3D Depth Reconstruction from Stereo Images

Jay Mohile

April 2020

I. INTRODUCTION

This algorithm extends upon the region matching approach described in the provided textbook [1]. Given two 2D images corresponding to a left and right eye view, this process generates a 3D reconstruction of the scene through three major steps:

1. Identifying feature pairs between the two images.
2. Using the lateral shift of these feature pairs to determine 3D world-space coordinates.
3. Post processing

II. PROCESS

1. Identifying Feature Pairs and Determining Left-Right Disparity

We desire a way to determine how far each feature has "shifted" when viewed from the right hand side ("RHS") camera relative to the left ("LHS"). This shift is called the feature's *disparity*. Identifying distinct features (e.g. book, cup) is a complex process in and of itself, so a simpler alternative is used: for each LHS pixel, a *conjugate-pair* is obtained by finding the most similar RHS pixel. Then, the difference in pixel-location is used as the disparity. This process is repeated across every LHS pixel to calculate a *disparity-map*, i.e a parallax shift at every location.

1.1 Evaluating similarity

For each potential LHS-RHS pair, a fixed-sized window (WL, WR) is extracted around each pixel. WL and WR are compared using the provided product-sum method [1] (eqn 11.9). For each of {WL, WR}, a new window {WL', WR'} is calculated by subtracting the average intensity of that window. As a result, within each window {WL', WR'}, a positive value indicates a pixel above the mean, and a negative value below the mean. When pixel-wise product summing these windows, complementary positive or negative value pairs will increase the sum, while an opposing positive-negative pair will reduce it. This result is normalized to a value between 0 and 1, providing a final LHS-RHS correlation coefficient.

1.2 Constraining similarity search

If the system were to evaluate similarity between every potential left-right pair, the time complexity would quickly grow impractical. Instead, a set of assumptions are made to narrow the search space. First, it is assumed that (given the left and right camera are at the same height), only a horizontal disparity exists. That way, for a given LHS pixel, only RHS pixels within the same row are considered; this reduces the complexity of this step from N^2 to N.

Next, referring to the diagram above, we realize that disparities must be all positive or all negative (based on whether left or right is the reference). That is, since the RHS camera's field of view extends further to the right, all of its pixels must be left-shifted compared to the LHS camera. As such, we can further constrain our similarity search to only consider RHS pixels on *one side* of the reference LHS pixel.

Finally, we realize that the closer an object is to the camera, the greater its disparity will be. As the image intuitively contains a 'closest feature', it must also have a 'maximum disparity'. By setting this maximum disparity to some reasonable conservative upper bound—for example, one quarter the width of the image—the set of conjugate pairs to evaluate is further reduced.

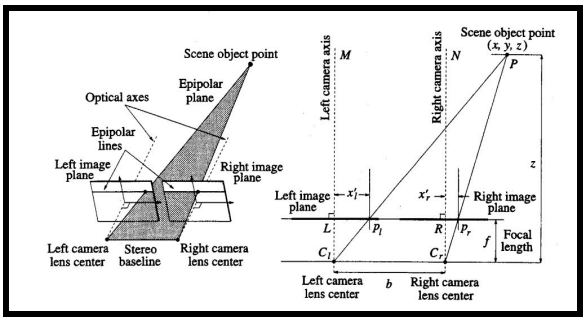
1.3 Improving region matching

The above represents a 'bare-minimum' approach to region matching. However, it is possible to improve its results. This implementation takes advantage of the intuitive limitation that each RHS pixel can only correspond to one LHS pixel, adapted from Stefano et al [2], with the following effect. Assume LHS pixel L1 (i, j) is matched with RHS pixel R1 (i', j') with a correlation of 0.6. Later, it's found that LHS pixel L2 (i + x, j) matches R1 with a correlation of 0.8. Even if R1 is not the best match for L2, it is clear that L1 was not the optimal match for R1. Therefore, the previous conjugate pair is removed, and L1 is added back to the evaluation queue.

This allows the matching algorithm to 'reject' a previous (greedy) feature mapping if a more accurate one is found later on.

2. Using Lateral Shift (Disparity) to Determine World Coordinates

Each pixel of the reference image corresponds to some point P in absolute "world-space". From the first step of this process, we are given the pixel offset for a given feature between the left and right eye views. Using this information, combined with known details about the camera setup, we construct the following geometry. The following diagram (Figure 1) and formulas are from the provided textbook [1].



PMC_l and p_lLC_l form similar triangles, as do PNC_r and p_rRC_r.

The left pair implies $\frac{x}{z} = \frac{x'_l}{f}$,

The right pair implies $\frac{x-(N-M)}{z} = \frac{x'_r}{f}$

Combined, these yield $z = \frac{f(N-M)}{(x'_r - x'_l)}$

$N - M$ is the distance in mm between camera axes, provided to us as the *baseline* distance. $x_r - x_l$ is the total left-to-right pixel offset, calculated as *disparity* + *doffs* (provided). Finally, using this calculated z, $x = z\frac{x'_L}{f}$ and $x = z\frac{y'}{f}$ using the x/y centerline offsets of P.

Image Rectification

For this implementation of stereo-matching, it was possible to assume both cameras were horizontal and parallel. However, this is not always the case (for example, with movable cameras). In this case, two images taken from different perspectives can be *rectified*, i.e warped such that equivalent points lie on the same horizontal lines [3]. Using this approach requires precisely measuring/calculating the relative location of the cameras.

3. Post Processing

While the algorithm described above has reasonable success in most parts of the target image, it suffers from some noise and distortion, especially around occluding edges. Furthermore, as disparities are quantized to pixel-differences, the resultant point cloud appears "sliced" in the Z axis like an MRI scan. To address this, the stereo-matching generated point cloud is post processed to improve the final rendering.

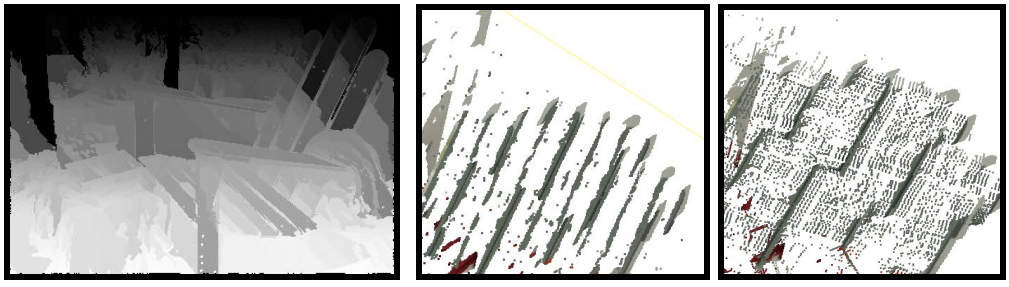
3.1: Smoothing

As discussed above, the original image suffers from quantized distances. In reality, it is expected that continuous surfaces from the target image also appear continuous in 3D space. To achieve this, the original point cloud is modified as follows. Around each point P, a fixed size window W is obtained (using the 2D projection of the cloud). The Z coordinate of P is set to the average of all *related* points from W.

Determining if two points are related

Two metrics are used to determine the relationship between a pair of points: (1) Z-distance, and (2) region. The first metric is true whenever two points are within some threshold Z-distance of one another. The second metric is a bit more complicated.

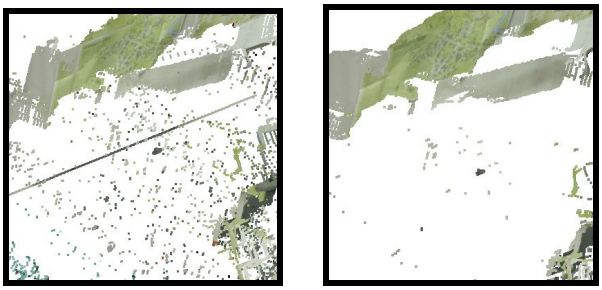
Prior to performing any stereo matching, the target image is preprocessed to identify contiguous *regions*. The image is traversed depth-first such that if two pixels are within a set color threshold, they will be assigned the same region. This very roughly breaks an image down into a set of contiguous features. Now while post-processing, two pixels satisfy this metric if they fall within the same region. Two points are considered related if they satisfy both relation metrics.



*(**Left: Figure 2**) Results of region matching. Each region is assigned a color, but two regions having a similar color has no significance. (**Center: Figure 3**) Section of point cloud before smoothing. (**Right: Figure 4**) Same section after smoothing.*

3.2: Denoising

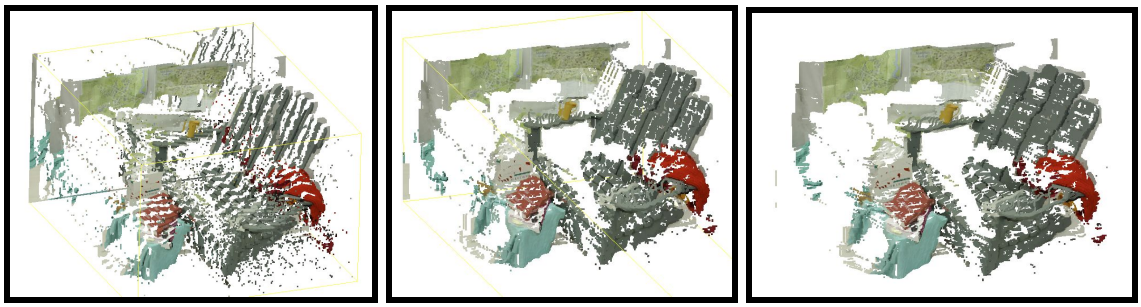
While the previous step smooths the cloud, it does not address (and in fact, amplifies) the noise present. Here, points that the algorithm does not successfully place near other points in their intuitively expected 'feature' are considered noise. To address this, the system considers a window W around each cloud-point P, with W defined the same way as above. The absolute Z-distances from P to each point in W are averaged. If this average is above a given threshold, P will be considered an outlier and removed from the cloud.



*(**Left: Figure 5**) A section from the point cloud before denoising. (**Right: Figure 6**) The same section after denoising. Note that the valid structures in the corners are unaffected.*

3.3: Iteration

One pass of the post-processor involves first smoothing, then denoising the cloud. During a given pass, the results are outputted into a new cloud without mutating the input. This is so that the order of traversal (i.e left-right, top-bottom) does not skew the results. However, the post-processing step is repeated N times, allowing each iteration to build on the results of the previous. This allows the system to 'gain confidence' in its results. For example, if a point P can survive multiple smoothing and denoising steps, it is highly likely that it is not noise.

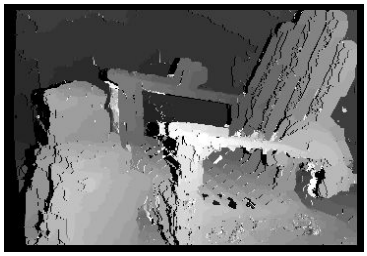
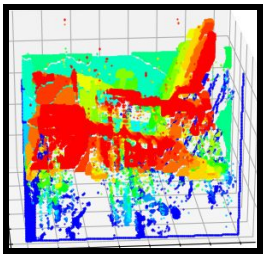


(**Left: Figure 7**) N=0 (Raw), (**Center: Figure 8**) N=1, (**Right: Figure 9**) N=3

III. RESULTS

Disparity Maps

These maps visualize the disparity calculated at each pixel of the input image. In general, smooth gradients are expected for contiguous surfaces and sharp edges for occluding ones. While the process appears to struggle with low-contrast occluding surfaces, it is overall fairly accurate.

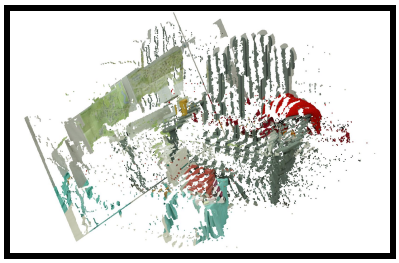
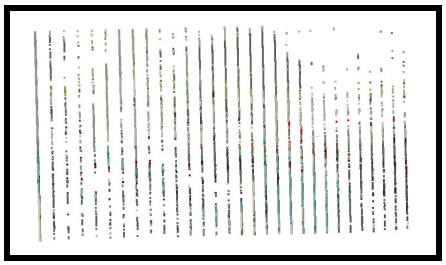




(Left: Figure 10) *Source image.* ***(Center: Figure 11)*** *Disparity map from stereo region matching, step 1 of the process described above. Lighter points are closer.* ***(Right: Figure 12)****: False color disparity map of reconstructed image, viewed from a slight angle. Red corresponds to near points, while green corresponds to those farther away.*

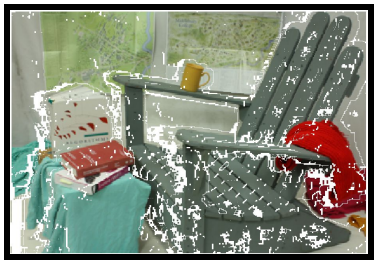
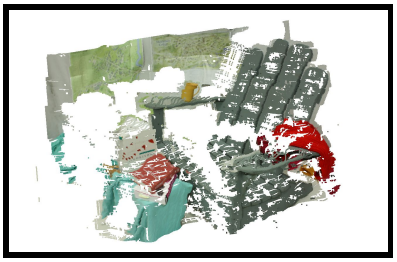
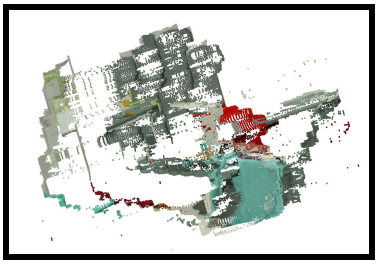
Point Clouds

These are the outputs of the algorithm, where each pixel is mapped to a 3D point. Below, the results with and without post processing are shown. For each, a 2D projection, 45° top-down, and 90° side view are provided. While the raw cloud better preserves the head-on projection (since denoising removes some points), the post-processed version is far better at preserving realistic depth and accurate surfaces.[1]

Raw

Post Processed

[1]Please note that the visualization tools used did not support axes. However, the rendering appears to be proportionate.

REFERENCES

[1]

Provided "Machine Vision" Textbook Chapter 11

[2]

Stefano, L., Marchionni, M. and Mattoccia, S., 2004. A fast area-based stereo matching algorithm. Image and Vision Computing, 22(12), pp.983-1005.

[3]

A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*