Google StreetView Stereo-Reconstruction using Graph Cuts: α-β Swap Algorithm

Timur Zhanabaev

M.Sc. Project - September 9, 2023

1 Introduction and Overview

In order to map a collection of 2D images onto a 3D object, methods of stereo-reconstruction can be used to estimate a point cloud of a scenery. For stereo-vision, these depth values (*disparities*) are generated from the pixel coordinates of matching pixels between a pair of images.

For the task at hand, images are collected using the Google StreetView API to produce stereo pairs from a set of panoramic view-points registered by Google along a street. The goal is then to produce a depth-map for a building, a street, or potentially a city block.

The main objective is then to develop an optimization method to estimate the disparity values. The method of choice is the energy function (MRFs) based method, using graph cuts as energy minimization moves. From the 2 popular methods; the α-β swap algorithm is chosen, with α-expansion being the latter. Such methods construct graphs from a set of labels and the labelled pixels to produce cuts where the new partitions correspond to new relabelling, while minimizing locally the energy of the system for the given labels in the move.

On top of the disparity estimation, further image processing methods are employed when successful, such as rectification to generate improved results.

2 Theoretical Background

2.1 Epipolar Geometry

The issue arises when estimating depth from a single image, being unable to gauge the distance between the camera and the object of the scenery. In order to estimate this distance, a pair of images are required at the very least. The images I_l and I_r located at points O_l and O_r are then produced, with the intention of capturing the same scenery at different view-points, as shown in fig. 1 as optical centers left and right, defining the epipolar geometry.

Then for some point of interest in the image, the *scene point* x_l (or *key-point*) that can be seen in I_l the depth can be estimated by matching it within I_r as point x_r (points x and x' in in fig. 1). In the un-rectified scenario, The search for the key-point is made along pixels (x,y). When the cameras lie on the same height and the camera pointing at the same lateral axis (*pitch*) the images are calibrated and can be assumed to be rectified, with corresponding key-points being found along image rows (also called *scan-lines*), leaving the disparity values to vary there only.

Then, as shown in fig. 2, having the points correctly matched, then with a known baseline distance

T and focal length f of the cameras, then the true distance Z can be inferred using the equation,

$$Z = \frac{1}{f}\frac{T}{d}$$

where d is the disparity value.

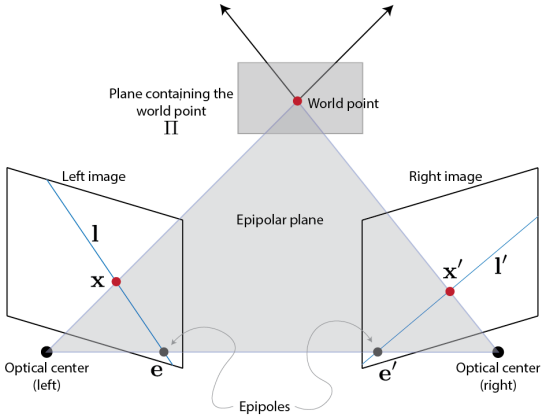

Figure 1: Epipolar geometry (from [1])

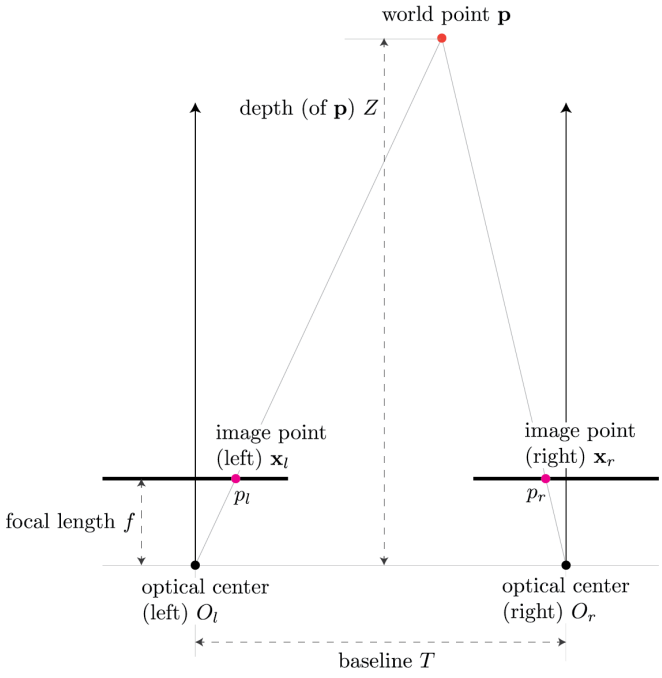

Figure 2: Rectified image pair (from [1])

2.2 Rectification

In order to rectify the images, a collection of key-points are generated for each image. The matching algorithm then finds the corresponding key-points in left image and right image, by some distance

metric and with the a correspondence uniqueness condition. Having these pairs, the relationship between the two cameras can be modelled with the fundamental matrix $F \in \mathbb{R}^{3\times3}$. That is we find a matrix F s.t. it maps a point x_l from image I_l to the *epiline* l_r on the right image,

$$l_r = Fp_l$$

An epiline l is a line on the image plane I connected by the key-point x and the line forming the baseline T.

When the camera calibration parameters are unknown, as in our case, the 8-point algorithm is used to estimate F [1]. It is then require to stack 8 rows of the 8 matching key-points $x_l = (x, y, 1)$ and $x_r = (x', y', 1)$ in the form,

$$a = (xx', yx', x', xy', yy', x, y, 1) \text{ with, } A \in \mathbb{R}^{8\times9} = (a_1, a_2, ..., a_8)^\top$$

and with components of F in $f \in \mathbb{R}^{9\times1}$

$$f = (f_{11}, f_{12}, ..., f_{33})^\top$$

Then F is found by solving,

$$Af = 0$$

2.3 Occlusion

A major issue arises is when the pixels in one image get hidden (*occluded*) in the next image, thus if key points are generated in those regions, it can lead to severely poor estimation of the fundamental matrix. As well, the occluded pixel should theoretically not contain any depth values, while most methods fill in the gaps by using the best match, leading to loss of the uniqueness of matches. Occlusion is present in all Google StreetView pairs of images, due to trees and sides of buildings, potentially ruining good key-points that would be matched well in the scenery, such as; stop signs, store signs, etc.

2.4 The Stereo Matching Steps

In general, the stereo-matching problem is composed of several building blocks; the matching cost calculation between pixels, the cost aggregation between a collection of pixels, disparity optimization and disparity refinement. For energy-based methods, the match cost and aggregation is often together in one step [2]. Each step is described briefly in the next sub-sections.

2.4.1 Matching Cost

The simplest **matching cost** for 2 pixels can be a simple difference of their pixel values. These common cost functions include, squared intensity differences (SD) and absolute intensity differences (AD), analogous to traditional measures squared error and absolute error. For more sophisticated methods, gradient-based measures can include values from edge detectors, non-parametric measures such as rank and census transforms, that are robust against image gain [3], that is when camera is exposed to greater intensity of light, increasing the apparent brightness of an image. [3].

2.4.2 Aggregation of Cost

Having a function of determining cost between pixel $p \in I$ (left iamge) compared to $p' \in I'$ (right image), a summation or weighted averaging is performed over a **support region** $\mathcal{S}$, such region can be defined as a set of neighboring pixels of p and their counter parts, the neighbors of p' of the right image. The total matching cost is then averaged, giving an aggregated cost that is more robust when taking account surrounding pixels [2].

2.4.3 Disparity

Knowing the aggregated cost for each pixel for all disparity values d, choosing the optimal disparity is the matter of choosing d s.t. it gives the minimum aggregated matching cost. The disparity value then determines the difference of pixels for reference (left) image corresponding to the point in the right image.

Looking for an optimal d is an NP-hard problem, and generating matching costs for all possible d values can be intractable or computationally infeasible [3].

This alternative way minimization uses MRFs and their energy function formulations to iteratively find local minimal energy with label swaps. For such methods, an energy function is formulated as,

$$E(f) = E_{data}(f) + \lambda E_s(f)$$

where,

$$E_d(f) = \sum_{p \in \mathcal{P}} D_p(f) \text{ and } E_s(f) = \sum_{(p,q) \in \mathcal{N}} V_{p,q}(f_p, f_q)$$

with f being the set of disparities assigned to pixel p, $\mathcal{P}$ denoting the set of all pixels, $\mathcal{N}$ the set of neighboring pixels.

As well with potential functions; the node $D_p(f)$ and edge $V_{p,q}(f_p, f_q)$ potentials, with λ controlling the smoothness term. In order to get better results, the potentials are to be considered wisely, having to consider trade-offs between faster convergence and disparity smoothness of the image [3].

2.4.4 Optimization Methods

Among the many ways the matching cost can be optimized, the project uses graph cuts (GC) as optimization steps. These methods are used on graphs constructed from images with additional *source* and *sink* nodes to find best partitions of binary labels, denoted as the source and sink. Then to expand cuts to multiple labels, the cuts are performed iteratively between a pair of labels until all labels have converged to their local minimum, with the global minimum only guaranteed up to some known constant [2]. In the project, the GC optimization method implements the α-β swap move algorithm. Where all the α-β pairings at each (inner) iteration are cut in a cycle, with termination occurring when no further local energy improvements are possible in all the label pairs [2]. The α-β swap method and it's *flow graph* construction is described in greatest detail in the next section.

2.4.5 Disparity Refinement

As the final part, as the disparity values are simply discrete values of pixel distances, the disparity refinement can be done to compute continuous values disparities that will give greater resolution to depth. Such, *sub-pixel* disparity estimates can be computed in a variety of ways, including iterative gradient descent and fitting a curve to the matching costs at discrete disparity levels [3].

2.5 The Optimization Method: α-β swap algorithm

The algorithm implemented for disparity estimation is the α-β swap algorithm. The method generates a local minimum with respect to very large moves. The favourable property of these moves is the guarantee of labeling being locally optimal for metric potentials [2]. As well as the solution is robust to various initial labeling, not changing significantly. The main idea behind alpha-beta swap is turning *some* α pixels in partition P_α with label β, and simultaneously turning *some* β pixels to α in partition P_β using the graph cut partitioning method [2].

The algorithm is described as such, first the graph is constructed as in sec. 2.5.1-2.

The α-β swap algorithm (as described in [2])

1. `initialize labelling` f
2. `success = 0`
3. `for each pair` α,β

   3.1 $\hat{f}$ `= minimum-cut(G`$_{\alpha,\beta}$`)`

   3.2 `if` $E(\hat{f}) < E(f)$`, set` $f = \hat{f}$ `and success = 1`
4. `if success == 1, go to 2.`
5. `return f`

As an optimization method, the graph cuts are used to efficiently find $\hat{f}$, using a single graph cut computation for each label pair. In each cycle of the alpha-beta algorithm (step 2-4), we iterate for every pair of labels. A cycle is successful when there is at least one pair of labels lowering the energy of the system, with the algorithm stopping only when all pairs of labels in the cycle have made no improvement. [2]

2.5.1 Graph Construction

In order to perform a cut on the graph that will generate optimal separation between two labels α,β, a graph is constructed from the pair of images for labels, with the following steps.

For each pair of labels, we define $G_{\alpha,\beta}$ to be an *undirected* graph with set of nodes $\mathcal{V}_{\alpha,\beta}$ and set of edges $\mathcal{E}_{\alpha,\beta}$. The set of nodes then consists of *terminal node* (source and sink) α and β and pixels p with labels α and β, where $p \in P_{\alpha,\beta}$, the partition of pixels with labels α or β from entire image, where $P_{\alpha,\beta} = P_\alpha \cup P_\beta$. The set of nodes is then $\mathcal{V}_{\alpha,\beta} = \{\alpha, \beta, P_{\alpha,\beta}\}$

The set of edges consists of edges formed between all pixels $p \in P_{\alpha,\beta}$ connected to the terminal

nodes α and β, denoted as t_p^α and t_p^β (called t-links), respectively. Then all pixels $\{p, q\}$ in $P_{\alpha,\beta}$ that are neighbours are connected by n-links, denoted as $e_{p,q}$.
The set of edges is then $\mathcal{E}_{\alpha,\beta} = \{t_p^\alpha, t_p^\beta : p \in P_{\alpha,\beta}\} \cup \{e_{p,q} : p, q \in P_{\alpha,\beta}, q \in \mathcal{N}_p\}$.

2.5.2 Energy functions

Having all nodes and edges, the weights are then assigned according to an energy functions with node and edge potentials components, of the form,

$$E(f_p) = \sum_{p \in P} D_p(f_p) + \sum_{(p,q) \in \mathcal{E}} V_{p,q}(f_p, f_q)$$

where $\mathcal{P}$ is the set of all pixels in the reference image, f_p is the disparity value of the pixel. The node potentials are calculated based on the matching cost between a pixel $p = (i, j)$ in the *reference* image and the pixel $p' = (i, j + f_p)$ in second image, that is being being away by α or β label value, which is the disparity value.

Using the energy function formulation, the weights assigned to edges in $G_{\alpha,\beta}$ are defined as such,

1. $t_p^\alpha = D_p(\alpha) + \sum_{q \in \mathcal{N}_p, q \notin P_{\alpha,\beta}} V(\alpha, f_q)$ where $p \in P_{\alpha,\beta}$, that is pixel q are not in the partition with labels α, β, but are neighbors of such pixels p.
2. $t_p^\beta = D_p(\beta) + \sum_{q \in \mathcal{N}_p, q \notin P_{\alpha,\beta}} V(\beta, f_q)$ where $p \in P_{\alpha,\beta}$
3. $e_{p,q} = V_{p,q}(\alpha, \beta)$ where $p, q \in P_{\alpha,\beta}$ and $q \in \mathcal{N}_p$. That is neighboring pixels within the label pair partition.

2.5.3 Graph cut

The general idea for graph cut follows from finding a separation between a set on nodes connected by edges, s.t. we get two partitions, P_a and P_b, where $P = P_a \cup P_b$. These partitions are then chosen such that all the severed edges between nodes connecting P_a and P_b give the minimal flow (edge weights) between them.

When designing the energy function, the node potential should be low for good disparity value. This would allow the edge to be cut and the label to be swapped to the better disparity value. Thus, a low potential tends to swap, and a high potential tends to keep the edge.

Adding on top the edge terms, usually using label and pixel intensity dissimilarities that increases flow. The disagreeing labels and pixel intensity dissimilarity will then not render the swap to not be suitable.

2.5.4 The Cost of the Graph Cut

One important ingredient that determines the termination of the algorithm is if the energy of the system is lowered after each cut. That is, if for the pair of labels the energy is not lowered, the cut is not considered to update the label of the nodes. As shown in [2], the cost of the cut is tied to the energy of the system with resulting set of labels f^C from cut, with the energy and the cost of the cut defined as,

$$cost(\mathcal{C}) = \sum_{(\alpha,p) \in \mathcal{C}} t_p^\alpha + \sum_{(\beta,p) \in \mathcal{C}} t_p^\beta + \sum_{(p,q) \in \mathcal{C}} e_{p,q}$$

where $\mathcal{C}$ is the set of edges that were cut from $G_{\alpha,\beta}$. With relation to energy function, $cost(\mathcal{C}) = E(f^C) - K$ where,

$$K = \sum_{p \notin P_{\alpha,\beta}} D_p(f_p) + \sum_{(p,q) \notin P_{\alpha,\beta}, q \in \mathcal{N}_p} V_{p,q}(f_p, f_q)$$

at each iteration, the cost of the cut can update the energy $E(f^C)$, for further decisions into whether update disparity values from the swap or not.

2.5.5 The Node and Edge Potentials

In order for the algorithm to guarantee the cut resulting in decreasing energy of the system, the proof assumes that edge potential $V_{p,q}$ is at least a *semi-metric* [2]. That is for any metric $V_{p,q}$, any local minimum generated by the swap moves is within a known factor of the global optimum. [2]

2.5.6 The Node Potential D_p

As discussed in sec. 2.5.1, the node potential or the *matching cost* computes the similarity between a set of pixels in the left to the right images. The node term D_p can be computed using windows over a *support region* $\mathcal{S}$ as in convolution operations to determine the cost of pixel p at disparity f_p in the right image. The window sizes included can vary from 3×3 or 5×5 with operations performed in the window include; squared intensity differences (SD) and absolute intensity differences (AD), analogous to traditional measures squared error and absolute error.

For the project, the 4-neighbor system of the image graph is used to compute the aggregated cost. The pixel values are aggregated and averaged to form the penalty $D_p(f_p)$. Let $\mathcal{S} = \{pixel, \mathcal{N}_{pixel}\}$, then,

$$C_p(f_p) = \frac{1}{|\mathcal{S}|} \sum_{p \in \mathcal{S}} f(I_p, I'_{p'})$$

where, $f(x,y) = \begin{cases} |x-y| & \text{Absolute Difference} \\ \sqrt{(x-y)^2} & \text{Squared Difference} \end{cases}$

$$D_p(f_p) = min(k, C_p(f_p))$$

where $p = (i,j), p' = (i, j + f_p)$

The k parameter, allows the penalty to be robust against outliers (noise in image) [2] This means that the if the pixels are too different due to noise in the right image, with the limit k, they will be more likely to be swapped than without the limit.

2.5.7 The Edge Potential

The first potential developed is the Pott's model, considered to be good for fronto-parallel surfaces due to the discontinuity preserving property, although has poor performance for large number of labels [2]. As well, the Potts model may incorporate the neighbouring pixel dependence, which has demonstrated improvement over models assuming independence and potentially even better results for higher order of neighborhood systems in images [2].

The potential is described as such,

$$V_{p,q}(f_p, f_q) = u_{p,q} \cdot \mathbb{1}(f_p \neq f_q)$$

That is whenever the labels do not match of neighboring pixels, the penalty increases by $u_{p,q}$, which is also can be customized by the user, where in [2], the potentials calculated are based on the pixel intensity differences within the left image.

Such that when $|I_p - I_q| \leq \textit{threshold}$, $u_{p,q} = 2K$, and when $|I_p - I_q| > \textit{threshold}$, $u_{p,q} = K$.

Thus when labels match with low AD, $V_{p,q} = 2K$, and with high AD, $V_{p,q} = K$. This encourages the matching pixels with lower AD to have to not be cut as greater weight preserves edges in the cut.

Other methods used to generate $u_{p,q}$ include the values from edge detection algorithm. The pixels that are found on the edges should be encouraged to have different disparities.

Another potential used is the linear disparity difference edge model, a piece-wise smooth model. The function works better for slanted surfaces, where the disparity changes in smaller steps. If the nodes p, q disparity difference is large, then there is incentive to make it in the cut, by assigning a greater $V_{p,q}$, such that it swaps to another label. The edge potential is then defined as,

$$V_{p,q}(f_p, f_q) = a \cdot \min(b, |f_p - f_q|)$$

where a and b are model parameters.

3 Algorithm Implementation and Development

3.1 Data Collection

Using the Google StreetView API, a collection of images are collected for panoramic viewpoints. The parameters chosen for panoramic view are listed as such;

1. fov = 110 (to capture most of scenery without too much fish-eye lens distortions)
2. pitch = 15 (degrees) looking up slightly to capture more of building and less of road.
3. heading = 43 (degree) selected to be perpendicular from to the road.
4. size = 640x640 (being the maximum resolution provided by the API)

The images are then collected as fronto-parallel buildings scenery from the road's viewpoints.

3.2 Image Pre-Processing

For key-point matching, the image pair s(I_l, I_r) are converted to gray-scale images $I_l^{(g)}, I_r^{(g)} \in [0, 1]$, since they are implemented as such for gray-scale images. Moreover, for α-β swap algorithm, the images are scaled down from $(640, 640)$ to $(96, 96)$ for faster performance with the implemented method.

3.3 Key-Point Generation

The first step of the process is generating key-points in the pair of images, with code taken from the `scikit-image` tutorials [5], the key-point generating algorithm used is `SIFT`, Scale Invariant Feature Transform.

The algorithm is deemed to be most popular with it's strengths being; invariant to image scaling, translation, and rotation, and partially in-variant to image gain and projections [5]. As output we get N key-points and their descriptors (embedding of 128 dimensions), which are then used to match from right image.

3.4 Key-Point Matching

In the scikit-image package, the matching process is done with the `match_descriptors()` function, a brute-force way of selecting best key points by a distances metric, L_1 or L_2 for the 128 dimensional embeddings.

For `SIFT`, the parameter `max_ratio` is advised to be set to 0.8 [6] (distances between first and second closest descriptor in the second set of descriptors), as well as `cross_check` is set to True, checking if both points agree on being optimal matches.

Among the various algorithms offered by `openCV` and `scikit-images`, the key-point matching algorithms have all been performing poorly for `Google StreetView` images, with key-points not matching correctly, leading to poor image rectifications, as shown in fig. 5.

In hopes of fixing the issue, the key-points are filtered by a distance threshold, the along pixel's row coordinates, such that the distance for key points are filtered to be parallel to each other. Since the StreetView car tends to travel 10 meters for each viewpoint, many objects in the scenery can be as far apart (on the pixel's column coordinates) as the image's width for the limited image size the API provides.

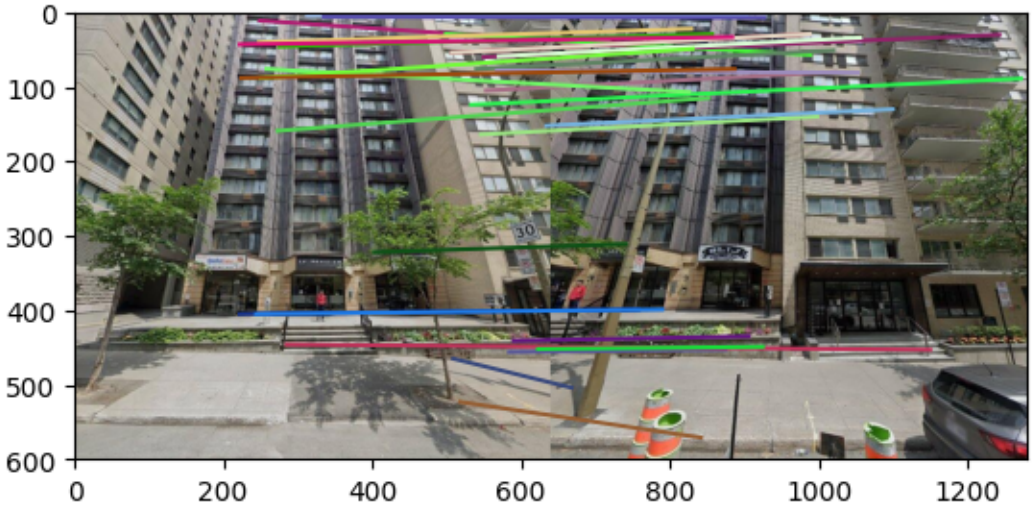

Figure 3: A sample of successfully matched key-points in a stereo-pair of Google StreetView images.

3.5 The Fundamental Matrix Estimation

With key-points matched, the next method used is `ransac()`, random sampling consensus. The method aims to robustly estimate a model's parameters from repeated random sub-sampling. Then with a large number of iterations, the model's parameters converges closer to the true ones while finding the outlier key-points. [5]

In `ransac`, the parameter `model_class` is required to be set to `FundamentalMatrixTransform`, the object class for the 8-point algorithm with model parameters being the F matrix. With the F matrix found, the rectified images are produced from image warping methods `stereoRectifyUncalibrated` and `warpPerspective` available in `openCV`.

3.6 Depth-map Initialization

In order for the swap moves to work, the depth-map has to be initialized with the range of all possible disparity values contained in the image. If only $\{1, 2\}$ are present in the depth-map, only $G_{1,2}$ can be constructed.

Then for a column j, the disparity values are initialized randomly to $d \in \{-j, w-j\}$, where w is the number of columns in image (width). The restriction is made, to prevent the labels being mapping to pixel's outside the right image's bounds. As well, by passing a smaller range of disparities {`min_d`,`max_d`}, the values will be initialized to $d \in \{max(-j,$ `min_d`$), min(w-j,$ `max_d`$)\}$.

3.7 The Alpha-Beta Swap Algorithm

Having generated a stereo-pair of rectified images, the pair of RGB images $I_l, I_r \in \{0, 255\}^3$ are then fed into `alpha_beta_swap` method, implemented for this project using `networkx`, a `python` library for graphical data structures and algorithms. The necessary algorithm being the `minimum_cut` algorithm with sink and source nodes partitioning.

The images are then transformed to grid graphs to store image information, including; RGB values and current disparity values to generate node and edge potentials for $G_{\alpha,\beta}$ graph constructions, as described in sec 2.5.1-2. *See code for more details of implementation.*

Having generated $G_{\alpha,\beta}$ graphs, the `minimum_cut` method can take several flow function as parameters, the one chosen here is the default method `preflow_push`, running in $O(n^2\sqrt{m})$ time, generating fastest results, with n being the number of nodes and m the number of edges.

For energy function parameters, these include; choosing potential metrics and their parameters (*L1* or *L2* for nodes and *linear* or *Potts* for edges), choosing disparity range, number of cycles to iterate for and methods of initializing depth-map.

4 Computational Results

4.1 Image Preprocessing

For the first part, the fundamental matrix estimation would not generate the best homographies for each stereo pair, resulting in projection that are unusable for the GC method. A common

problem encountered was the right epipole being either visible in the left image or is too far away, as shown in fig. 4. This location is supposed be the right camera's position in the left image. These incorrect projections then produce faulty rectifications, as shown in fig. 5. In the end, the rectifications are not used as an automated way to fix the images. The pairs are then instead collected in a manner, by tuning the StreetView API parameters, so that they appear somewhat parallel.

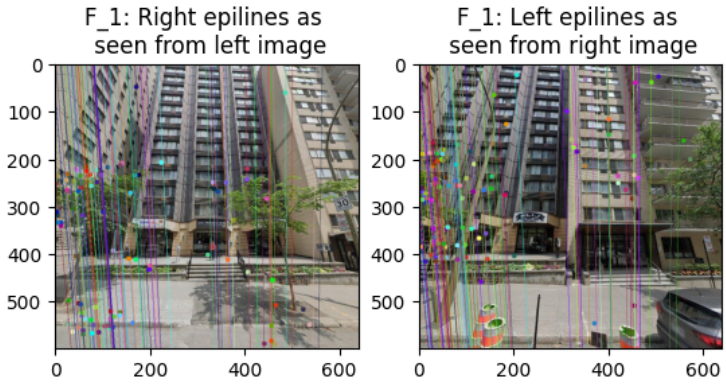

Figure 4: Unsuccessful homographies: the epipoles are within the image when they should be slightly outside, where the paired viewpoint is supposed to be.

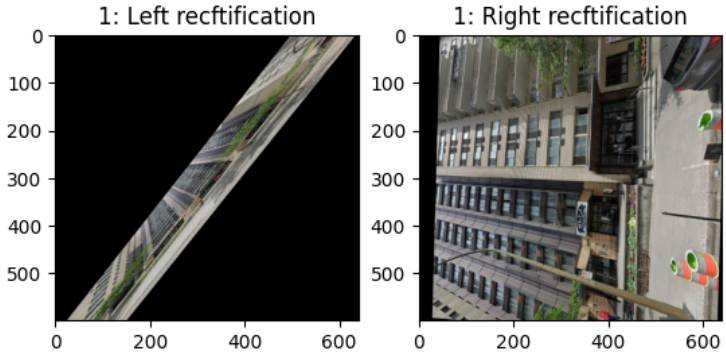

Figure 5: Unsuccessful rectifications

4.2 Stereo Matching

For the second part, the stereo-pairs are used despite the rectification short-comings. Meanwhile, as an initial test, the swap method is used on a pair of rectified images from *sci-kit images* dataset, shown in fig. 6. This is done to test for the best energy function model and parameters.

4.2.1 Hyper-parameter Tuning

The initial issue with the implementation is the running time. With N disparity values, each cycle scales by $O(N^2)$ iterations. For an image of shape $(75, 111)$ (scaled to 15% from $(500, 741)$) and a range of 30 disparity values, each cycle takes around 2 minutes to generate a depth map.

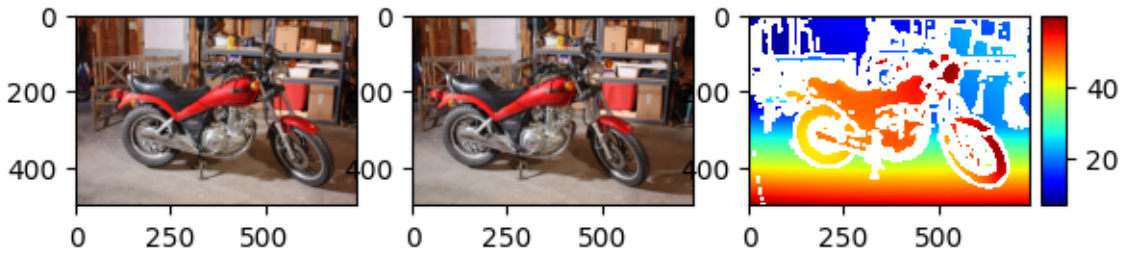

Figure 6: Left and Right rectified images of a motorcycle. The right most picture are the true depth values.

Since the library used was `networkx` where all data manipulation is being done over python objects, thus running time is 40 to 250 times slower than what it should have been with other graphical libraries that are python wrapper for code written in `C++` [4] such as; `graph-tool`, `networkit` and `maxflow`. The methods should be rewritten in those libraries for better performance.

Then testing for disparity values from -15 to 15 at the reduced image shape, which is key to making the work at all. There must be a good disparity range for the method to correctly find the best matching pixels. Without a good range, the image labels would degrade in quality with each cycle.

The method is tested for node potentials with L_1 and L_2 metric with values $k = \{10, 50, 100\}$. For Potts edge potential, $a = \{25\}$ and $b = \{10, 100\}$. For linear edge potential, $a = \{5, 15, 50\}$ and $b = \{3, 10, 30\}$. The resulting depth-maps are presented next,

Varying a parameter in linear V_{pq} with L_2 metric in D_p ($k = 50$)

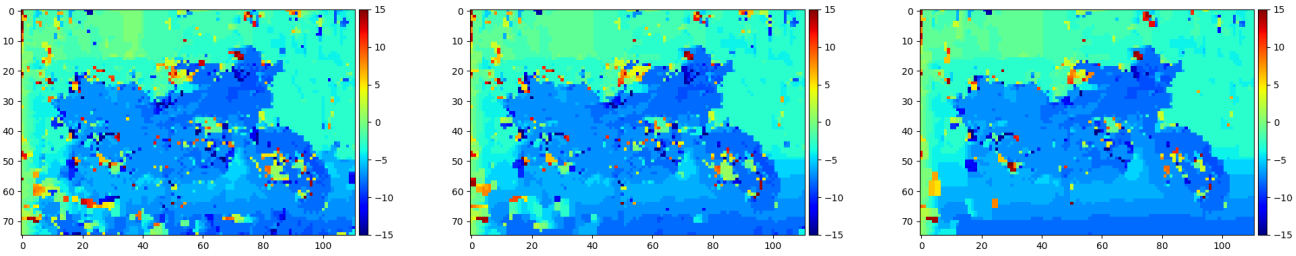

Figure 7: **For linear edge potential, varying a with $b = 3$ fixed**: (left) $a = 5$, (middle) $a = 15$ and (right) $a = 50$. Increased smoothing with greater a. The speckles are present due to iteration stopping at 5 cycles.

Varying b parameter in linear V_{pq} with L_2 metric in D_p ($k = 50$)

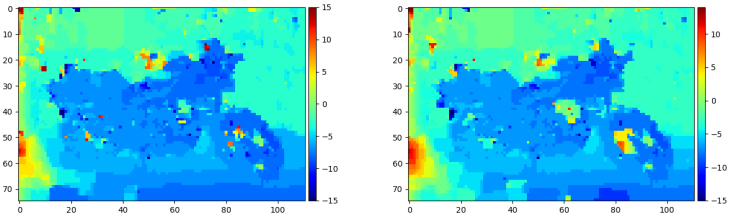

Figure 8: **For linear edge potential, varying b with $a = 50$ fixed**: (left) $b = 10$, (right) $b = 30$. Increasing b over smooths the image labels further, although the quality of the depth-map appearing to be lesser.

Varying k parameter in linear V_{pq} with L_1 and L_2 metrics in D_p

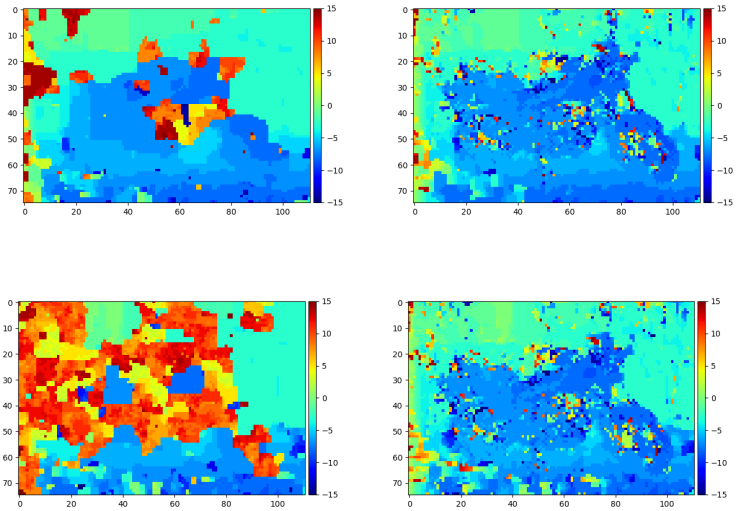

Figure 9: **Varying k and node metrics L_1 and L_2 with $a = 15, b = 3$**: L_2 - (top left) $k = 10$, (top right) $k = 100$. Tuning k determine how much node potential dominates the smoothing effect, with low k there is too much smoothing so the image is potentially unable to converge to the matching points, while with a high k there more speckles. L_1 - (bottom left) $k = 10$, (bottom right) $k = 100$. Changing L_1 or L_2 has no noticeable effect compared to a, b, k parameters.

Testing the Potts model for V_{pq} with L_2 metric in D_p

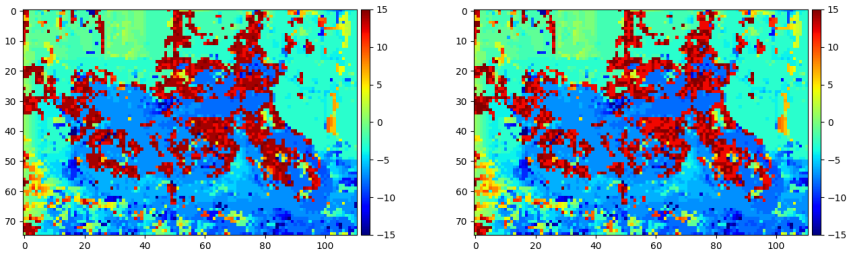

Figure 10: **Potts edge potential**: (left) $a = 25, b = 10$, (right) $a = 25, b = 100$. Increasing b sets a higher threshold for pixel difference, and a scales the penalty from the difference as in linear model.

The best result for the test are L_2 node metric for $k = 50$ and linear edge metric for $a = 50$ and $b = 10$, providing the best trade-off for depth smoothness. In general, the data term should dominate the smoothing term, or else the method will not converge.

Although, the ground-truth disparity values of the images lie in the $[7, 60]$, and it is clear that the right image's matching points should give negative values for the left image by observing the column numbers, thus the algorithm is giving the correct disparity values, and could be adjusted to fit the $[7, 60]$ range for the disparity refinement step (*not done here*).

As the last example, the α-β swap algorithm is used on a pair of StreetView images, with a disparity range from -30 to 30. The images were scaled from (640,640) to (96,96), in order to have realistic computation time. (*with (640,640) it would take several hours to produce the depth-map*).

The best model applied to a Google StreetView image pair

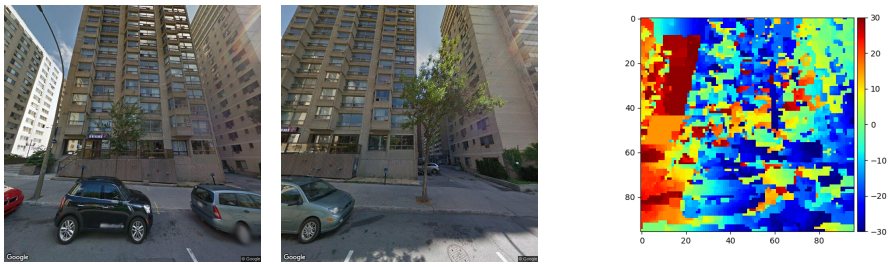

Figure 11: (*left*) left viewpoint image, (*middle*) right viewpoint image and (*right*) disparity estimates for left viewpoint. The disparity estimates are subpar, with first most reason being the depth values range being too small for the great displacements observed in both images.

Finally, the energy after each iteration is plotted for the StreetView example, showing how the energy is decreasing after each successful graph cut. Here is also shown how in the first cycle, most of iterations are accepted, with diminishing numbers for next ones. Then for the 5th cycle, the algorithm usually accepts almost none of the moves, indicating that the algorithm has converged for it's given parameters.

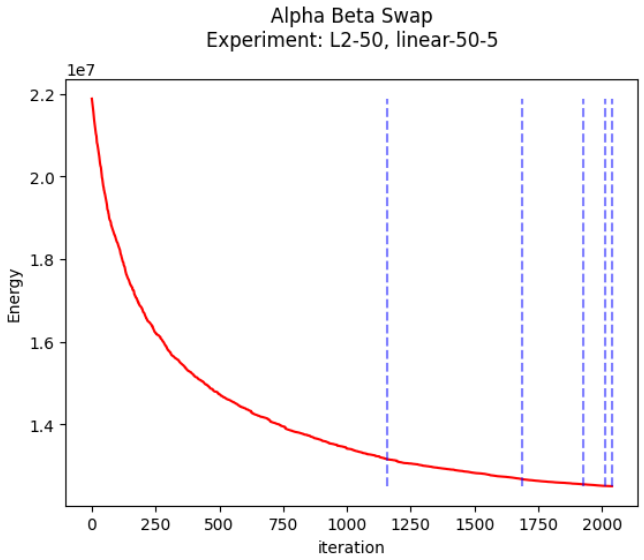

Figure 12: **StreetView example**: plotting the successful iterations of graph cut energy minimization. (each blue line denotes end of a cycle, with each cycle accepting less cuts)

5 Summary and Conclusions

In the end, the α-β swap algorithm converges successfullt to it's optimal disparity values of the rectified image pair, provided the disparity range to look through is sufficient. As well, the method is flexible to design alternative energy functions, with the issue only being the run time with the chosen python network library. For the StreetView images, more work is required to develop energy functions that search for disparities in both axes, and in faster network libraries to make the problem feasible for network analysis at such scale.

Since the rectification methods did not provide satisfactory results, there is more incentive now to develop and energy function that looks for disparity along the y axis as well, as used in structure from motion methods in [2].

The ability to design potentials opens up room to more exotic energy functions that could perform better on the stereo-correspondence task. Such improvements come from incorporating other types of information for edge weights, including information from *edge detectors* and occluded pixels penalties [7]. Since it is important to get good estimates at discontinuities for stereo-vision, treating the occlusion problem is much needed, since it is at the boundaries of such areas that that greater disparity discontinuities occur [7].

References

[1] http://csundergrad.science.uoit.ca/courses/cv-notes/notebooks/21-epipolar-geometry.html

[2] Y. Boykov, O. Veksler, and R. Zabih. *Fast Approximate Energy Minimization via Graph Cuts*, IEEE TPAMI, 23(11):1222–1239, 2001

[3] Daniel Scharstein and Richard Szeliski. *A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms.*

[4] https://graph-tool.skewed.de/performance

[5] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. *scikit-image: Image processing in Python.*

[6] D.G. Lowe, "*Distinctive Image Features from Scale-Invariant Keypoints*", International Journal of Computer Vision, 2004.

[7] V. Kolmogorov and R. Zabih. *Computing Visual Correspondence with Occlusions via Graph Cuts*, Proc. Int'l Conf. Computer Vision, vol. II, pp. 508-515, 2001