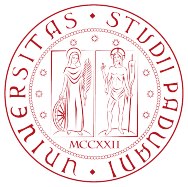


University of Padova

3D AUGMENTED REALITY

Stereo disparity computation from projected random dot pattern

Luca Attanasio 1206749, Savio Francesco 1206747

supervised by

Simone Milani

January 23, 2019

Contents

1 Introduction **2**

2 Image processing **2**
2.1 Image rectification 2
2.2 Stereo system formalization 3
2.3 Disparity estimation 4

3 Block matching **4**
3.1 Disparity evaluation in MATLAB 4

4 Solution **5**

5 Pseudocode **6**

6 Results **7**
6.1 Model 1 results 8
6.2 Model 2 results 8
6.3 Model 3 results 9
6.4 Model 1 images 10
6.5 Model 2 images 11
6.6 Model 3 images 12

1 Introduction

In the following project, given a couple of images from an *Aquifi camera*, the disparity is computed using a multi-layer hierarchical approach.

The multi-layer hierarchical approach applied in this work consists in computing the disparity on multiple reduced resolution versions of the two images, both left and right. The reduced resolution layers consist in three decomposition layers: 1/2, 1/4, 1/8. By decomposing the image in layer 1/2, the image is meant to be resized in width and height by 1/2. The same method is used with the other layers.

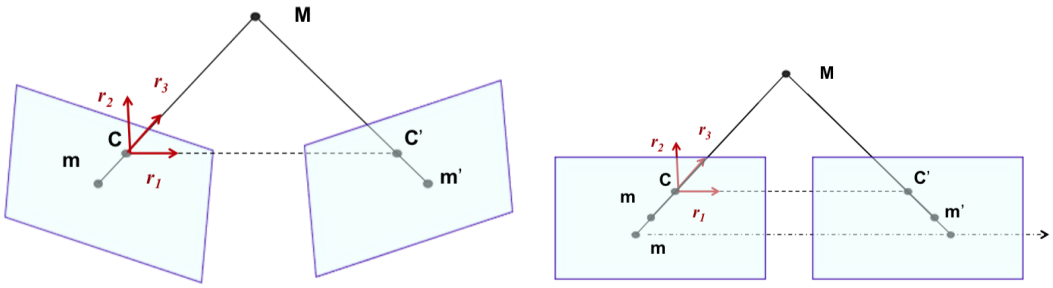
The results of these computations, in terms of accuracy estimators, are employed on the original image.

The dataset used for the images is available at: http://www.dei.unipd.it/ sim1mil/CVcourse/imagesStereo.zip. In the dataset, there are three sets of couples of images (*models*) that will be later analyzed to yield the results.

2 Image processing

Before explaining the solution of the assignment given, there are two processes involved to produce the results, taking into account the two images: image rectification and disparity computation. The technical details of these processes will be analyzed in this section, providing also mathematical insights into the stereo system architecture.

2.1 Image rectification



(a) Original images. (b) Rectified images.

Figure 1: Image rectification.

Given our two raw images as in Figure 1a, the first step consists in aligning their reference system. The *image rectification* process allows to rotate one image around the center of projection **C** or **C'** and translate it in order to translate its reference system and match the other. The aim of this computation is to align the two images so that the two image planes are parallel and the vertical coordinates of conjugate points are the same. By performing this step, the amount of possible conjugate points is reduced since the target of our work is to find possible conjugate points. Conjugate points are the same points matching in relation to the two different images. From a mathematical perspective, there are two points lying on the two different input images: $\mathbf{m} = (u, v)$ and $\mathbf{m'} = (u', v')$. After image rectification, the aim is to get $v' = v$ for each of the points, **m** and **m'** lying on the two planes.

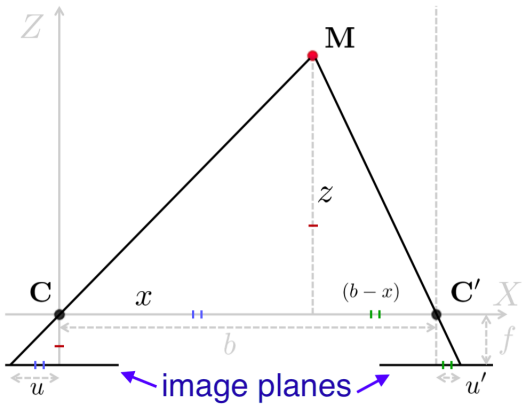


Figure 2: Rectified stereo system.

To achieve the rotation, a rotation matrix needs to be specified:

$$R = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix} \begin{cases} \mathbf{r}_1 = \dfrac{\tilde{\boldsymbol{C}}' - \tilde{\boldsymbol{C}}}{\left\| \tilde{\mathbf{C}}' - \tilde{\mathbf{C}} \right\|} \\ \mathbf{r}_2 = \mathbf{k} \times \mathbf{r}_1 \\ \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \end{cases} \tag{1}$$

where $\mathbf{k}$ is defined as an arbitrary vector, $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ are the rows of matrix R.

If, after a rotation $v \neq v'$, then a translation is applied, translating the reference system of one of the two images by operating on v'_0 of intrinsic matrix K'. After this operation, the two images are rectified as in Figure 1b. The images given in the dataset were rectified following the process explained above and are available in the *imagesStereo/modelX/file_ rectified/* folder.

2.2 Stereo system formalization

After applying image rectification, the solution proposed relies on stereo disparity computation. The disparity is the difference between the coordinates of the first and second image related to an axis. From a mathematical viewpoint, $d = u - u'$, with u and u' being the first coordinates of points **m** and **m'**. The two pixels **m**, left camera, and **m'**, right camera, are the projection of points **M** on the two image planes. $\mathbf{M} = [x, y, z]^T$ is projected on the two image planes, passing through the two centers of projections: **C** and **C'**. Both cameras have an equal focal length f, which is the distance from the image planes to their respective center of projection. f is related to intrinsic parameters. Furthermore, the two center of projections are at distance b, which is referred to as *baseline* and is related to an extrinsic parameter, in particular to the translation of the camera.

The aim of this model is to observe that, after computing the disparity, the coordinates of the 3D point $\mathbf{M} = [x, y, z]^T$ can be constructed.

As a matter of fact, from *projective equations* which are non linear in Cartesian coordinates. These equations link the 3D point **M** to the 2D points m and m':

$$\begin{cases} u = -\dfrac{f}{z}x \\ v = -\dfrac{f}{z}y \end{cases} \quad \begin{cases} u' = -\dfrac{f}{z}(x - b) \\ v' = v \end{cases} \tag{2}$$

We can now derive the x and z coordinates of $\mathbf{M}$. From u and u' we can derive z. $-\frac{u}{x} = -\frac{u'}{x-b}$ which implies $(x-b)u = xu'$. Lastly, the following can be written:

$$\begin{cases} x = -\frac{bu}{u'-u} = -\frac{bu}{d} \\ y = -\frac{bv}{d} = -\frac{bv'}{d} \\ z = \frac{bf}{u'-u} = \frac{bf}{d} \end{cases} \tag{3}$$

From these latter equations, it ensues that the coordinates of point $\mathbf{M}$ can be estimated. In particular, the z coordinate of point $\mathbf{M}$ can be calculated, owing to the disparity d and the product bf that can be given by camera calibration.

2.3 Disparity estimation

After computing image rectification, it is possible to associate a point **m** to a point **m'** through a process referred to as *disparity estimation*. There are two classes of strategies for disparity computation:

1. **Global methods:** which impose constraints on a small number of local pixels around the pixel to conjugate.
2. **Local methods:** which impose constraints on the whole scan line or image.

We will mainly focus on *block matching* method that is a local method used in *MATLAB* which resorts to *Sum of Absolute Differences (SAD)* as a coupling metric.

3 Block matching

Block matching algorithm is computed after image rectification and it is defined by the following steps:

1. Select a rectangular block of $(2n+1)(2m+1)$ pixels, or a square $(n = m)$ around each pixel $\mathbf{m} = (u, v)$ and its conjugate candidate $\mathbf{m'} = (u, v) = (u + d, v)$. It can be noticed that **m** and **m'** differ only by the u-axis coordinate by the disparity value d since the images were previously rectified.
2. It computes a coupling metric (later defined) between the block around **m** and the block around **m'**.
3. The operations above are repeated for different disparity values d in range $[d_{min}, d_{max}]$. d_{max} is the maximum disparity value, depending on the proximity of the closest object. d_{min} depends on the largest distance z to be measured. Usually d_{min} is set to 0, although it is possible to set it to non-zero so as to reduce the complexity of the estimation.
4. The best estimation of the disparity value $d_o(u, v)$ for point **m** is the value that minimizes (or maximizes, depending on the coupling metric) the specified coupling metric.

3.1 Disparity evaluation in MATLAB

In *MATALAB*, disparity evaluation can be computed using the following method:

```
disparityMap = disparity(I1,I2)
```

which returns the disparity map, gray scale image, for a pair of stereo images, I_1 and I_2 which need to be gray scale inputs. The size of the disparity map coincides with the size of the input images. In particular, *block matching* algorithm's square size (described in Step 1 of the previous section) can be set as an optional parameter. The *disparity range* can be set as well (as described in Step 3 of the previous section).

For instance, a $blockSize = 15$ and a $disparityRange = [0, 160]$ can be specified. The *MATLAB* method becomes:

```
disparityMap = disparity(I1,I2,'BlockSize',blockSize, ...
'DisparityRange',disparityRange);
```

MATLAB uses Sum of Absolute Differences (SAD) in its *block matching* algorithm as a coupling metric. *SAD* is a coupling metric based on image intensity differences. It is a measure of similarity between blocks of an image. *SAD* is calculated by taking the absolute difference between each pixel in the original block and the corresponding block in the other image. The resulting values of the matrix corresponding to each pixel which contain the differences are then summed. Formally, it can be defined as

$$SAD(u, v, d) = \sum_{k,l} |I_1(u + k, v + l) - I_2(u + k + d, v + l)| \tag{4}$$

SAD requires its output to be minimized so that the best disparity of point m is found by the following rule:

$$d_o(u, v) = \underset{d \in [d_{min}, d_{max}]}{\arg\min} \; SAD(u, v, d) \tag{5}$$

The advantages of using *SAD*, instead of other coupling metrics, consist in the following elements: it is less sensitive to impulsive noise, requiring a lower complexity (i.e. w.r.t *SSD* since *SAD* is parallelizable), effective on wide motion. Nevertheless, *SAD* may be unreliable in some cases due to: lighting, color, viewing directions.

In the following example, it is shown why *SAD* may be unreliable, in some cases.

$$I_1^{block} = \begin{bmatrix} 3 & 4 \\ 7 & 9 \end{bmatrix}, I_2^{block} = \begin{bmatrix} 3 & 7 & 4 \\ 9 & 7 & 9 \end{bmatrix} \tag{6}$$

$$I_1^{block} - I_2^{leftblock} = \begin{bmatrix} 0 & 3 \\ 2 & 2 \end{bmatrix}, I_1^{block} - I_2^{rightblock} = \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix} \tag{7}$$

In this example, $SAD(I_1^{block} - I_2^{leftblock}) = 7$ and $SAD(I_1^{block} - I_2^{rightblock}) = 4$, and since *SAD* is to be minimized, the left block is preferred over the right block. However, it can be noticed that if in the right block, there was one pixel (i.e. in position (1,2) of I_2^{block}) with a value $I_2^{block}(1, 2) > 10$ and all the other pixels matching with I_1^{block}, the estimation of *SAD* would probably be wrong.

4 Solution

To solve the problem, the disparity map, w.r.t a given *Ground Truth (GT)* (available in the dataset at imagesStereo/model1/GT/Labels/), of the original image was computed in order to have a numerical reference of the best accuracy and complexity. To do this, the disparity map was computed multiple times by changing the values of the *disparityRange* and the *blockSize*. The best values in terms of accuracy were then stored to obtain such reference. Subsequently, for each reduced resolution image, the same approach was adopted and the best values in terms of accuracy were stored corresponding to the specified *blockSize* and *disparityRange*. These two values, for each reduced resolution image, were later used in order to recompute the disparity map of the original image. This allows to compare the accuracy results given by the disparity map computed with reduced resolution *disparityRange* and *blockSize* and the reference accuracy. The complexity reduction was also measured and compared to the reference complexity, to obtain a better understanding of the usefulness of reducing the image resolution. Figure 3 shows the architecture of the solution.

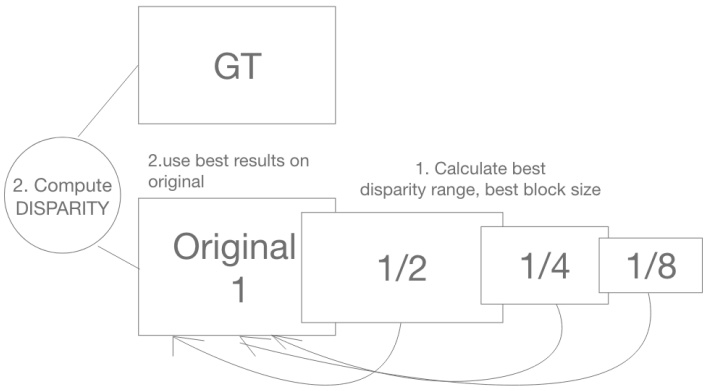


Figure 3: Solution architecture

5 Pseudocode

Data: Left and right rectified images
Result: Measure complexity reduction and accuracy
initialization;
resize = [1, 1/2, 1/4, 1/8];
originalRange = [64:16:160];
InitBlockSize = [41, 21, 11, 7];
for *i=1:4* **do**
  imResizeL = imresize(imL, resize(i));
  imResizeR = imresize(imR, resize(i));
  range = originalRange*resize(i);
  range = range(mod(range, 16) == 0);
  for *blockSize=5:2:InitBlockSize(i)* **do**
    for *disparityRangeEnd=range* **do**
      disparityRange = [0 disparityRangeEnd];
      disparityMap = disparity(imResizeL, imResizeR, 'BlockSize', blockSize ,'DisparityRange', disparityRange,);
      GTgrayResize = imresize(GTgray, resize(i));
      acc = sqrt(mean((GTgrayResize(indexCommon)-disparityMap(indexCommon))^2)
      save best results based on the accuracy;
    end
  end
end
for *i=1:4* **do**
  image = imresize(imL, resize(i));
  imSize = size(image);
  complexity(i) = bestRange(i)*imSize(1)*imSize(2);
end
disparityRange = [0 bestRange(i)/resize(i)];
disparityMap = disparity(imL, imR, 'BlockSize', bestBlockSize(i)/resize(i) - 1,'DisparityRange', disparityRange);
NewAcc = sqrt(mean((GTgray(indexCommon)-disparityMap(indexCommon))^2))
mseReduction = 100 - bestAcc(1)*100/NewAcc;
reduction = 100 - complexity(i)*100/complexity(1);

Algorithm 1: Stereo disparity computation from projected random dot pattern

6 Results

Before explaining how results were obtained, some prior knowledge needs to be taken into account. For proper results, the *ground truth* had to be rescaled by a scale factor of 3.5 for $model1$ and 2.5 for $model2$ and $model3$.

In the problem analysed, the *mean squared error (MSE)* is used as a measure to compare the results of the GT and the disparity maps computed. In statistics, the MSE of an estimator measures the average of the squares of the errors that is equivalent to the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. The MSE is also referred to as accuracy in this work. In this project, the MSE is calculated as

$$MSE = \frac{1}{h * w} \sum_{i=1}^{h*w} (GT_i - DM_i)^2 \tag{8}$$

where $GT = GTgrayResize$ and $DM = disparityMap$ in the *pseudocode* of the previous section. In addition, h and w are the height and width of the image, respectively.

The complexity, instead, refers to the amount of operations performed by the algorithm when computing the *disparity map*. It is calculated as:

$$C = disparityRange * h * w; \tag{9}$$

An analysis was made of the results for three versions of the images given in the dataset available in each of the three models. Corresponding tables, disparity map plots and heat map plots were produced in the following pages.

From the results, it can be stated that for $model1$, using the first version, the accuracy calculated for the original image is close to the image rescaled by 1/2. As a matter of fact, the MSE increases only by 1.97%, but more importantly, the complexity computation is reduced by 87.50%. The same cannot be stated for the other reduced versions of the images because the MSE increases by a factor of 1/4, even though we obtain a complexity reduction of almost 100%. Likewise, the same applies for the other versions in $model1$ since worse results are achieved in terms of MSE just for the images reduced by 1/8.

Considering the first version in $model2$, the same results were not achieved as in $model1$. The values of the images rescaled by 1/4 perform better than the one rescaled by 1/2. In fact, the MSE increases, for the one rescaled by 1/4, by 19.01% and for the one rescaled by 1/2, by 27.22%. It has to be noticed that in the first version, the image resized by 1/2 has a significantly lower complexity reduction than in the other versions for the same image resize.

In $model3$, other different results were obtained. The images rescaled by 1/2 and 1/4 perform more or less similarly in all versions. The MSE increases by almost 10% and there is a complexity reduction above 95%.

To conclude, in all models, the images rescaled by 1/8 have the worst performances in terms of MSE, as expected. In the first model the best results are obtained by reducing the image size by 1/2. In the second model, reducing the image size by 1/4 gives better results both considering the tradeoff between MSE increase and the complexity reduction. In the third model, the best results are given by either the resize by 1/2 or 1/4, depending on the aim, i.e. to achieve better MSE or better complexity reduction, respectively.

6.1 Model 1 results

version	best block	best range	MSE	complexity
1	23	160	18.805925	49152000.00
2	25	48	18.861580	14745600.00
3	23	160	18.785856	49152000.00

Table 1: Model 1: reference values.

version	resize	best block	best range	MSE	MSE increase	complexity reduction
1	0.50	21	80	19.183420	1.97 %	87.50 %
1	0.25	11	32	25.222807	25.44 %	98.75 %
1	0.12	5	16	24.959181	24.65 %	99.84 %
2	0.50	21	80	19.244629	1.99 %	58.33 %
2	0.25	11	32	25.401165	25.75 %	95.83 %
2	0.12	7	16	33.756184	44.12 %	99.48 %
3	0.50	21	80	19.240854	2.36 %	87.50 %
3	0.25	11	32	24.759109	24.13 %	98.75 %
3	0.12	7	16	33.770298	44.37 %	99.84 %

Table 2: Model 1: reduced resolution images.

6.2 Model 2 results

version	best block	best range	MSE	complexity
1	25	48	19.821085	14745600.00
2	25	48	19.853476	14745600.00
3	25	48	19.797880	14745600.00

Table 3: Model 2: reference values.

version	resize	best block	best range	MSE	MSE increase	complexity reduction
1	0.50	21	80	27.234734	27.22 %	58.33 %
1	0.25	11	16	24.474241	19.01 %	97.92 %
1	0.12	7	16	41.277653	51.98 %	99.48 %
2	0.50	21	32	25.729816	22.84 %	83.33 %
2	0.25	11	16	24.447748	18.79 %	97.92 %
2	0.12	7	16	41.246197	51.87 %	99.48 %
3	0.50	21	32	25.792824	23.24 %	83.33 %
3	0.25	11	16	24.456842	19.05 %	97.92 %
3	0.12	7	16	41.248436	52.00 %	99.48 %

Table 4: Model 2: reduced resolution images.

6.3 Model 3 results

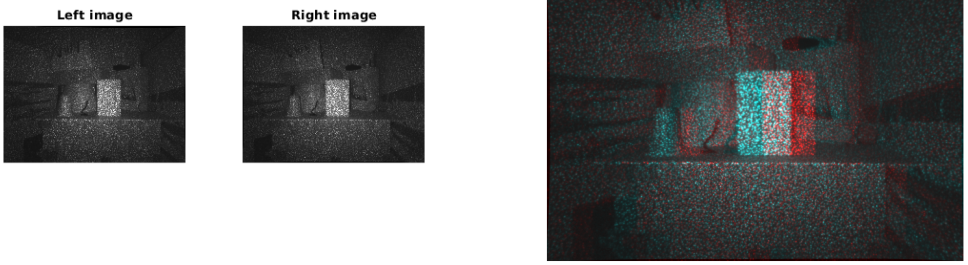
version	best block	best range	MSE	complexity
1	23	160	20.276314	49152000.00
2	23	160	20.142399	49152000.00
3	23	160	20.181196	49152000.00

Table 5: Model 3: reference values.

version	resize	best block	best range	MSE	MSE increase	complexity reduction
1	0.50	7	32	22.636173	10.43 %	95.00 %
1	0.25	11	16	23.067793	12.10 %	99.38 %
1	0.12	7	16	28.922523	29.89 %	99.84 %
2	0.50	7	32	22.678537	11.18 %	95.00 %
2	0.25	11	16	23.074764	12.71 %	99.38 %
2	0.12	7	16	28.875351	30.24 %	99.84 %
3	0.50	7	32	22.630680	10.82 %	95.00 %
3	0.25	11	16	23.046076	12.43 %	99.38 %
3	0.12	7	16	28.946297	30.28 %	99.84 %

Table 6: Model 3: reduced resolution images.

6.4 Model 1 images



(a) Left and right images.

(b) Stereo anaglyph.

Figure 4: Starting point.

(a) Computed with resize 1 best values.

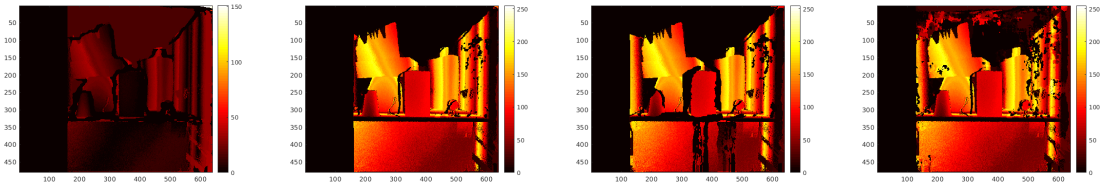
(b) Computed with resize 1/2 best values.

Figure 5: First disparity set.

(a) Computed with resize 1/4 best values.

(b) Computed with resize 1/8 best values.

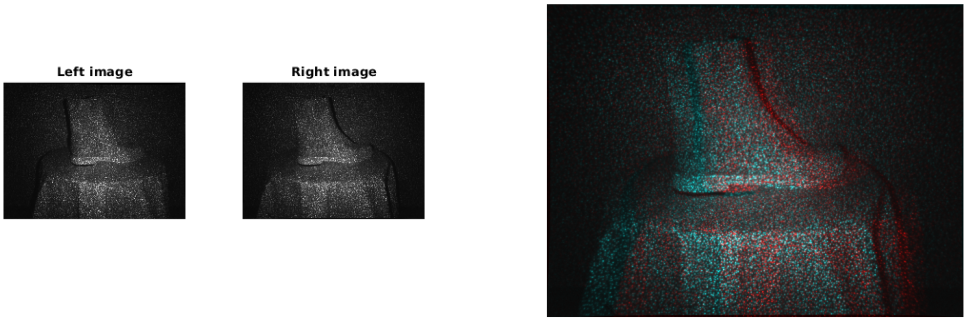
Figure 6: Second disparity set.



(a) Using resize 1 best value. (b) Using resize 1/2 best value. (c) Using resize 1/4 best value. (d) Using resize 1/8 best value.

Figure 7: Heat maps.

6.5 Model 2 images



(a) Left and right images. (b) Stereo anaglyph.

Figure 8: Starting point.

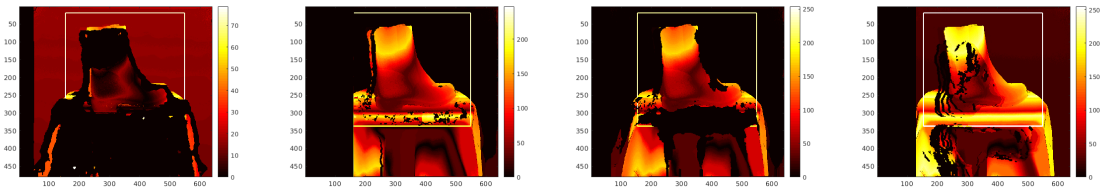
(a) Computed with resize 1 best values. (b) Computed with resize 1/2 best values.

Figure 9: First disparity set.

(a) Computed with resize 1/4 best values.

(b) Computed with resize 1/8 best values.

Figure 10: Second disparity set.



(a) Using resize 1 best value.

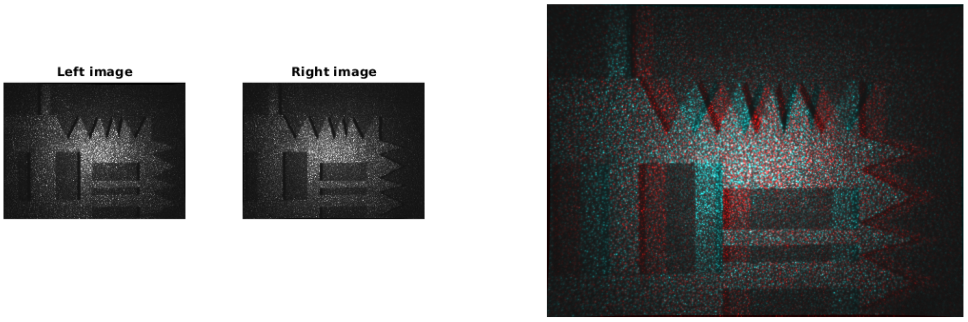
(b) Using resize 1/2 best value.

(c) Using resize 1/4 best value.

(d) Using resize 1/8 best value.

Figure 11: Heat maps.

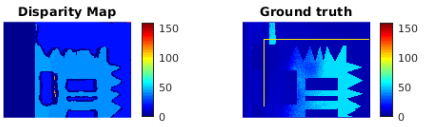
6.6 Model 3 images



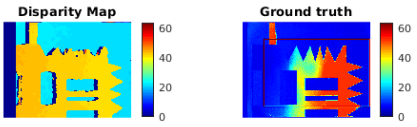
(a) Left and right images.

(b) Stereo anaglyph.

Figure 12: Starting point.

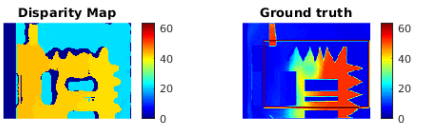


(a) Computed with resize 1 best values.

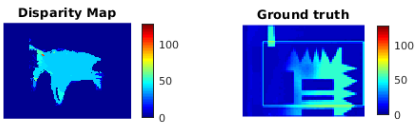


(b) Computed with resize 1/2 best values.

Figure 13: First disparity set.

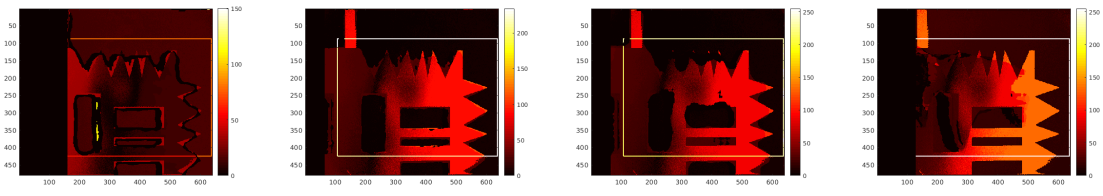


(a) Computed with resize 1/4 best values.



(b) Computed with resize 1/8 best values.

Figure 14: Second disparity set.



(a) Using resize 1 best value. (b) Using resize 1/2 best value. (c) Using resize 1/4 best value. (d) Using resize 1/8 best value.

Figure 15: Heat maps.