
# ================================================
# 1.2 Disparity Map
def getDisparityMap(imL, imR, numDisparities, blockSize):
    """
    Get the disparity map of the scene

    imL: the left image
    imR: the right image
    numDisparities: the maximum disparity minus minimum disparity
    blockSize: the size of the window used to match pixels
    """
    # Ensure blockSize is odd, within 5..255, and not larger than image width or height
    # blockSize = max(5, min(255, blockSize))
    # if blockSize % 2 == 0:
    #     blockSize += 1
    # blockSize = min(blockSize, imL.shape[1], imL.shape[0])

    stereo = cv2.StereoBM_create(numDisparities=numDisparities, blockSize=blockSize)

    disparity = stereo.compute(imL, imR)
    disparity = disparity - disparity.min() + 1 # Add 1 so we don't get a zero depth, later
    disparity = disparity.astype(np.float32) / 16.0 # Map is fixed point int with 4 fractiona

    return disparity # floating point image
#
```

Focal Length Calculation

```
# ================================================
# 1.1 Focal Length Calculation
def calcRealFocalLength(sensor_size, image_size, focal_length):
    """
    Calculate the real focal length of the camera

    The formula is: f = (sensor_size / resolution) * focal_length
    sensor_size: the size of the camera sensor in mm
    image_size: the size of the image in pixels
    focal_length: the focal length of the camera in pixels
    """
    return (sensor_size / image_size) * focal_length

def getEdgeMap(img, threshold1, threshold2):
    """
    Get the edge map of the image using Canny edge detection

    img: the image to detect edges in
    threshold1: the first threshold for the hysteresis procedure
    threshold2: the second threshold for the hysteresis procedure
    """
    imgCanny = cv2.Canny(img, threshold1, threshold2)
    return imgCanny
#
```

Depth Map

```python
# ================================================
#  1.3 Views of the Scene
def calcDepthMap(disparityMap, f=FOCAL_LENGTH_PIXELS, DOFFS=DOFFS, Z=BASELINE):
    """
    Calculate the depth map of the scene

    disparityMap: the disparity map of the scene

    baseLine: the distance between the two cameras in mm
    f: the focal length of the camera in pixels
    d: the disparity map of the scene in pixels
    Z: the depth map of the scene in mm

    The formula is: Z = baseLine * (f / (d + DOFFS))
    """

    Z = BASELINE * (f / (disparityMap + DOFFS))
    return Z
```

Depth Calculation

```python
# ================================================
# 2 Selective Focus
def calcDepth(disparity, k):
    """
    Calculate the depth map from the disparity map
    """
    depth = 1 / (disparity + k)
    return depth

def backgroundBlur(img, depth, gaussian_size=25, depth_threshold=0.2):
    """
    Blur the background of the image based on the depth map

    img: the image to blur
    depth: the depth map of the scene
    gaussian_size: the size of the Gaussian kernel
    depth_threshold: the threshold for the depth map
    """
    foreground_depth = depth.min() + depth_threshold * (depth.max() - depth.min())
    depth_normalized = (depth - depth.min()) / (depth.max() - foreground_depth)
    depth_normalized = np.clip(depth_normalized, 0, 1)
    blurred = cv2.GaussianBlur(img, (gaussian_size, gaussian_size), 0, borderType=cv2.BORDER_REFLECT).astype(np.float64)

    output = blurred * depth_normalized + img * (1 - depth_normalized)
    return output.astype(np.uint8)
```