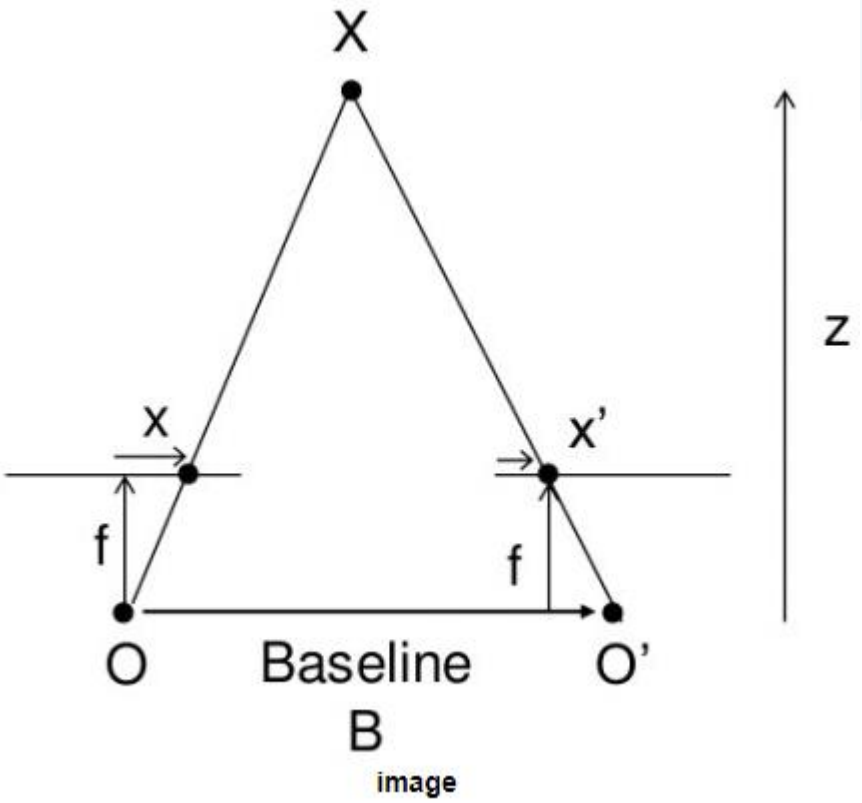
الجلسة الثامنة

Stereo Vision

1.1 Disparity and depth generation

سنتعلم في جلسة اليوم عن كيفية توليد صور العمق Depth باستخدام صورتين ثنائيتي البعد 2D ملتقطتان لنفس المشهد بإزاحة مختلفة.

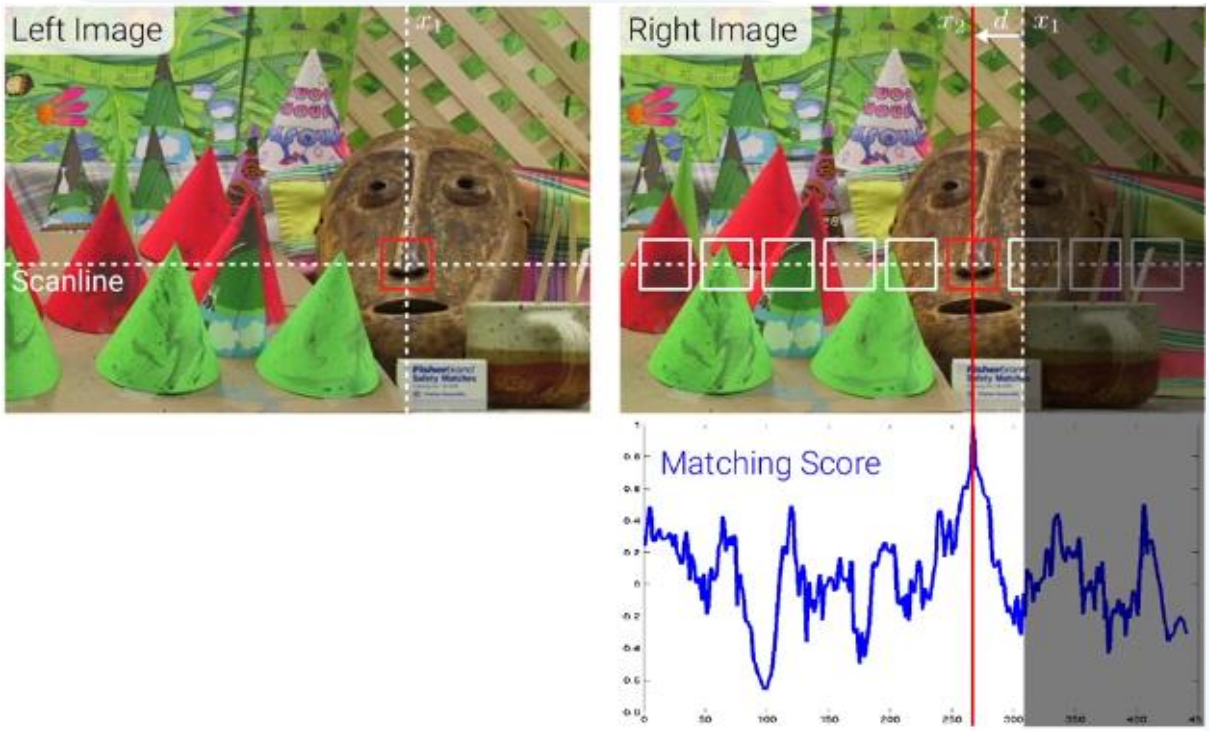
تعتمد فكرة توليد العمق أو النمط ثلاثي البعد على الفكرة الآتية:



حيث O, O' مركز الكاميرا اليمينية واليسارية والنقطة X هي نقطة من المشهد المراد تصويره (نقطة ضمن الفضاء ثلاثي البعد Z).

المستوي الواصل بين X,O,O' هو مستوي Epipolar plane أما البعدين F,F' فيمثلان البعدين البؤريين بين BaseLine ومستويي الصورة اليسارية واليمينية (بالتالي النقطتان X, X' تمثلان البعد الذي تبدأ عنده مكونات الصورة بالوضوح).

سنقوم بحساب Disparity matrix من الصورة اليمينة واليسارية وذلك لإظهار صورة العمق حيث ستبدو المكونات القريبة من الكاميرا بلون فاتح والبعيدة عنها بلون أغمق ما يجعلنا ندرك العمق Depth.



تعتمد خوارزمية حساب العمق على مفهوم BlockMatching حيث يتم أخذ نافذة حول البكسل المدروس بحجم محدد BlockSize ثم يتم دراسة جميع النوافذ المقابلة في الصورة اليمينية والمتواجدة على نفس خط Epiploar line ويتم حساب disparity score بين النافذة المدروسة في الصورة اليسارية وجميع النوافذ المقابلة لها في الصورة اليمينية والنافذة التي تحقق أعلى درجة مطابقة تكون هي النافذة المقابلة. بعدها يتم أخذ عدد مستويات Disparity المرادة فرضاً 16 مستوى (عدد مستويات أكبر يعني دقة أفضل لكن زمن تنفيذ أطول) وضمن نافذة المطابقة يتم حساب Disparity بين البكسل ومقابله في الصورة الثانية وكلما كانت Disparity أعلى كلما كان البكسل أقرب للكاميرا (عمق أقل) وبالتالي يظهر بأعلى مستوى Disparity (أعلى إضاءة).

1.1.1 التنفيذ العملي:

نفذ الكود التالي:

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load the stereo pair of images

left_image = cv2.imread('left.png', 0)  #left image path

right_image = cv2.imread('right.png', 0)  #right image path
```

قراءة الصورتين اليمينية واليسارية بصيغتهما الرمادية

```
print(left_image.shape)
print(right_image.shape)
left_image=cv2.resize(left_image,(400,300 ))
right_image=cv2.resize(right_image,(400,300 ))
```

تحجيم الصورتين لحجم واحد ومحدد 300x400

```
# Create a StereoBM object
stereo = cv2.StereoBM_create(numDisparities =16,blockSize = 15)
```

إنشاء كائن StereoBM بپارامترين numDisparities بقيمة 16 وblockSize بقيمة 15 يمثل هذا الكائن تطبيق خوارزمية Stereo Block Matching لحساب العمق بين صورتين يمينية ويسارية. يمثل بارامتر numDisparities عدد مستويات الفرق Disparity levels (الفرق مقصود به الفرق في الإحداثيات بين نقطتين في الصورة اليسار والصورة اليمين)، بارامتر blockSize يمثل حجم البلوك او عدد المجاورات المأخوذة بعين الاعتبار لدى دراسة البكسل الواحد

```
# Compute the disparity map
disparity_map = stereo.compute(left_image, right_image)
```

حساب مصفوفة disparity map (صورة العمق) بين الصورتين اليسارية واليمينية

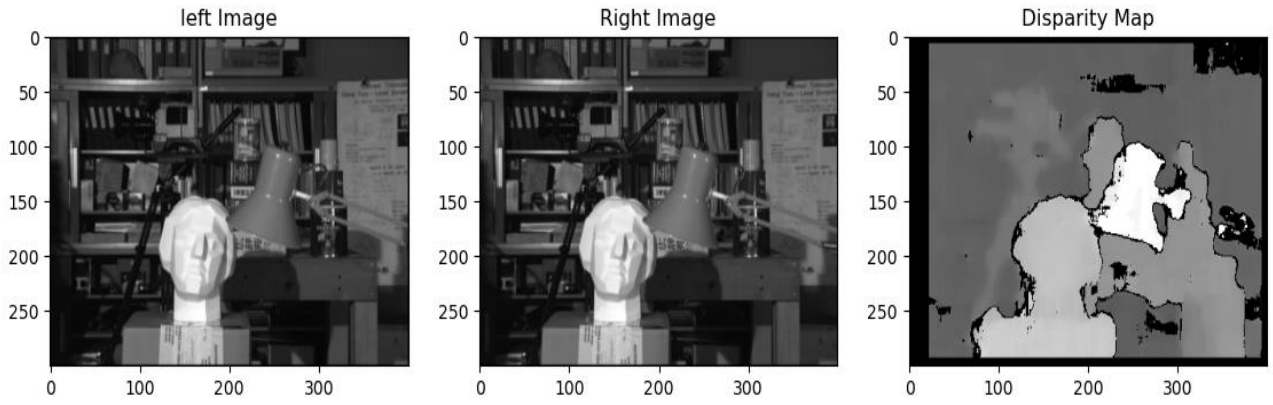
```
# Create a figure with 3 subplots
fig, axs = plt.subplots(nrows=1, ncols=3,  figsize=(18, 6))
```

عرض الصورتين اليسارية واليمينية وصورة العمق

```
# Display the original image
axs[0].imshow(left_image, cmap='gray')
axs[0].set_title('left Image')
axs[1].imshow(right_image, cmap='gray')
axs[1].set_title('Right Image')

# Display the disparity map
axs[2].imshow(disparity_map, cmap='gray')
axs[2].set_title('Disparity Map')
plt.show()
```

النتيجة:



يظهر الجزء الأقرب للكاميرا بلون أبيض وهو الضوء في حين يأتي بعده في العمق صورة التمثال حيث تبدو بلون فاتح لكن ليس أبيض. تستمر السويات بالاتجاه نحو اللون الداكن حتى الوصول لأعمق نقطة في الصورة حيث يظهر اللون الأسود.

1.2 كشف الحركة في الفيديو:

يعتمد كشف الحركة على حساب القيمة المتوسطة لمجموعة إطارات متتالية واعتبار الصورة الناتجة عن عملية المتوسط صورة خلفية Background يتم بعدها مقارنة كل إطار مع صورة الخلفية من خلال عملية طرح بحيث يتم اعتبار وجود حركة في حال كان ناتج الطرح أكبر من عتبة ما. يمكن استخدام عمليات لاحقة لتصحيح ناتج الكشف مثل إزالة المناطق الصغيرة التي لا يتجاوز عدد بكسلاتها عتبة ما وبالتالي يتم التخلص من المناطق الضجيجية التي لا تمثل حركة والتي قد يكون سببها تغير إضاءة بسيط في المشهد.

1.2.1 كشف الحركة – تطبيق عملي

نفذ الكود الآتي:

```
import numpy as np

import cv2

# Create a background subtractor object

fgbg = cv2.createBackgroundSubtractorMOG2(varThreshold=60, history=20)


# Open a video capture stream (you can replace this with your own video file)

cap = cv2.VideoCapture('test.mp4')

while True:
```

إنشاء كائن لكشف الخلفية بالبارامترات التالية: عتبة كشف سويات 60، عدد الإطارات المراد أخذها بالحسبان لحساب صورة المتوسط = 20

قراءة مقطع الفيديو

```
# Read a frame from the video
ret, frame = cap.read()
if not ret:
    break
# Apply background subtraction to get the foreground mask
fgmask = fgbg.apply(frame)
# Remove small regions using bwareaopen
min_area_threshold = 20  # Adjust as needed
# Find connected components and filter out small regions
num_labels, labels, stats, _ = cv2.connectedComponentsWithStats(fgmask, connectivity=8)
# Create an array of indices for regions to be removed
small_regions_indices = np.where(stats[:, cv2.CC_STAT_AREA] < min_area_threshold)[0]
# Set the pixels corresponding to small regions to 0
fgmask_processed = np.isin(labels, small_regions_indices, invert=True).astype(np.uint8) * fgmask
# Display the processed foreground mask
cv2.imshow('Processed Foreground Mask', fgmask_processed)
# Display the original frame
cv2.imshow('Original Frame', frame)
# Press 'q' to exit
if cv2.waitKey(30) & 0xFF == ord('q'):
    break
# Release the video capture and close all windows
cap.release()
cv2.destroyAllWindows()
```

تنفيذ حلقة مستمرة يتم فيها قراءة إطار واحد من الفيديو في كل تكرار حتى انتهاء الإطارات تتوقف الحلقة
يتم تطبيق عملية طرح بين الإطار وبين صورة المتوسط ويتم حفظ الصورة الناتجة في fgmask
بعدها يتم تطبيق خوارزمية لإزالة المناطق التي حجمها أقل من 20 بكسل

عرض الإطار الأصلي ونتيجة motion detection
والانتظار حتى انتهاء كل الإطارات في الفيديو أو ضغط Q من لوحة المفاتيح