Stereo Matching – Distance Calculation

Presenters: Roman Revzin and Shlomo Sandowski

Introduction

Objective

In this project we will demonstrate an application of "**Stereo Image Matching**".

Computer stereo vision is the extraction of 3D information from digital images, such as those obtained by a CCD camera. By comparing information about a scene from two vantage points, 3D information can be extracted by examining the relative positions of objects in the two panels. This is similar to the biological process of stereopsis.

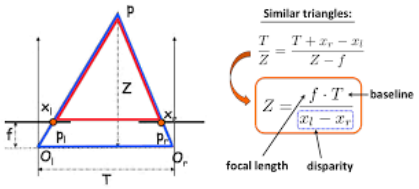
We will strive to automate the process of obtaining a depth map with minimum error from two images captured by a single camera from parallel view points and measured distance.

Overview

Pre-processing :

- Camera calibration - extracting the cameras intrinsic parameters.
  The images may be taken from two preset cameras or a single camera from a measured distance.
- Image rectification – the projection of the images to a common plane.

Processing :



- Disparity Map extraction – find matching pixels in both rectified images and calculate the disparity.
- Disparity Map post-processing – the initial disparity map may contain errors due to wrongly matched pixels or image distortion from lighting or the sort.
  We will try and remove these by filtering and applying Morphological binary operations.
- Disparity to Depth conversion – using the measured values of camera distances(or image distance) and focal length of camera we can calculate the depth from disparity using geometric properties $depth = \frac{focal\ length * base\ distance}{disparity}$.

Assumptions

In order to achieve optimal results, the following factors should be minimized :

- Monochromatic uniform objects and few objects(not a lot of changes in the images).
- Image lens distortion
- Uneven lighting
- Images non-overlapping area
- Images of different sizes
- Distance of objects in relation to base distance and focal length

Similar Projects and inspiration

Image rectification from what we have seen is usually done more or less the same.
The main difference in the process of rectification was the algorithm used for feature extraction. The algorithms relied on edges and changes within objects corner points, orientation, locality .

Disparity calculation was also very similar in most applications we saw. The algorithms generally ran a window of a certain size along epipolar lines in the images and found the best match using lowest cost calculations, correlation and local disparity.
Another option is to use in addition global information, but this may come at the expense of
higher complexity.

During our research we came across simple projects of finding the depth of an object in images containing a single object to very complex and high accuracy full depth maps.
The majority of projects had very accurate calibration and high-end equipment.

After studying and replicating MATLAB's disparity map and rectification examples we tried to apply them to various datasets online with great success.

We also compared different algorithms with MATLAB's examples to find the most suitable algorithms for our dataset and restrictions.

Project

Stage I

The first stage involved camera calibration and finding the cameras intrinsic values particularly the focal length.

We used MATLAB's single camera calibration applications since we did not have the equipment for a dual camera setup.

We prepared a data set of image pairs taking into account the disturbances the distance between the cameras and the distance to the objects.

Stage II

In this stage we projected the image pair onto a single plain.

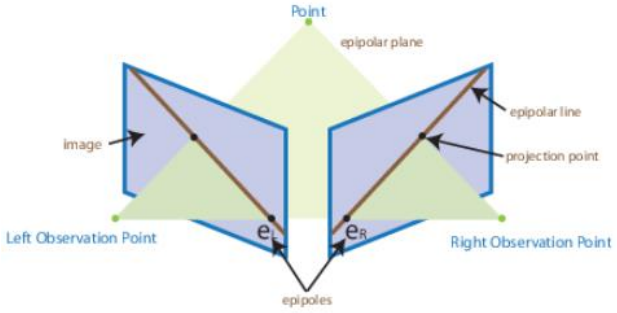
In order to do this, we needed to extract unique features to construct the Fundamental matrix (global warping).

The matrix is a 3X3 matrix that maps the points from one image to the other.

Most of the algorithms we saw for feature extractions utilized detections of points containing unique edges, orientation and corners in particular for example :SURF(speeded up robust features),
SIFT, FREAK, SURF, ORB .

After the feature extraction we matched the points with similar features in both images and constructed the fundamental matrix.

Once we had the fundamental matrix, we were able to project the images onto a single plane.



MATLAB code:

RectifyImages:

MATLAB functions use and explanation found in code comments:

- *detectSURFFeatures*
- *extractFeatures*
- *matchFeatures*
- *estimateFundamentalMatrix*
- *isEpipoleInImage*
- *showMatchedFeatures*
- *estimateUncalibratedRectification*
- *rectifyStereoImages*

```
function [I1Rect,I2Rect,img] = rectifyImages(I1,I2)
%RECTIFYIMAGES performes stereo image rectification using matching detected SURF
%points in both images, calculating the fundumental matrix and applying the
%affine transformation retained from it on both images.
% input I1,I1: left, right stereo images.
% accepts rgb or grayscale
% output I1Rect,I2Rect : rectified images I1,I2 respectfully
%        img : 3D anaglyph from images with matching points
%  the function requires at least 8 matching points in order to return a
%  result.

% convert images to grayscale
I1gray = rgb2gray(I1);
I2gray = rgb2gray(I2);

% detect points of interest on both pictures using SURF algorithm
% metric threshold is set to 2000 from trial and error.
% represents strongest feature threshold (the larger the threshold less
% points are found)
blobs1 = detectSURFFeatures(I1gray, 'MetricThreshold', 2000);
blobs2 = detectSURFFeatures(I2gray, 'MetricThreshold', 2000);

% extrace valid points with their features from image
[features1, validBlobs1] = extractFeatures(I1gray, blobs1);
[features2, validBlobs2] = extractFeatures(I2gray, blobs2);

% find corresponding pairs of indexes in both feature arrays
% using sum of absolute differences
% metric threshold represents the percent of the distance from a perfect
% match
indexPairs = matchFeatures(features1, features2, 'Metric', 'SAD', ...
  'MatchThreshold', 5);

% get corresponding pixels of both images
matchedPoints1 = validBlobs1(indexPairs(:,1),:);
matchedPoints2 = validBlobs2(indexPairs(:,2),:);

% get estimadted fundamental matrix for projection onto the same plain
```

```
% using a random sample consesus method with 10000 itterations
% distance threshold and confidance used for RANSAC algorithem which also
% requires at least 8 points
[fMatrix, epipolarInliers, status] = estimateFundamentalMatrix(...
  matchedPoints1, matchedPoints2, 'Method', 'RANSAC', ...
  'NumTrials', 10000, 'DistanceThreshold', 0.1, 'Confidence', 99.99);

% check if there are enough points to perform rectification if epipole is
% in image (using parallel cameras the epiples should be outside of images)
if status ~= 0 || isEpipoleInImage(fMatrix, size(I1)) ...
  || isEpipoleInImage(fMatrix', size(I2))
  error(['Either not enough matching points were found or '...
         'the epipoles are inside the images. You may need to '...
         'inspect and improve the quality of detected features ',...
         'and/or improve the quality of your images.']);
end

% get corresponding points to build visualization
inlierPoints1 = matchedPoints1(epipolarInliers, :);
inlierPoints2 = matchedPoints2(epipolarInliers, :);

% disallow showing created figures
set(0,'DefaultFigureVisible','off')
% returns handler, which has no arrows on the image
% also, opens figure, which is suppresed by previous
showMatchedFeatures(I1, I2, inlierPoints1, inlierPoints2);
% save axes as image for passing image with arrows
img = frame2im(getframe(gca));
% close invisible figure
close(gcf)
% restore default behaviour
set(0,'DefaultFigureVisible','on')

% get projective transformations of non calibrated stereo images
% for rectification
[t1, t2] = estimateUncalibratedRectification(fMatrix, ...
  inlierPoints1.Location, inlierPoints2.Location, size(I2));

% perform rectification of stereo images
tform1 = projective2d(t1);
tform2 = projective2d(t2);
[I1Rect, I2Rect] = rectifyStereoImages(I1, I2, tform1, tform2);
end
```

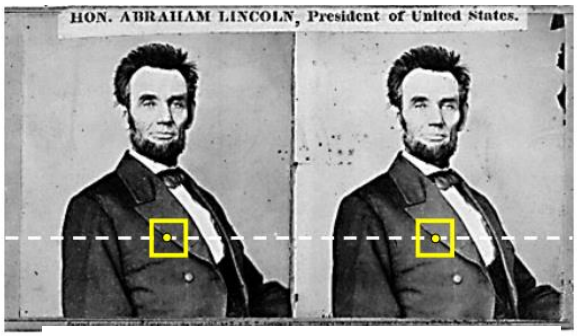
Stage III

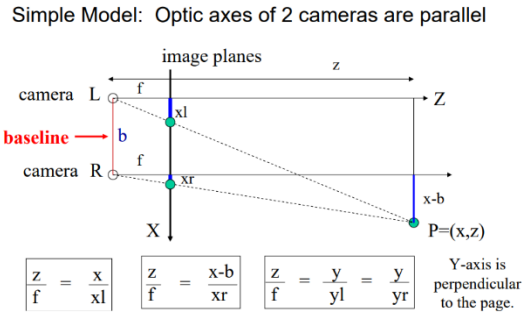
Now that we have the rectified image pair, we can use epipolar geometry and instead of finding where each pixel is positioned in both images over their entire surface ,we can search on parallel lines instead.

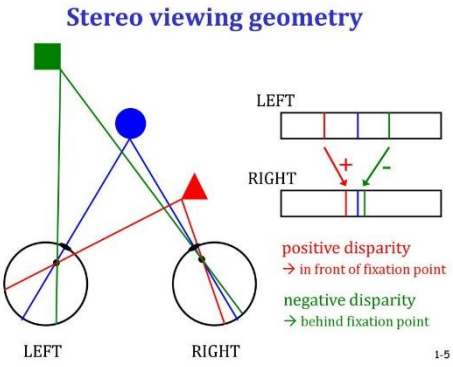
Finding matching points along the epipolar lines can be done using window/ block matching with differing window sizes and with different constraints on uniqueness of the block features in its area and minimum cost functions.

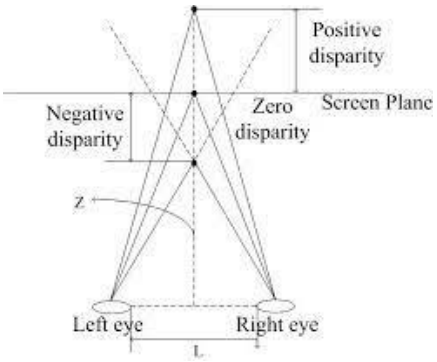
For each matching pixel we calculate the disparity and create a disparity map.

From here we can extract the Depth Map using geometry based on the pinhole camera utilizing the focal length of the camera and the distance between the cameras or in our case between both image taking locations.









MATLAB code:

Disparity:

MATLAB functions use and explanation found in code comments:

- *disparitySGM*
- *medfilt2, imopen, imclose, imfill*

```
function disparityMap = disparity(rect1,rect2)
%DISPARITY calculates the disparity map from two rectified images
% input rect1,rect2 : left, right rectified stereo images.
% accepts rgb or grayscale
% output disparityMap : single nxm matrix with disparity values for each
% pixel

% convert images to grayscale
J1 = rgb2gray(rect1);
J2 = rgb2gray(rect2);

% pick disparity range
% because the rectification was done using uncalibrated rectification we
% get negative disparity values
disparityRange = [-64 64];

% compute disparity map with semi-global matching
% uniquenessThreshold obtained from trial and error dictates the uniqueness
% threshold for retaining the disparity
disparityMap =
disparitySGM(J1,J2,'DisparityRange',disparityRange,'UniquenessThreshold',20);

% shift disparity range to positive range
disparityMap = disparityMap - min(disparityMap,[],"all");

% filter salt and pepper noise
% disparity values that are very localized and therefore unreliable
disparityMap = medfilt2(disparityMap,[3 3]);

% opening and closing to close large areas with same disparity under the
% assumption that disparity values are probably similar in very near
% proximity
disparityMap = imopen(disparityMap,strel("square",10));
disparityMap = imclose(disparityMap,strel("square",10));

% replace NaN values or unreliable disparity values with zeros
disparityMap(isnan(disparityMap)) = 0;

% fill holes in objects on the map
disparityMap = imfill(disparityMap,'holes');
end
```

Disparity2depth:

```
function depthMap = disparity2depth(base_distance,focal_length, disparityMap)
%DIPARITY@DEPTH convert each disparity value to distance/depth based on
% camera intrinsic parameters and distance between cameras
% input base_distance : the distance between cameras in length units
%       focal_lenght : the cameras focal length in pixels assuming both images
%           were takin with same camera(type)
%       disparityMap : stereo images calulated disparity map
% output depthMap : a matrix of same dimentions as disparity map with
%           pixels calculated depth distance
    depthMap = base_distance*focal_length./disparityMap;
end
```

Results

Algorithm Name	Execution Time
rectifyImages	0.86854 s
disparitySGM	0.31082 s
shiftMean	0.0012515 s
medfilt2	0.016691 s
imopen	0.011958 s
imclose	0.008418 s
isnan	0.0037455 s
imfill	0.021518 s
Total	1.243 s

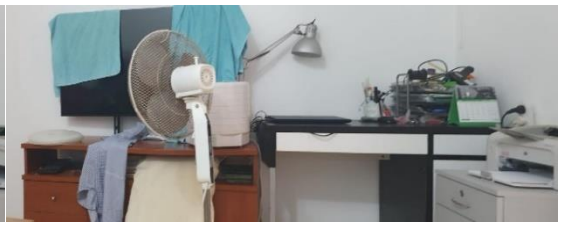
First set:

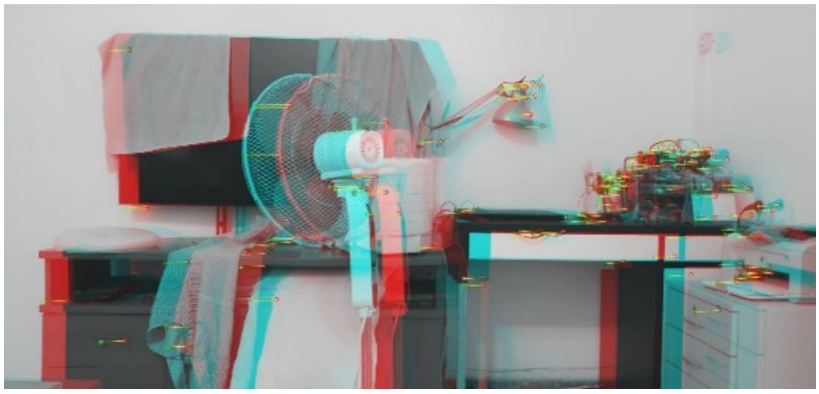
Size: 1600x757

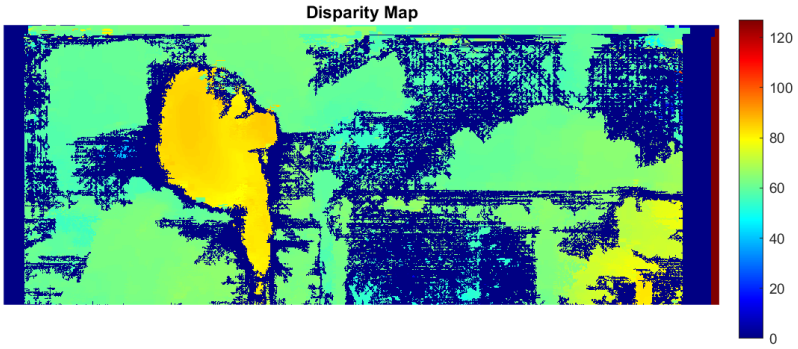
Focal length: 1300 pixel

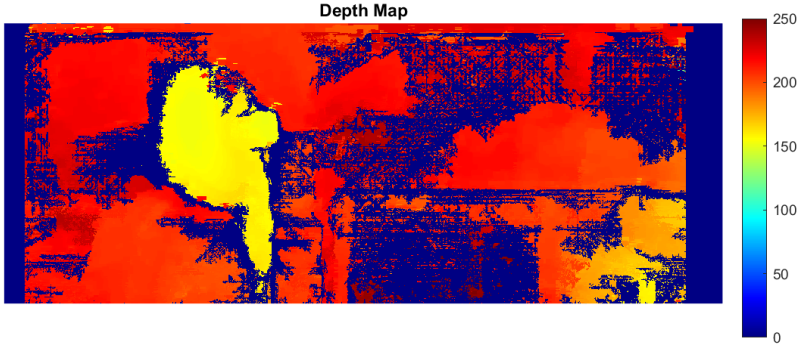
Base distance:10 cm

This set is characterized by having many changes that added to the number of matched features in the images for both the image rectification and disparity processes.









Algorithm Name	Execution Time
rectifyImages	0.57566 s
disparitySGM	0.29808 s
shiftMean	0.0011686 s
medfilt2	0.0016352 s
imopen	0.0097473 s
imclose	0.0075734 s
isnan	0.0035495 s
imfill	0.019609 s
Total	0.917 s

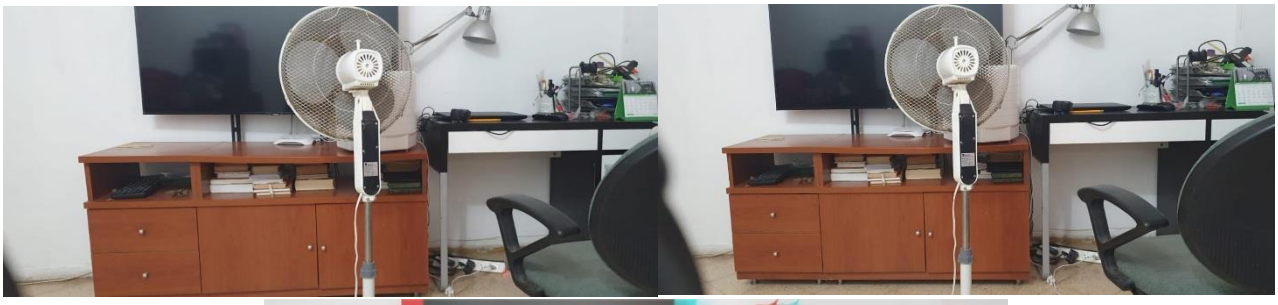
Second set:

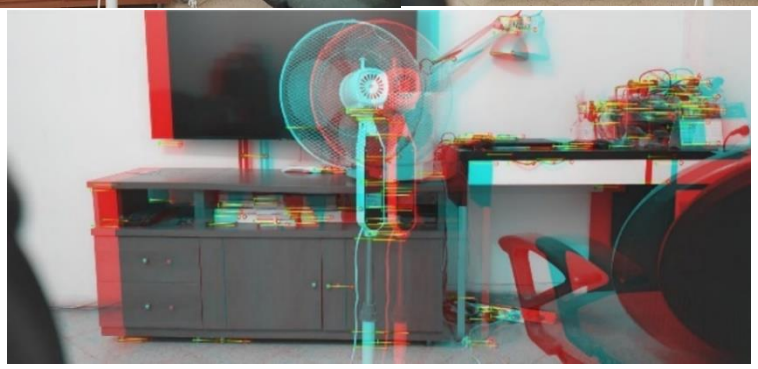
Size: 1600x757

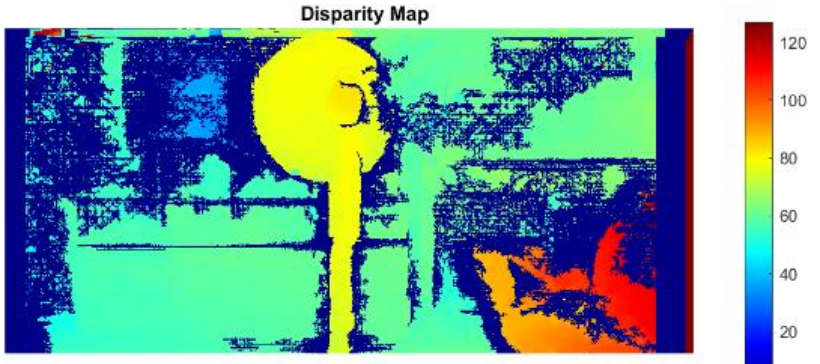
Focal length: 1300 pixel

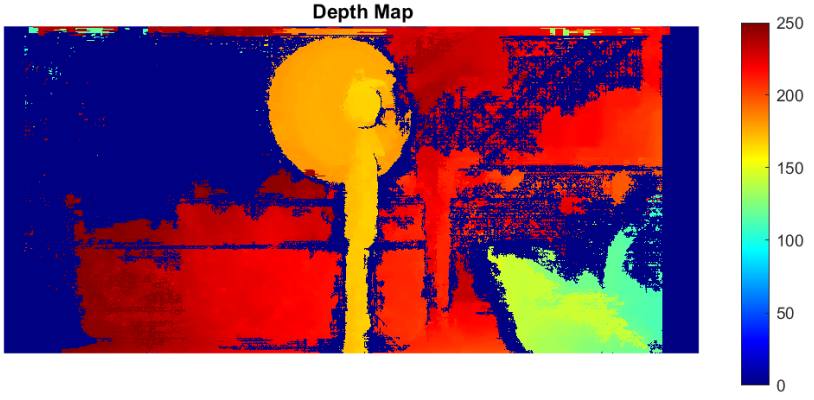
Base distance:10 cm

This set is characterized by having less changes then the first reducing to the number of matched features in the images for both the image rectification and disparity processes.









Algorithm Name	Execution Time
rectifyImages	0.61943 s
disparitySGM	0.30111 s
shiftMean	0.001496 s
medfilt2	0.0015684 s
imopen	0.010208 s
imclose	0.012323 s
isnan	0.0037257 s
imfill	0.021253 s
Total	0.971 s

Third set:

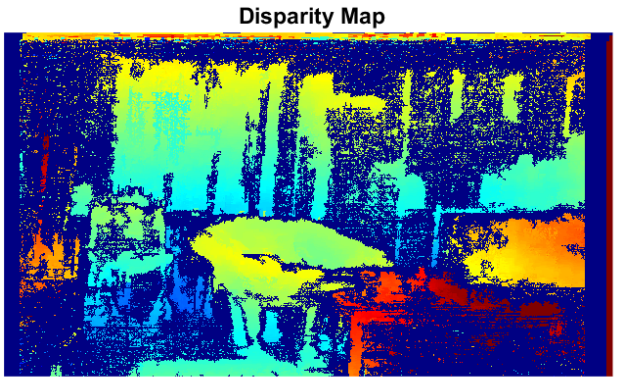
Size: 1920x1080

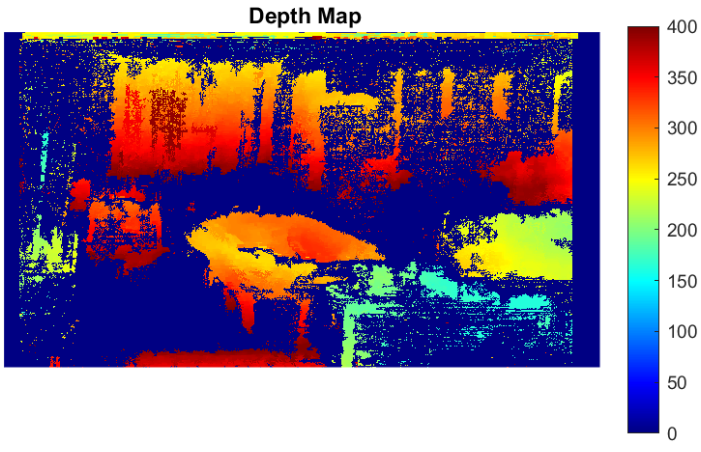
Focal length: 1300 pixel

Base distance:16 cm

This set is characterized by having many changes but similar coloring of objects ,uneven lighting and larger dimensions.

The coloring and lighting reduced the number of matched pixels although the size all of which affected the rectification and disparity processes.





Summarizing discussion

The purpose of the project was to create a reliable Depth-Map from two stereo images.

After trying multiple approaches and algorithms we managed to create for our three pair dataset a Depth-Map that shows with minimum error the depth of most objects in the images(due to the fact that we do not have an exact depth map we discerned errors from a couple of objects whose distance we measured manually).
Each of set in our dataset emphasizes different limitation and obstacles that correspond to are original assumptions.
It is clear from the runtime measurement that the bulk of the processing time is due to image rectification.
This time can be greatly reduced with pre-calibrated fixed stereo cameras due to the fact that we wouldn't need to calculate to fundamental matrix with each pair of images with feature extraction.
The camera lens distortion and angles would be constant.

Once rectified obtaining the Disparity Map presented a trade of situation. Scanning both images for matching pixels with a small window reduces complexity on the other hand we reduce errors by finding the most accurate match.
Another observation was that for different depths ranges we needed different distances between the cameras.
The further the object is from the cameras to smaller the disparity and therefore as can be seen in the disparity to depth equation $depth = \frac{focal\_lenght * base\_distance}{disparity}$ unless the cameras focal length is variable, we are left with increasing the base distance to get accurate values.

To summarize the two most significant factor affecting the quality of acquiring a depth map using stereo imaging are:

1. The number of changes in the image affected by the number of objects and their uniformity, increases dramatically the ability to ascertain disparities and therefore depth.
   In our application we saw that lack of changes led to either inability to create the map or more errors and undefined areas.
2. Using a single camera as apposed to a fixed structure of stereo cameras increases the complexity and reduces accuracy of process.

Sources

- https://www.mathworks.com/help/
- Course lectures and assignments
- https://www.wikipedia.org
- Random lectures and projects online
- https://www.cs.cmu.edu/~16385/s17/Slides/13.2_Stereo_Matching.pdf

Appendices

Main file :

The rest of the functions called in main are shown in previous sections.

To run the main file make sure that the function files and the images folder are in the same directory.

```
clear,clc,close all force;
% main file for testing

%hold algorithm times
allTimes = zeros(1, 8);


I1 = imread('.\images\GoodOneLeft.jpeg');
I2 = imread('.\images\GoodOneRight.jpeg');

tic
% image rectification
[rect1, rect2] = rectifyImages(I1,I2);
allTimes(1) = toc;

J1 = rgb2gray(rect1);
J2 = rgb2gray(rect2);
disparityRange = [-64 64];

tic
% disparity map calculation
disparityMap =
disparitySGM(J1,J2,'DisparityRange',disparityRange,'UniquenessThreshold',20);
allTimes(2) = toc;

tic
% shifting disparity range to positive values due to uncalibrated
% rectification
disparityMap = disparityMap - min(disparityMap,[],"all");
allTimes(3) = toc;
```

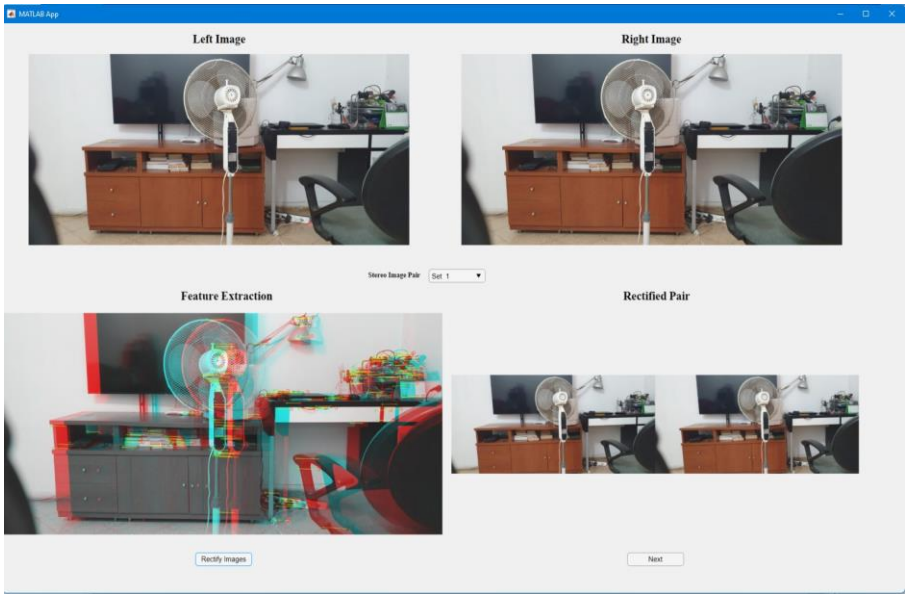
```
tic
% removing errors in disparity map due to faulty algorithm mistakes and
% image distortions
disparityMap = medfilt2(disparityMap,[3 3]);
allTimes(4) = toc;
tic
disparityMap = imopen(disparityMap,strel("square",10));
allTimes(5) = toc;

tic
disparityMap = imclose(disparityMap,strel("square",10));
allTimes(6) = toc;

tic
disparityMap(isnan(disparityMap)) = 0;
allTimes(7) = toc;

tic
disparityMap = imfill(disparityMap,'holes');
allTimes(8) = toc;


figure
imshow(disparityMap,[]);
title('Disparity Map')
colormap jet
colorbar
% camera and image location values
base_distance = 10;
focal_length = 1300;
depth_map = base_distance*focal_length./disparityMap;
figure
max_dist = 400;
depth_map(depth_map>max_dist)=nan;
imshow(depth_map,[0 400]);
title('Depth Map')
colormap jet
colorbar
figure
show = zeros(size(J2));
bottom_range = 250;
top_range = 330;
range = find(depth_map < top_range & depth_map > bottom_range);
show(range) = rect2(range);
imshow(show,[])

algLabels = {'rectifyImages','disparitySGM','shiftMean','medfilt2',...
             'imopen', 'imclose', 'isnan', 'imfill'};
times = strcat(string(allTimes)',' s');

t = table(algLabels',strcat(string(allTimes)',' s'),'VariableNames',...
    {'Algorithm Name', 'Execution Time'})

writetable(t, 'executionTime3.csv')
```

Stereo imaging App :

To run the app from MATLAB you must be in the same directory as Stereo_Image_App.m, Stereo_Image_App_SecondStep.m

and images folder.

First page:



Second page: