Depth Map from Stereo Images

Goal

In this session,

- We will learn to create a depth map from stereo images.

Basics

In the last session, we saw basic concepts like epipolar constraints and other related terms. We also saw that if we have two images of same scene, we can get depth information from that in an intuitive way. Below is an image and some simple mathematical formulas which prove that intuition. (Image Courtesy :
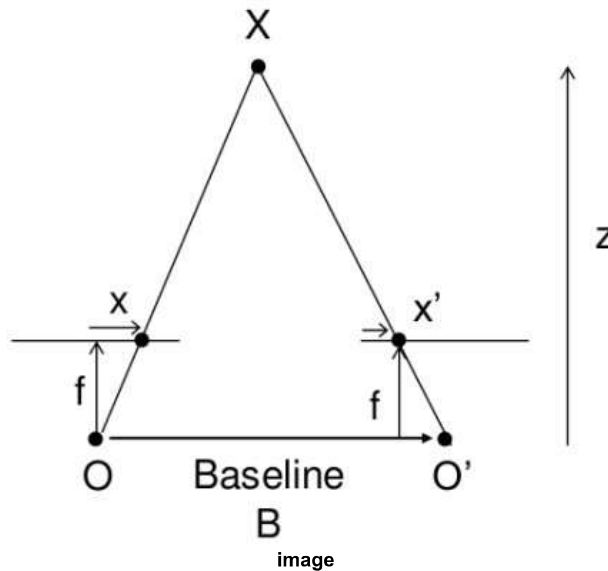
image

The above diagram contains equivalent triangles. Writing their equivalent equations will yield us following result:

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

x and x' are the distance between points in image plane corresponding to the scene point 3D and their camera center. B is the distance between two cameras (which we know) and f is the focal length of camera (already known). So in short, the above equation says that the depth of a point in a scene is inversely proportional to the difference in distance of corresponding image points and their camera centers. So with this information, we can derive the depth of all pixels in an image.

So it finds corresponding matches between two images. We have already seen how epiline constraint make this operation faster and accurate. Once it finds matches, it finds the disparity. Let's see how we can do it with OpenCV.

Code

Below code snippet shows a simple procedure to create a disparity map.

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

imgL = cv.imread('tsukuba_l.png', cv.IMREAD_GRAYSCALE)
imgR = cv.imread('tsukuba_r.png', cv.IMREAD_GRAYSCALE)

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

Below image contains the original image (left) and its disparity map (right). As you can see, the result is contaminated with high degree of noise. By adjusting the values of numDisparities and blockSize, you can get a better result.
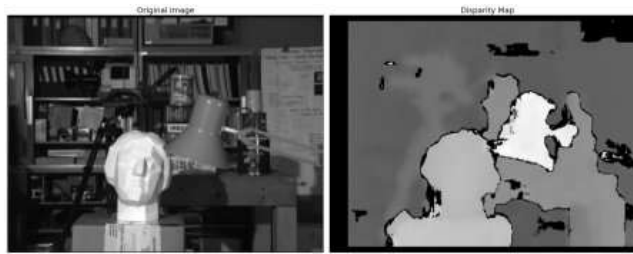
image

There are some parameters when you get familiar with StereoBM, and you may need to fine tune the parameters to get better and smooth results. Parameters:

- texture_threshold: filters out areas that don't have enough texture for reliable matching
- Speckle range and size: Block-based matchers often produce "speckles" near the boundaries of objects, where the matching window catches the foreground on one side and the background on the other. In this scene it appears that the matcher is also finding small spurious matches in the projected texture on the table. To get rid of these artifacts we post-process the disparity image with a speckle filter controlled by the speckle_size and speckle_range parameters. speckle_size is the number of pixels below which a disparity blob is dismissed as "speckle." speckle_range controls how close in value disparities must be to be considered part of the same blob.
- Number of disparities: How many pixels to slide the window over. The larger it is, the larger the range of visible depths, but more computation is required.
- min_disparity: the offset from the x-position of the left pixel at which to begin searching.
- uniqueness_ratio: Another post-filtering step. If the best matching disparity is not sufficiently better than every other disparity in the search range, the pixel is filtered out. You can try tweaking this if texture_threshold and the speckle filtering are still letting through spurious matches.
- prefilter_size and prefilter_cap: The pre-filtering phase, which normalizes image brightness and enhances texture in preparation for block matching. Normally you should not need to adjust these.

These parameters are set with dedicated setters and getters after the algorithm initialization, such as `setTextureThreshold`, `setSpeckleRange`, `setUniquenessRatio`, and more. See **cv::StereoBM** documentation for details.

Additional Resources

- Ros stereo img processing wiki page

Exercises

1. OpenCV samples contain an example of generating disparity map and its 3D reconstruction. Check stereo_match.py in OpenCV-Python samples.

Generated on Mon Sep 1 2025 03:41:09 for OpenCV by doxygen 1.12.0