Stereo Vision and Depth Perception in Computer Vision

Sumitkrsharma · Follow

7 min read · Oct 5, 2023

Welcome to post on stereo vision and depth perception in computer vision! In the fascinating world of artificial intelligence and image processing, these concepts play a pivotal role in enabling machines to perceive the three-dimensional beauty of our surroundings, just like our human eyes do. Join us on this journey as we explore the technology behind stereo vision and depth perception, unraveling the secrets of how computers gain a sense of depth, distance, and spatial understanding from 2D images. Whether you're a seasoned computer vision enthusiast or a curious newcomer, prepare to embark on an enlightening adventure into the world of 3D vision in the digital realm. Let's dive in!

What is stereo vision and depth perception in computer vision?

Stereo vision and depth perception are important concepts in computer vision that aim to mimic the human ability to perceive depth and 3D structure from visual information. They are often used in applications like robotics, autonomous vehicles, and augmented reality.

Stereo Vision

- Stereo vision, also known as stereopsis or binocular vision, is a technique used to perceive depth in a scene by capturing and analyzing images from two or more cameras placed slightly apart, mimicking the way human eyes work.
- The basic principle behind stereo vision is triangulation. When two cameras (or "stereo cameras") capture images of the same scene from slightly different viewpoints, the resulting pair of images, called stereo pairs, contains disparities or differences in the positions of corresponding points in the two images.
- By analyzing these disparities, a computer vision system can calculate the depth information of objects in the scene. Objects that are closer to

the cameras will have larger disparities, while objects farther away will have smaller disparities.

- Stereo vision algorithms often involve techniques such as feature matching, disparity mapping, and epipolar geometry to compute the depth map or a 3D representation of the scene.

Depth Perception

- Depth perception in computer vision refers to the ability of a system to understand and estimate the distance of objects in a 3D scene from a single or multiple 2D images or video frames.
- In addition to stereo vision, depth perception can be achieved using other methods, including:
- Monocular cues: These are depth cues that can be perceived with a single camera or image. Examples include perspective, texture gradient, shading, and occlusion. These cues can help estimate depth even when stereo vision is not available.
- LiDAR (Light Detection and Ranging): LiDAR sensors use laser beams to measure the distance to objects in a scene, providing precise depth information in the form of a point cloud. This information can be fused with visual data for more accurate depth perception.
- Structured Light: Structured light involves projecting a known pattern onto a scene and analyzing how the pattern deforms on objects in the scene. This deformation can be used to calculate depth information.
- Time-of-Flight (ToF) Cameras: ToF cameras measure the time it takes for light to bounce off objects and return to the camera. This information is used to estimate depth.

Depth perception is crucial for tasks such as obstacle avoidance, object recognition, 3D reconstruction, and scene understanding in computer vision applications.

Components in Stereo vision and depth perception in Computer Vision

Stereo Cameras: Stereo vision relies on the use of two or more cameras (stereo cameras) that are placed at a known distance from each other. These cameras capture images of the same scene from slightly different viewpoints, simulating the way human eyes perceive depth.

Image Capture: The cameras capture images or video frames of the scene. These images are typically referred to as the left image (from the left camera) and the right image (from the right camera).

Calibration: To accurately compute depth information, stereo cameras must be calibrated. This process involves determining camera parameters such as intrinsic matrix, distortion coefficients, and extrinsic parameters (rotation and translation between cameras). Calibration ensures that the images from both cameras can be correctly rectified and matched.

Rectification: Rectification is a geometric transformation applied to the captured images to align corresponding features along epipolar lines. It simplifies the stereo matching process by making the disparities more predictable.

Stereo Matching: Stereo matching is the process of finding correspondences or matching points between the left and right images. Disparities, which represent the horizontal shift of features between images, are calculated for each pixel. Various stereo matching algorithms are available, including block matching, semi-global matching, and graph cuts, to find these correspondences.

Disparity Map: The disparity map is a grayscale image where each pixel's intensity value corresponds to the disparity or depth at that point in the scene. Objects that are closer to the camera have larger disparities, while objects farther away have smaller disparities.

Depth Map: It calculates the depth of each pixel in real-world units (e.g., meters) based on the disparities obtained from the stereo images, using the known baseline and focal length of the cameras.

Visualization: Depth and disparity maps are often visualized to provide a human-readable representation of the scene's 3D structure. These maps can be displayed as grayscale images or converted into point clouds for 3D visualization.

Some Hardware In addition to cameras, specialized hardware like depth-sensing cameras (e.g., Microsoft Kinect, Intel RealSense) or LiDAR (Light Detection and Ranging) sensors can be used to obtain depth information. These sensors provide depth directly, without the need for stereo matching.

Python example implementation of stereo vision and depth perception in computer vision

```
import cv2
import numpy as np

# Create two video capture objects for left and right cameras (adjust device IDs as needed)
left_camera = cv2.VideoCapture(0)
right_camera = cv2.VideoCapture(1)

# Set camera resolution (adjust as needed)
width = 640
height = 480
left_camera.set(cv2.CAP_PROP_FRAME_WIDTH, width)
left_camera.set(cv2.CAP_PROP_FRAME_HEIGHT, height)
right_camera.set(cv2.CAP_PROP_FRAME_WIDTH, width)
right_camera.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

# Load stereo calibration data (you need to calibrate your stereo camera setup first)
```

```
stereo_calibration_file = 'stereo_calibration.yml'
calibration_data = cv2.FileStorage(stereo_calibration_file,
cv2.FILE_STORAGE_READ)

if not calibration_data.isOpened():
print("Calibration file not found.")
exit()

camera_matrix_left = calibration_data.getNode('cameraMatrixLeft').mat()
camera_matrix_right = calibration_data.getNode('cameraMatrixRight').mat()
distortion_coeff_left = calibration_data.getNode('distCoeffsLeft').mat()
distortion_coeff_right = calibration_data.getNode('distCoeffsRight').mat()
R = calibration_data.getNode('R').mat()
T = calibration_data.getNode('T').mat()

calibration_data.release()

# Create stereo rectification maps
R1, R2, P1, P2, Q, _, _ = cv2.stereoRectify(
camera_matrix_left, distortion_coeff_left,
camera_matrix_right, distortion_coeff_right,
(width, height), R, T
)

left_map1, left_map2 = cv2.initUndistortRectifyMap(
camera_matrix_left, distortion_coeff_left, R1, P1, (width, height), cv2.CV_32FC1
)
right_map1, right_map2 = cv2.initUndistortRectifyMap(
camera_matrix_right, distortion_coeff_right, R2, P2, (width, height),
cv2.CV_32FC1
)

while True:
# Capture frames from left and right cameras
```

```
ret1, left_frame = left_camera.read()
ret2, right_frame = right_camera.read()

if not ret1 or not ret2:
print("Failed to capture frames.")
break

# Undistort and rectify frames
left_frame_rectified = cv2.remap(left_frame, left_map1, left_map2,
interpolation=cv2.INTER_LINEAR)
right_frame_rectified = cv2.remap(right_frame, right_map1, right_map2,
interpolation=cv2.INTER_LINEAR)

# Convert frames to grayscale
left_gray = cv2.cvtColor(left_frame_rectified, cv2.COLOR_BGR2GRAY)
right_gray = cv2.cvtColor(right_frame_rectified, cv2.COLOR_BGR2GRAY)

# Perform stereo matching to calculate depth map (adjust parameters as needed)
stereo = cv2.StereoBM_create(numDisparities=16, blockSize=15)
disparity = stereo.compute(left_gray, right_gray)

# Normalize the disparity map for visualization
disparity_normalized = cv2.normalize(disparity, None, alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8U)

# Display the disparity map
cv2.imshow('Disparity Map', disparity_normalized)

if cv2.waitKey(1) & 0xFF == ord('q'):
break

# Release resources
left_camera.release()
```

```
right_camera.release()
cv2.destroyAllWindows()
```

Note: camera calibration is required for stereo camera setup and save the calibration data in .yml file , put path in example location.

Applications

The depth information obtained through stereo vision and depth perception can be used in various computer vision applications, including:

- **3D scene reconstruction**
- **Object detection and tracking**
- **Autonomous navigation for robots and vehicles**
- **Augmented and virtual reality**
- **Gesture recognition**

Limitations

Here are some of the key limitations:

1. **Dependence on Camera Calibration**: Stereo vision systems require precise calibration of the cameras used. Accurate calibration is critical to ensure that the depth information is correctly computed. Any errors in calibration can lead to inaccurate depth perception.
2. **Limited Field of View**: Stereo vision systems have a limited field of view based on the baseline distance between the two cameras. This can result in blind spots or difficulty in perceiving objects that are outside the field of view of both cameras.
3. **Textureless and Featureless Surfaces**: Stereo matching algorithms rely on finding corresponding features in the left and right images. Surfaces that lack texture or distinctive features, such as smooth walls or uniform backgrounds, can be challenging to match accurately, leading to depth estimation errors.

4. **Occlusions:** Objects that occlude one another in the scene can cause difficulties for stereo vision. When one object partially hides another, determining the depth of the occluded region can be problematic.
5. **Limited Range and Resolution:** The accuracy of depth perception with stereo vision diminishes as the distance from the cameras increases. Additionally, the resolution of depth measurements decreases with distance, making it challenging to perceive fine details in distant objects.
6. **Sensitivity to Lighting Conditions:** Changes in lighting conditions, such as variations in ambient light or shadows, can affect the accuracy of stereo vision. Inconsistent lighting can make it difficult to find correspondences between the left and right images.
7. **Computational Resources:** Stereo matching algorithms can be computationally intensive, especially when working with high-resolution images or real-time video streams. Real-time applications may require powerful hardware for efficient processing.
8. **Cost and Complexity:** Setting up a stereo vision system with calibrated cameras can be expensive and time-consuming. The hardware requirements, including cameras and calibration equipment, can be a barrier for some applications.
9. **Inaccuracy in Transparent or Reflective Objects:** Transparent or highly reflective surfaces can cause errors in stereo vision, as these materials may not reflect light in a way that is suitable for depth perception.
10. **Dynamic Scenes:** Stereo vision assumes that the scene is static during image capture. In dynamic scenes with moving objects or camera motion, maintaining correspondences between the left and right images can be challenging, leading to inaccurate depth estimates.
11. **Limited Outdoor Use:** Stereo vision systems may struggle in outdoor environments with bright sunlight or in scenes with a lack of texture, such as clear skies.

In conclusion, stereo vision and depth perception in computer vision open up a world of possibilities for machines to interact with and understand the three-dimensional richness of our environment. As we've explored in this post, these technologies are at the heart of applications ranging from robotics and autonomous vehicles to augmented reality and medical imaging.

Thank you readers !! Stay connected !

Computer Vision | Artificial Intelligence | Image Processing | Machine Learning

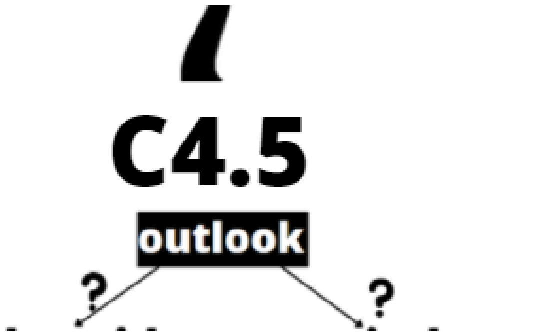
Written by Sumitkrsharma

Follow

121 Followers

Data scientist | ML researcher

More from Sumitkrsharma



Sumitkrsharma

Image Registration in Computer Vision

In the ever-evolving realm of computer vision, where pixels transform into intelligence and…

7 min read · Sep 2, 2023

18

Sumitkrsharma

Understanding C4.5 Decision tree algorithm

C4.5 algorithm is improvement over ID3 algorithm, where “C” is shows algorithm is…

3 min read · Jan 25, 2022

37

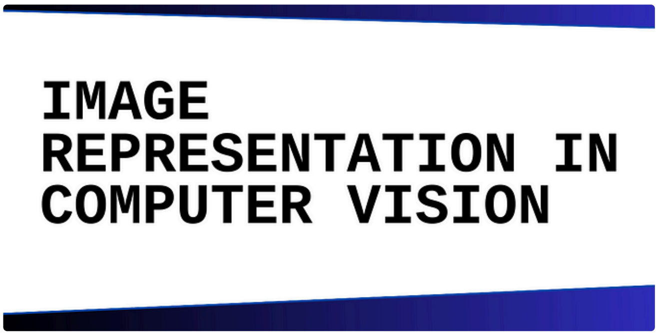
Sumitkrsharma

Image Filtering in Computer vision

This post is about, brief understanding of image filtering step in computer vision, what…

6 min read · Jul 27, 2023

30



Sumitkrsharma

Image representation in Computer Vision

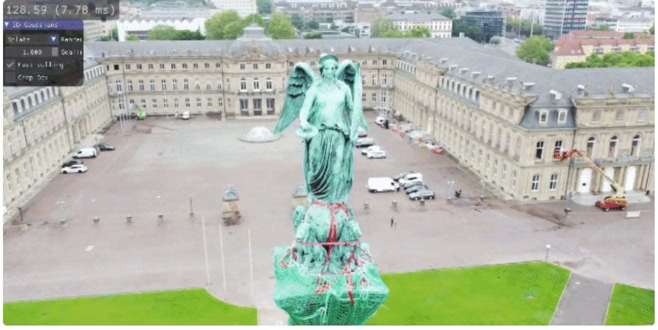
This post is about brief understanding of different techniques used for image…

5 min read · Jul 20, 2023

8

See all from Sumitkrsharma

Recommended from Medium



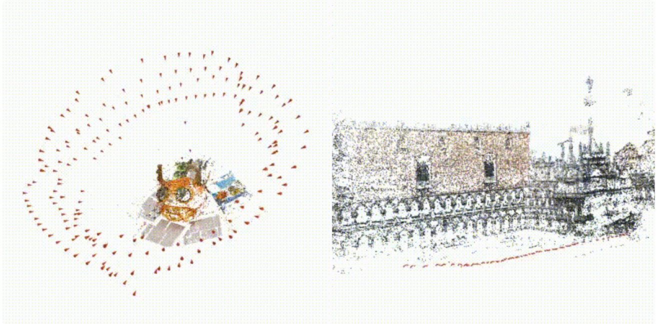
Loges Siva

Understanding and Exploring 3D Gaussian Splatting: A...

Blending Pixels and Realities with no compromise in Speed

9 min read · Dec 28, 2023

52



Teresa Lobo

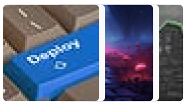
Understanding Structure From Motion Algorithms

Obtaining the geometry of 3D scenes from 2D images.

6 min read · Dec 18, 2023

303

Lists

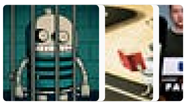


Predictive Modeling w/ Python

20 stories · 975 saves

Natural Language Processing

1258 stories · 748 saves

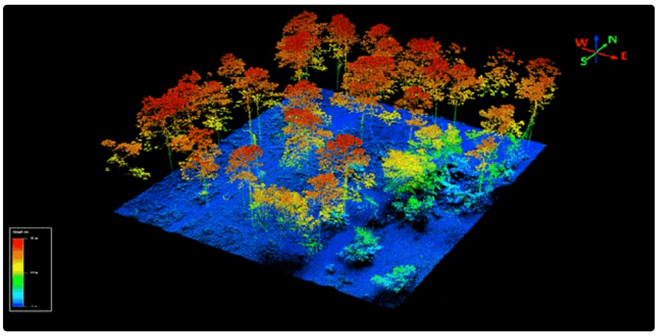


AI Regulation

6 stories · 351 saves

Practical Guides to Machine Learning

10 stories · 1161 saves



Matt Sharon

Demystifying LiDAR Point Cloud Data: Comprehensive Guide

Sumitkrsharma

Unveiling the World in 3D: The Power of Computer Vision...

LiDAR, short for Light Detection and Ranging, has revolutionized data acquisition across...

4 min read · Feb 2, 2024

60 1

In the realm of computer vision, 3D reconstruction stands as a transformative...

7 min read · Oct 16, 2023

78

Isaac Berrios

Introduction to Motion Detection: Part 2

Using Optical Flow to Detect Motion

6 min read · Nov 1, 2023

31 1

Abdul Rehman in Red Buffer

Beyond the Surface: Advanced 3D Mesh Generation from 2D Images...

Reconstructing a 3D object from 2D images is a very interesting task and has many real-...

17 min read · Feb 16, 2024

191 2

See more recommendations