A black background with orange and blue text

Description automatically generated



**Assignment Report  
CPV301 – AI1801**

**STEREO DEPTH ESTIMATION**

**Mentor: Nguyen Dinh Vinh**

**Can Tho, 11/2024**

**Group Project**

**Group 2:**

| **Name of group member** | **Contribution (%)** |
| --- | --- |
| 1. Nguyen Van Phong (Leader) | 100% |
| 2. Huynh Anh Phuong | 100% |
| 3. Dao Anh Khoa | 100% |
| 4. Huynh Ngoc Nhu Quynh | 100% |
| 5. Tran Trung Nhan | 100% |

**Table of Contents**

[**I. Introduction** 4](#_heading=h.gjdgxs)

[**1. Introduction to the Problem** 4](#_heading=h.1fob9te)

[**2. Practical Applications of Depth Estimation** 4](#_heading=h.2et92p0)

[**3. Related Research and Achievements in the Past 5-7 Years** 4](#_heading=h.3dy6vkm)

[3.1.](#_heading=h.4d34og8) Research Trends 4

[3.2.](#_heading=h.17dp8vu) Key Results and Achievements 6

[**II. Mathematical background** 7](#_heading=h.26in1rg)

[**1. Epipolar Geometry** 7](#_heading=h.35nkun2)

[1.1. Definition 8](#_heading=h.44sinio)

[1.2. Epipole or epipolar point 9](#_heading=h.z337ya)

[1.3. Epipolar line 9](#_heading=h.1y810tw)

[1.4. Epipolar plane 10](#_heading=h.2xcytpi)

[1.5. Epipolar constraint 10](#_heading=h.3whwml4)

[1.6. Disparity and Depth map 11](#_heading=h.qsh70q)

[**2. Essential matrix** 12](#_heading=h.1pxezwc)

[2.1. Coordinate representation 12](#_heading=h.2p2csry)

[2.2. Set up camera framework 13](#_heading=h.3o7alnk)

[2.3. Essential matrix derivation 13](#_heading=h.ihv636)

[2.4. Essential matrix mapping 14](#_heading=h.1hmsyys)

[2.5. Essential matrix kernel 15](#_heading=h.2grqrue)

[**3. Fundamental matrix** 15](#_heading=h.3fwokq0)

[3.1. Camera matrix 15](#_heading=h.4f1mdlm)

[3.2. Fundamental matrix derivation 15](#_heading=h.19c6y18)

[3.3. Properties of fundamental matrix 16](#_heading=h.28h4qwu)

[**4. The Eight-Point algorithm** 16](#_heading=h.37m2jsg)

[4.1. Formulating a homogeneous linear equation 16](#_heading=h.46r0co2)

[4.2. Solving the equation 17](#_heading=h.111kx3o)

[4.3. Enforcing the internal constraint 18](#_heading=h.206ipza)

[4.4. Normalized algorithm 18](#_heading=h.2zbgiuw)

[**III. Solution** 20](#_heading=h.sqyw64)

[**1. calibration.py** 21](#_heading=h.1rvwp1q)

[1.1. def draw\_keypoints\_and\_match(img1: MatLike, img2: MatLike, nfeatures: int = 500) -> Tuple[NDArray, NDArray, MatLike]: … 22](#_heading=h.2r0uhxc)

[1.2. def compute\_Fundamental\_matrix(kp1\_list: NDArray, kp2\_list: NDArray) -> NDArray: … 22](#_heading=h.3q5sasy)

[1.3. def RANSAC\_F\_mat(kp1\_list: NDArray, kp2\_list: NDArray, max\_inliers=20, threshold=0.05, max\_iter=1000) -> NDArray: … 22](#_heading=h.kgcv8k)

[1.4. def compute\_Essential\_matrix(F\_mat: NDArray, K1: NDArray, K2: NDArray) -> NDArray: … 22](#_heading=h.1jlao46)

[1.5. def drawlines(img1src: MatLike, img2src: MatLike, lines, pts1src, pts2src, random\_seed=0) -> tuple[MatLike, MatLike]: … 22](#_heading=h.2iq8gzs)

[**2. correspondence.py** 23](#_heading=h.3hv69ve)

[2.1. def sum\_of\_squared\_diff(pixel\_vals\_1, pixel\_vals\_2): 23](#_heading=h.4h042r0)

[2.2. def block\_comparison(y, x, block\_left, right\_array, block\_size, x\_search\_block\_size, y\_search\_block\_size): 23](#_heading=h.1baon6m)

[2.3. def ssd\_correspondence(img1, img2): 23](#_heading=h.2afmg28)

[**3. depth.py** 23](#_heading=h.39kk8xu)

[3.1. def disparity\_to\_depth(baseline, f, img): 23](#_heading=h.48pi1tg)

[**4. Notebooks:** 23](#_heading=h.1302m92)

[**IV. Evaluation** 24](#_heading=h.2250f4o)

[**1. The results achieved** 24](#_heading=h.319y80a)

[1.1 Dataset used (Middle Blurry Q-Size) 24](#_heading=h.40ew0vw)

[1.2. Estimating Fundamental matrix 24](#_heading=h.upglbi)

[1.3. Disparity map 25](#_heading=h.1tuee74)

[**2. Compare with the results of the group on the same topic** 27](#_heading=h.2szc72q)

[**3. Previous studies** 27](#_heading=h.3s49zyc)

[3.1. Calculating correspondences (SGM) 27](#_heading=h.meukdy)

[3.2. Disparity post Filtering (Fast Global Smoother Filter) 30](#_heading=h.1ljsd9k)

[3.3. Cencus Transform 32](#_heading=h.2koq656)

[**V: Conclusion:** 34](#_heading=h.3jtnz0s)

[**References** 34](#_heading=h.1yyy98l)

# **I. Introduction**

## **1. Introduction to the Problem**

The problem of depth estimation from images is a core challenge in the field of Computer Vision, with widespread applications across automated systems, self-driving cars, robotics, virtual reality (VR), augmented reality (AR), and more. By analyzing and determining the depth of objects within images, systems can better understand the spatial structure and depth of the real-world environment, allowing for intelligent decision-making and effective interactive responses.

Depth estimation from a single image (monocular depth estimation) is challenging because a single image provides only 2D data, lacking the stereo information available to human vision. Multi-image (stereo or multi-view depth estimation) approaches, on the other hand, leverage differences between viewpoints to reconstruct spatial depth. Both approaches demand complex and accurate algorithms to achieve optimal performance.

## **2. Practical Applications of Depth Estimation**

* Self-Driving Cars: Depth estimation enables cars to detect the distance to objects, pedestrians, and other vehicles, assisting with collision avoidance and precise navigation.
* Robotics: Robots use depth information to navigate and manipulate objects accurately, essential for tasks across industrial, healthcare, and home environments.
* AR/VR: In VR and AR, depth perception enhances realism by positioning objects accurately in 3D space, allowing for more natural and immersive interactions.

## **3. Related Research and Achievements in the Past 5-7 Years**

### Research Trends

**3.1.1. Deep Learning-Based Approaches**

* 1. CNN-Based Monocular Depth Estimation: Convolutional Neural Networks (CNNs) have significantly advanced single-image depth estimation. Architectures like U-Net, ResNet, and DenseNet help models learn crucial image features, producing detailed and accurate depth maps.
  2. Transformer-Based Models: In recent years, Transformers have been applied to depth estimation, with models like Vision Transformer (ViT) and Swin Transformer. These models leverage the spatial processing power of Transformers, improving depth estimation accuracy.
  3. Hybrid Approaches: Combining CNN and Transformer models has allowed for the best of both techniques: CNN’s spatial feature extraction and Transformers’ contextual processing capabilities.
     1. **Unsupervised and Self-Supervised Learning**
  4. Self-Supervised Depth Estimation: A prominent research direction is self-supervised learning, where models learn from image pairs or videos without requiring ground-truth depth labels. This approach reduces the dependency on manually labeled data, making it adaptable to various environments and datasets.
  5. Photometric Consistency Loss: A popular technique in self-supervised learning is using photometric consistency loss, based on matching pixel intensities between images from different views. This improves model accuracy without requiring labeled depth data.
     1. **Multi-Image Depth Estimation (Stereo/Multi-View)**
  6. Stereo Matching Techniques: For systems that capture multiple images, stereo matching techniques such as Semi-Global Matching (SGM), PatchMatch, and Cost Volume Processing allow models to extract depth information by comparing disparities between images.
  7. Multi-View Stereo (MVS): This technique uses multiple perspectives to determine object depth more accurately than monocular methods. Popular methods include Patch-based Multi-View Stereo (PMVS) and COLMAP.

### Key Results and Achievements

**3.2.1 Improvements in Accuracy and Performance**

Newer models using CNNs and Transformers have achieved higher accuracy while reducing processing time and optimizing memory use, enabling depth estimation systems to perform efficiently on mobile and low-power devices. For example, advanced models like MiDaS and DPT (Depth Prediction Transformer) have successfully applied single-image depth estimation.

**3.2.2 Real-World Applications**

Depth estimation algorithms have been successfully integrated into real-world applications, including Tesla’s self-driving technology, Apple’s Face ID, and Google’s ARCore, allowing for realistic face recognition and depth simulation features for users.

**3.2.3 Challenges and Potential Solutions**

Complex scenarios such as low light, transparent objects, or highly reflective surfaces still present challenges for depth estimation models. Current research focuses on developing advanced techniques like Image Denoising, Regularization, and Data Augmentation to enhance accuracy in complex environments.

# **II. Mathematical background**

A person posing for a picture with a leaning tower with Leaning Tower of Pisa in the background

Description automatically generated

## **1. Epipolar Geometry**

Epipolar geometry is the geometry of stereo vision. When two cameras view a 3D scene from two distinct positions, there are several geometric relations between the 3D points and their projections onto the 2D images that lead to constraints between the image points. These relations are derived based on the assumption that the cameras can be approximated by the pinhole camera model.

A group of white rectangular objects with circles and a orange ball

Description automatically generated

### 1.1. Definition

The figure below depicts two pinhole cameras looking at point X. In real cameras, the image plane is behind the focal center. However, we simplified the problem by placing a *virtual* image plane in front of the camera, since the real plane is symmetric about the focal center of the lens.

Each camera captures a 2D image of the 3D world. This conversion from 3D to 2D is referred to as a perspective projection and is described by the pinhole camera model.

A diagram of a complex math problem

Description automatically generated with medium confidence

### 1.2. Epipole or epipolar point

Each center projects onto a distinct point into the other camera’s image plane. These two image points, denoted by and , are called epipoles or epipolar points.

Ep`ipolar point is the intersection of line connecting two camera’s centers and with their image plane.

### 1.3. Epipolar line

Line projected on right camera creating a line called the epipolar line. Symmetrically, the line is seen by the right camera as a point and is seen as epipolar line by the left camera.

A collage of a model of a tower

Description automatically generated

The red lines are epipolar lines

When two image planes are parallel then the epipoles and are located at infinity. Then the epipolar lines are parallel to axis of image.

A diagram of a diagram of a diagram

Description automatically generated with medium confidence

### 1.4. Epipolar plane

form a plane, called epipolar plane. The epipolar plane and all epipolar lines intersect the epipoles regardless of where is located.

### 1.5. Epipolar constraint

If the relative position of the two cameras is known, this leads to two important observations:

Assume the projection point , the epipolar line and the point projects into the right image is known. A point which must lie on this particular epipolar line.

This provides an epipolar constraint: the projection of on the right camera plane xR must be contained in the epipolar line. All points e.g.  on the line will verify that constraint.

Epipolar constraints can also be described by the essential matrix or the fundamental matrix between the two cameras.

### 1.6. Disparity and Depth map

A diagram of a triangle with lines and points

Description automatically generated

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

So the depth would be:

Where:

* and are the distance between image points and their corresponding camera center.
* is the baseline, distance between two camera centers.
* is the focal length of both cameras (they should have the same).

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.

## **2. Essential matrix**

### 2.1. Coordinate representation

This derivation follows the paper by Longuet-Higgins.

For simplicity, we assume all the cameras are normalized and project the 3D world onto their respective image planes. i.e .

Let the 3D coordinates of a point P be and relative to each cmera’s coordinate system.

The mapping from the coordinates of a 3D point P to the 2D image coordinates of the point’s projection onto the image plane, according to the pinhole camera model, is given by:

More compactly as:

Where are the image 2D coordinate, are real proper 3D cooordinates but in two different coordinate systems.

### 2.2. Set up camera framework

A diagram of a complex structure

Description automatically generated with medium confidence

Let further assume world reference coordinate is associated with the first camera with the second camera offset by a rotaion matrix and 3-dimensional translation matrix . This implies:

And the camera matrix will be:

Or (because of cameras are normalized):

### 2.3. Essential matrix derivation

Since the vectors and lie in the epipolar plane, then if we take the cross product of , we will get a vector normal to the epipolar plane. This also means that which lies in the epipolar plane is normal to , giving us the constraint that their dot product is zero:

Reminder from linear algebra: The cross product between any two vectors a and b as a matrix-vector multiplication:

Combining this expression with Equation 6, we can convert the cross-product term into matrix multiplication, giving:

Then, the Essential Matrix is creating a com- pact expression for the epipolar constraint:

The Essential matrix is a 3 × 3 matrix that contains 5 degrees of freedom. It has rank 2 and is singular.

### 2.4. Essential matrix mapping

Different to a homography which maps a point to a point, an essential matrix maps a point to a line. Furthermore, let’s consider an epipolar line , with the form of , or in vector form:

Then it is easy to see that, from Equation 8:

### 2.5. Essential matrix kernel

Since every line on image plane pass epipolar. So , combine with Equation 9, we have . Furthermore, thus is normal with .

In short, essential kernel defines the epipole:

(points in normalized camera coordinates)

## **3. Fundamental matrix**

### 3.1. Camera matrix

How do you generalize to uncalibrated cameras? Recall the Equation 3:

First, we must define is a point from 3d world. We get two projected point on cameras:

Say we have canonical cameras transform space by a general homography matrix , then we have projections of to the corresponding camera images.

### 3.2. Fundamental matrix derivation

Recall that in the canonical case from Equation 7:

By substituting in the values of and , we get:

Let the matrix as the Fundamental Matrix which acts the same to the Essential matrix from previous but also encodes information about the camera matrices and and the relative translation T and rotation R between the cameras.

Therefore, it is also useful in computing the epipolar lines associated with p and p′, even when the camera matrices K, K′ and the transformation R, T are unknown.

### 3.3. Properties of fundamental matrix

Similar to the Essential matrix, we can compute the epipolar lines and from just the fundamental matrix and the corresponding points.

Fundamental matrix contains 7 degrees of freedom, while Essential matrix’s 5 degrees of freedom.

If we know the fundamental matrix, then simply knowing a point in an image gives us an easy constraint (the epipolar line) of the corresponding point in the other image. Therefore, without knowing the actual position of in 3D space, or any of the extrinsic or intrinsic characteristics of the cameras, we can establish a relationship between any and .

## **4. The Eight-Point algorithm**

### 4.1. Formulating a homogeneous linear equation

With each correspondent and

The constraint can be rewritten as:

That is represents the flatten Fundamental matrix vector and this vector must be orthogonal to vector .

Each pair of corresponding image points produces a vector . Given a set of 3D points corresponding to a set of vectors and all of them must satisfy:

Collect vector as the row of matrix and:

Where is a matrix with .

### 4.2. Solving the equation

In practice, there is noise, so solution vector f is defined only up to an unknown scale. So, it is better to use more than eight correspondences and create a larger . Furthermore, is often rank-deficient, so we approximate by Linear least squares:

The subject is to avoid the trivial solution f.

The solution to this optimization problem can be found by Singular Value Decomposition (SVD). is the right singular vector corresponding to the smallest singular value of . A reshape of this into matrix give result called as .

### 4.3. Enforcing the internal constraint

An important property of the fundamental matrix is that it is singular, in fact of rank 2. Furthermore, the left and right null spaces ( and ) of are generated by the vectors representing the two epipoles in the images i.e . However, often, dealing with noisy image gives the result from Equation 14 usually does not have rank 2.

We find a best rank-2 matrix approximation of F by the mean of:

The constrain is to make is singular.

This problem is solved again by SVD, where then the best rank-2 approximation is found by:

### 4.4. Normalized algorithm

#### 4.4.1. Problems

The problem of the standard algorithm is that is often ill-conditioned for SVD. For SVD to work properly, should have differences between singular values not too large.

However, correspondences coordinate will often have extremely large values in the first and second compared to the third due to large pixel range of modern digital camera.

Furthermore, if the image points used to construct lie in a relatively small region of the image , then and are relatively similar, resulting in has one ery large singular value, with the rest relatively small.

#### 4.4.2. Solution

To solve this, map each coordinate system of two images independently into a new system satisfying two conditions:

* The origin of the new system should be at the centroid (center of gravity) of the image points. This is accomplished by translating the original origin to new one.
* After the translation, the coordinates have to be uniformed so that the mean of distance from each point to the origin equals . This can be done by the scaling factor for each respective image

Afterwards, a distinct coordinate transformation for each of the two images. We obtain a new homogeneous image:

This normalization is only dependent on the image points which are used in a single image and is, in general, distinct from normalized image coordinates produced by a normalized camera.

Note that we overload the notations because of the obvious relations.

The epipolar constraint based on the fundamental matrix can now be rewritten as:

Where .

This means that it is possible to use the normalized homogeneous image coordinates, and , to estimate the transformed fundamental matrix using the basic eight-point algorithm described above.

The solution is now more well-defined from the homogeneous equation than is relative to . Once has been determined, we can de-normalized to give by:

# **III. Solution**

Most of the algorithms used in our codes are described above. First, we downsize the image (optional, to improve running time). Then using Oriented FAST and Rotated BRIEF (ORB) descriptor from *cv2.ORB\_create* to find and match feature points between two images. From the matched feature points, we use RANSAC to randomize the inputs to calculate Fundamental matrix by [*8-points algorithm*](#bookmark=id.2y3w247) to reduce noises.

We calculate disparity by [*Sum of Squared Differences*](#bookmark=id.1d96cc0), then using the baseline and focal length from data calibration file to calculate the depth image.

A diagram of a flowchart

Description automatically generated

For better running time, we also use numba to optimize Python functions with decorator *@jit(nopython=True)*, which speeds up running time from 5 minutes to 1 minute compared to pure python.

## **1. calibration.py**

In this file, we compute the feature points, then estimate the fundamental and essential matrix from them. Furthermore, we find the correct camera pose (Rotation and Translation) from the matrix.

### 1.1. def draw\_keypoints\_and\_match(img1: MatLike, img2: MatLike, nfeatures: int = 500) -> Tuple[NDArray, NDArray, MatLike]: …

Finding keypoints using ORB feature detection and descriptors in the image and find best matches using brute-force based matcher.

### 1.2. def compute\_Fundamental\_matrix(kp1\_list: NDArray, kp2\_list: NDArray) -> NDArray: …

Calculate the matrix from a set of 8 points using SVD. Furthermore, the rank of matrix is reduced from 3 to 2 to make the epilines converge.

More details at section [*8 points algorithm*](#bookmark=id.3x8tuzt)

### 1.3. def RANSAC\_F\_mat(kp1\_list: NDArray, kp2\_list: NDArray, max\_inliers=20, threshold=0.05, max\_iter=1000) -> NDArray: …

Shortlist the best F matrix using RANSAC based on the number of inliers.

RANSAC randomize 8 feature points to compute and calculate errors in estimating the points. Then we classify inliers with predefined threshold. Best is one with the most inliers.

### 1.4. def compute\_Essential\_matrix(F\_mat: NDArray, K1: NDArray, K2: NDArray) -> NDArray: …

Calculation of the Essential matrix with , and camera intrinsic.

More details at section [*Essential-matrix*](#bookmark=id.2ce457m)

### 1.5. def drawlines(img1src: MatLike, img2src: MatLike, lines, pts1src, pts2src, random\_seed=0) -> tuple[MatLike, MatLike]: …

Visualize the epilines on both the images

## **2. correspondence.py**

### 2.1. def sum\_of\_squared\_diff(pixel\_vals\_1, pixel\_vals\_2):

We use Sum of Squared Differences (SSD) formular:

### 2.2. def block\_comparison(y, x, block\_left, right\_array, block\_size, x\_search\_block\_size, y\_search\_block\_size):

Block comparison function to find minimum SSD match. Two image blocks are correspodence if their SSD is the minimum.

### 2.3. def ssd\_correspondence(img1, img2):

Correspondence applied on the whole image to compute the disparity map and finally disparity map is scaled by min max scaling.

## **3. depth.py**

### 3.1. def disparity\_to\_depth(baseline, f, img):

More details at section [*Depth Map*](#bookmark=id.rjefff)

## **4. Notebooks:**

Mainly used for testing/ reviewing and producing image results.

# **IV. Evaluation**

## **1. The results achieved**

### 1.1 Dataset used (Middle Blurry Q-Size)

We here use the middle blurry, quarter-size datasets to accommodate Python’s performance. From the looks of features matching, we can see that the cameras are calibrated, and two image planes has been already parallel, so no rectification is needed:

A blue motorcycle parked in a room

Description automatically generated

Our main error measurement for two given matrix is RMSE:

Where:

* and are the dimensions of the image.
* is the actual pixel value at position
* is the predicted pixel value at position

### 1.2. Estimating Fundamental matrix

Our estimation of Fundamental matrix is very close to *cv2.findFundamentalMat* for the first 5 datasets: mean = 0.053111; min = 0.00034; max = 0.120961. For the last dataset, the RMSE error is 7e+21:

A shelf with items on it

Description automatically generated

We suspect the logo on the bag creates too many features outliers that are too close to each other, thus yield very distorted results:

*>>> cv2.findFundamentalMat(list\_kp1, list\_kp2)[0]  
array([[-5.12793973e-08, -1.42405116e-04, 3.62484045e-02],  
 [ 1.42119735e-04, -5.70761988e-06, 1.85833981e+11],  
 [-3.59749826e-02, -1.85833981e+11, 1.00000000e+00]])   
  
>>> RANSAC\_F\_mat(list\_kp1, list\_kp2)  
array([[ 1.42808893e-17, 2.60663832e-09, -1.00222640e-06],  
 [-2.49459138e-09, 9.15801324e-11, 7.07106677e-01],  
 [ 9.55271472e-07, -7.07106882e-01, 6.59325435e-05]])*

Since our algorithm is based on randomizing matched feature points, the noises produced from that did not affect the results.

### 1.3. Disparity map

For images with not many sudden depth and the majority is smooth surface, our RMSE error is quite low:

A comparison of a heat map

Description automatically generated with medium confidence

While many uneven, discontinuous surface like on a bike when its engine has many smaller compartments inside:

A comparison of images of a person's body

Description automatically generated

Overall, our RMSE ranges from 14221 to 3445. The average score in 6 datasets is 7532.

There are a lot of noises in depth-discontinuity area, and a border 15 padding because of SSD block-matching algorithm. The noises can be rectified by using Fast Global Smoother Filter for post-filtering, the reasons for this problem will be discussed later.

Note: All of the images, results and further details can be found at *evaluation.ipynb*.

Note: The number on the image label is RMSE error compared to ground truth

Note Any values in ground truth has to be set to *ndisp* to avoid results.

## **2. Compare with the results of the group on the same topic**

## **3. Previous studies**

### 3.1. Calculating correspondences (Semi Global Matching)

We used Sum of Squared Differences (SSD) to calculate the corresponding block. However, this function inherently assumes there are no sudden depth variations and occlusions. Furthermore, this assumes continuous or functionally variable disparities within the correlation window. Thus, often produce very noisy results, jumping and gap pixels.

A diagram of a line with lines and letters

Description automatically generated with medium confidence

One portion of the scene may obscure other portions in either or both cameras. Ex: .

The Semi-Global Matching (SGM) implements Mutual Information cost function. The pixelwise cost and the smoothness constraints are expressed by defining the energy E(D) that depends on the disparity image D:

Where:

* represents the disparity map.
* is the data term that measures the fidelity of the disparity map at pixel .
* denotes the neighborhood of pixel .
* is an indicator function.
* First term is the similarity cost, the sum of all pixels matching costs for the disparities of .
* penalty for small disparity changes (1 pixel).
* penalty is constant and larger for all larger disparity changes.

Using a lower penalty for small changes permits an adaptation to slanted or curved surfaces. The constant penalty for all larger changes (i.e. independent of their size) preserves discontinuities. While SSD is too simple to capture these characters.

Generally, SGM outperforms us, here is the comparison:

A graph with blue lines and orange lines

Description automatically generated

It is clearly at our algorithm is sensitive to noise. At data 2, our algorithm outperforms SGM when the surface are generally smooth and there are no sudden depth.

A rainbow colored image of a piano and a guitar

Description automatically generated

While at data 3, there is a lot of pipe obstructing each other, meaning there is a lot of sudden depth.

A colorful image of a building

Description automatically generated with medium confidence

This module is available at *cv2.StereoBM\_create*

### 3.2. Disparity post Filtering (Fast Global Smoother Filter)

Its objective is to enhance the quality of the input low-resolution depth map by increasing its spatial resolution. A low-resolution depth map and its associated high-resolution color image are used as inputs. It can preserve edges, which is important because one object should have a continuous disparity.

Fast Global Smoother Filter first take the left image as guidance image , an input image and produces a desired output . It tries to minimize the following Weighted Least Square (WSL) equation:

Where:

* controls the balance between the two terms. Increasing results in more smoothing in the output u.
* The first term expresses the similarity between output and input.
* are neighbor pixels of pixel .
* expresses the “inverse” similarity between pixel and . For example:
* The second term expresses the similarity between similarity between pixel and its neighbor in output .

When neighbor in is large (possibly edge), is small and thus, neighbour in can be large without affecting the overal energy. While neighbour in is small, forcing neighbour in to be also small. Overall, this tries to preserve the edges while smoothing low-variance, smooth surfaces.

Here, we set parameter *sigma\_color = 1.5* while using a grid search strategy to find the best *lambda\_value = 8273000*. The noise reduction is obivious:

A colorful image of a building

Description automatically generated with medium confidence

Comparing with Gaussian filter and Mean filter, we got:

A graph of a line graph

Description automatically generated with medium confidence

The most impactful scenario is Data 3, where there are a lot of sudden depth:

A close-up of a heat loss

Description automatically generated

* Gaussian blurs edges, which we do not want because one object should has continuous and similar disparity, which helps them stand out.
* Median filter is not inherently continuous while the noises are usually big enough to be spilled further.

This module is available at *cv2.ximgproc.createFastGlobalSmootherFilter*

### 3.3. Cencus Transform

Cencus transform represents a block as the relationship between the center and its neighbors for matching instead of the original. Then hamming distance is used to calculate the similarity between blocks. Our implementation of cencus transform each neighbor pixel of (in grayscale) to:

* if
* if
* if

For example:

Their Hamming distance is 3.

Here are some comparisons of results between our algorithm and cencus. We used block size of 5, any larger block size would yield much more noises:

A screenshot of a heat map

Description automatically generated

A heat map of a person

Description automatically generated with medium confidence

Unfiltered cencus performs the worst because of noises, thus when being smoothed by Fast Global Smoothing, its accuracy is improved drastically, but not as much as our algorithm.

A graph of a graph with lines and numbers

Description automatically generated with medium confidence

# **V: Conclusion:**

In this paper, we evaluated our depth estimation algorithm using various datasets and compared it with other methods. Our approach showed promising results, especially in scenarios with smooth surfaces and minimal depth discontinuities. We found that our SSD-based algorithm is sensitive to noise, particularly in areas with sudden depth changes and occlusions. The Fast Global Smoother Filter significantly improved accuracy by reducing noise and preserving edges. However, challenges remain in handling complex scenes with transparent or highly reflective surfaces. Comparisons with methods like Semi-Global Matching (SGM) and Census Transform highlighted our algorithm's strengths and weaknesses. While SGM generally outperformed our approach, the Census Transform, combined with Fast Global Smoothing, also showed improved accuracy. Overall, our depth estimation algorithm works well in in-door smooth surface and ideal brightness. Future work will focus on rectification images and enhancing robustness and accuracy in challenging environments, including using neural network-based approaches for block matching to further improve performance.

# **References**

* K. Kutulakos, “Epipolar Geometry,” Stanford University, [Online]. Available: https://web.stanford.edu/class/cs231a/course\_notes/03-epipolar-geometry.pdf. [Accessed: Nov. 3, 2024].
* “Structure from Motion,” [Online]. Available: https://cmsc426.github.io/sfm/#essential. [Accessed: Nov. 3, 2024].
* “Epipolar Geometry,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Epipolar\_geometry. [Accessed: Nov. 3, 2024].
* “Essential Matrix,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Essential\_matrix. [Accessed: Nov. 3, 2024].
* “Fundamental Matrix (Computer Vision),” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Fundamental\_matrix\_(computer\_vision). [Accessed: Nov. 3, 2024].
* W. Fitzgibbon and A. Zisserman, “Automatic Camera Recovery for Closed or Open Image Sequences,” [Online]. Available: https://core.ac.uk/download/pdf/11134866.pdf. [Accessed: Nov. 3, 2024].
* Kak, “A Tutorial on the Semi-Global Matching Algorithm,” Purdue University, [Online]. Available: https://engineering.purdue.edu/kak/Tutorials/SemiGlobalMatching.pdf. [Accessed: Nov. 3, 2024].
* S. H. Lu and C. Chen, “Fast Global Smoothing via Weighted Least Squares,” University of Illinois, [Online]. Available: https://publish.illinois.edu/visual-modeling-and-analytics/files/2014/10/FGS-TIP.pdf. [Accessed: Nov. 3, 2024].