

# 3D Reconstruction with Stereo Images — Part 4: Depth Map

**AI/HUB**

AI/HUB

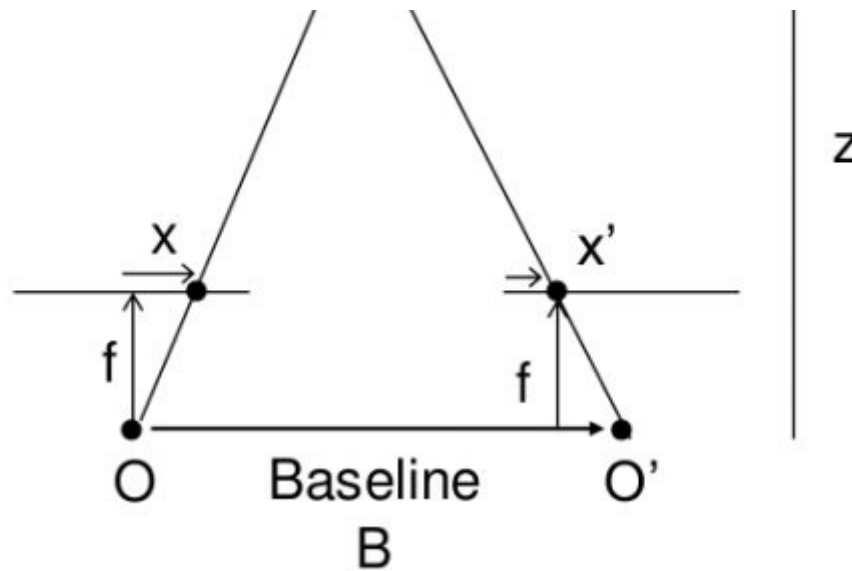
Jan 24, 2019 · 3 min read



The following will wrap up our series on 3D reconstruction. Our goal will be to visualize the depth of objects found in a set of stereo images. Essentially we will produce a gray scale heat map whereby lighter shades of gray will signify objects close to the camera lens with progressively darker shades to distinguish objects further away.

To build upon our last post we can now proceed to pull depth information from our stereo images using a novel approach. The depth of a point in real world space can be calculated through the use of mathematical formulas and functions that incorporate the principles of epipolar geometry. The image below illustrates a proof of these concepts in action.





So let's break down the diagram above. First we should note the optical centers of both our cameras at O and O' in addition to the point in 3D space whose depth we are trying to find; point X. Length f is the focal length of our cameras which we know from our intrinsic values found in the first part of this series. Now x and x' are the points on the 2D image plane where point X will appear in each image respectively. The idea here is that the equivalent triangles as denoted Ofx and O'fx' can be compared to each other in order to extract the depth Z of point X. The equation that allows us to accomplish this is shown below.

$$\text{disparity} = x - x' = \frac{Bf}{Z}$$

To describe the equation in as plain terms as possible, it tells us that the distance Z of point X away from our cameras' optical centers is inversely proportional to the difference in distance of points x and x' (the image points of our object as they appear on the 2D plane).

In summary Open CV is being used to find matched points in each of our images, greatly aided and expedited by the principles of an epipolar constraint. Subsequently it proceeds to calculate the disparity between the matched points therefore giving us the depth of the object in question. This process can be repeated for every pixel in our stereo images offering our concluding solution to 3D reconstruction.

Below we can see the code in action which processes a set of stereo photos found in the OpenCV package. The disparity values between the images are found by calling the

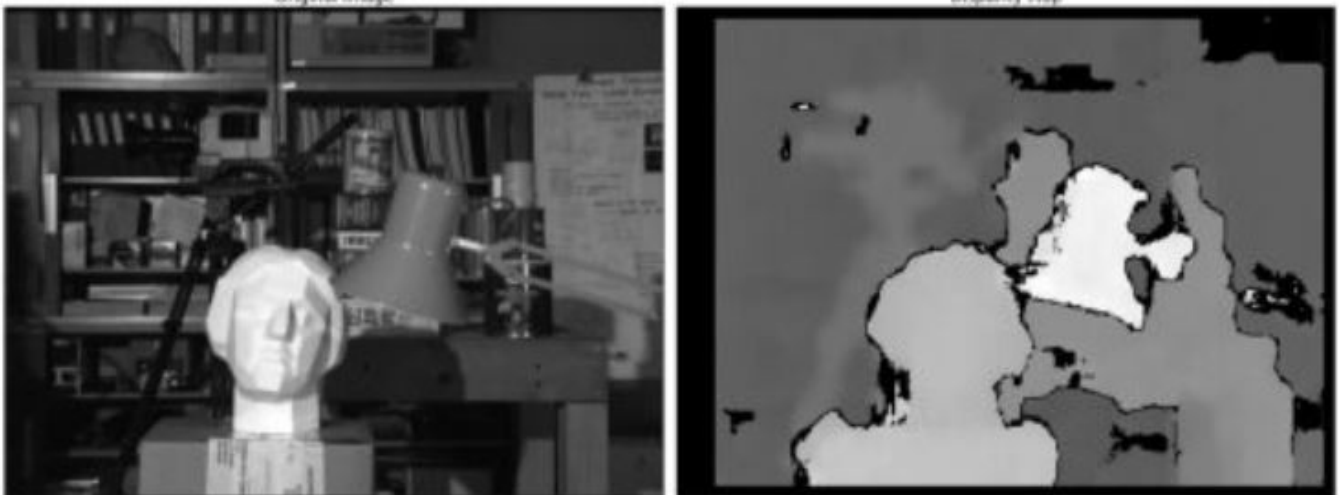
compute method of a stereo object we've instantiated on the previous line.

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

imgL = cv2.imread('tsukuba_l.png',0)
imgR = cv2.imread('tsukuba_r.png',0)

stereo = cv2.createStereoBM(numDisparities=16, blockSize=15)
disparity = stereo.compute(imgL,imgR)
plt.imshow(disparity,'gray')
plt.show()
```

And finally in the last two lines of our code we produce what exemplifies what can be accomplished with OpenCV and Stereo images; the original photo along with an image visualizing the distance of objects found within.



Thanks for reading I hope you've learned something from this series as I certainly have. Although I cannot say I have perfect knowledge on the subject, the process of working through the original tutorials found in OpenCV documentation (link posted below) greatly advanced my skills in this area. With these resources I'd be confident to tackle my own application using 3D Reconstruction and Stereo Images; an effective technology with undoubtedly a multitude of beneficial applications in our day to day lives.

**Camera Calibration and 3D Reconstruction - OpenCV 3.0.0-dev documentation**

[Edit description](#)

docs.opencv.org

Written by: Keenan James under supervision of Professor Amit Maraj

Virtual Reality

Computer Vision


Opencv


About

Help

Legal

Get the Medium app

A button that says 'Download on the App Store', and if clicked it will lead you to the iOS App store

A button that says 'Get it on, Google Play', and if clicked it will lead you to the Google Play store