# LS-ELAS: Line Segment based Efficient Large Scale Stereo Matching

Radouane Ait-Jellal, Manuel Lange, Benjamin Wassermann, Andreas Schilling and Andreas Zell

***Abstract*— We present LS-ELAS, a line segment extension to the ELAS algorithm, which increases the performance and robustness. LS-ELAS is a binocular dense stereo matching algorithm, which computes the disparities in constant time for most of the pixels in the image and in linear time for a small subset of the pixels (support points). Our approach is based on line segments to determine the support points instead of uniformly selecting them over the image range. This way we find very informative support points which preserve the depth discontinuity. The prior of our Bayesian stereo matching method is based on a set of line segments and a set of support points. Both sets are plugged into a constrained Delaunay triangulation to generate a triangulation mesh which is aware of possible depth discontinuities. We further increased the accuracy by using an adaptive method to sample candidate points along edge segments.**

## I. INTRODUCTION

In this paper we address the problem of estimating disparity maps at high frame rates and using low computational resources. Depth estimation is an important task in mobile robotics. It is essential for many high level and medium level tasks such as 3D reconstruction, obstacle avoidance, navigation, recognition and object grasping. The introduction of RGBD sensors[1] with their sufficiently accurate depth map and their real time capability has accelerated the research on indoor mobile robotics. Since these Kinect-style RGBD sensors are based on the projection and capture of structured infra-red light, they are not designed for outdoor usage with substantial sunlight. There is a need for more adequate sensors. Stereo cameras are the most adequate depth estimation sensors to be used for a wide range of outdoor and indoor applications. They provide data at a high frame rates, they are lightweight, passive, energy efficient and customizable. This makes the hardware side of a stereo system very convenient. The software side is still problematic because of the trade-off between accuracy and efficiency. In this work we propose an efficient and accurate dense stereo algorithm which can facilitate the migration of systems initially designed to work in indoor environments using Kinect-style RGBD sensors, to operate outdoors. The need for high resolution depth maps involves considering large disparity ranges. In this case, even stereo algorithms with linear time $\mathcal{O}(d)$ complexity (such as block matching) become inefficient. Here we drop the number of the pixels $(width \times height)$ in the complexity and we analyse the complexity only with respect to the disparity $d$. We present four contributions in this work: (1) An efficient method for edge extraction, which provides the connectivity of the edges components as well. (2) A new method based on edge features for computing the support matches. This method is selective and does find support points which are more informative and allows to preserve the depth discontinuity (See Fig. 1). It is more efficient as well since the candidate support points that we use are more likely to have robust correspondences. (3) A new prior for the Bayesian inference approach proposed by ELAS. Our prior better represents the probability distribution of the disparity given the set of support points, the set of line segments and the observations. (4) An adaptive method for sampling support points along the edge segments.

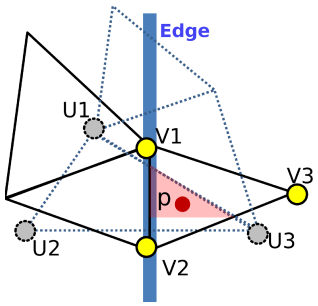



Fig. 1. Illustration of the difference between LS-ELAS (black triangles) and ELAS (dashed triangles) triangle meshes.

## II. RELATED WORK

Dense stereo matching algorithms can be divided into two categories [1]: local stereo matching and global stereo matching. This classification is based on the way the disparity optimization is done. Since we are interested in real-time algorithms, we focus on the classification based on the computational time complexity. Most local stereo algorithms have a linear time with respect to the disparity $\mathcal{O}(d)$ and most global stereo algorithms have a quadratic time with respect to the disparity $\mathcal{O}(d^2)$. Algorithms with higher order complexities are usually impractical for robotics applications. Algorithms, which smartly restrict the search domain to a small interval, such as [2] [3] [4], are becoming more popular and more, because these algorithms deal efficiently with large scale images. While [2] involves a global optimization stage, ELAS [4] is a near constant time local stereo matching algorithm. ELAS starts by finding robust matches for some pixels called support points and generates from them a 2D

R. Ait-Jellal is with the Chair of Cognitive Systems,headed by Prof. A. Zell, M. Lange and B. Wassermann are with the Chair of Visual Computing, headed by Prof. A. Schilling, Computer Science Department, University of Tübingen, Sand 13, D-72076 Tübingen, Germany {radouane.ait-jellal, manuel.lange, benjamin.wassermann, andreas.zell, schilling@uni-tuebingen.de}

[1]e.g. Microsoft Kinect, Asus Xtion Pro or Intel RealSense

mesh by triangulation. The matching of these triangles corner is done in linear time $\mathcal{O}(d)$. Afterwards the algorithm uses Bayesian inference with the assumption that the disparity of a pixel is independent of all other pixels on the reference image given the disparities of the triangle corners it belongs to. For the pixels inside the triangles the matching is done in constant time $\mathcal{O}(1)$.

In local stereo matching [5] [6] [7] [8], the disparities are computed for each pixel individually, with the cost aggregated over a local correlation window. The choice of the size and the shape of the correlation window is both crucial and challenging. A correlation window with a large size can lead to edge blurring at the depth discontinuities. A correlation window with a small size, on the other hand, can significantly decrease the signal to noise ratio. This problem increases the ambiguity of the candidate disparities and as a result it increases the amount of outliers. A customized shape and size of the correlation window [7] for each pixel individually comes at the cost of increasing the computational time. In this case it is not surprising that some adaptive window local stereo matching algorithms are even slower than some efficient global stereo matching algorithms. [3] proposed slanted support windows. They compute 3D planes at the level of pixels and projects the support region onto the local 3D plane. In global stereo matching [9] [10] [11] [12], the whole pixels of the image contribute to the computation of the disparity of a given pixel. In addition to the data term the energy function does include a smoothness term which penalizes the disparities which do not agree with the disparities of the neighbors. The complexity of these algorithms is usually $\mathcal{O}(d^c)$, where $d$ is the disparity range and $c$ is the size of the maximum clique potential. $c$ is usually set to $c = 2$ for speed-up the global stereo algorithms. Nevertheless, global stereo algorithm remain very slow for mobile robotics applications. An efficient global stereo algorithm is proposed by Felzenszwalb et al. [13]. They come up with a method for updating the messages in linear time $\mathcal{O}(d)$ for the three models of smoothness term: the potts model, the linear model and the quadratic model. However, the huge memory requirement needed for storing the messages and the fact that there is no guarantee for convergence for loopy graphs are two major drawbacks of this method. Another efficient algorithm is the semi-global stereo matching (SGM) firstly proposed by Hirschmüller et al. [14]. In SGM the costs are first computed using local stereo matching in linear time $\mathcal{O}(d)$. Then, a smoothness step is applied for updating the costs by penalizing the disparities that disagree with those of the neighbours. It considers interactions that are potentially very long range along image rows, columns and diagonals to minimize a cost function. The semi-global characteristic is due to the fact that only a small subset of possible interaction paths is considered.

## III. Line Segment based Efficient Large Scale Stereo Matching

In this section we describe our algorithm in detail. We start by describing our method for extracting the edge segments and the way we compute the set of support points and the set of line segments. Then, we show the mathematical derivation of our Bayesian approach for the stereo problem.

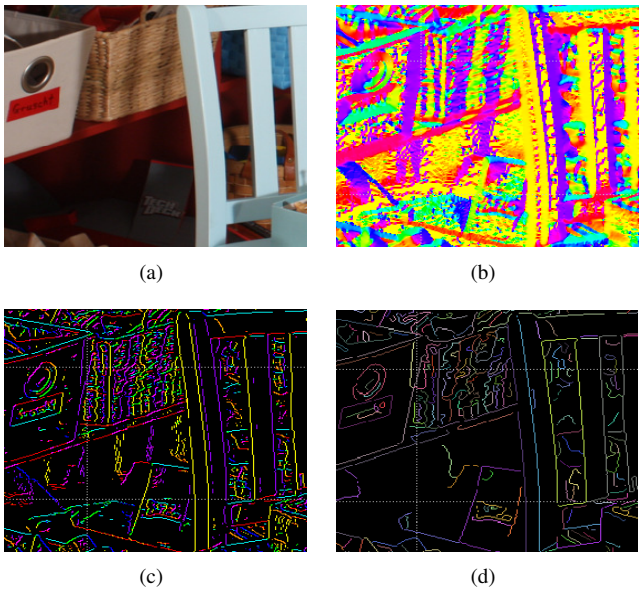


Fig. 2. Intermediate steps of our edge segments extraction method. (b) shows the dirMap (gradient directions). (c) shows the edgeMap. The color encodes the gradient direction (same as in dirMap). (d) shows the final edge segments list. Each edge segment is drawn by a random color individually.

| | | |
|---|---|---|
| ↖ 1 | ↑ 2 | 3 ↗ |
| ← 0 | x | 4 → |
| ↙ 7 | ↓ 6 | 5 ↘ |

| | | |
|---|---|---|
| 5 | 6 | 7 |
| 4 | x | 0 |
| 3 | 2 | 1 |

Fig. 3. The left table contains the direction labels for the 8 adjacent pixels around the pixel x. The table on the right contains the inverse direction labels that are used for an inverse search.

### A. Fast Edge Segment Detection

In ELAS [4] the support point candidates are sampled uniformly over the image (See Fig. 5(e)). This method is motivated by the need for an efficient way to compute the support points. In this paper we present a new method for extracting support point candidates based on edge features. Our approach is motivated by our efficient edge extraction method. We show that our method for computing the support matches is both, robust and efficient. Furthermore, it allows us to use the line segments between two consecutive support points to generate a better triangulation (see Sec. III-C.2).

**Algorithm 1:** Main loop for extracting edge segments.

```
Data: seedList, edgeMap, dirMap, dir2Index
Result: edgeSegmentList
foreach seed in seedList do
    Check if seed is valid?
    segForward = extractSeg(seed);
    reverse(dir2Index);
    segReverse = reverse(extractSeg(seed));
    edgeSegmentList.push(segReverse + segForward);
```

**Algorithm 2:** adjacentIndex: Find adjacent edge pixel.

**Input**: index, direction
**Data**: edgeMap, dir2Index
**Result**: adjIndex
**for** *each adjIndex pixel* **do**
  **if** *edgeMap[adjIndex] > 0* **then**
    // Invalidates the pixel from *seedList*
    edgeMap[adjIndex] = -1;
    **return** *adjIndex*;
**return** *-1*;

**Algorithm 3:** extractSeg: Extract single edge segment.

**Input**: index
**Data**: edgeMap, dirMap
**Result**: edgeSegment
**while** *index >= 0* **do**
  edgeSegment.push(index) ;
  index = adjacentIndex(index, dirMap[index]) ;
**return** *edgeSegment*;

Our fast edge segment extraction method is similar to the Canny edge detector approach [15]:

1) The input image is smoothed by a Gaussian filter to reduce noise.
2) The intensity gradients of the image are computed.
3) Non-maximum suppression is applied for thinning the spread edge responses to the maximum responses.
4) Two thresholds are applied to determine potential strong and weak edge pixels.
5) The edges are tracked by hysteresis to suppress weak edges that are not connected to strong edges.

The result of the classical Canny method only provides an edge pixel map without the connected edge components. We want to have a data structure which holds all pixels which belong to the same edge segment (See Fig. 2(d)). We need to split the edge map to individually determine the edge segments. So, we extended the Canny method as follows:

1) Within the third step of Canny the gradient orientation for each pixel has to be known. Fast algorithms compute a rough approximation for the 8 possible neighbouring pixels. In our method we store this information in a map for later use.
2) In our approach all edge responses over the lower threshold are marked in an edge map (see Fig. 2(c)). The indices of the edge responses over the higher threshold are stored in a seed list.
3) The hysteresis step is replaced by the connected edge components search (See Fig. 3).

Starting from a seed point and its initial direction (Canny modification 2 and 1), a connected edge component is extracted by traversing the adjacent edge points (see Algorithm 3). Instead of testing all 8 adjacent pixels, we use the stored direction to predict the next adjacent edge point as shown in Algorithm 2. The seed point could be the start point or the end point of a segment. For an end point the search direction has to be reversed. It is also possible that the seed is a point in between the segment, which requires a reverse and a forward search. The ordered connected edge points are stored in an edge segment list. For more details see Algorithm 1.

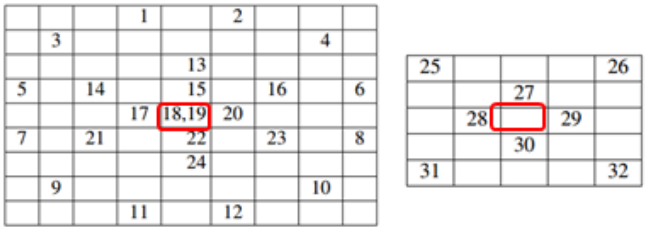
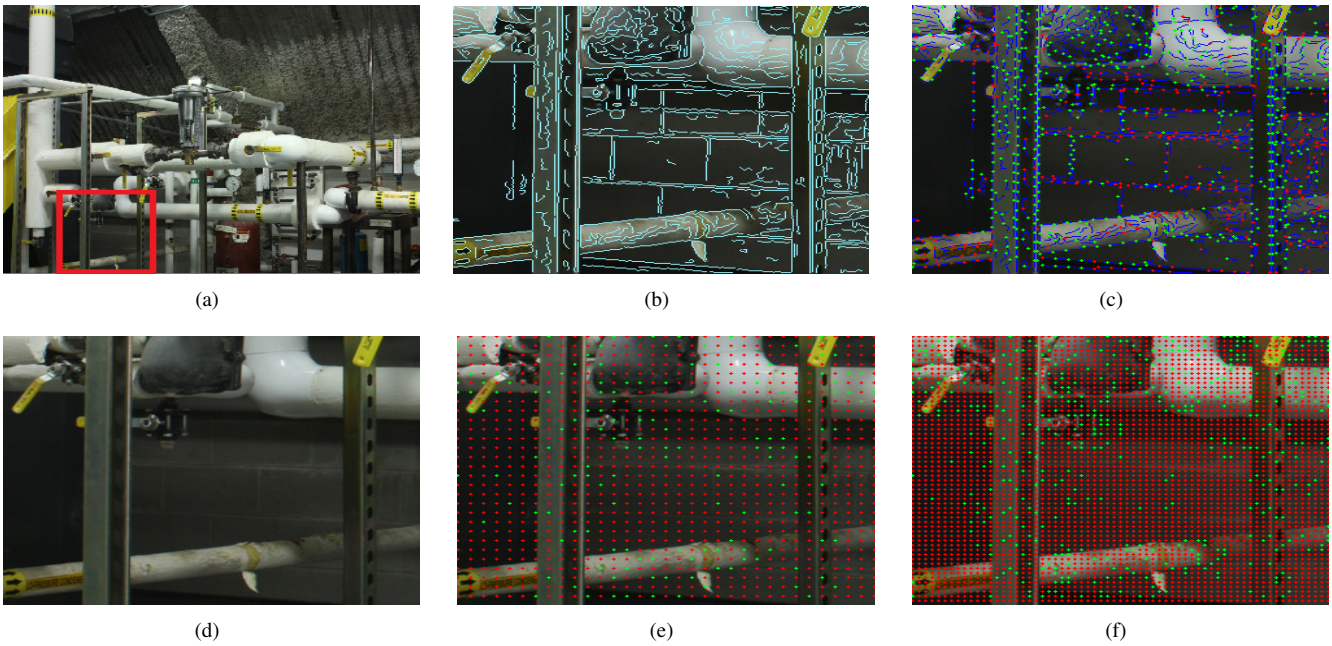



Fig. 4. Locations of the 32 elements of the feature vector on the X-Sobel (left) and Y-Sobel (right) gradient images. The candidate support point is at the center. The Sobel responses are normalized and shifted to have the size of 8 Bits and values between 0 and 255.

### B. Fast Support Points Matching

The process of computing the support points is illustrated in Fig. 5. We start by sampling candidate support points along the edge segments. A candidate support point includes the pixel coordinates $(u, v)$ and the disparity $d$ ($d$ is initially set as invalid). The set of support points is $S$. We set a feature vector for each candidate support point based on the gradients in X and Y direction around it. Our feature descriptor is shown in Fig. 4. To match the support points candidate we use the sum of absolute differences to compute the cost. We perform a left-right consistency check to filter the outliers. Since the feature vector is 32 elements of 8 bits each, the 256 bits feature vector does fit in the AVX2 (Advanced Vector Extensions 2 SIMD instructions) CPU registers and the computation is accelerated using the AVX2 intrinsics. Table IV shows a comparison of the percentage of successfully matched support points. The AVX2 is used only for support points matching. For pixels inside triangles we use a 128 Bits feature vector and SSE2 SIMD instructions as in ELAS.

We present two different methods for sampling candidate support points. In our first method, we uniformly sample candidate support points along the edges. A constant step separates two consecutive candidate support points. This constant step is calculated from the image diagonal length. We present also a more elegant method for sampling the support point candidates. This sampling method is an adaptive method. Here we sample more points on curved parts of edges than on straight parts. We use an iterative process where we walk along the edge and consider the straight line between the last recently sampled support point candidate (start) and the current point (end). When the "curvature" (the projection distance, of any of the points from the currently checked part, onto the straight line which lies between the current start and the current point) is over a certain threshold, then a new support point candidate is set. This point is the new start for the next steps of the walk. In case the curvature



(a) (b) (c)

(d) (e) (f)

Fig. 5. (a) shows the left image of the Middleburry 2014 Pipes dataset from which we select the region defined by the red rectangle. (d) shows this area. (b) shows the edges (in light blue) detected with our method. (c) (e) (f) shows the support point candidates $c_{ij}$ ($red$ and $green$) and the valid support point $s_{ij}$ (green). (c) shows the set $S$ we get using our algorithm. (e) (f) using ELAS method with a step of 10 and 5 pixels respectively.

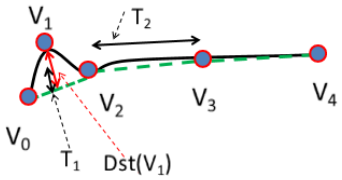



Fig. 6. Illustration of the adaptive sampling. The points $V_i$ are sampled along the (black) edge segment. We sample a candidate support point when the distance exceeds the threshold $T_1$ for curved edge regions or the threshold $T_2$ for relatively flat regions.

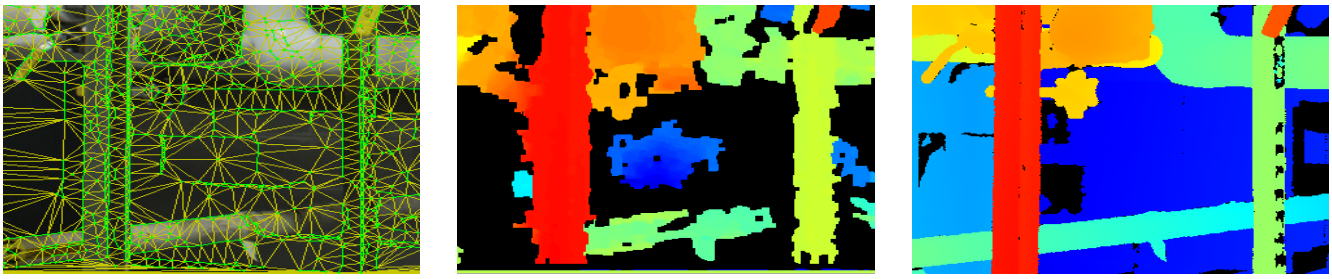
is low, we also set a new support point candidate with a constant step. This process is illustrated in Fig. 6.

### C. Bayesian inference of the disparity map

*1) Notation:* We use the following notation in this work:

- $d_n$: Disparity of the pixel $n$.
- $E = \{e_j\}$: List of edge segments.
- $C = \{c_{ij}\}$: List of candidate support points sampled along the edges. $c_{ij}$ candidate support point $i$ sampled on edge segment $e_j$.
- $S = \{s_{ij}\}$: List of support points. These are elements of $C$ which are successfully matched on the target image.
- $s_{ij} = (u_{ij}, v_{ij}, d_n)$: Include the pixel coordinates $(u_{ij}, v_{ij})$ and its disparity $d_n$.
- $L = \{l_{i_1 j - i_2 j}\}$: List of straight line segments between two consecutive support points $s_{i_1 j}$ and $s_{i_2 j}$ which belong to the same edge $e_j$.
- $O = \{o_n\}$: Image observations. $o_n = (u_n, v_n, f_n)$. Observations on the left image (reference image) are denoted by $o_n^{(l)}$ and $o_n^{(r)}$ for the right image.
- $f_n$: 16-dimensional feature vector around the pixel $n$ computed base on the image gradients. It is used to compute the likelihood. It is similar to the one in Fig. 4 but has smaller size. It has 128 Bits (16x8) instead of 256 Bits (32x8). SSE2 SIMD instructions are used to make parallel operations (addition, subtraction, etc) on these feature vectors.
- $\sigma, \gamma, \beta$: Constant parameters.
- $N_s = \bigcup_{i=1,2,3} \{d_{pi} - 1, d_{pi}, d_{pi} + 1\}$: Where $p1$,$p2$ and $p3$ are the vertices of the triangle to which the pixel belongs to. $d_{pi}$ are the disparities of the triangle vertices.
- $\mu(S, L, o_n^{(l)})$: The mean disparity (the prior). We get it by interpolating the disparities of the triangle corners.

*2) Prior based on constrained Delaunay triangulation:* We extend the Bayesian approach introduced by Geiger et al. in ELAS [4] by including a list $L = \{l_{i_1 j - i_2 j}\}$ of straight line segments in the probability model. $l_{i_1 j - i_2 j}$ approximate curved edges between two consecutive (matched) support points $s_{i_1 j}$ and $s_{i_2 j}$. The set of (matched) support points $S$ and the set of line segments $L$ are plugged into a constrained Delaunay triangulation algorithm to compute a mesh of triangles. In other words the input to our triangulation is the set of all support points that have valid matches (this includes the isolated edge support points whose predecessors and successors on the edge have been deleted) and the set of line segments $L$. The constrained Delaunay triangulation is a triangulation, which is forced to include the input edges (our line segments list $L$). This way we nicely preserve the object boundaries. Fig. 7(a) shows the 2D mesh created by our method LS-ELAS. The line segments are represented with the green color. Equation 1 shows the mathematical



(a) (b) (c)

Fig. 7. (a) shows the 2D mesh generated using our method for the example in Fig. 5. The discontinuity edges are nicely preserved. (b) shows the disparity map estimated using our algorithm (LS-ELAS). No hole filling post processing is done and most black regions corresponds to occluded regions (and they should be black). We refer to the occlusion map for the Pipes dataset which is available at the website of the Middlebury stereo benchmark.

formulation of our prior:

$$p(d_n \mid S, L, o_n^{(l)}) = \begin{cases} \gamma + \exp\left(-\frac{(d_n - \mu(S,L,o_n^{(l)}))^2}{2\sigma^2}\right), \\ \text{if } |d_n - \mu| < 3\sigma \vee d_n \in N_S \\ \\ 0, \text{otherwise} \end{cases} \tag{1}$$

It includes a Gaussian part and a uniform part. The mean $\mu(S, L, o_n^{(l)}))$ in the Gaussian part depends on the set of line segments $L$. Using the constrained Delanay triangulation to generate the mesh might result in a mesh which does violate the Delaunay property locally for some triangles. This is a result of enforcing the constrained Delaunay triangulation to include the set $L$ of line segments. Remember that the Delaunay property means that no point is contained inside the circumcircle of any triangle in the mesh. This means avoiding skinny triangles with large height to base ratio. We prefer to have a mesh which preserves the discontinuity rather than a mesh which conforms to Delaunay. The search domain is restricted to a small set (radius of $3\sigma$ pixels) of candidates disparities around the mean $\mu(S, L, o_n^{(l)}))$. For example we can restrict the search domain to 7 disparities $(\sigma = 1)$ instead of a full range of 800 disparities. To account for depth discontinuity, the search domain is extended to include the disparities of the corners $N_s$.

*3) Maximum A-Posteriori Estimation:* The Maximum A-Posteriori estimate of the depth map $d_n^{MAP}$ is then given by the Equation 2.

$$d_n^{MAP} = \arg\max p(d_n \mid o_n^{(l)}, o_1^{(r)}, ..., o_N^{(r)}, S, L) \tag{2}$$

It depends on the set of support points $S$, the set of line segments $L$ and the observations on the target image $o_1^{(r)}, ..., o_N^{(r)}$. Note that we consider only the observations along the epipolar line. We model the likelihood using a constrained Laplace distribution (See Equation 3). The Laplace distribution is robust against outliers. We can safely ignore the outliers on the data.

$$p(o_n^{(r)} \mid d_n, o_n^{(l)}) = \begin{cases} exp(-\beta \left\| f_n^{(l)} - f_n^{(r)} \right\|_1), \\ \text{if } \begin{pmatrix} u_n^{(l)} \\ v_n^{(l)} \end{pmatrix} = \begin{pmatrix} u_n^{(r)} + d_n \\ v_n^{(r)} \end{pmatrix} \\ \\ 0, \text{otherwise} \end{cases} \tag{3}$$

We see in this equation that for a given disparity candidate $d_n$ only one observation is non-zero. The posterior in Equation 2 can be simplified as shown in Equation 4.

$$\begin{aligned} d_n^{MAP} &= \arg\max p(d_n \mid S, L, o_n^{(l)}) p(o_1^{(r)}, ..., o_N^{(r)} \mid o_n^{(l)}, d_n) \\ &= \arg\max p(d_n \mid S, L, o_n^{(l)}) \sum_{i=1}^{N} p(o_i^{(r)} \mid o_n^{(l)}, d_n) \end{aligned} \tag{4}$$

Maximizing the posterior probability is equivalent to minimizing the negative logarithmic energy. To find the winning disparity, we minimize the energy given by Equation 5.

$$\begin{cases} E(d) = \beta \left\| f_n^{(l)} - f_n^{(r)}(d) \right\|_1 \\ \qquad - \log\left[\gamma + \exp\left(-\frac{(d_n - \mu(S,L,o_n^{(l)}))^2}{2\sigma^2}\right)\right], \\ \qquad \text{if } |d_n - \mu| < 3\sigma \vee d_n \in N_S \\ \\ +\infty, \text{otherwise} \end{cases} \tag{5}$$

where $f_n^{(r)}(d)$ is the feature vector at $(u^{(l)} - d, v^{(l)})$ on the right (target) image.

## IV. RESULTS

### A. Evaluation on the Middlebury Benchmark

We performed experiments on the Middlebury Stereo Evaluation Dataset - Version 3 from 2014 [16] to evaluate our adaptive LS-ELAS algorithm. We have submitted our results to this benchmark, they are listed as "LS-ELAS" for broader comparison with other algorithms[2]. Our results are particularly good in the sparse datasets, because we did

[2]http://vision.middlebury.edu/stereo/eval3/

TABLE I

PERFORMANCE EVALUATION ON THE MIDDLEBURY STEREO EVALUATION DATASET - VERSION 3, BAD 2.0, TEST SPARSE

| Algorithm | Avg | Bad 2.0 | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Aust | AustP | Bic2 | Clas | ClasE | Comp | Crus | CrusP | Djem | DjemL | Hoop | Livgr | Nkub | Plant | Stair |
| SGM (F) | 3.33 | 11.7 | 1.64 | 2.04 | 2.01 | 3.04 | 3.56 | 6.15 | 3.41 | 2.45 | 2.19 | 4.10 | 3.39 | 2.35 | 3.61 | 1.01 |
| SNCC (H) | 6.24 | 17.3 | 3.32 | 3.61 | 4.45 | 5.48 | 7.39 | 13.3 | 9.40 | 3.49 | 3.40 | 6.46 | 4.10 | 3.99 | 7.07 | 3.32 |
| **LS-ELAS** (F) | **8.82** | **14.9** | **4.43** | **7.94** | **4.89** | **2.92** | **7.20** | **10.9** | **6.50** | **5.27** | **4.97** | **10.5** | **9.78** | **8.55** | **20.3** | **15.7** |
| SGBM1 (F) | 9.44 | 24.0 | 5.55 | 5.63 | 7.17 | 8.85 | 7.65 | 14.7 | 9.97 | 3.82 | 10.9 | 16.0 | 9.93 | 8.25 | 9.55 | 11.9 |
| Cens5 (H) | 9.48 | 23.7 | 4.44 | 4.60 | 6.21 | 9.23 | 9.21 | 20.3 | 18.1 | 4.77 | 6.50 | 8.87 | 6.55 | 6.02 | 11.9 | 4.57 |
| ELAS (F) | 16.4 | 24.9 | 5.44 | 7.55 | 10.7 | 15.5 | 9.85 | 21.1 | 17.2 | 7.07 | 11.5 | 19.9 | 15.6 | 16.6 | 42.2 | 30.7 |
| LPS (F) | 20.3 | 6.72 | 6.06 | 9.72 | 9.87 | 94.3 | 14.1 | 11.2 | 11.2 | 5.88 | 89.3 | 36.0 | 20.5 | 23.8 | 16.0 | 25.4 |
| TSGO (F) | 39.1 | 34.1 | 16.9 | 20.0 | 43.3 | 55.4 | 14.3 | 54.1 | 49.2 | 33.9 | 66.2 | 45.9 | 39.8 | 42.6 | 47.2 | 52.6 |

TABLE II

PERFORMANCE EVALUATION ON THE MIDDLEBURY STEREO EVALUATION DATASET - VERSION 3, TIME IN SECONDS / MP, TEST SPARSE

| Algorithm | Avg | Time per Megapixel | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Aust | AustP | Bic2 | Clas | ClasE | Comp | Crus | CrusP | Djem | DjemL | Hoop | Livgr | Nkub | Plant | Stair |
| **LS-ELAS** (F) | **0.50** | **0.49** | **0.51** | **0.61** | **0.49** | **0.46** | **0.48** | **0.50** | **0.50** | **0.50** | **0.49** | **0.47** | **0.48** | **0.50** | **0.47** | **0.45** |
| ELAS (F) | 0.56 | 0.57 | 0.56 | 0.54 | 0.59 | 0.52 | 0.48 | 0.65 | 0.65 | 0.56 | 0.54 | 0.56 | 0.53 | 0.59 | 0.53 | 0.53 |
| SNCC (H) | 1.02 | 0.73 | 0.69 | 0.68 | 1.25 | 1.23 | 0.68 | 1.57 | 1.62 | 0.77 | 0.80 | 0.95 | 1.16 | 1.24 | 0.77 | 1.01 |
| Cens5 (H) | 1.35 | 1.03 | 1.01 | 0.85 | 1.83 | 1.84 | 0.82 | 2.21 | 2.20 | 1.00 | 1.02 | 1.24 | 1.02 | 1.64 | 1.01 | 1.42 |
| SGBM1 (F) | 3.69 | 2.44 | 2.45 | 2.31 | 4.67 | 4.61 | 1.84 | 6.27 | 6.28 | 3.01 | 2.95 | 3.56 | 2.99 | 4.80 | 2.97 | 3.51 |
| LPS (F) | 5.28 | 5.97 | 4.30 | 4.42 | 4.22 | 4.17 | 11.4 | 4.90 | 4.91 | 4.83 | 4.37 | 5.85 | 4.24 | 4.38 | 5.79 | 4.96 |
| TSGO (F) | 8.26 | 11.6 | 11.5 | 9.94 | 6.53 | 6.98 | 22.4 | 8.34 | 8.63 | 4.21 | 4.45 | 4.88 | 4.95 | 6.09 | 4.12 | 5.27 |
| SGM (F) | 13.4 | 10.0 | 9.32 | 8.20 | 16.1 | 18.4 | 8.03 | 23.0 | 22.6 | 9.83 | 10.4 | 12.7 | 10.2 | 16.8 | 10.4 | 13.9 |

TABLE III

PERFORMANCE EVALUATION ON THE MIDDLEBURY STEREO EVALUATION DATASET - VERSION 3, LS-ELAS VS ELAS

| Algorithm | LS-ELAS vs ELAS: Percentage of wrong matches averaged for all images. Set: test sparse, Mask: nonocc | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | bad 0.5 | bad 1.0 | bad 2.0 | bad 4.0 | avgerr | rms | A50 | A90 | A95 | A99 | time | time/MP | time/GD |
| **LS-ELAS** (F) | **30.9** | **15.8** | **8.82** | **5.97** | **7.36** | **22.8** | **0.74** | **19.7** | **32.0** | **98.0** | **2.63** | **0.50** | **1.33** |
| ELAS (F) | 45.2 | 26.4 | 16.4 | 11.5 | 8.53 | 24.6 | 1.55 | 20.1 | 34.5 | 102 | 3.00 | 0.56 | 1.45 |

not implement a good post-processing method for correcting invalid matches (hole filling). In the rest of this section, we focus on the sparse evaluation. Remember that in Middlebury benchmark, sparse disparity map means a "dense" disparity map for which the hole filling step is not performed. Most of the holes (filtered pixels) occur at occlusions, in that case the stereo matching is an ill-posed problem. The Middlebury Stereo Evaluation Dataset contains a variety of different kinds of scenes. This new Middlebury dataset is more challenging compared to the previous datasets. It includes images taken in industrial environments, such as the Pipes dataset and images taken in indoor environments. We attached a video showing intermediate results of our algorithm when applied on Middlebury dataset. Also the disparity range of up to 800 disparities for the full (F) size images and up to 400 for half (H) size images is higher than in previous datasets. The Resolution is up to $3000\times2000$ pixels on the full resolution images. We conducted all our experiments on a notebook computer equipped with 16 GB RAM and an Intel Core i7 4700MQ, 2.4 GHz 6 MB cache. All results are single core performance. We selected the most popular stereo algorithms in the Middlebury Stereo Evaluation Dataset to compare them to our method. The training and test datasets contain 15 images each. The average is calculated by weighting the images. The weighting differs and the weights are set by Middlebury. Table I shows an excerpt of the Middlebury Evaluation table for the test sparse dataset. We chose *test sparse* as set and *nonocc* as mask. A significantly lower error value of our method compared to ELAS is clearly visible. LS-ELAS is also more accurate than SGBM1 [17], Cens5 [18], LPS [2] and TSGO [19]. While Semi-Global Matching [14] yields a lower error value, it is a much slower method. The same applies to SNCC [20]. Table II shows a speed comparison of the algorithms. Our algorithm is the fastest.

### *B. Comparing LS-ELAS with ELAS*

In this subsection we compare the sub-part of the algorithm concerning support point matching, then we compare the whole algorithms for all different *metrics* of the Middlebury Benchmark. In Fig. 8 the time for the support point calculation and matching is drawn for the images of the training (H) dataset. As ELAS places the support points on a uniform grid ($5 \times 5$) over the image, calculating the position of the support point candidates is done immediately. We first have to detect the edge segments in the image, then sample our support point candidates along them. This detection takes some time, which is visualized as *Edge detection*. After having placed our support point candidates along the edge segments, their matches in the corresponding stereo image are calculated. That time is plotted as *Match candidates*. The sum of both our steps is drawn as LS-ELAS to compare

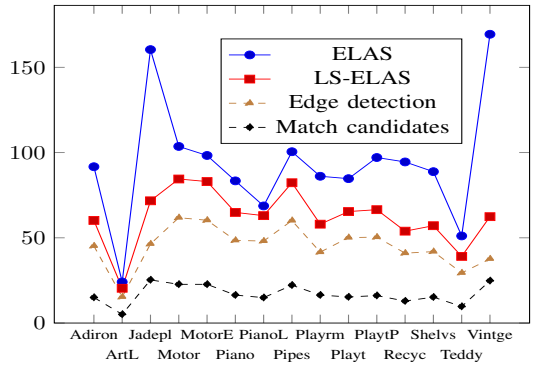



Fig. 8. X-axis: the different dataset images. Y-axis: the time in ms for computing the support matches. ELAS (blue) and LS-ELAS (red). For LS-ELAS the time is the sum of the edge detection (brown) and the support candidate matching (black). The results are reported for the training data set at half resolution. Our algorithm is even more efficient compared to ELAS on the full resolution. See Fig. 9.

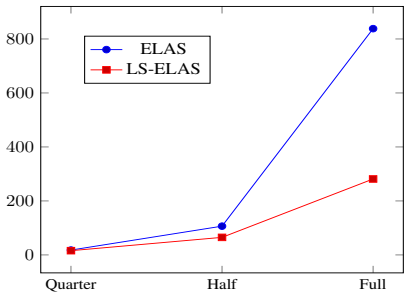



Fig. 9. Evolution of the average time (in ms) for computing the support matches with respect to the image resolution. Time averaged for 15 stereo-pairs of the training H dataset. X-axis: image resolution (Full = 2888×1920 pixels). Y-axis: disparity range in pixels.

against the time ELAS takes in its corresponding steps. We can see that our method is faster, even though it performs the additional edge detection. This is because our candidate support points are much likely to successfully match and the size of the candidate support points set is considerably smaller see Table IV. Fig. 9 shows the running time comparison, of the support point matching step, between LS-ELAS and ELAS. The y-axis is the average time in ms of the images of the Middlebury Stereo Evaluation Dataset. The image size is on the x-axis. *Q* stands for quarter resolution, *H* and *F* for half and full resolution correspondingly. The average time for the quarter resolution images is $15.7\,ms$ for our method and $17,9\,ms$ for ELAS. While the $2.2\,ms$ difference is not significant, it takes $64\%$ more time on the half resolution images. This increases to three times the calculation time on the full size images for the corresponding steps in ELAS. We can see that increasing the picture size affects ELAS much more than LS-ELAS. With a tendency to larger images, this difference becomes more relevant. The Table III shows the comparison results for all the Middlebury *metrics* on the *test sparse* dataset in full resolution. Our algorithm outperforms ELAS on all *metrics*. All the results above are from the adaptive sampling version of our algorithm. The version with uniform sampling is less accurate but slightly faster. The

TABLE IV
PERCENTAGE OF VALID SUPPORT POINTS

| Dataset | Algorithm | Candidates | Matched | Percent |
|---|---|---|---|---|
| Pipes F | LS-ELAS | 23311 | 15845 | 68% |
| | ELAS | 228144 | 20834 | 9% |
| ArtL F | LS-ELAS | 10797 | 6064 | 56% |
| | ELAS | 61716 | 6529 | 10% |

evaluation on the training sparse dataset showed that it is still more accurate than ELAS and most competing algorithms listed on Table I. For example: "bad 0.5" = 37.3, "bad 2.0" = 11.2 and "avgerr" = 3.56.

## REFERENCES


[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *IJCV*, vol. 47, no. 1-3, pp. 7–42, 2002.

[2] S. Sinha, D. Scharstein, and R. Szeliski, "Efficient high-resolution stereo matching using local plane sweeps," *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR*, 2014.

[3] M. Bleyer, C. Rhemann, and C. Rother, "Patchmatch stereo - stereo matching with slanted support windows," 2011, pp. 14.1–14.11.

[4] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Proceedings of the 10th Asian Conference on Computer Vision - Volume Part I*, ser. ACCV'10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 25–38.

[5] K.-J. Yoon and I.-S. Kweon, "Adaptive support-weight approach for correspondence search," *IEEE TPAMI*, vol. 28, no. 4, pp. 650–656, 2006.

[6] Z. Ma, K. He, Y. Wei, J. Sun, and E. Wu, "Constant time weighted median filtering for stereo matching and beyond," *ICCV*, 2013.

[7] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," *TPAMI*, vol. 16, no. 9, pp. 920–932, 1994.

[8] R. Ait Jellal and A. Zell, "A fast dense stereo matching algorithm with an application to 3d occupancy mapping using quadrocopters," *17th International Conference on Advanced Robotics*, 2015.

[9] M. Mozerov and J. van Weijer, "Accurate stereo matching by two step global optimization," *TIP*, 2014.

[10] R. Boykov, Veksler, and Zabih, "Graph cuts using alpha-beta swaps," *PAMI*, 2001.

[11] Q. Yang, W. L., and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," *CVPR*, 2010.

[12] A. Klaus, M. Sormann, and K. K., "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," *ICPR*, pp. 15–18, 2006.

[13] P. Felzenszwalb and H. D., "Efficient belief propagation for early vision," *International Journal of Computer Vision*, vol. 70, no. 1, pp. 41–54, 2006.

[14] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 2, pp. 328–341, 2008.

[15] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, June 1986.

[16] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Pattern Recognition*. Springer, 2014, pp. 31–42.

[17] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[18] H. Hirschmüller, P. R. Innocent, and J. Garibaldi, "Real-time correlation-based stereo vision with reduced border errors," *International Journal of Computer Vision*, pp. 229–246, 2002.

[19] M. G. Mozerov and J. van de Weijer, "Accurate stereo matching by two-step energy minimization," *Image Processing, IEEE Transactions on*, vol. 24, no. 3, pp. 1153–1163, 2015.

[20] N. Einecke and J. Eggert, "A two-stage correlation method for stereoscopic depth estimation," in *Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on*. IEEE, 2010, pp. 227–234.