Line Assisted Light Field Triangulation and Stereo Matching

Zhan Yu[1] Xinqing Guo[1] Haibin Ling[2] Andrew Lumsdaine[3] Jingyi Yu[1]
[1]University of Delaware, Newark, DE 19716, USA, {yshmzhan,xinqing,jingyiyu}@udel.edu
[2]Temple University, Philadelphia, PA 19122, USA, hbling@temple.edu
[3]Indiana University, Bloomington, IN 47405, USA, lums@osl.iu.edu

Abstract

Light fields are image-based representations that use densely sampled rays as a scene description. In this paper, we explore geometric structures of 3D lines in ray space for improving light field triangulation and stereo matching. The triangulation problem aims to fill in the ray space with continuous and non-overlapping simplices anchored at sampled points (rays). Such a triangulation provides a piecewise-linear interpolant useful for light field super-resolution. We show that the light field space is largely bilinear due to 3D line segments in the scene, and direct triangulation of these bilinear subspaces leads to large errors. We instead present a simple but effective algorithm to first map bilinear subspaces to line constraints and then apply Constrained Delaunay Triangulation (CDT). Based on our analysis, we further develop a novel line-assisted graph-cut (LAGC) algorithm that effectively encodes 3D line constraints into light field stereo matching. Experiments on synthetic and real data show that both our triangulation and LAGC algorithms outperform state-of-the-art solutions in accuracy and visual quality.

1. Introduction

Rays are directed lines in 3D space. They represent the visual information about a scene by their associated radiance function [1]. A light field (LF) [17, 9] captures a dense set of rays as scene descriptions in place of geometry. To represent each ray, a LF uses a two-plane parametrization (2PP). Every ray is parameterized by its intersections with two parallel planes: $[s, t]$ as the intersection with the first plane Π_{st} and $[u, v]$ as the second with Π_{uv}. Rays in a LF hence form a 4D space.

This paper explores ray geometry of a common primitive, 3D line segments. Previous studies show that the LF space is largely linear: a 3D scene point maps to a 2D ray hyperplane [39, 38]. This indicates that a LF can be "triangulated", i.e., the 4D can be partitioned into a set of space filling and non-overlapping simplices. For a 2D Epipolar-Plane Image (EPI), the simplices are 2D triangles; for a 3D LF, they are tetrahedra; and for the complete 4D LF, they are pentatopes. The triangulation provides a natural anisotropic reconstruction kernel: any point in the space can be approximated using a convex combination of the enclosing simplex's vertices (samples).

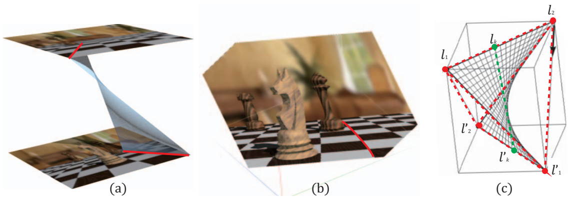

Figure 1. Bilinear ray structures. (a) A 3D line segment l maps to a bilinear subspace in a LF; (b) l maps to a curve on a diagonal cut; (c) Brute-force triangulation creates volume.

The simplest triangulation method is to apply high-dimensional Delauney Triangulation [8]. Such triangulations produce simplices (or pentatopes if in 4D) of "good shapes". However, triangulating the LF as such leads to severe aliasing, as shown in Fig. 3. A better approach is to align simplices with ray geometry of 3D scene. For example, we can first estimate the disparity (depth) of the feature pixels (rays), map them to the hyperplane constraints, and apply Constrained Delaunay Triangulation (CDT) [27]. We show this approach is still insufficient to produce high quality triangulations: the LF space contains a large amount of non-linear, or more precisely, bilinear substructures that correspond to 3D line segments. Brute-force triangulation of these bilinear structures leads to large errors and visual artifacts. We instead present a new solution that combines the bilinear and hyperplane constraints for CDT.

Our ray geometry analysis of 3D lines also leads to a new LF stereo algorithm. We first introduce a new $\mathcal{F}^3$ energy term to preserve disparity consistency along line segments. We then modify the binocular stereo graph via the general purpose graph construction framework [15] and solve it using the extended Quadratic Pseudo-Boolean Optimization algorithm [25]. We validate our approach on both Middlebury datasets, Stanford LF datasets [32] and real LF data acquired by the Lytro camera [19]. Experiments show that both our LF triangulation and stereo matching algorithms outperform state-of-the-art solutions in accuracy and visual quality.

2. Related Work

LF Space. The LF ray space is a vector space. Any linear combination of the $[s, t, u, v]$ coordinate of two rays is still a valid ray. This contrasts with the 6D $Plücker$ coordinates [10] which do not form closed vector space. Ponce [22] applies projective geometry to analyze ray structures. Yu and McMillan [39] use General Linear Cameras (GLCs) to analyze all 2D linear structures (hyperplanes) in 4D LF ray space. Their studies have shown that the LF ray space is largely linear and hence suitable for triangulation: scene geometry such as 3D points or parallel directions maps to GLCs. We show that special care needs to be taken to handle non-linear (bilinear) ray structures.

LF Acquisition. Our work is also motivated by recent advances on LF acquisition where reliable depth estimation and LF super-resolution are in urgent needs. Earlier camera-array based systems [32] are bulky and expensive, although they can acquire high (spatial) resolution LFs. More recent approaches aim to capture the LF using a single commodity camera. Ng [20] designs a hand-held LF camera that places a microlens array in front of the camera sensor to separate converging rays. This design has led to the commercial LF camera Lytro [19]. Lumsdaine et al. [18] introduce a slightly different design by focusing the microlens array on a virtual plane inside camera. In this case each microlens image captures more spatial samples but fewer angular samples on the refocusing plane. Levin and Durand [16] use the dimensionality gap prior to recover the 4D LF from a 3D focal stack without depth estimation.

LF Stereo. The availability of LF cameras has also renewed the interest on multi-view reconstruction. The seminal work by Kolmogorov and Zabih [14] extend the binocular graph-cut solution to multi-view stereo. In addition to the data and the smoothness terms, they add an occlusion term for handling complex occlusions. Based on the graph framework, Woodford et al. [37] further incorporate the second order smoothness priors and optimize the non-submodular objective function via Quadratic Pseudo-Boolean Optimization (QPBO) [25]. Bleyer et al. [3] impose soft segmentation and minimum description length as priors to solve for a non-submodular objective function. Most recently, Wanner and Goldlücke [26, 35] apply structure tensor to measure each pixel's direction in 2D EPI. They then encode the estimated edge directions into dense stereo matching with consistency check. Most previous algorithms, however, do not explicitly consider or aim to preserve the geometry of 3D lines.

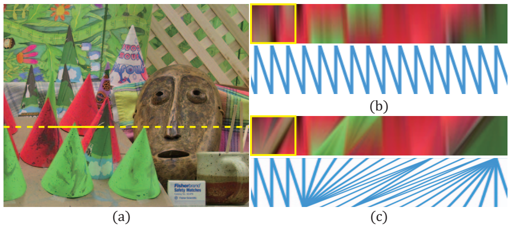

Figure 2. Triangulating a 2D LF (an EPI). (a) A scanline from a stereo pair; (b) RG Delaunay triangulation (bottom) performs poorly on LF super-resolution (top); (c) Using disparity as additional edge constraints, Constrained Delaunay triangulation significantly improves LF super-resolution.

3. Ray Geometry of 3D Lines

We first briefly reiterate the ray geometry of 3D lines [38, 23]. If a 3D line l is parallel to Π_{uv} and Π_{st}, we can represent it with a point $\dot{P} = [P^x, P^y, P^z]$ on l and its direction $[\gamma^x, \gamma^y, 0]$. All rays passing through l satisfy the following linear constraint:

$$As + Bt + Cu + Dv + E = 0, \tag{1}$$

where $A = \gamma^y - \gamma^y P^z, B = \gamma^x P^z - \gamma^x, C = \gamma^y P^z, D = -\gamma^x P^z, E = \gamma^x P^y - \gamma^y P^x$. This reveals that lines in the 3D scene that are parallel to Π_{uv} will map to linear subspaces in the LF and hence can be triangulated.

If l is not parallel to Π_{uv}, it then can be directly parameterized by a ray under 2PP as $[u_0, v_0, s_0, t_0]$. All rays passing through l satisfy the following *bilinear* constraint:

$$\frac{s - s_0}{u - u_0} = \frac{t - t_0}{v - v_0}. \tag{2}$$

The bilinear ray geometry is particularly important since a real scene usually contains many linear structures unparallel to the image plane. This reveals that the LF ray space contains a large amount of bilinear structures. In Fig. 1 (a), we construct a 3D LF by stacking a row of LF images and cut it using the videocube tool [33]. Fig. 1 (b) shows a cut through a volume where 3D lines on the checkerboard appear curved due to their bilinearity.

4. Light Field Triangulation

The simplest LF triangulation is regular-grid (RG) triangulation. Given a regularly sampled LF, we can first build 4D hypercubes using two corners (s, t, u, v) and $(s+1, t+1, u+1, v+1)$ and then triangulate each hypercube. Let us consider a 2D LF, an EPI formed by the same horizontal scanlines in a row of LF images. Fig. 2 (b) shows the RG triangulation of the EPI. If we use this triangulation to super-resolve the LF, i.e., by rasterizing the triangles, the result exhibits severe aliasing. This is because RG triangulation is analogous to bilinear interpolation and does not consider scene geometry (e.g., object depth or disparity).

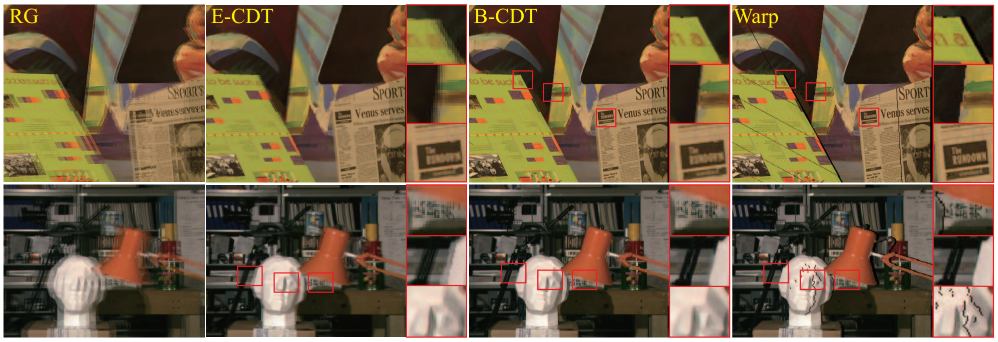

Figure 3. View interpolation using a triangulated 3D LF. We use the same set of feature points for RG, E-CDT, and B-CDT (ours). B-CDT produces comparable results to image warping but preserves continuity (no holes).

4.1. Constrained Delaunay Triangulation

To improve RG triangulation, we can add epipolar constraints. Using stereo matching, we can first estimate every pixel's disparity and map it to a 2D hyperplane [38, 23] as a constraint. In the 2D EPI case, each pixel maps to an edge where the slope of the edge corresponds to its disparity (depth). We can then apply Constrained Delaunay Triangulation (CDT) [27]. We call this scheme EPI-CDT or E-CDT. Fig. 2 (c) shows an E-CDT triangulation and its super-resolution result. Specifically, we first detect 47 salient feature points along the scanline and add their corresponding EPI constraints. Our triangulation applies CDT to all pixels with these edge constraints. Fig. 2 (c) show the closeup views of the triangulation. E-CDT greatly reduces aliasing while providing a continuous interpolant.

An apparent question is whether we can directly apply E-CDT to triangulating higher dimensional LFs. The second column of Fig. 3 shows the E-CDT result of two images forming a 3D LF from the Middlebury Venus dataset. Specifically, we detect $10,132$ feature points (6% of total pixel) and use their disparities as edge constraints. We use the Tetgen [29] to conduct Constrained Delaunay Tetrahedralization. To illustrate its quality, we synthesize a new intermediate view between two source views by rasterizing the tetrahedralized 3D LF. The new view improves RG at non-occlusion regions but exhibits strong aliasing near linear occlusion boundaries.

To analyze the cause of aliasing, let us consider a 3D line segment l whose image is $(l_1^x, l_1^y)-(l_2^x, l_2^y)$ in LF view (s,t). Assume the disparity of l_1 and l_2 are d_1 and d_2 respectively. If $d_1 \neq d_2$, by Eqn. 2, l maps to a bilinear surface S formed by four corners (s, l_1^x, l_1^y), (s, l_2^x, l_2^y), $(s+1, l_1^x+d_1, l_1^y)$, and $(s+1, l_2^x+d_2, l_2^y)$ in 3D (s,u,v) LF space. In geometric modeling, it is well known that any direct triangulation of S from the four vertices of S will introduce large error: S is a surface that does not occupy any volume. However, a triangulation of the four vertices will turn S into a tetrahedron which will occupy large volume when $|d_1 - d_2|$ is large, as shown in Fig. 1 (c). The tetrahedron will "erode" into neighboring space, i.e., nearby pixels will be forced to use this tetrahedron as the interpolant. Therefore it is important to add additional constraints onto the bilinear structure.

4.2. CDT with 3D Edge Constraints

We present a simple but effective scheme that directly maps bilinear ray structures of 3D lines into the CDT framework. Specifically, we apply a subdivision scheme [21] by discretizing the bilinear surface into slim bilinear patches and then triangulate each patch. Finally, we use edges of bilinear patches and disparity hyperplanes as constraints for CDT. We call this scheme Bilinear CDT or B-CDT.

3D LFs. For a 3D LF, B-CDT can be effectively implemented using Tetgen [29]. In the Venus example (first row of Fig. 3), we detect additional 303 line segments in the reference view, subdivide their corresponding bilinear surfaces, and add them as constraints for conducting B-CDT. The new triangulation significantly improves the E-CDT result: it preserves most sharp edges and exhibits very little aliasing near occlusion boundaries. Compared with image warping that results in missing pixels or holes, the B-CDT provides a continuous representation of the LF where any new view corresponds to a valid 2D triangulated LF without holes. Notice that the texts on the newspaper are slightly blurred since they are not selected as constraints.

Fig. 3 row 2 shows our result on the Tsukuba dataset containing fewer linear structures. In this example, we select $15,748$ feature points (14% of total pixel) from the reference image and detect 120 line segments. Same as the Venus example, we compare E-CDT and B-CDT by rendering an new view between the two reference views. E-CDT preserves non-boundary contents but exhibits strong aliasing near the boundary pixels such as the tripod, the light edges, and the bust. In contrast, B-CDT preserves both boundary and non-boundary contents. The Tsukuba scene has a relatively large disparity range and direct warping produces many holes. To patch these holes, one can use 2D interpolation schemes such as Delaunay triangulation.

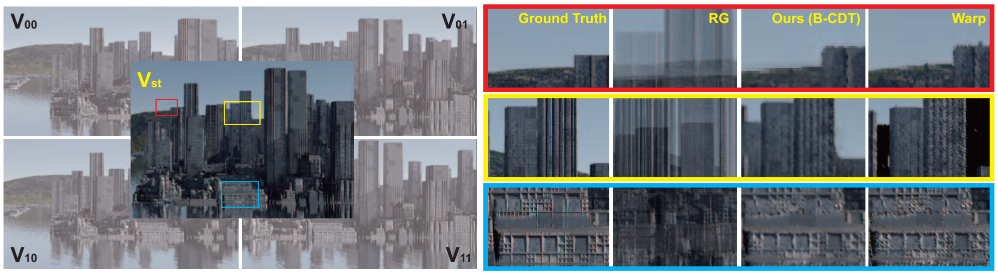

Figure 4. New view (central) synthesis from a 4D LF. Left: a LF of a skyscraper scene. Right: Closeup views of the synthesized results using different schemes.

Such interpolation, however, is different from B-CDT: B-CDT provides a consistent triangulation throughout the LF volume while warping followed by hole patching produces an ad-hoc triangulation on each slice; Further, B-CDT only needs to be conducted once while hole patch needs to be conducted whenever rendering a new view.

4D LFs. Finally, we extend the B-CDT scheme to 4D LFs. In computational geometry, high dimensional CDTs [27] remains as an open problem for two reasons. First, a plausible solution may require inserting a large number of auxiliary vertices. This also occurs in 3D CDT although the number of inserted vertices is much smaller. Second, the computational complexity grows rapidly with respect to dimensionality [2]. To our knowledge, no practical 4-dimensional CDT is currently available to the public.

Our solution is to convert the 4D problem to 3D. Specifically, to synthesize a new view V_{st} in the 4D LF with four sample views indexed as V_{00}, V_{01}, V_{10}, V_{11}, we first detect 3D line segments and apply 3D B-CDT to synthesize two new views V_{s0} and V_{s1} from 3D LFs $V_{00} - V_{10}$ and $V_{01} - V_{11}$, respectively. Next, we use the same 3D line constraints and B-CDT to triangulate a 3D LF $V_{s0}-V_{s1}$ for synthesizing V_{st}. Fig. 4 shows an skyscraper LF with disparity range [0,300]. Results using RG exhibit severe aliasing where directly warping produces holes and discontinuity. Next, we select $90,269$ feature points (11% of total pixel) and 2092 line segments and apply the pseudo 4D CDT. Our results exhibits little aliasing while preserving smoothness.

5. Light Field Stereo Matching

Our LF triangulation also provides useful insights on incorporating 3D line constraints into multi-view stereo.

5.1. Disparity Interpolant

We first prove the linearity of disparity along a line segment, i.e., given two endpoints l_1 and l_2 of a 3D line segment l with disparity d_1 and d_2, the disparity d_k of any intermediate point $l_k = \lambda_k l_1 + (1-\lambda_k)l_2$ is $\lambda_k d_1 + (1-\lambda_k)d_2$. This property is well known, e.g., in perspective geometry in computer vision and in projective texture mapping in computer graphics. We present a different proof based on bilinear ray geometry of line l.

If l is parallel to Π_{uv} and Π_{st}, then the proof is trivial since $d_1 = d_2 = d_k$.

If l is not parallel to Π_{uv} and Π_{st}, l can be represented as a ray (s_0, t_0, u_0, v_0). Consider a specific pixel (s, t) in camera (u, v) that observes a point P on line l and pixel $s + \Delta s$ in a neighbor camera $(u + \Delta u, v)$ that also observes P. Both ray (s, t, u, v) and $(s + \Delta s, t, u + \Delta u, v)$ satisfy the bilinear ray constraint (Eqn. 2):

$$\frac{s + \Delta s - s_0}{u + \Delta u - u_0} = \frac{s - s_0}{u - u_0} = \frac{t - t_0}{v - v_0}. \tag{3}$$

Therefore, $\frac{\Delta s}{\Delta u} = \frac{t - t_0}{v - v_0}$. This reveals that disparity $\frac{\Delta s}{\Delta u}$ is a linear function in t along l, i.e., we can linearly interpolate the disparity along l.

5.2. Line-Assisted Graph Cut (LAGC)

To incorporate the linear disparity constraint into multi-view stereo matching, the most direct approach is to first detect line segments in the captured LF, then estimate their disparities, and use them as hard constraints in the graph-cut algorithm. The top row of Fig. 5 shows the result of this brute-force approach on a city scene. We render a LF of the scene (17×17 views at 1024×768 resolution). We detect line segments using the state-of-the-art line segment detector (LSD) [34] for each view (around $1100 \times 17 \times 17$ line segments). For the endpoints of each line segment l, we iterate over all possible disparities and interpolate the disparity for all intermediate points. Finally, we find the optimal disparity assignments to the endpoints that yield to highest consistency of all intermediate points. The results are then used as hard constraints for the multi-view graph-cut (MVGC) [14]. Fig. 5 shows improvements near edges and rich texture regions compared with MVGC. However, if the disparity of the line segment is incorrectly assigned, it will lead to large errors, e.g., on one of the chimneys on the building, as shown in Fig. 5.

Next, we study how to explicitly encode the disparity constraint of line segments into MVGC. MVGC aims to find the optimal disparity label that minimizes the energy

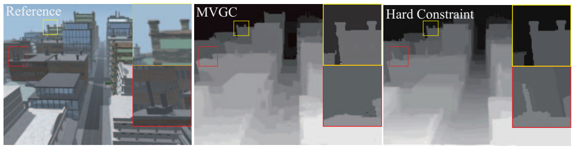

Figure 5. Encoding 3D line segments as hard constraints improves MVGC but misses important details, e.g. the chimney on the building.

function $E_{conventional} = E_{data} + E_{smooth} + E_{occ}$,where

$$E_{data} = \sum_{P,Q} E_d(P,Q), \quad E_d(P,Q) = ||I(P) - I'(Q)||_2 - K,$$
$$E_{smooth} = \sum_{P,P_N \in \mathcal{N}} E_s(P,P_N), \quad E_s(P,P_N) = min(||d_P - d_{P_N}||, T_c), \tag{4}$$

where P and Q correspond to the same 3D point given a disparity, $\mathcal{N}$ is the neighborhood of P, T_c is the truncation threshold, and K is a constant. The occlusion term E_{occ} measures if occlusion is correctly preserved when warping the disparity from I to I' [14].

We add the fourth *line constraint* term. Our key observation is that when assigning disparity labels to the two endpoints, every intermediate point along the line should check occlusion consistency. Specifically, given the two endpoints (pixels) l_i and l_j of line segment l and an intermediate pixel $l_k = \lambda_k l_i + (1 - \lambda_k) l_j$, we define

$$E_{line} = \sum_l \sum_{l_k \in [l_i, l_j]} E_l(l_i, l_j, l_k),$$
$$E_l(l_i, l_j, l_k) = ||\lambda_k d_{l_i} - d_{l_k} + (1 - \lambda_k) d_{l_j}||. \tag{5}$$

Our goal is to minimize the new energy function $E_{conventional} + E_{line}$.

The work by Boykov et al. [5] and Kolmogorov and Zabih[13, 14] show that one can minimize $E_{conventional}$ by consecutively solving the two-label problem: at each iteration, a new disparity label is added and the algorithm decides whether each pixel should keep the old disparity or switch to the new disparity. We follow their convention to use *0* for keeping the old label and *1* for using the new label. To solve for the two label problem with $alpha$-expansion [5], the energy function needs to be regular. For example, E_{data}, E_{smooth} and $E_{occlusion}$ (the two-variable functions) are all regular.

Notice that our E_{line} (Eqn. 5) is a three-variable ($\mathcal{F}^3$) term, i.e., the endpoints and any intermediate point individually choose to relabel or not. E_{line} can be viewed as a general second order smoothness prior and is generally non-submodular. Therefore, $alpha$-expansion is not directly applicable to minimize E_{line}. We instead adopt the extended QPBO approaches proposed by Rother et al. [25]. To briefly reiterate, the QPBO algorithm [12] splits each node in the graph into two subnodes; when both subnodes are assigned to the source or sink after min-cut, they will be assigned the corresponding label. Otherwise, they will be treated as unlabeled. Theoretically, QPBO can potentially result in a large number pixels assigned unlabeled. In our experiments, we observe that QPBO results in about 10% or less unlabeled pixels. This is because an unlabel pixel is caused by non-submodularity which, in our case, occurs mostly on the line segments since E_{line} is non-submodular. For natural scenes, line segments are generally sparse and therefore only a small percentage of pixels are unlabeled in QPBO.

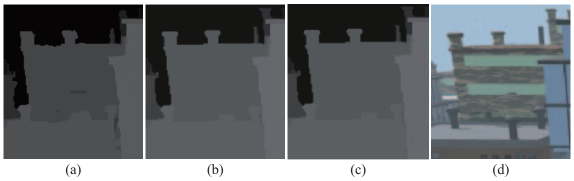

Figure 6. Comparison using different optimization schemes. (a) $alpha$-expasion.(b) QPBO-I. (c) QPBO-P (d) Reference Image.

Extensions of QPBO such as QPBO-P and QPBO-I [4, 25] as well as the more complex QPBO-R [37] have shown great success on reducing the unlabeled pixels. For example, QPBO-I uses additional geometry priors (called the proposals) to improve optimization. We use fronto-parallel surface priors as proposals. Fig. 6 shows the results on the city scene using QPBO-P, QPBO-I (with fronto-parallel surfaces as proposals), and standard $alpha$-expansion. QPBO-P and QPBO-I produce comparable results while $alpha$-expansion produces noticeable artifacts such as inaccurate edges.

5.3. Graph Construction

Next, we construct the graph so that we can reuse min-cut/max-flow algorithm to minimize our $\mathcal{F}^3$ energy function. We follow the general-purpose graph construction framework by Kolmogorov and Zabih [15]: each pixel corresponds to a graph node. We then add the source s node for label *0*, the sink node t for label *1*, the t-links from the graph nodes to s or t, and the n-links between the graph nodes using 4-connectivity. We decompose the two variable term E_d and E_s to the corresponding t-links and n-links. For example, for two neighboring nodes n_i and n_{i+1}, we assign weights $E_s(1,0) - E_s(0,0)$ and $E_s(1,0) - E_s(1,1)$ to t-links (s, n_i) and (n_{i+1}, t) respectively, and weight $E_s(0,1) + E_s(1,0) - E_s(1,1) - E_s(0,0)$ to n-link (n_i, n_{i+1}). The similar scheme can be applied for handling E_{occ}.

Different from E_s and E_{occ}, E_l is $\mathcal{F}^3$ and auxiliary nodes and links need to be added to the graph [15]. Specifically, for each edge tuple (l_i, l_j, l_k) (i, j the endpoints and k the intermediate point), we add three auxiliary n-links (an-links) $l_i - l_j$, $l_i - l_k$ and $l_k - l_j$, as shown in Fig. 7 (b). We also add an auxiliary sink/source node n_k^* and the corresponding auxiliary t-links (at-links) using one of the two possible assignments: either (l_i, n_k^*), (l_j, n_k^*), (l_k, n_k^*), and (n_k^*, t) (Group at_1) or (n_k^*, l_i), (n_k^*, l_j), (n_k^*, l_k), and (s, n_k^*) (Group at_2), as shown in Fig. 7 (c). The selection of the

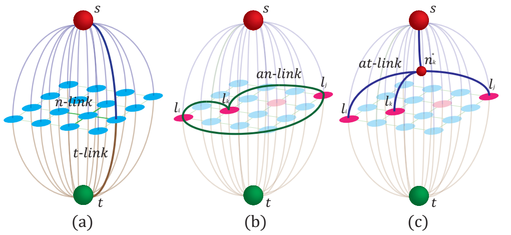

Figure 7. Graph construction for our LAGC algorithm. (a) The conventional graph for two-view stereo matching.(b) For a line segment (pink), we add auxiliary n-links (green). (c)We also add an auxiliary node n_k^* and auxiliary t-links (dark blue).

group depends on the weight decomposition of E_l. Specifically, we follow the same procedure in [15] and rename all 8 cases of E_l as shown in the Appendix A. We call our solution the line-assisted graph-cut (LAGC).

5.4. Evaluation

All experiments were conducted on a PC with Intel Core i7 3.2GHz CPU and 8GB memory. We first evaluate our algorithm on binocular stereo using the Tsukuba dataset. Fig. 8 compares the ground truth, global stereo reconstruction under second order smoothness priors (SOSP) [37], adaptive ground control point (GCP) [28] (rank 1 for all four Middlebury datasets [6]), MVGC [14], and our LAGC. Table 1 lists the percentage of bad pixels and the ranking (in subscripts) for each method. Compared with MVGC, LAGC effectively reduces errors and outranks MVGC (13 vs. 35 for non-occlusion, 14 vs. 38 for boundary, and 14 vs. 50 for all pixels). LAGC also preserves edges such as the feet of the table and the tripod and the arm of the lamp.

Next, we apply LAGC to the LF datasets. The detected line segments are highlighted in red. We first test on a synthetic LF of a city scene composed of one million triangles. We render the scene using the POV-Ray ray-tracer [24] to generate an array of 17 × 17 images, each with a resolution of 1024 × 768. The scene has a disparity range from 0 - 16 pixels. Notice that the city scene exhibits repeated line patterns. Certain regions lack textures while the others contain complex textures. The scene hence is challenging for classical stereo matching. For this and the following examples, we fine-tune the parameters for both algorithms and compare only their best results.

Algorithm	non-occlusion	all	discontinuity
LAGC	1.00_{13}	1.41_{14}	5.39_{14}
MVGC	1.27_{35}	1.99_{50}	6.48_{38}
SOSP	2.91_{103}	3.56_{92}	7.33_{57}
GCP	1.03_{14}	1.29_{5}	5.60_{16}

Table 1. Stereo matching using LAGC (ours), MVGC [14], SOSP [37], and GCP [28] on Tsukuba. We show both the percentage of bad pixels and the algorithm's ranking (in subscripts)

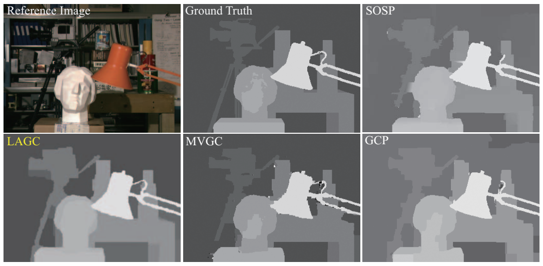

Figure 8. Stereo matching on the Tsukuba dataset. Our LAGC outperforms MVGC [14] and SOSP [37] but is slightly worse than GCP [28]. However, it better preserves edges, e.g., the left foot of the tripod. See Table 1 for numerical comparison.

We compare our scheme with the recent globally consistent depth labeling (GCDL) scheme using the source code posted by the author of [26, 36]. The top row of Fig. 9 compares the disparity maps computed by GCDL and LAGC. Recall that GCDL uses the structure tensor to measure local EPI structures. Therefore, it requires ultra-densely sampled LFs with a small disparity range (usually between -2 to 2). As a result, GCDL misses fine details such as the contours of the chimneys and is highly noisy on surfaces with rich textures. Its error is also larger on distant buildings. In contrast, LAGC accurately preserves most fine details and robustly handles both distant and close objects. Although a real scene may not contain as many linear structures, our result demonstrates that LAGC is robust enough to handle such complex scenes. We then experiment on real LF data. Fig. 9 row 2 shows the comparison on the Stanford Lego Gantry dataset [32] composed of 17 × 17 views at a resolution of 1280 × 960 of a Lego gantry crane model. The disparity range is between -3 to 5 pixels and we discretize it into 16 labels. On continuous regions such as the ground, LAGC produces much smoother disparity transitions whereas the result from GCDL contains large discontinuities. LAGC is particularly good at preserving edges, as shown on the hoist rope from the crane, the contours of the headlights and windows, etc. Fig. 9 row 3 shows the amethyst dataset which is expected to be challenging to LAGC: it lacks long linear structures but contains strong view-dependent features. The disparity range is small, from -3 to 3 pixels. We consider subpixel disparity with step size 0.2. LAGC can robustly handle this challenging scene and slightly outperforms GCDL, e.g., by better preserving the facets of the amethyst.

Finally, we test on a real LF acquired by the Lytro camera [19]. Lytro uses an array of 328 × 328 microlenses, each with 10 × 10 pixel resolution. We first resample the LF to a 11 × 11 LF at 800 × 800. The disparity range is ultra small (between -1 to 1 pixels). We discretize the disparity range using 0.1 step (20 disparity label). The original GCDL code [26] was not directly applicable to process this data. At our request the authors of [26] sent us the results using their modified GCDL. Both LAGC and the revised GCDL produce reasonable results despite structure and texture similarities across the scene. Our results, however, better preserves linear structure such as the narrow vertical wall in the background. Additional results and our source code can be found at the project's website [40].

6. Limitations and Future Work

We have presented a LF triangulation and stereo matching framework by imposing ray geometry of 3D line segments as constraints. For LF triangulation, we utilize Constrained Delaunay Triangulation and by far our solution is restricted to 3D and pseudo 4D LFs since 4D CDT is still an open problem in computational geometry. An immediate future direction is to experiment our scheme on irregularly sampled LF, e.g., the ones captured by a catadioptric mirror array [31] or by a hand-held camera [7]. Our current super-resolution scheme requires rasterizing ray simplices into voxels. An alternative approach is to use a walk-through algorithm that picks one face of the ray simplex at a time and does the orientation test for locating the simplex, a process can be accelerated using parallel processing on the graphics hardware. Finally, the triangulated LF can be potentially compressed via geometric compression. For example, half-edge collapse operator in progressive meshes [30, 11] can be used to remove edges and vertices while maintaining a continuous simplex-tiled structure.

For stereo matching, we have experimented on synthetic, pre-acquired LF, and Lytro acquired LFs. An important following step is test our scheme on the Raytrix data which have a larger disparity range. In addition, given the increasing interest in LF imaging and the availability of commercial LF cameras, we also plan to build a LF stereo benchmark of real scenes analogous to the Middlebury Stereo Portal[6], for evaluating LF stereo matching algorithms. Finally, it remains an open problem on how to handle view-dependent objects in both binocular and multi-view stereo. In the future, we will investigate robust algorithms for detecting and reconstructing these objects via LF analysis.

7. Acknowledgments

We thank Sven Wanner and Bastian Goldlüecke for providing the GCDL code and providing the results on the Lytro data. This project was supported by the National Science Foundation under grants IIS-RI-1016395 and IIS-RI-121856.

References

[1] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, pages 3–20. MIT Press, 1991.

[2] D. Attali and J.-D. Boissonnat. Complexity of the delaunay triangulation of points on polyhedral surfaces. *Discrete & Computational Geometry*, 30(3):437–452, 2003.

[3] M. Bleyer, C. Rother, and P. Kohli. Surface stereo with soft segmentation. In *CVPR 2010.*

[4] E. Boros, P. L. Hammer, and G. Tavares. Preprocessing of unconstrained quadratic binary optimization. Technical report, 2006.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE TPAMI*, 23:2001, 2001.

[6] M. S. Datasets. http://vision.middlebury.edu/stereo/data/.

[7] A. Davis, M. Levoy, and F. Durand. Unstructured light fields. In *Proceedings of Eurographics*, 2012.

[8] B. N. Delaunay. Sur la sphére vide. In *Bulletin of Academy of Sciences of the USSR*, pages 793–800, 1934.

[9] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Proceedings of ACM SIGGRAPH*, pages 43–54, 1996.

[10] W. V. D. Hodge and D. Pedoe. Methods of algebraic geometry, volume i (book ii), 1994.

[11] H. Hoppe. Efficient implementation of progressive meshes. *Computers and Graphics*, 1998.

[12] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts-a review. *TPAMI*, 29(7):1274–1279, 2007.

[13] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *ICCV 2001.*

[14] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *Proceedings of the ECCV*, 2002.

[15] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *PAMI, IEEE Transactions on*, 26(2):147 –159, feb. 2004.

[16] A. Levin and F. Durand. Linear view synthesis using a dimensionality gap light field prior. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1831–1838, 2010.

[17] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of ACM SIGGRAPH*, pages 31–42, 1996.

[18] A. Lumsdaine and T. Georgiev. The focused plenoptic camera. In *IEEE ICCP*, pages 1–8, 2009.

[19] Lytro. www.lytro.com.

[20] R. Ng, M. Levoy, M. Brdif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. *Stanford University Computer Science Tech Report*, 2:1–11, 2005.

[21] J. Peters and U. Reif. The simplest subdivision scheme for smoothing polyhedra. *ACM Trans. Graph.*, 16(4), Oct. 1997.

[22] J. Ponce. What is a camera? In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[23] J. Ponce and Y. Genc. Epipolar geometry and linear subspace methods: A new approach to weak calibration. *IJCV 1996.*

[24] POV-Ray. www.povray.org.

[25] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary mrfs via extended roof duality. In *CVPR 2007.*

[26] B. G. S. Wanner. Globally consistent depth labeling of 4d light fields. In *Proceedings of IEEE CVPR*, 2012.

[27] J. R. Shewchuk. General-dimensional constrained delaunay and constrained regular triangulations i: Combinatorial properties. In *Discrete and Computational Geometry*, 2005.

[28] C. Shi, G. Wang, X. Pei, H. Bei, and X. Lin. High-accuracy stereo matching based on adaptive ground control points. *Submitted to IEEE TIP*, 2012.

[29] H. Si. Tetgen: A 3d delaunay triangulator.

[30] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, Aug. 2004.

[31] Y. Taguchi, A. Agrawal, A. Veeraraghavan, S. Ramalingam, and R. Raskar. Axial-cones: modeling spherical catadioptric cameras for wide-angle light field rendering. In *ACM SIGGRAPH Asia*, 2010.

[32] S. University. Stanford light field.

[33] Videocube. Microsoft.

[34] R. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *TPAMI*, 2010.

[35] S. Wanner and B. Goldlüecke. Spatial and angular variational super-resolution of 4d light fields. 2012.

[36] S. Wanner and B. Goldlüecke. Variational light field analysis for disparity estimation and super-resolution. 2013.

[37] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. In *CVPR*, june 2008.

[38] J. Yu. *General linear cameras: theory and applications*. PhD thesis, 2005.

[39] J. Yu and L. McMillan. General linear cameras. In *ECCV (2)*, pages 14–27, 2004.

[40] Z. Yu and J. Yu. http://www.eecis.udel.edu/ zyu/iccv2013/.

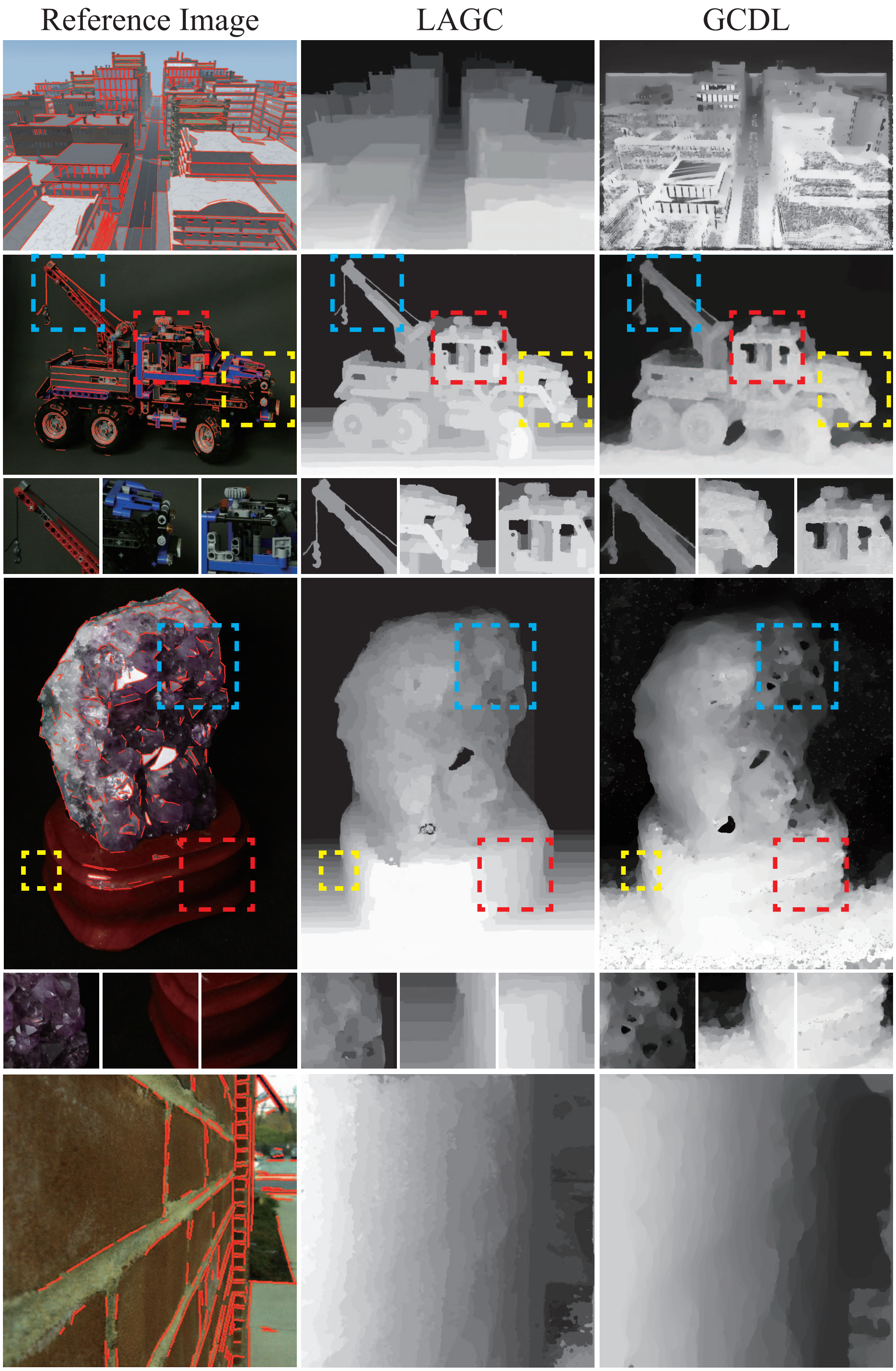

Figure 9. LAGC vs. GCDL [26] in LF. From top to bottom: a city scene LF (17 × 17 × 1024 × 768) rendered using POV-Ray, the Stanford Gantry LF (17 × 17 × 1280 × 960) and Amethyst LF (17 × 17 × 768 × 1024), and a real LF captured by Lytro [19]. The Lytro result was generated by the authors of [26] using the modified GCDL.

Appendices

A. Decomposition of Line Constraint Term

$$E_l = \begin{array}{|c|c|}\hline E(0,0,0) & E(0,0,1) \\ \hline E(0,1,0) & E(0,1,1) \\ \hline E(1,0,0) & E(1,0,1) \\ \hline E(1,1,0) & E(1,1,1) \\ \hline \end{array} = \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline e & f \\ \hline g & h \\ \hline \end{array} = \begin{array}{l} A + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline p_1 & p_1 \\ \hline p_1 & p_1 \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline p_2 & p_2 \\ \hline 0 & 0 \\ \hline p_2 & p_2 \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & p_3 \\ \hline 0 & p_3 \\ \hline 0 & p_3 \\ \hline 0 & p_{32} \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & p_{23} \\ \hline 0 & 0 \\ \hline 0 & p_{23} \\ \hline 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline p_{31} & 0 \\ \hline p_{31} & 0 \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline p_{12} & p_{12} \\ \hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline 0 & 0 \\ \hline 0 & -p \\ \hline \end{array} \quad \begin{array}{ll} p_1 = f - b & p_{23} = b + c - a - d \\ p_2 = g - e & p_{31} = b + e - a - f \\ p_3 = d - c & p_{12} = c + e - a - g \\ \multicolumn{2}{l}{p = (a + d + f + g) - (b + c + e + h) \geq 0} \end{array} \\ \\ H + \begin{array}{|c|c|}\hline p_1 & p_1 \\ \hline p_1 & p_1 \\ \hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|}\hline p_2 & p_2 \\ \hline 0 & 0 \\ \hline p_2 & p_2 \\ \hline 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|}\hline p_3 & 0 \\ \hline p_3 & 0 \\ \hline p_3 & 0 \\ \hline p_3 & 0 \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline p_{32} & 0 \\ \hline 0 & 0 \\ \hline p_{32} & 0 \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & p_{13} \\ \hline 0 & p_{13} \\ \hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline p_{21} & p_{21} \\ \hline 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|}\hline p & 0 \\ \hline 0 & 0 \\ \hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \quad \begin{array}{ll} p_1 = c - g & p_{32} = f + g - e - h \\ p_2 = b - d & p_{13} = d + g - c - h \\ p_3 = e - f & p_{21} = d + f - b - h \\ \multicolumn{2}{l}{p = (a + d + f + g) - (b + c + e + h) < 0} \end{array} \end{array}$$

Table 2. We follow the $\mathcal{F}^3$ decomposition scheme from [15](Table 7 and 9) for E_l

We follow their decomposition and edge assignment schemes for $\mathcal{F}^3$. Here we briefly reiterate this process. There are two possible decompositions of E_l. We first compute $p = (a + d + f + g) - (b + c + e + h)$. If $p \geq 0$, E_l can be decomposed using the upper branch of Table 2. We then assign the weights to edges as follows: 1) assign p to all four at-links in group at_1; 2) Assign p_1, p_2, p_3 to t-links for l_i, l_j, and l_k respectively, and finally assign p_{12}, p_{23}, p_{31} to an-links (n_i, n_j), (n_j, n_k), and (n_k, n_i). If $p > 0$, we can decompose the table in a similar fashion as shown in the lower branch of Table 2 and assign the weights to edges accordingly.