Learning Binary Feature Descriptors for use in Stereo Matching

Andrew Milson

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Computing and Mathematical Sciences with Honours (BCMS(Hons)) at The University of Waikato.

Abstract

Census transform is one of the most widely used practical methods for computing matching costs due to it's ability to deal with camera gain and bias. This project shows that random binary descriptors outperform census transform on the 2015 KITTI data set and shows that binary descriptors optimised with genetic algorithms significantly improve accuracy over conventional methods. Additionally an open source efficient CUDA implementation, pseudo-code and an ablation study is provided for a total generalised variation regularisation method.

Acknowledgements

First and foremost, I would like to thank my supervisor, Henry Gouk, for the inspiration and guidance he has given me. What started out at the beginning of the year as a small discussion in the tea room about computer vision related honours projects has turned into topic I have become passionate about and want to pursue further.

I am also grateful to the other ML students and faculty of staff for letting me hog the ML servers to collect the results needed for this report.

Contents

Chapter 1

Introduction

There are a wide range of applications that require depth information to function, such as augmented reality which needs to reconstruct a 3D representation of its environment to work. Self Driving cars need to detect potentially hazardous objects to keep passengers safe, which requires depth of a scene to work. Depth information has also come to my attention for its use in the forestry industry. For example New Zealand's Crown research institute SCION reads depth information from flying drones over the tops of forest to get canopy height and density, allowing them to know when and what trees to harvest.

Currently there are two main categories of methods for estimating depth from an environment. These being "active" methods which receive light after bouncing it off the environment, and "passive" methods that do not emit anything and just receive light from arbitrary external sources such as lights in a room, or the sun. In general, active methods are fast and accurate although expensive and heavy. Passive methods require fewer components which reduces the cost and weight.

Examples of active methods are time of flight (LIDAR 1.1, Microsoft's second-generation Kinect) and structured light (Microsoft's first-generation Kinect). Time of flight gathers depth information by measuring the time of flight of a light signal between the camera and the subject for each point of the image. Structured light methods project a pattern onto an environment, and software estimates depth based on how this pattern is deformed. Both structured light and time of flight suffer from not working in bright environments. Also active methods are not suited to all environments. For example in under water environments, the water affects the scattering and absorption of the propagation of light waves which degrade the quality of readings.

Stereo Matching is a passive method that requires two cameras and processor. Since it does not need to emit anything it uses less power than active methods and requires fewer components than active imaging systems, making stereo implementations cheaper and more lightweight. Also it more adaptable to environments where light emission has issues, such as under water. Since it does not emit anything it could be suited to undetectable military application.

Figure 1.1: Image of a Velodyne LIDAR camera [7], the primary active Depth sensor used for Google's Self Driving cars. It uses 64 laser beams spinning around 10 times per second to capture 1.3 million depth readings of its environment. This unit costs $75,000 and weighs 12kg so if we want cheap self driving cars we need a better solution. Also, strapping a 12kg Velodyne camera to a drone doesn't sound like the best idea.

Additionally depending on the algorithms and environment it can outperform the accuracy of active methods.

Stereo algorithms come with their challenges though. Accurate algorithms have large memory and computational requirements making efficient embedded implementations difficult. Embedded platforms generally have far less memory and computational resources than desktop systems. Binary based matching cost methods are one way to reduce the computational and memory requirement of these algorithms. As a result I want to investigate learning binary methods that outperform conventional binary approaches.

1.1 Structure of report

A summary of stereo matching will first be given first to provide context behind the technology and previous research related to the project. The method chapter will include by an in depth explanation and pseudo code for the specific methods investigated for the stereo matching and genetic algorithm components of this project. Following this will be an analysis of conventional approaches compared to my method. I also provide an ablation study of stereo processing steps in section 5.3 to show the effect on accuracy and performance from removing steps from the pipeline.

1.2 Supplementary material

I have open sourced the Stereo Matching pipeline discussed in this project at `https://github.com/andrewmilson/total-generalized-variation-stereo-matching-pipeline`. I believe this is the only open source stereo matching code that includes a total generalised variation (TGV) method implementation. The code-base is written in CUDA C, a heterogeneous programming language developed by Nvidia for general-purpose GPU (GPGPU) programming. The code has been optimised and as a result is difficult to read. When analysing, the pseudo code provided in the Chapter 3 should be used as a reference. Animations of the TGV process and definitions for binary descriptors in Figure 5.4 have also been given in the repo's README.

Chapter 2

Background

This chapter gives an overview of stereo matching and introduces the previous work related to my project.

2.1 Overview

Stereo matching is an important problem in computer vision. The goal of stereo matching is to find a mapping between all pixels between two images taken at different viewpoints in a scene. This is known as the correspondence problem. The images I will be working with in this project will be rectified. Rectified images are horizontally aligned so that the search for correspondence is reduced from two dimensions (row and column) to one dimension (row). There are algorithmic approaches to rectify non-rectified images, however these will not be investigated.

Once rectified images are obtained it is given that corresponding pixels will be located on the same Y coordinate between images. The X component of this correspondence is estimated through a disparity function. The term disparity was introduced from human vision literature to describe the difference in location between features seen in the left and right eye. In computer vision, disparity can be defined as a projective transformation between pixels in the reference and target images. Reference and target are the names assigned to the input images. Either one of the images can be the reference and the other the target, although disparities represent the transformation, i.e. the shift along the X axis from the reference image to their corresponding pixels in the target image.

The output of the stereo matching algorithm I will be producing is a dense disparity map, also known as a disparity image. Each pixel value of the disparity image is the disparity between the corresponding pixel locations in the reference and target images.

There are two main types of stereo matching algorithms: local and global. Local methods use a window around the pixel of interest when estimating a disparity value. In contrast, global methods either scan lines or the whole image to determine the disparity value associated with a single pixel. Global stereo matching algorithms tend to produce more accurate representations at the expense of generally being more computationally intensive.

Stereo Matching algorithms usually involve these four steps [14]:

1. Computing matching cost;
2. Cost aggregation;
3. Computing disparity; and
4. Disparity refinement.

The remainder of this section describes these steps in more detail.

2.1.1 Computing matching cost

Suppose we have a reference image I_r. Let the target image I_t be it's stereo pair taken from a different viewpoint. For a given pixel (x_r, y_r) in I_r, the corresponding pixel will be (x_t, y_r) in I_t. The disparity of pixel (x_r, y_r) is $x_r - x_t$. To compute matching costs we compute a cost for all x_t such that $x_r - x_t \in [0, d_{max}]$, where d_{max} represents the max disparity search range. The result of this process is a cost volume, which is a three dimensional array containing matching costs for each image location and potential disparity value. There are several window based methods for computing matching costs. These can be divided into two groups, parametric and non-parametric methods [5].

Parametric methods calculate costs by comparing magnitudes of pixel intensities. Techniques for this are borrowed from statistics. Examples of such methods are Sum of Squared Differences (SSD) and Sum of Absolute Differences (SAD). Non-parametric methods use local ordering of pixel intensities to calculate costs. Non-parametric methods were originally introduced to be more robust than parametric methods against outliers that occur at object boundaries [17], and also handing differences in illumination and photometric calibration between images.

Descriptors

Descriptors are at the core of computer vision, due to their effectiveness in the application of object recognition and 3D reconstruction, among other applications. Descriptors are functions used to produce feature vectors of image patches that are more suitable for determining similarity than the raw pixel values themselves. A feature vector is a numerical description of a feature. Several methods exist for comparing similarity between features but euclidean and hamming distance are common choices. Descriptors can be designed to be invariant to gain, scale and rotation.

A commonly used example of such a descriptor is the gradient based method SIFT [9]. While effective, it is computationally expensive compared to binary based methods [3][12] and since computing and comparing descriptors take up a large computational chunk of many algorithms, it makes binary descriptors a more favourable choice.

Binary descriptors involve a subset of these three steps:

1. Sampling pattern - where to sample points in the patch.
2. Orientation adjustment - computation enabling rotational robustness.
3. Sampling pairs - what sampling pattern points to compare.

The output of a binary descriptor is a binary string where values represent comparisons between pixel intensities within the same patch. Similarity between strings can then be computed using hamming distance.

An example of a basic binary descriptor is the BRIEF [3] method. This involves a smoothing step and an arbitrary selection of sampling pairs. Since samples are of individual pixels of the patch and not sub regions, this makes pixel wise samples very noise sensitive. By smoothing this reduces the sensitivity to high frequency. By generating a length n binary vector, large values of n gives many options to choose sampling pairs. The paper [3] gives results for different sampling distributions and concludes that choosing random sampling pairs gives an advantage over symmetrical and grid based sampling pairs.

It is important to note that BRIEF has been extended to incorporate orientation and scale invariance by ORB [12] so that it can compete with methods such as SIFT. The ORB method will not be investigated by this project since I am performing stereo matching under known camera geometry with minimal perspective distortion, thus scale and orientation invariance would offer little benefit to accuracy, and decrease run-time.

Census transform

The census transform [17] is an example of a non-parametric local transform. The census transform is defined as the following. Let Ω represent the image space, i.e. the set of all possible image coordinated, then $p \in \Omega$ be a pixel and $N(p) \subset \Omega$ be the set of pixels within a square window of diameter D surrounding p. The intensity of a given pixel, since I am only investigating grey-scale images, is defined as the single valued function $I(p) \in \mathbb{R}$. All non-parametric local transforms involve performing comparisons between p and pixels within $N(p)$. Define τ as

$$\tau(p, p') = \begin{cases} 1 & I(p') < I(p) \\ 0 & otherwise \end{cases}$$

let $\otimes$ denote concatenation with a constant implied ordering. The census transform can be defined as the following binary feature string,

$$CT(p) = \bigotimes_{\forall p' \in N(p)} \tau(p, p')$$

To compute the matching costs both stereo images are transformed with the census transform. The cost is defined as the hamming distance between two feature strings from the census transformed images.

The hamming distance is a common calculation performed to compute the similarity between binary feature strings. It measures the minimum number of substitutions to change one string into another. Let S and S' be two binary strings of length k. Let S_i denote the binary value at index i in S and $\oplus$ denote the XOR operation on two binary values. The hamming distance can be specified,

$$H(S, S') = \sum_{i=1}^{k} S_i \oplus S'_i$$

Census transform is one of the most widely used functions involved in the calculation of matching costs due to its ability to deal with differences in camera gain and bias [14]. Costs are computed using hamming distance. Since this involves binary comparisons it makes hardware implementations trivial and fast. Additionally it performs fast on other architectures since both the SSE [6] and NEON [11] instruction sets have instructions to perform XOR and bit counting operations.

Note how the census transform can be described as a BRIEF-style descriptor around a central pixel where sampling pairs are taken on a square grid (Figure 5.4a).

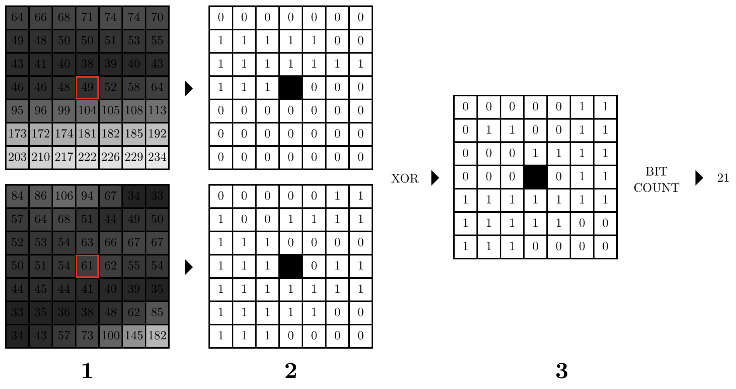


Figure 2.1: Overview of census transform - in stage one, pixel intensities are collected from a patch in either image. In stage 2, patch pixel intensities are compared to the central pixel to give a binary vector. In stage 3, patch similarity is given a score based on hamming distance between the two binary vectors. The higher the score the less similar the patches, similarly the lower the score the more similar the patches.

2.1.2 Cost aggregation

The cost aggregation stage is usually performed by local methods only. When the local regions of images are similar (i.e. noise or repetitive patterns) correspondence can be difficult to calculate using local methods due to their lack of regional information. As a result incorrect matches can easily have lower cost than correct ones due to noise etc. Cost aggregation aims to reduce this ambiguity by aggregating over a local support region in the cost volume. Cost aggregation can be seen as a class of methods for regularising cost volumes. As an example, a three dimensional median filter could be applied to the cost volume to improve global reasoning.

2.1.3 Computing disparity and optimisations

Local methods perform a "winner-take-all" (WTA) optimisation on disparities at each pixel which makes computing the final disparity image trivial. A disparity associated with the minimum matching cost value is selected for each pixel.

Global methods perform most of their computation during this stage. Many implementations use an energy minimisation framework. Their method for computing the disparity uses a disparity function that minimises global energy. A variety of algorithms can be used to find local minimum such as simulated annealing, gradient descent, Monte Carlo methods and dynamic programming.

2.1.4 Disparity refinement

Simply performing the previous steps results in integer disparity values representing correspondences. As a result, the disparity map appears coarse. To rectify this, sub-pixel refinements can be made to smooth disparities in the disparity image.

Several methods exist to calculate sub-pixel disparities, or more appropriately called floating point disparities. One such method is curve fitting. Consider taking a pixel in the disparity image p and its left and right neighbours of ± 1 disparity p_l and p_r. Fit a parabola to the winning costs of p_l, p and p_r. The disparity of pixel p is replaced with the x coordinate of the supremum or infimum if the curvature is positive or negative respectively.

2.2 Related work

The goal of this section is to introduce global disparity estimation algorithms related to the ones investigated in this project.

2.2.1 Semi-global matching

Semi-global matching is a stereo method which involves computing a pixel wise cost volume and approximating a global, 2D smoothness constraint by combining many 1D constraints. Any arbitrary cost function can be used to calculate the initial pixel wise cost volume. To improve accuracy the cost function should be strong and robust against camera gain and bias, such as the census transform.

Semi-global matching differs from other methods in its aggregation and optimisation stages. The method re-frames the correspondence problem as finding a disparity image D that optimises this energy function,

$$\begin{aligned} E(D) = \sum_{p} & C(p, D_p) + \\ & \sum_{q \in N(p)} P_1 T[|D_p - D_q| = 1] + \\ & \sum_{q \in N(p)} P_2 T[|D_p - D_q| > 1] \end{aligned} \tag{2.1}$$

The first term represents the sum of all matching costs for disparities D. The second term adds a penalty P_1 for pixels in the neighbourhood of pixel p defined by $N(p)$ for which the disparity changes by 1 pixel. The third term penalty P_2 adds a greater penalty than P_1 for disparity changes larger than 1 pixel. This effectively allows lower penalties for slight slants or curves and larger penalties for unwanted noise.

Minimisation of $E(D)$ is NP-complete for 2D values of D, although 1D values of D (e.g. rows, columns and diagonals) can be solved in polynomial time using dynamic programming.

Solving rows in 1D leaves the resulting disparity image suffering from streaking [14], since there are strong constraints across rows with weak constraints along columns. The semi-global method is to rather aggregate by summing the costs of 1D paths in multiple arbitrary directions. The final cost $\theta(p, d)$ for pixel p at disparity d is defined as the sum of costs of the paths θ' in all directions R,

$$\begin{aligned} \theta'(p, d, r) = & C(p, d) + min(\theta'(p - r, d), \\ & \theta'(p - r, d - 1) + P_1, \\ & \theta'(p - r, d + 1) + P_1, \\ & min_i \theta'(p - r, i) + P_2)) \end{aligned}$$

$$\theta(p, d) = \sum_{\forall r \in R} \theta'(p, d, r)$$

Taking eight to 20 paths gives good coverage of the image and is a heuristic method of achieving the behaviour of equation (2.1). The final disparity image is then found by performing a WTA optimisation over the aggregated costs.

2.2.2 Stereo Reconstruction using Variational Methods

By treating an image as a three dimensional non-linear surface, the total variation of it can be defined as the area of the surface. Images with high amounts of fine detail, such as noise, will have a larger overall total variation. Total Variation Regularisation (TV) is a technique to reduce total variation whilst subjecting the image to be close to its original. TV formulas [13] were introduced as a noise reduction technique in the 90s. They can be applied to stereo matching problems and be solved globally optimally. TV methods favour piecewise constant solutions. This leads to fronto-parallel artifacts in the estimated disparity image [10]. TGV [1] is a generalisation of TV and is a better performing variational method than TV for stereo due to it composing solutions of piecewise polynomials. In effect TGV relaxes the smoothness constraint and in turn allows for fronto-parallel, affine and quadratic surfaces in the estimated disparity image. TGV problems can only be solved optimally if the data terms are convex. With a non convex data term such as a cost volume the TGV solution will be only locally optimal.

Several techniques have been proposed to solve this, such as using a convex approximation of the data term in a coarse to fine warping approach [2]. In this approach the resolution of the disparity image to estimate for has been lowered significantly (course) such that the data term represented by a convex function. This convex data term in combination with a TGV regulariser can then be optimised to estimate the corresponding low resolution disparity image. This process is repeated by initialising the data term by using an incrementally scaled up version the finest available disparity image. This course to fine approach is repeated until the disparity image reaches a sufficiently fine level. Using this method leads to a solution that lies in a better local minima. The downside of this approach is that fine scene details not visible in low resolution disparity images are highly likely to be missing in the final reconstruction.

This project uses an augmented Lagrangian based method [8] for optimising a non-convex data term in combination with a TGV regulariser that boasts an improvement at preserving fine scene details and a speedup by a factor of two over course-to-fine based methods. The smoothness and data term are optimised in tandem using iterative gradient descent method for solving an energy functional that reads

$$E = \int_{\Omega} \{\lambda_s |G(\nabla u(x) - v(x))| + \lambda_a |\nabla v(x)| + \lambda_d C(u(x))\} dx, \tag{2.2}$$

where $u(x) \to \mathbb{R}$ represents the disparity map to solve for. The values λ_s, λ_a, λ_d balance the penalties given to smooth, affine and data term respectively. The effect of finding $u(x)$ is a disparity image which penalises certain types of noise while preserving the original data. Overall this results in a vast accuracy improvement from reducing overall noise. In many variational methods symbols are defined on continuous intervals such as $\mathbb{R}$ since images are treated as differentiable functions rather than arrays.

An added benefit to TGV based algorithms is that they are well suited for parallelism. With the ease of GPU programming nowadays on platforms such as CUDA and OpenCL, given the right constraints, TGV enables efficient and accurate algorithms for dense stereo estimation.

Chapter 3

Stereo Matching Pipeline

The proposed algorithm involves three distinct steps: computing matching costs, cost aggregation and disparity estimation. Each step feeds into the next. A disparity image can be produced at each subsequent step, although all three are required to yield disparity maps that are usable in practice. Algorithm 11 outlines how the steps integrate with one another.

3.1 Matching Costs

Suppose we want to calculate correspondence between a reference image I_r and target image I_t with the same dimensions $W \times H$. To do so, we must first compute a cost volume. Computation of the cost volume is a two step process. Firstly, matching images are transformed with a BRIEF-style descriptor (Algorithm 1). Secondly, costs are computed using the hamming distance between pixels in the transformed images (Algorithm 2). The disparity image can the be generated by performing a WTA optimisation (Algorithm 3) on the cost volume. This WTA optimisation works on a cost volume by choosing a disparity for each pixel associated with the lowest cost. The output it a disparity image.

Note that in Algorithm 1, $\otimes$ denotes concatenation, P represents BRIEF sampling pairs where $(p_1, p_2) \in P$ each represent sampling positions $(x_1, y_1) = p_1$ and $(x_2, y_2) = p_2$ relative to a central pixel of interest.

3.2 Cost Aggregations

Image ambiguity is one of the biggest issues of creating accurate disparity maps. Repetitive textures make it difficult to get good disparities estimates using matching costs alone. As a result aggregation is a common step carried out by stereo algorithms to yield usable disparity maps in practice. This project uses [16] to perform cost aggregation. It has been described here for completeness.

Algorithm 1 BRIEF Transform

```
function TransformBRIEF($I, P$)    ▷ Image $I$, sampling pair vector $P$
    Initialize the transformed image $I_t$ with empty strings
    for all positions $p \in I$ do
        for all sampling pair positions $(p_1, p_2) \in P$ do
            $I[p + p_1] \leftarrow 0$ if position $p + p_1$ does not exist in $I$
            $I[p + p_2] \leftarrow 0$ if position $p + p_2$ does not exist in $I$
            $\tau \leftarrow \begin{cases} 1 & I[p+p_1] < I[p+p_2] \\ 0 & \textit{otherwise} \end{cases}$
            $I_t[p] \leftarrow I_t[p] \otimes \tau$
    return $I_t$
```

Algorithm 2 Matching cost volume

```
function Costs($I_r, I_t, D$)    ▷ Matching images $I_r, I_t$, max disparity $D$
    Initialize cost volume $C \subset \mathbb{R}^3$ of dimensions $W \times H \times D$
    $I'_r \leftarrow$ TransformBRIEF($I_r$)
    $I'_t \leftarrow$ TransformBRIEF($I_t$)
    for all positions $(x, y) \in I'_r$ do
        $s_r \leftarrow I'_r[x, y]$
        for all $d \in [0, D]$ do
            $s_t \leftarrow I'_t[x - d, y]$
            $C[x, y, d] \leftarrow$ HammingDistance($s_r, s_t$)
    return $C$

function HammingDistance($s_1, s_2$)    ▷ Binary strings $s_1, s_2$
    $s \leftarrow s_1 \oplus s_2$
    $n \leftarrow 0$
    $i \leftarrow 0$
    repeat
        if $s_0 = 1$ then
            $n \leftarrow n + 1$
        Right shift $s$ by 1
        $i \leftarrow i + 1$
    until $i = |s|$
    return $n$
```

Algorithm 3 Winner-Take-All (WTA)

```
1:  function WTA($C$)                                  ▷ Cost volume $C$
2:      Initialize disparity image $I_d$
3:      for all positions $(x, y) \in C$ do
4:          $c_{min} \leftarrow C[x, y, 0]$
5:          $d_{min} \leftarrow 0$
6:          $d \leftarrow 1$
7:          while $d < |C[x, y]|$ do
8:              if $C[x, y, d] < c_{min}$ then
9:                  $c_{min} \leftarrow C[x, y, d]$
10:                 $d_{min} \leftarrow d$
11:             $d \leftarrow d + 1$
12:         $I_d[x, y] \leftarrow d_{min}$
```

The methods outlined in [16] take inspiration from the Gestalt Grouping principals of similarity and proximity to compute "support weights" for values in the cost volume. The support weight of a pixel is defined as

$$SW(p, q) = W_{sim}(p, q) \times W_{prox}(p, q)$$

Where $W_{sim}(p, q)$ and $W_{prox}(p, q)$ represent the weight based on grouping pixels p and q by similarity and proximity respectively. Let $\Delta I(p, q)$ represent the difference in intensity between pixel p and pixel q

$$W_{sim}(p, q) = 1 - exp\left(-\frac{\Delta I(p, q)}{\gamma_{sim}}\right)$$

The behaviour of this is that $W_{sim}(p, q)$ grows inversely proportional to the intensity difference between p and q. It is important to note that the original formula calculates the similarity of two pixels by taking the euclidean distance between pixel colour values in the CIELab colour space. Since this project is investigating grey scale imagery this has been replaced with absolute difference between the intensities of two pixels. This does not have a negative effect on the algorithm since grey-scale values in the CIELab colour space are proportional to intensity values. Let $\Delta D(p, q)$ represent the euclidean distance between pixels p and q.

$$W_{prox}(p, q) = exp(-\frac{\Delta D(p, q)}{\gamma_{prox}})$$

The behaviour is that $W_{prox}(p, q)$ decreases inversely proportionally as the euclidean distance between pixels p and q increase.

The effect both these functions have when multiplied together in $SW(p, q)$ is that the similarity weight has less of an effect the further the pixels' proximity between each other are. Similarly the proximity weight has less of an effect if the pixels are less similar.

The dissimilarity between pixels p and q'_d, $E(p, p'_d)$ can now be represented by

$$E(p, p'_d) = \frac{\sum_{q \in N_p, q'_d \in N_{q'_d}} SW(p, q) SW(p'_d, q'_d) e(q, q'_d)}{\sum_{q \in N_p, q'_d \in N_{q'_d}} SW(p, q) SW(p'_d, q'_d)}$$

where p'_d and q'_d are the corresponding pixels in the matching image when pixels p and q have a disparity of d. $e(q, q'_d)$ represents the pixel-based raw matching cost between q and q'_d.

The effect this function has when computed over all disparity values for pixels in the reference image is a local 3D window based smoothing over the cost volume. Neighbouring cost values in a support window are added together and weighted by their similarity and proximity to the reference pixel. The end result is a cost volume with less overall ambiguity. The technique is also efficiently implemented on a GPU, since support-weights can be computed in parallel.

Pseudo code for the similarity and proximity functions are in Algorithm 4. Pseudo code for performing support-weight aggregation over a cost volume is given in Algorithm 5.

Algorithm 4 Support Weight

```
1: function SW($I, p, q$)                                   ▷ Image $I$, positions $p, q$
2:     return $exp(-(\Delta C(I, p, q)/\gamma_c + \Delta G(p, q)/\gamma_p))$

3: function $\Delta$C($I, p, q$)                            ▷ Image $I$, positions $p, q$
4:     $I[p] \leftarrow 0$ if position $p$ does not exist in $I$
5:     $I[q] \leftarrow 0$ if position $q$ does not exist in $I$
6:     return $|I[p] - I[q]|$

7: function $\Delta$G($p, q$)                               ▷ Positions $p, q$
8:     $(x_p, y_p) = p$
9:     $(x_q, y_q) = q$
10:    $x \leftarrow x_p - x_q$
11:    $y \leftarrow y_p - y_q$
12:    return $\sqrt{x^2 + y^2}$
```

3.3 Disparity Estimation

The main stereo algorithm utilised by this project is based on TGV regularisation. The algorithm proposed in Algorithm 9 is used to solve the energy minimisation problem given in equation 2.2. It is a gradient descent based method that optimises a smoothness and data term in tandem. The algorithm outlined in [8] is vague since there is information missing about initialisation

Algorithm 5 Support Weight Aggregation

```
    ▷ Raw matching costs $C$, window radius $r$, matching images $I_r, I_t$
 1: function AGGREGATE($C, r, I_r, I_t$)
 2:     Initialise aggregated cost volume $C_a$ to the size of $C$
 3:     for all positions $(x, y, d) \in C$ do
 4:         $p \leftarrow (x, y)$
 5:         $p' \leftarrow (x - d, y)$
 6:         $S_u \leftarrow S_l \leftarrow 0$
 7:         $y_d \leftarrow x_d \leftarrow -r$
 8:         for $y_d \leq r$ do
 9:             for $x_d \leq r$ do
10:                 $q \leftarrow p + (x_d, y_d)$
11:                 $q' \leftarrow q + (x_d, y_d)$
12:                 $sw \leftarrow SW(I_r, p, q) * SW(I_t, p', q')$
13:                 $S_u \leftarrow S_u + sw * C[x, y, d]$
14:                 $S_l \leftarrow S_l + sw$
15:                 $x_d \leftarrow x_d + 1$
16:             $y_d \leftarrow y_d + 1$
17:         $C_a[x, y, d] \leftarrow S_u / S_l$
18:     return $C_a$
```

of certain variables. Also no information was provided about computing divergence. No source code was provided either, making implementation with this lack of information difficult. As a result a more detailed outline and pseudocode has been provided for the method in this project to aid understanding and aid further implementation efforts.

The constants $\tau_u, \tau_p, \tau_v, \tau_q$ in Algorithm 10 limit the maximum size of the gradient ascents/descents by bounding them to $\tau_u = \tau_p = \frac{1}{\sqrt{12}}$ and $\tau_v = \tau_q = \frac{1}{\sqrt{8}}$. In Algorithm 9, C_{max} represents the maximum cost in the cost volume.

An animation of the gradient descent process of TGV can be visualised on the GitHub page for this report `https://github.com/andrewmilson/total-generalized-variation-stereo-matching-pipeline`.

Algorithm 6 Computes the finite forward differences of an image using Neumann boundary conditions. A technique to compute an image's discrete derivative.

```
function ForwardDifferences($I$)                ▷ $n$ channel image $I$
    Initialise $2n$ channel image $I_{grad}$
    for all positions $(x, y) \in I$ do
        $i \leftarrow 0$
        while $i < n$ do
            $V \leftarrow I[x, y, i]$
            $V_{east} \leftarrow I[x + 1, y, i]$ or $V$ if undefined
            $V_{south} \leftarrow I[x, y + 1, i]$ or $V$ if undefined
            $I_{grad}[x, y, 2i] \leftarrow V_{east} - V$
            $I_{grad}[x, y, 2i + 1] \leftarrow V_{south} - V$
            $i \leftarrow i + 1$
    return $I_{grad}$
```

Algorithm 7 Computes the finite backward differences of an image using Dirichlet boundary conditions. A technique to compute an image's discrete derivative.

```
function BackwardDifferences($I$)               ▷ $n$ channel image $I$
    Initialize $2n$ channel image $I_{grad}$
    for all positions $(x, y) \in I$ do
        $i \leftarrow 0$
        while $i < n$ do
            $V \leftarrow I[x, y, i]$
            $V_{west} \leftarrow I[x - 1, y, i]$ or 0 if undefined
            $V_{north} \leftarrow I[x, y - 1, i]$ or 0 if undefined
            $I_{grad}[x, y, 2i] \leftarrow V - V_{west}$
            $I_{grad}[x, y, 2i + 1] \leftarrow V - V_{north}$
            $i \leftarrow i + 1$
    return $I_{grad}$
```

Algorithm 8 Computes divergence using finite backward differences

```
Require:
    $2n$ channel image $I$

function DIVERGENCE($I$)
    Initialize $n$ channel image $I_{div}$
    $\nabla I \leftarrow$ BACKWARDDIFFERENCES($I$)
    for all positions $(x, y) \in \nabla I$ do
        $i \leftarrow 0$
        while $i < n$ do
            $I_{div}[x, y, i] \leftarrow \nabla I[x, y, 4i] - \nabla I[x, y, 4i + 3]$
            $i \leftarrow i + 1$
    return $I_{div}$
```

Algorithm 9 Total Generalized Variation (TGV)

```
function TGV($C, N$)                                ▷ Cost volume $C$, iterations $N$
    Initialize two channel images $p, v, \hat{v}$ with 0s
    Initialize four channel image $q$ with 0s
    $L \leftarrow a \leftarrow u \leftarrow \hat{u} \leftarrow \mathrm{WTA}(C)/D$
    $\theta \leftarrow 1$
    $n \leftarrow 0$
    repeat
        SMOOTH($L, a, u, \hat{u}, v, \hat{v}, p, q, \theta$)
        $C_a \leftarrow$ UPDATECOSTS($C, L, a, u, \theta$)
        $a \leftarrow \mathrm{WTA}(C_a)/D$                       ▷ Max disparity $D$
        $L \leftarrow L + \frac{1}{2\theta}(u - a)$
        $\theta \leftarrow \theta(1 - \beta n)$
        $n \leftarrow n + 1$
    until $n = N$
    return $a * D$

function UPDATECOSTS($C, L, a, u, \theta$)
    Initialise cost volume $C_u$
    for all positions $(x, y, d) \in C$ do
        $\delta \leftarrow u[x, y] - d/D$                       ▷ Max disparity $D$
        $C_u[x, y, d] \leftarrow \lambda_D C[x, y, d] + C_{max}(L[x, y] * \delta + \frac{\delta^2}{2\theta})$
    return $C_u$
```

Algorithm 10 TGV Smoothing

```
procedure SMOOTH($L, a, u, \hat{u}, v, \hat{v}, p, q, \theta$)
    $i \leftarrow 0$
    repeat
        $\nabla\hat{u} \leftarrow$ FORWARDDIFFERENCES($\hat{u}$)
        $\nabla\hat{v} \leftarrow$ FORWARDDIFFERENCES($\hat{v}$)
        for all positions $(x, y) \in u$ do
            $\nabla\hat{v}[x, y, 1] \leftarrow \nabla\hat{v}[x, y, 2] \leftarrow \frac{1}{2}\nabla\hat{v}[x, y, 2]$
            $p[x, y] \leftarrow p[x, y] + \tau_p(\nabla\hat{u}[x, y] - \hat{v}[x, y])$
            $\|p\| \leftarrow min(1, \lambda_s/p[x, y]_{max})$
            $p[x, y] \leftarrow p[x, y] * \|p\|$
            $q[x, y] \leftarrow q[x, y] + \tau_q\nabla\hat{v}[x, y]$
            $\|q\| \leftarrow min(1, \lambda_a/q[x, y]_{max})$
            $q[x, y] \leftarrow q[x, y] * \|q\|$
        $p_{div} \leftarrow$ DIVERGENCE($p$)
        $q_{div} \leftarrow$ DIVERGENCE($q$)
        $u_{tmp} \leftarrow u$
        $v_{tmp} \leftarrow v$
        for all positions $(x, y) \in u$ do
            $u[x, y] \leftarrow \frac{u_{tmp}[x,y]+\tau_u p_{div}[x,y]+\frac{\tau_u}{\theta}a[x,y]}{1+\frac{\tau_u}{\theta}}$
            $\hat{u}[x, y] \leftarrow 2u[x, y] - u_{tmp}[x, y]$
            $v[x, y] \leftarrow \tau_v(p[x, y] + q_{div}[x, y])$
            $\hat{v}[x, y] \leftarrow 2v[x, y] - v_{tmp}[x, y]$
        $i \leftarrow i + 1$
    until $i = 150$
```

Algorithm 11 Stereo matching pipeline

```
Require:
    Binary descriptor $f$
    Reference image $I_r$, Target image $I_t$
    Maximum disparity $D$
    Aggregation window radius $r$

procedure STEREOMATCHING($f, I_r, I_t, D, r$)
    Initialise an empty disparity image $I_d$
    $C \leftarrow$ COSTS($I_r, I_t, D$)
    if performing cost aggregation then
        $C \leftarrow$ AGGREGATE($C, r, I_r, I_t$)
    if performing TGV then
        $I_d \leftarrow TGV(C, 80)$
    else
        $I_d \leftarrow WTA(C)$
    return $I_d$
```

Chapter 4

Optimising Binary Feature Extractors with Genetic Algorithms

Genetic algorithms (GA) are meta-heuristic algorithms in computer science inspired by Charles Darwin's theory of natural evolution. They can be designed to optimise a function by using biologically inspired operations such as mutation, crossover and selection. This project uses a genetic algorithm to optimise a BRIEF-style binary descriptor for use in stereo matching. This section will outline the methods I use to calculate fitness, perform crossover, apply mutations, and select members of the population to carry over into the next generation. Then I will outline how these methods are synthesised to simulate evolution.

Chromosomes for the GA are BRIEF-style binary feature extractors. Each feature extractor can be represented as a vector G where $(b_1, b_2) \in G$ each represent sampling positions $(x_1, y_1) = b_1$ and $(x_2, y_2) = b_2$ relative to a central pixel of interest.

Mutation of a chromosome is defined in Algorithm 12. The effect mutation has on a chromosome is changing any number of the sampling pair positions. Since mutations are probabilistic they are not guaranteed to occur. Each x or y coordinate present in the chromosome to be replaced, with probability p_m, by a random number sampled from a uniform distribution. Using this scheme allows for sampling pairs to change direction, shift position or change length. For example a vertical sampling pair could be mutated into a horizontal sampling pair.

Crossover of two parent chromosomes P, P' is defined in Algorithm 13. If crossover occurs, a proportion of parent chromosomes sampling pairs are combined with one another to create two offspring chromosomes that contain resemblances of both parents. Specifically, parent chromosomes are split into two parts, head and tail, at a uniform random sampling pair index, with probability p_c. Offspring are the concatenation of a parents head with the other

Algorithm 12 Mutation

```
    ▷ Chromosome $C$, mutation probability $p_m \in [0,1]$
1: procedure MUTATE($C, p_m$)
2:     for all pairs $(b_1, b_2) \in C$; do
3:         if uniformly random $r \in [0,1]; r < p_m$ then
4:             Assign new position $b_1$
5:         if uniformly random $r \in [0,1]; r < p_m$ then
6:             Assign new position $b_2$
```

parents tail. If crossover occurs, the parents are removed from the gene pool and the offspring take their place in the population. If crossover does not occur the parents are passed on "as is" to the next generation.

Algorithm 13 Crossover

```
Require:
    Parents $P, P' : \|P\| = \|P'\|$
    Crossover rate $p_c \in [0,1]$

1: function CROSSOVER($P, P', p_c$)
2:     if uniformly random $r \in [0,1]; < p_c$ then
3:         Split position $s \in [2, 3, ..., \|P\| - 1]$
4:         $C \leftarrow \{P_1, ..., P_s, P'_{s+1}, ..., P'_{s+m}\}$        ▷ $s + m = \|P\|$
5:         $C' \leftarrow \{P'_1, ..., P'_s, P_{s+1}, ..., P_{s+m}\}$
6:         return $C, C'$
7:     return $P, P'$
```

Fitness $\bar{f}$ of chromosome G is defined in Algorithm 14. It is calculated as the reciprocal of the average error accumulated over a set of training images. Each training set item consist of a reference image I_r , target image I_t and ground truth disparity image I_{GTD}.

Selection of the population is the process of deciding what chromosomes make it through to the following generation. This selection occurs after fitness has been evaluated for all members of the current population. The population is sorted based on fitness and truncated to the population size such that the members of the population with the lowest fitness are removed.

The synthesisation of these methods is as follows. First an population is chosen arbitrarily. The fitness of the population is calculated and selection takes place to determine what members make it through to the second generation. Each member of the selected population performs crossover and mutation. The fitness of this new population is calculated and selection takes place. This process is repeated an arbitrary number of times although is halted if convergence occurs. Meaning the fitness of the population halts for a prolonged number of generations.

Algorithm 14 Fitness

```
Require:
    Error threshold $thresh$

 1: function FITNESS($C$)
 2:     $E \leftarrow 0$
        ▷ Reference $I_r$, target $I_t$ and GT disparity $I_{\text{GTD}}$ images
 3:     for all Images $(I_r, I_t, I_{\text{GTD}}) \in S$ do
            ▷ Generate disparity image $I_d$ from $I_r$ and $I_t$
            ▷ Max disparity $D$ is data set specific
 4:         $I_d \leftarrow$ STEREOMATCHING($C, I_r, I_t, D, 7$) with binary descriptor $C$
            ▷ Evaluate accuracy of $I_d$ against $I_{\text{GTD}}$
 5:         $e \leftarrow$ EVALUATE($I_d, I_{\text{GTD}}, thresh$)
 6:         $E \leftarrow E + e$
 7:     $\bar{f} \leftarrow \frac{1}{E}$
 8:     return $\bar{f}$

 9: function EVALUATE($I_d, I_{\text{GTD}}, thresh$)
10:     $n_{tot} \leftarrow n_{thresh} \leftarrow 0$
11:     for all Positions $(x, y) \in I_{\text{GTD}}$ do
12:         continue if $I_{\text{GTD}}[x, y] = 0$
13:         $n_{tot} \leftarrow n_{tot} + 1$
14:         $diff \leftarrow |I_{\text{GTD}}[x, y] - I_d[x, y]|$
15:         if $diff > thresh$ then
16:             $n_{thresh} \leftarrow n_{thresh} + 1$
17:     return $\frac{n_{thresh}}{n_{tot}} * 100$
```

Chapter 5

Evaluation

5.1 Experiments

The goal of this section is to quantify the performance of evolving BRIEF-style binary descriptors using genetic algorithms. I will be running two genetic algorithm experiments. Both will have the same hyper-parameters and initial populations will be randomly generated of the same size. One of the populations will be seeded with census transform (Figure 5.4a) and census wide transform (Figure 5.4b) to see if optimal solutions borrow from these methods. This means that the population is mostly random with only one instance of census and one instance of census wide.

After the genetic algorithms converge, top performing descriptors from both will be compared to conventional methods on 100 test scenes from the 2015 KITTI data set to evaluate performance. Further details about the data set used for training and algorithm parameters used are discussed below. An ablation study has also been performed to measure the effect on performance and accuracy of removing steps from the stereo pipeline.

5.1.1 Data Sets

This project uses two data sets for measurements. The 2015 KITTI [4] stereo evaluation data set and a combination of the 2001 and 2003 Middlebury [15] data sets.

The KITTI data set consists of 200 training and 200 test scenes captured by driving around rural areas and on highways. The test images do not provide ground truth disparity images, therefore we will not be using them for any form of evaluation in this project. I have split the KITTI training data into two subsets. Training set will be the first 100 images from the KITTI training set. The second 100 images will be our test set. The training set will be used to calculate fitness for our genetic algorithm. The test set will be used to evaluate our final descriptors. The ground truth for this data set has been collected by combining a variety of sensors. These sensors collect ground truth information for regions of the disparity map that are not obtainable using stereo matching. For example, pixels on the far left of the left image do not have corresponding pixels in the right image. Therefore the KITTI ground truth comes with two categories of ground truth information. One that excludes ground truth information falling outside the image plane. The other has all ground truth available from the sensors. The ground truth with excluded information is marked as “Non-Occ” and where all ground truth available is marked as “All”.

The data I am using from Middlebury is a combination from their 2001 and 2003 data sets. Specifically the scenes include cones (2003), teddy (2003), Sawtooth (2001), Venus (2001), Bull (2001), Poster (2001), Barn 1 (2001) and Barn 2 (2001). The 2001 scenes are of piecewise planar scenes and 2003 scenes comprise of complex geometry (Mugs, sticks, soft toys). While Middlebury provides newer data the TGV[8] method we are using provides parameters they found effective on items from the 2001 and 2003 data sets and not newer ones.

I have decided to use KITTI for training the genetic algorithm since scenes contain more realistic environments where stereo matching will be used. The more basic Middlebury data set will play no role in the evolution process simulated by the genetic algorithm although it will be for testing.

5.1.2 Matching Cost Parameters

For all matching cost computations I am using binary descriptors with a window size of 17×17. By operating on such a large window we open up solutions to use information from far away from the central pixel. This also does not have an adverse effect on solutions within a small diameter of the central pixel when trained for long enough. Choosing a census window size of 7×7 was found to be effective in [8]. This setup results in feature vectors of length $7 * 7 - 1 = 48$. This project uses feature vectors of length $\|V\| = 64$ so that features can fit into 64 bit integers for performance and memory reasons.

5.1.3 Aggregation Parameters

A window size of 7×7 has been selected to use for aggregation. In [8] a window size of seven was found to reduce the effect of foreground flattening while not putting too much fronto-parallel assumption on the cost volume. Proximity γ_p and similarity γ_c have been set to 14 and 8 respectively. These values we found to work best with the KITTI data set by experimentation.

5.1.4 TGV

We have used the same regularisation parameters outlined in [8] for TGV since they provide parameter sets for the same data sets we will be using in this project. For KITTI the parameter sets are $\{\lambda_d = 1.0, \lambda_s = 0.2, \lambda_a = 8\lambda_s\}$ and the Middlebury parameter set is $\{\lambda_d = 0.4, \lambda_s = 1.0, \lambda_a = 8\lambda_s\}$.

Level	Approximations	Coefficients
8	1, 1, 2, 5, 9, 10, 13, 15	
4	1, 3.5, 9.5, 14	0, -1.5, -0.5, -1
2	2.5, 11.7	0, -1.5, -0.5, -1

Table 5.1: Comparison based on using different stereo matching pipeline stages. The table illustrates the mean non-occluded 3 pixel error on the 100 test scenes from the KITTI 2015 stereo evaluation set. Comparison is made between five different BRIEF-style descriptors being census (C, Figure 5.4a), census wide (CW, Figure 5.4b), Random (R, Figure 5.4c), top performing descriptor from genetic algorithm optimised with random population (GA-R, Figure 5.4e), top performing descriptor from genetic algorithm optimised with random population with census seed (GA-RC, Figure 5.4d)

5.1.5 Genetic Algorithm Hyper-parameters

A larger population size results in being able to search more paths simultaneously, although this increases the time needed to execute the algorithm, due to more chromosomes needing their fitness calculated. This project finds 100 to be a suitable population size to enable multiple paths to be searched simultaneously while making the GA run in a realistic amount of time. A mutation probability of $p_m = 0.02$ has been used. This is marginally more than $\frac{1}{\|V\|} \approx 0.016$ allowing for slightly more than two sampling pair positions to be mutated per chromosome each generation on average. A crossover rate of $p_c = 0.8$ was found from experimentation to be effective at yielding good results.

The genetic algorithms calculates fitness concurrently across six machines each running two instances of the Stereo Matching program. Using this setup each member of the population took ∼2 minutes to calculate and collect fitness over the 100 training scenes. Therefore it took ∼3 hours to get all fitnesses for one generation. Each genetic algorithm was run for 45 generations each. In total that is about 11 days of this setup running continuously and a total of 2,640 GTX 1080 Ti GPU hours. Both genetic algorithms (Figures 5.1, 5.2) had not converged due to time constraints. This is why a grid search was not used to find optimal hyper-parameters: it would have been too slow to collect results without spending large amounts of money on cloud GPU instances. A grid search was attempted using a stochastic fitness function, which sampled a random 10 training images although there was too much noise in the data set for this to have been viable. The limited computation budget also deterred me from trying other meta heuristic algorithms.

5.2 Comparison to Conventional Approaches

Neural nets are “state-of-the-art” for computing matching costs although their memory requirements make them infeasible for embedded applications. Census transform is the “state-of-the-art” when it comes to non-parametric methods

	KITTI					
	> 2 pixel		> 3 pixel		> 4 pixel	
	Non-Occ	All	Non-Occ	All	Non-Occ	All
C[17]	12.93	14.36	9.50	10.96	7.60	9.07
CW	10.78	12.25	7.75	9.25	6.23	7.73
R	11.63	13.10	7.56	9.08	5.97	7.51
GA-R	10.01	11.50	6.74	8.26	5.44	6.97
GA-RC	9.51	11.00	6.63	8.14	5.38	6.89
GA-I1	11.39	12.84	8.15	9.63	6.65	8.13
GA-I2	12.56	13.99	9.35	10.82	7.60	9.06
GA-I3	14.82	16.22	11.38	12.82	9.75	11.20

C = Census transform (Figure 5.4a)	R = Random (Figure 5.4c)
CW = Census spread transform (Figure 5.4b)	GA-I1 = GA inspired (Figure 5.4g)
GA-R = GA optimised (Figure 5.4e)	GA-I2 = GA inspired (Figure 5.4h)
GA-RC = GA optimised, census seed (Figure 5.4d)	GA-I3 = GA inspired (Figure 5.4i)

Table 5.2: Comparison of the mean error % of different descriptors across several thresholds on 100 test scenes from the KITTI 2015 data set.

	Middlebury		
	> 1 pixel	> 2 pixel	> 3 pixel
C[17]	11.76	10.31	9.325
CW	11.80	10.38	9.39
R	14.48	12.91	11.51
GA-R	13.10	11.65	10.38
GA-RC	12.39	11.01	9.84
GA-I1	11.69	10.46	9.42
GA-I2	11.63	10.35	9.45
GA-I3	14.34	12.44	11.20

Table 5.3: Comparison of the mean error % of different descriptors across several thresholds on scenes from the 2003 and 2001 Middlebury data sets. Refer to the descriptor key in Table 5.2.

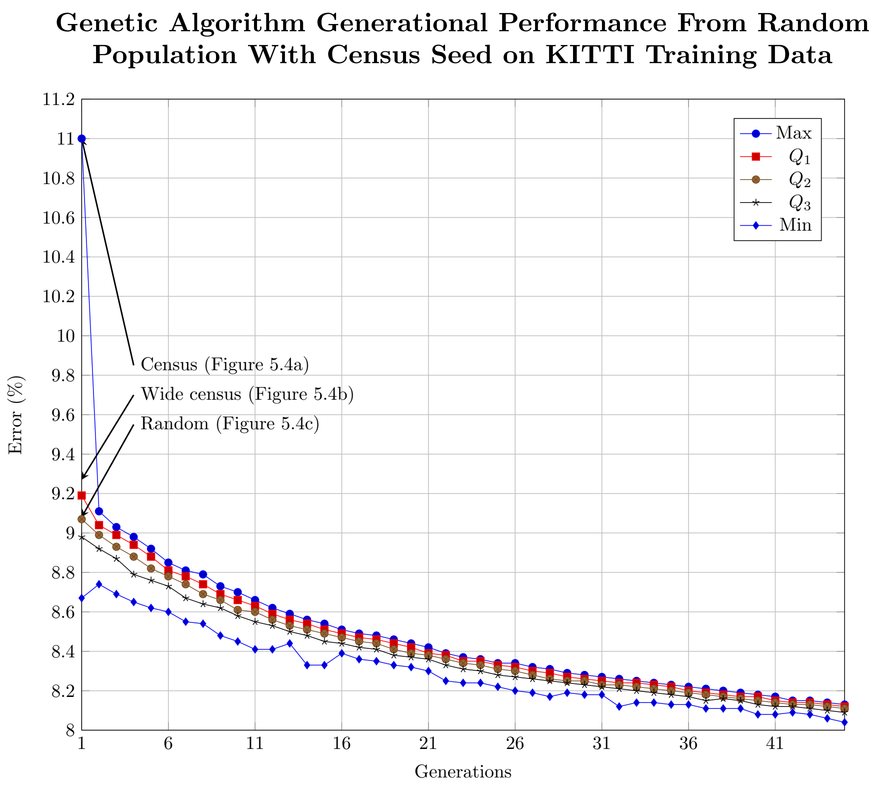


Figure 5.1: Error % represented by mean error from 3-pixel threshold across 100 training scenes from the 2015 KITTI training data. The “All”ground truth data was used for evaluation. Q_1, Q_2, Q_3 represent the lower quartile, median and upper quartile respectively. The initial population was randomly sampled although seeded with standard census transform (Figure 5.4a) and wide census transform (Figure 5.4b)

and out performs parametric methods by computing matching costs based on local orderings of pixel intensities (also a property of using binary descriptors for stereo matching) making it invariant to camera gain and bias. I have shown in Table 5.2 that random based binary descriptors outperform the census transform on all threshold levels by large margins. The top performing genetic optimised binary descriptor achieves a 3-pixel stereo error of 8.14%, whereas census transform obtains 10.96%. A 2.82% increase in accuracy with no decrease in performance. All random based binary descriptors perform worse than census transform on Middlebury but I am not as concerned about this since that data set contains a lot of fronto-parallel surfaces and is not a good representation of the kinds of scenes practical applications of stereo will encounter, such as the scenes in the 2015 KITTI data set.

The GA optimised binary descriptors appeared to have common features.

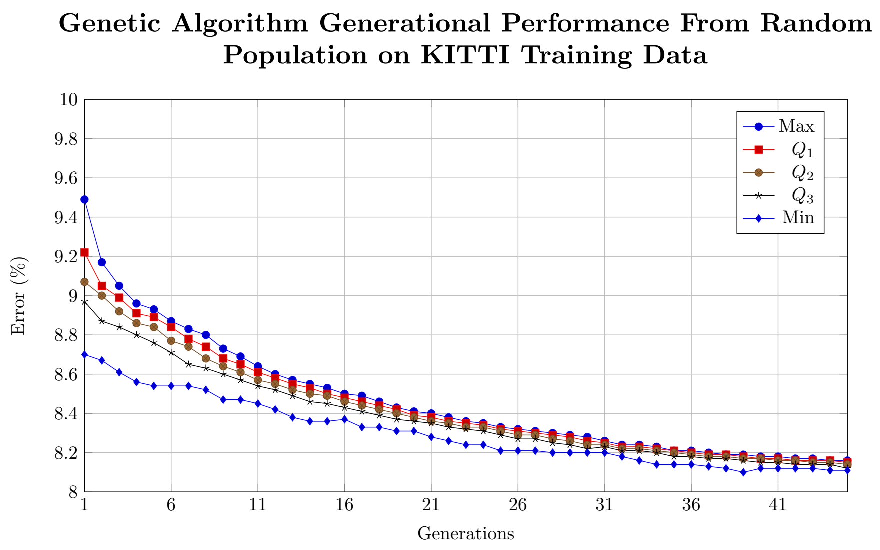


Figure 5.2: Error % represented by mean error from 3-pixel threshold across 100 training scenes from the 2015 KITTI training data. The "All"ground truth data was used for evaluation. Q_1, Q_2, Q_3 represent the lower quartile, median and upper quartile respectively. The initial population was randomly sampled.

As a result I was tempted to engineer some binary descriptors of my own using inspiration from the top performers from the GA. After analysing these top performers I tried to derive a set of criteria needed to create a generalised descriptor that was able to get higher accuracy than all methods across both data sets.

The first criteria was a focus around the central pixel. While both GA top performers performed similarly on KITTI, there was a significant increase in performance of the census seeded top performer on the Middlebury data set. I accredited this due to it still having the top row of sampling pairs from the census transform, since census transform was the top performer on Middlebury. I postulated its success to the fact that the focus to sampling pairs being focused around the central pixel allowed it to better capture objects in the foreground of the image (Figure 5.5). The second criteria was increasing lateral sampling pairs. As the fitness of the genetic algorithm population grew it became clear that top performers had an increased number of lateral sampling pairs. This can be seen in Figure 5.4d and 5.4e were these higher performing descriptors had more lateral sampling pairs than the lower performing random descriptors Figure 5.4c and 5.4f. The third criteria was increasing sampling pair distances. I postulated that census transform (Figure 5.4a) did not do as well as the census spread transform (Figure 5.4b) because comparisons were too close together and as a result noise interference caused more inaccuracies if sampling pair positions where close together. It is also clear that in all the random based descriptors

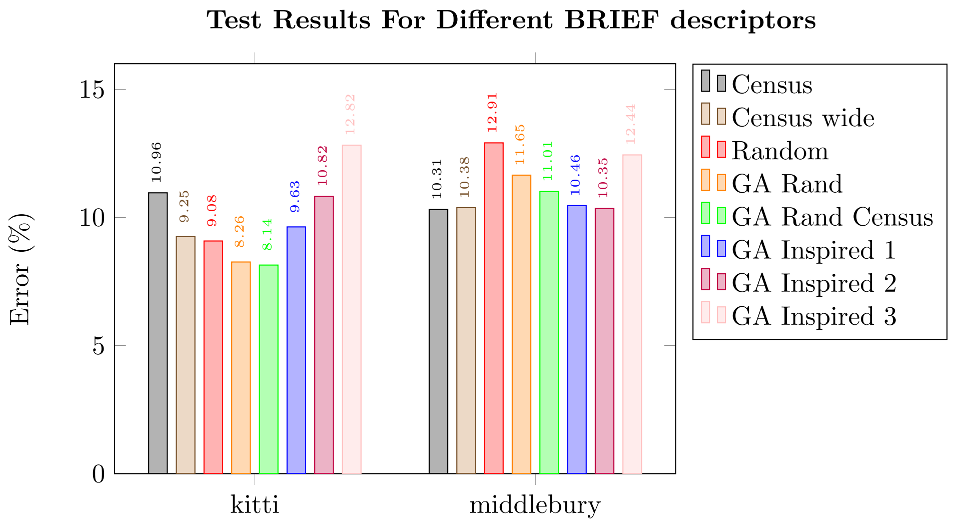
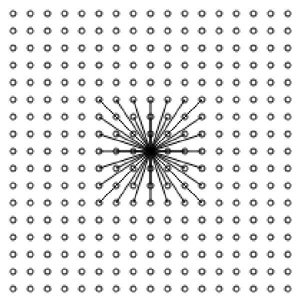


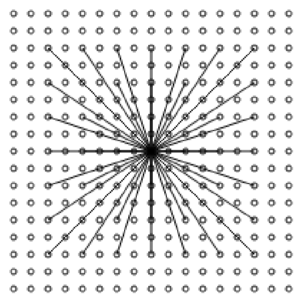
Figure 5.3: For KITTI error % is represented by mean error using a 3-pixel threshold across 100 test scenes taken from the 2015 KITTI training data. The "All" ground truth data was used for evaluation. For Middlebury error % is represented by mean error using a 2-pixel threshold across scenes from the 2003 and 2001 data sets.

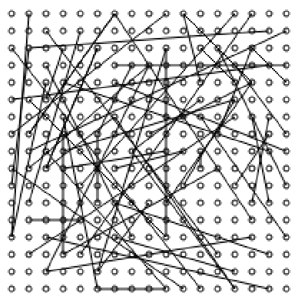
the sampling pair positions are mostly >4 pixels apart.

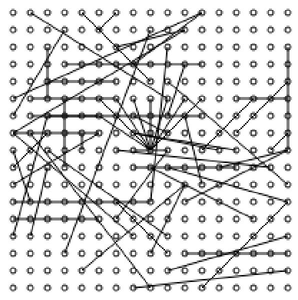
My first GA inspired descriptor (GA-I1, Figure 5.4g) had two sampling pairs per row each comparing against the pixel that lied on the central column of that row. Sampling pair positions were taken 5 pixels apart. With the extra sampling pairs available, I added a 5 × 5 census inspired square grid of sampling pairs around the central pixel to alleviate the inaccuracy on the Middlebury data set. My second GA inspired descriptor (GA-I2, Figure 5.4h) was to be similar to a cross hair. Multiple sampling pairs are taken on the central row and column against the central pixel. I reduced the vertical sampling pairs to allow for more lateral sampling pairs near the central pixel. My third GA inspired descriptor (GA-I3, Figure 5.4i) was taken by plotting lateral sampling pairs of length 4 in a random Gaussian distribution around the central pixel to archive a level of randomness while still accounting for my criteria.

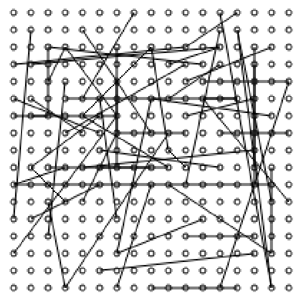
Results for these descriptors can be seen in Figure 5.3, Table 5.3 and Table 5.2. Accuracy of these GA inspired descriptors was worse than I anticipated. They all performed worse than random descriptors for 3-pixel and 4-pixel threshold error values on KITTI. GA-I1 was did fractionally better than random on the 2-pixel threshold. On Middlebury the performance was reasonable. GA-I1 and GA-I2 rivalled the census transform although GA-I3 performance was second worse to the random descriptor. the sub par results of hand engineered descriptors could be why better descriptors for computing matching costs have not been found. The best descriptors seem to have a counter intuitive nature to

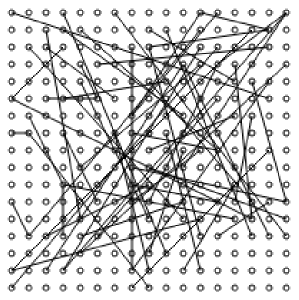

(a) Census transform[17]

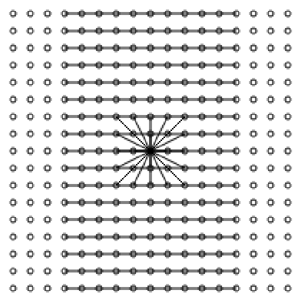

(b) Wide census transform

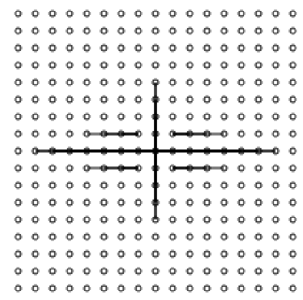

(c) Random


(d) GA optimised


(e) GA optimised


(f) Random


(g) GA inspired


(h) GA inspired

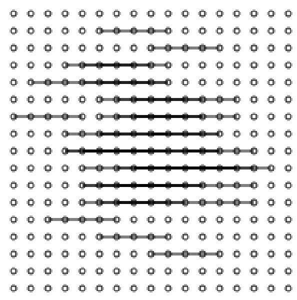
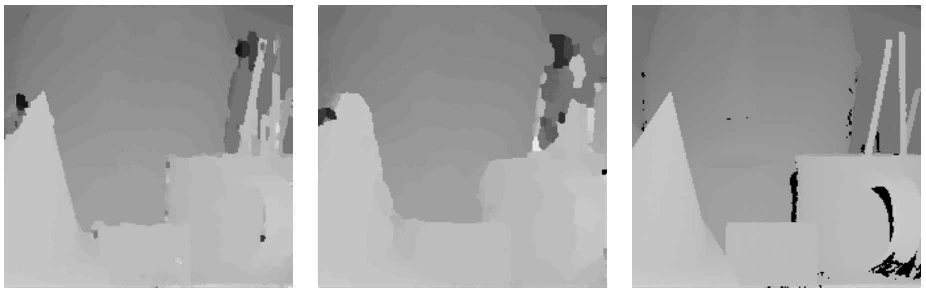

(i) GA inspired

Figure 5.4: Examples of BRIEF-style descriptors. Lines represent sampling pairs. Sampling pairs for both 5.4a and 5.4b are defined on a square grid around a central pixel. Sampling pairs in 5.4c and 5.4f are uniformly randomly sampled. Sampling pairs in 5.4d and 5.4e are the highest performing descriptors having been learnt from a genetic algorithm optimisation on random initial populations. The initial population for 5.4d was seeded with one instance of both census transforms 5.4a, 5.4b. Remains of the census transform's sampling pairs can be seen at the central pixel in 5.4d. Sampling pairs in 5.4g,5.4h,5.4i where build to incorporate aspects from the top performing GA optimised descriptors such as increased lateral comparisons. Sampling pairs are difficult to see when overlapping occurs. This is particularly significant in 5.4h and 5.4i



(a) Census (Figure 5.4a) (b) Random (Figure 5.4c) (c) GT

Figure 5.5: Examples where not comparing against central pixels has its downside. Comparison is on the ”cones” scene from the 2003 Middlebury data set. Fine scene details in the foreground are missing or highly inaccurate (paintbrushes, top right) with random based binary descriptors (Figure 5.5b) and edges are jagged (edge of the cone on the far left). This is because a higher number of comparisons take place with pixels in the background, rather than comparisons to a central pixel (i.e. census). Only the bottom left section of the disparity images have been shown to bring emphasise to the issue

them. Such as the top GA performing descriptors having sampling pairs push right up against the edges of the 17×17 descriptor window.

The estimated disparity images and their errors can be visualised for both data sets. Scenes from both 2003 and 2001 Middlebury data set can be seen in Figure 5.7. A scene from the 2015 KITTI data set can be seen in Figure 5.8.

5.3 Ablation Study

An ablation study in this project (Table 5.1) shows the effect of removing processing steps from the Stereo Matching pipeline, outlined in Chapter 3, and recording how it effects accuracy and performance. Experiments were performed on several Nvidia GTX 1080 Ti GPUs, Nvidia TITAN X GPUs and an eight core Intel i7 4GHz CPU. The time represents mean run time over the 100 executions and only measures the run-time of the pipeline stages and not any time dedicated to reading and writing images. As when processing the KITTI images we observe that census and census wide transform perform worse than random descriptors on all levels. Additionally, random feature extractors after just matching costs perform roughly as well as census transform after matching costs and cost aggregation. It is also interesting to see that the TGV[8] disparity estimation is able to deal with cost volumes with many inaccuracies since census with just MC+TGV performs far better than census with MC+CA. Although for the random filters MC with TGV performs roughly the same as MC+CA. Performance wise MC+CA have far less impact than TGV. Adding TGV into the pipeline creates a >10x slow down. The descriptors learnt using genetic

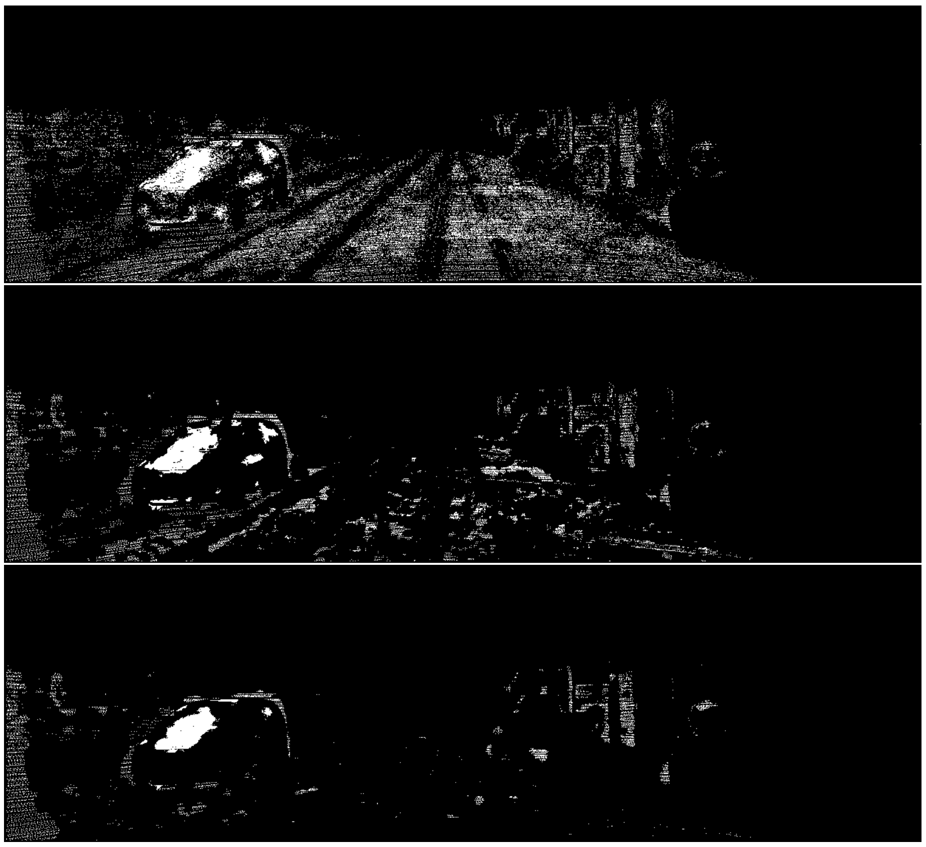


Figure 5.6: 3-pixel threshold errors marked in white on a scene from the 2015 KITTI data set: (top) MC, (middle) MC+AC, (bottom) MC+AC+TGV

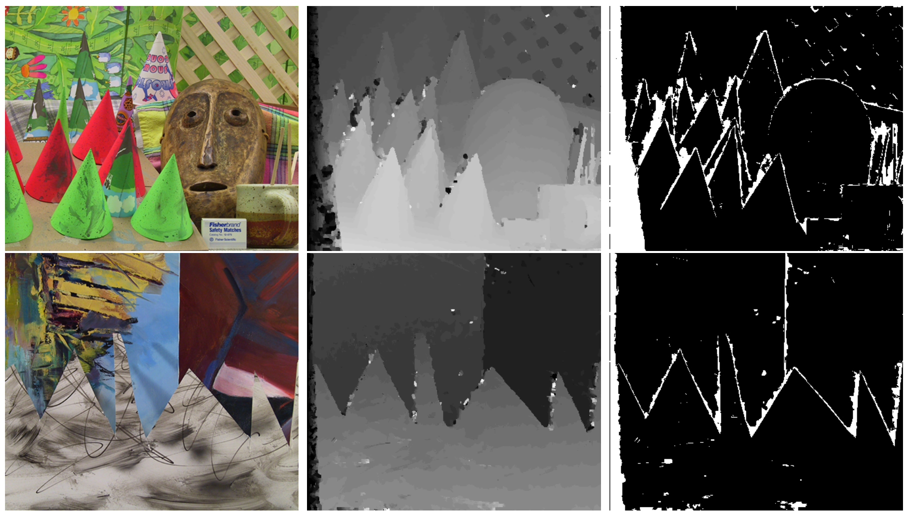


Figure 5.7: Results using census transform and all stereo matching pipeline stages on (top) cones from 2003 Middlebury dataset and (bottom) sawtooth from 2001 Middlebury dataset. (left) reference image, (middle) disparity estimate, (right) 2-pixel threshold errors marked in white.

algorithms are significantly more accurate than all other descriptors. At each stage the range of error between descriptors is reduced. When all pipeline stages are used GA optimised descriptors reduce error rate by $> 10\%$ from the next best performing descriptor. Most interesting is that GA optimised descriptors error rate is over a 50% reduction of census' on MC alone. Figure 5.6 shows how the estimated disparity image error increases as each step is removed.

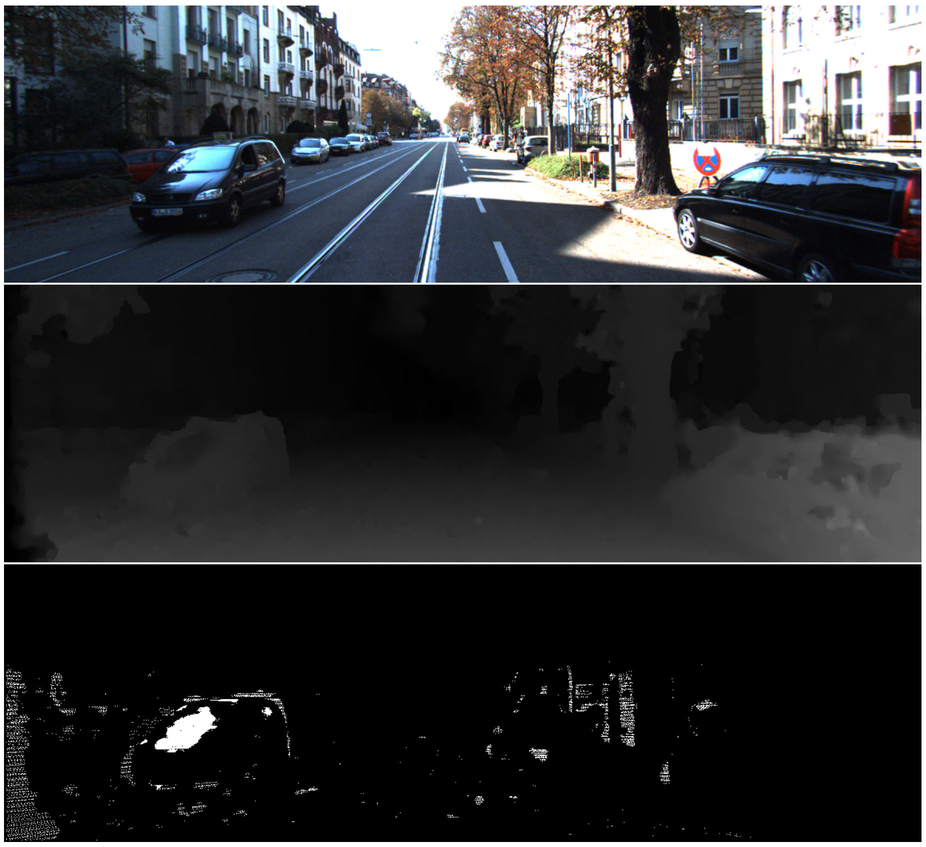


Figure 5.8: Results using census seeded GA top performer (Figure 5.4d) on a scene from the 2015 KITTI data set. (top) reference image, (middle) disparity estimate, (bottom) 3-pixel threshold errors marked in white.

Chapter 6

Conclusion

Dear past, thank you for your lessons. Dear future, I am ready!

Unknown

The results presented in Chapter 5 show that GA optimised BRIEF-style binary descriptors significantly outperform census transform on the 2015 KITTI data set. While random based descriptors where not found to be more accuracy than census transform on Middlebury data set GA optimised descriptors did outperform random descriptors proving that genetic algorithms are an effective tool to learn binary descriptors for stereo matching. With this, stereo matching algorithms that compute matching costs can gain a noticeable increase in accuracy at no impact to performance. This brings us a step closer to practical stereo vision on embedded devices. The primary contributions of this work are as follows:

1. Genetic algorithms offer an effective way to learn binary descriptors for use in stereo matching. Top performing binary descriptors where able to outperform census transform on 3-pixel threshold error by 2.82% on the 2015 KITTI data set.
2. Random binary descriptors outperform census transform on the 2015 KITTI data set.
3. Open source efficient CUDA implementation, Pseudo-code and an ablation study was provided for total generalised variation regularisation[8].

6.1 Possible future work

There are a multitude of ways in which this work can be further improved upon. This section describes several possible directions that could be taken:

1. The hyper-parameter selection of the genetic algorithm could be improved by performing a grid search using a validation set. Additionally, alternative meta-heuristic algorithms such as simulated annealing or guided local search could be explored as improved methods over GAs for discovering descriptors;

2. The pipeline was originally designed to be "Embarrassingly parallel" and use good memory access patterns to make for efficient embedded device implementations, such as low power GPUs or FPGAs. Investigating these implementation on such devices would be interesting as this would enable small autonomous devices such as drones to gain more information about their surroundings. This is currently not possible due to the large memory and computational power required to perform stereo matching at a practically accurate level;

6.1.1 Highly Condensed Partial Cost Volumes for Lazy Stereo Estimation

Lazy evaluation is an evaluation strategy which holds the evaluation of an expression until its value is needed. This evaluation strategy could be a solution to enabling implementations of stereo on low power embedded devices.

From the result it is clear that binary descriptors where sampling pair positions are on the same horizontal line perform better than conventional methods. By taking sampling pairs where sampling positions are at a fixed distance and direction the resulting cost volume can be highly condensed. Consider the following example.

Matching costs are computed with a binary descriptor B_1 that has n sampling pairs, where sampling pair positions are at a fixed distance d and fixed direction r. Traditionally computing the cost volume would start by transforming both images. These transformed images required $2 \times W \times H \times n$ bits to represent. Then costs would be computed from the transformed images with an XOR and bit counting step. This would yield a cost volume with a memory requirement of $W \times H \times D \times ln(n)$ bits. This cost volume memory requirement grows cubicly and practical values for these parameters eliminated the possibility of embedded device implementations.

An approach to further investigate that would achieve the previous result with significantly less memory overhead would be to use a binary descriptor B_2 with a single sampling pair. The first position in the pair is the central pixel, the other is at a fixed distance d from the central pixel in direction r. The images would then be transformed yielding a memory requirement of $2 \times W \times H$. Then only the XOR operations is performed to calculate a partial cost value that requires a single bit to represent. The resulting partial cost volume requires $W \times H \times D$ bits to represent. A resulting cost equivalent to one given in the previous example can now be lazily evaluated by performing a bit counting operation from surrounding pixels. The worst case trade off is having to perform n more memory operations although the memory overhead of the cost volume is reduced by a factor of $ln(n)$. E.g. if there where 64 sampling pairs this would result in a memory reduction by a factor of 6 but require a maximum of 63 additional memory fetches. Since bit counting operations are hardware accelerated on many architectures the computational overhead would be minimal.

The proposed method's construction of the partial cost volumes could be investigated to reduce the additional memory fetches required allowing the possibility of stereo matching applications on devices with highly constrained memory requirements such as small drones.

Bibliography

[1] Kristian Bredies, Karl Kunisch, and Thomas Pock. “Total generalized variation”. In: *SIAM Journal on Imaging Sciences* 3.3 (2010), pp. 492–526.

[2] Thomas Brox et al. “High accuracy optical flow estimation based on a theory for warping”. In: *European conference on computer vision*. Springer. 2004, pp. 25–36.

[3] Michael Calonder et al. “Brief: Binary robust independent elementary features”. In: *European conference on computer vision*. Springer. 2010, pp. 778–792.

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for autonomous driving? the kitti vision benchmark suite”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 3354–3361.

[5] Heiko Hirschmuller and Daniel Scharstein. “Evaluation of stereo matching costs on images with radiometric differences”. In: *IEEE transactions on pattern analysis and machine intelligence* 31.9 (2009), pp. 1582–1599.

[6] *Intel SSE4 Programming Reference*. English. Intel Corporation. URL: `https://software.intel.com/sites/default/files/m/0/3/c/d/4/18187-d9156103.pdf`.

[7] Steve Jurvetson. *Velodyne High-Def LIDAR*. 2009. URL: `https://www.flickr.com/photos/44124348109@N01/4042817787`.

[8] Georg Kuschk and Daniel Cremers. “Fast and accurate large-scale stereo reconstruction using variational methods”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2013, pp. 700–707.

[9] David G Lowe. “Object recognition from local scale-invariant features”. In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[10] Rene Ranftl et al. “Pushing the limits of stereo using variational stereo estimation”. In: *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE. 2012, pp. 401–407.

[11] *RealView Compilation Tools*. English. Version 4.0. ARM. 2010. URL: http://infocenter.arm.com/help/topic/com.arm.doc.dui0203j/DUI0203J_rvct_developer_guide.pdf.

[12] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE. 2011, pp. 2564–2571.

[13] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[14] Daniel Scharstein and Richard Szeliski. "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms". In: *International journal of computer vision* 47.1-3 (2002), pp. 7–42.

[15] Daniel Scharstein and Richard Szeliski. "High-accuracy stereo depth maps using structured light". In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2003, pp. I–I.

[16] Kuk-Jin Yoon and In So Kweon. "Adaptive support-weight approach for correspondence search". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.4 (2006), pp. 650–656.

[17] Ramin Zabih and John Woodfill. "Non-parametric local transforms for computing visual correspondence". In: *European conference on computer vision*. Springer. 1994, pp. 151–158.