LOCK-FREE MULTITHREADED SEMI-GLOBAL MATCHING WITH AN ARBITRARY NUMBER OF PATH DIRECTIONS

Dirk Frommholz

DLR Institute of Optical Sensor Systems, Berlin, Germany - dirk.frommholz@dlr.de

Commission II, WG II/2

KEY WORDS: Stereo Matching, SGM, Line Rasterization, Multithreading, OpenMP, SIMD

ABSTRACT:

This paper describes an efficient implementation of the semi-global matching (SGM) algorithm on multi-core processors that allows a nearly arbitrary number of path directions for the cost aggregation stage. The scanlines for each orientation are discretized iteratively once, and the regular substructures of the obtained template are reused and shifted to concurrently sum up the path cost in at most two sweeps per direction over the disparity space image. Since path overlaps do not occur at any time, no expensive thread synchronization will be needed. To further reduce the runtime on high counts of path directions, pixel-wise disparity gating is applied, and both the cost function and disparity loop of SGM are optimized using current single instruction multiple data (SIMD) intrinsics for two major CPU architectures. Performance evaluation of the proposed implementation on synthetic ground truth reveals a reduced height error if the number of aggregation directions is significantly increased or when the paths start with an angular offset. Overall runtime shows a speedup that is nearly linear to the number of available processors.

1. INTRODUCTION

Solving the correspondence problem is fundamental to photogrammetry as it will allow the reconstruction of 3D points from 2D bitmaps if the underlying camera geometry is known. Numerous algorithms have been proposed to find identical pixels in two calibrated images among which semi-global matching (SGM) (Hirschmüller, 2008) has been outstandingly successful in terms of speed and quality of the resulting disparity maps (Scharstein et al., 2014). Since the proposal of the classic algorithm and the availability of its original single-threaded implementation, many contributions have been made to improve and accelerate SGM, and some of them discharged into scientific and commercial-grade software packages.

Notably, the tSGM algorithm of (Rothermel, 2016) reduces both the number of calculations and the amount of memory required for the cost aggregation stage of SGM. This is achieved through dynamic bounds on the disparity ranges for each image pixel. The bounds are obtained from matching subsampled copies of the input stereo pair and propagating the disparity estimates to higher levels of the image pyramid. In its multithreaded commercial incarnation named SURE, the cost aggregation scheme of tSGM is identical to classic SGM and relies on a fixed number of typically 8 or 16 path directions (nFrames GmbH, 2014). Thus, in practice, homogeneous areas sometimes lack point coverage, and low-amplitude artifacts from the approximate "semi-global" minimization of the energy function involved in the matching process can be observed in the produced disparity maps. Characteristic quality issues related to SGM are specifically addressed by (Facciolo et al., 2015) in their more global matching (MGM) approach. In MGM, information is transferred between adjacent cost aggregation paths of a particular direction resembling belief propagation in graphs. It is demonstrated that the technique promotes smoothness and consistency in the generated disparity maps. However, the introduced data dependency comes at the price of a speed penalty, and it is likely that parallelization will be adversely impacted. Other modifications to SGM target its cost function and smoothness term to enhance the accuracy and coverage of the resulting depth estimates leaving the actual cost aggregation strategy untouched. This for instance includes the use of weighted similarity metrics (Miclea, Nedevschi, 2017), extended discontinuity penalties (Michael et al., 2013), their automatic adaptation to the image content (Karkalou et al., 2017), and combinations. Lately, machine learning techniques to predict SGM parameters have been discussed (Seki, Pollefeys, 2017).

This paper describes an implementation of SGM that supports a nearly arbitrary positive number of path directions for the cost aggregation stage. Hence, as a brute-force alternative to MGM, the user is given full control over the degree of approximation of the global energy minimization problem to be solved during stereo matching. In the implementation, for each direction, cost aggregation is performed along linear paths. The shapes of the paths are precomputed using an iterative line rasterization algorithm. Path-wise cost summing repeatedly reuses and shifts the resulting regular substructures, or runs, from the rasterizer, which happens in at most two partial sweeps per direction over the entire disparity space image. Since adjacent paths are properly aligned and sweeps are coordinated, it is guaranteed that the discrete scanlines do not overlap. Because there is no data dependency, path-wise multithreading can be applied without the need for time-consuming synchronization primitives. Computational efficiency is further improved by dynamically gating the disparities to a small range estimated from subsampled versions of the input images like in tSGM. On the machine level, the expensive cost calculation and aggregation stages of the proposed algorithm are parallelized using single instruction multiple data (SIMD) commands for the omnipresent Intel x86-64 and ARM CPU architectures.

The performance regarding both the runtime and disparity map quality of the new SGM implementation is evaluated on a synthetic 3D scene which gets rendered into error-free oriented pinhole camera images and depth ground truth. Following stereo

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2020-143-2020

matching, the reconstructed points are compared to the known scene geometry. The influence of high path direction counts and asymmetry on the height error and point coverage is examined for detailed scene elements and critical low-texture objects. Extensive optimization of the SGM penalty functions or disparity map refinement techniques remain unconsidered, however, these may complement the proposed aggregation scheme.

2. REVIEW OF THE BASELINE SGM ALGORITHM

The classic SGM algorithm comprises four basic processing stages. Assuming two oriented rectified images with parallel epipolar lines on the same vertical coordinate (Fusiello et al., 2000), the matching cost is initially computed for each pixel $\mathbf{p}$ at position (x, y) and disparity value d within the allowed range for the stereo shift. The cost calculation stage yields a disparity space image (DSI) which actually is a three-dimensional box-like volume $C(\mathbf{p}, d)$ storing the similarity between potentially corresponding pixel pairs $\mathbf{p}(x, y)$ and $\mathbf{p}'(x + d, y)$. A popular choice for the cost function serving as a similarity measure is the census transform because it is tolerant to radiometric differences between the images and can be efficiently calculated (Zabih, Woodfill, 1994).

Global stereo matching algorithms now attempt to optimally fit a 2D surface through the disparity space image minimizing the total cost, or energy. However, due to ambiguities in the DSI caused by homogeneous and repetitive areas in the input bitmaps, the neighborhood of a pixel must be incorporated using dynamic smoothness terms. This makes finding the desired surface and thus the corresponding pixels NP-complete and prohibitively time-consuming on current computers, at least when energy minimization is performed in two or more dimensions (Boykov et al., 2001). For one-dimensional structures, the optimal solution can be derived efficiently using dynamic programming (Cox et al., 1996) which leads to "semi-global" matching.

In SGM, the two-dimensional surface is approximated by minimizing equation 1 along a set of paths of direction $\mathbf{r}$. Here the cost $L_\mathbf{r}$ of pixel $\mathbf{p}$ at disparity d comprises the initial cost C from the DSI plus the minimum of the costs at the previous pixel along the path at the same disparity, at the disparity differing by one and at the disparity differing by more than one. A disparity change during the transition from $\mathbf{p} - \mathbf{r}$ to $\mathbf{p}$ however gets penalized by P_1 and $P_2 \geq P_1$. This introduces a smoothness constraint which permits small depth changes and discourages depth jumps. The last term subtracted in the equation ensures $L_\mathbf{r} \leq C_{max} + P_2$ to avoid numerical overflows with C_{max} denoting the maximum DSI value possible.

$$\begin{aligned} L_\mathbf{r}(\mathbf{p}, d) =& C(\mathbf{p}, d) \\ &+ \min(L_\mathbf{r}(\mathbf{p} - \mathbf{r}, d), L_\mathbf{r}(\mathbf{p} - \mathbf{r}, d - 1) + P_1, \\ &\quad L_\mathbf{r}(\mathbf{p} - \mathbf{r}, d + 1) + P_1, \min_i L_\mathbf{r}(\mathbf{p} - \mathbf{r}, i) + P_2) \\ &- \min_k L_\mathbf{r}(\mathbf{p} - \mathbf{r}, k) \end{aligned} \tag{1}$$

The final cost per pixel and disparity is the sum of the $L_\mathbf{r}$ over all path directions

$$S(\mathbf{p}, d) = \sum_\mathbf{r} L_\mathbf{r}(\mathbf{p}, d) \tag{2}$$

where the number of directions often is fixed to 8 or 16. In the first case, only paths running strictly horizontally, vertically and diagonally must be considered. The next pixel on the line is computed by simply incrementing either the x or y coordinate, or both simultaneously. For sixteen orientations, two pixel steps in the major line direction are followed by one pixel step in the minor line direction. For n_{dir} directions, the maximum aggregated cost to be kept in memory will be $n_{dir}(C_{max} + P_2)$ which becomes relevant when the data type for the $S(\mathbf{p}, d)$ is to be chosen. If equation 1 is implemented reusing the previously computed $L_\mathbf{r}$ along the respective path, the time and space complexity of the described cost aggregation step for a DSI of w by h pixels and d disparities is $O(whd)$ for any constant value n_{dir}.

In a third stage, to eventually derive the pixel matches, the stereo shift d'_{min} with the minimum cost is taken from the set of $S(\mathbf{p}, d)$. Subpixel precision can be obtained by for instance fitting a curve through the costs at the disparities next to d'_{min} and recalculating the final d_{min} from the minimum function value. The last stage comprises the consistency check where erroneous stereo shifts due to mismatches or occlusions are eliminated. This often is achieved by reversing the role of the input images, re-matching the bitmaps and testing the new disparities against their counterparts from forward matching. Optionally, before being written to a bitmap, the surviving consistent disparity values can be filtered for example with a local median operator to eliminate outliers.

3. SGM WITH AN ARBITRARY DIRECTION COUNT

To support any positive number of directions in SGM, the simple discretization scheme for 8 and 16 path orientations must be abandoned. Instead, Bresenham's four-connected component line rasterizer is used to construct the paths for cost aggregation (Bresenham, 1965). This algorithm, which originates from computer graphics, supports arbitrary slopes and uses efficient integer additions and subtractions only. Cost aggregation now comprises at most two coordinated sweeps over the disparity space image as shown in figure 1.

3.1 Primary sweep

For the primary sweep performed first, the line shape of direction $\mathbf{r}$ is rasterized by running the Bresenham algorithm exactly once over the full horizontal or vertical extent of the xy plane of the DSI depending on the slope. Thus, path sampling must begin in one of the DSI corners at the minimum disparity, i.e. at $(\mathbf{p_0}, d_0)$. As a result, the longest sequence of linear segments, or runs, in which the horizontal or vertical coordinate remains constant is obtained for $\mathbf{r}$ (figure 2). The runs in general have a variable length and characterize the fast-changing major direction of the line to be discretized.

To sum up the cost along a particular SGM path, the run sequence archetype is followed starting at the DSI boundary at position $(\mathbf{p}, d) = (\mathbf{p_0}, d_0)$ according to the sweep scheme. For each run pixel, the cost $L_\mathbf{r}$ as of equation 1 is computed. The obtained value gets added to cell $(\mathbf{p}, d)$ of the aggregation volume S which has the same dimensions as the disparity space image. Position $\mathbf{p}$ is incremented or decremented by one in the major path direction. When the current run of the precalculated sequence is exhausted, the pixel position $\mathbf{p}$ will be incremented or decremented with respect to the minor path direction, and calculation will continue with the next run. If the DSI is eventually left, a new path of direction $\mathbf{r}$ next to the current one will be processed. The position $\mathbf{p}$ is reset to the DSI boundary and

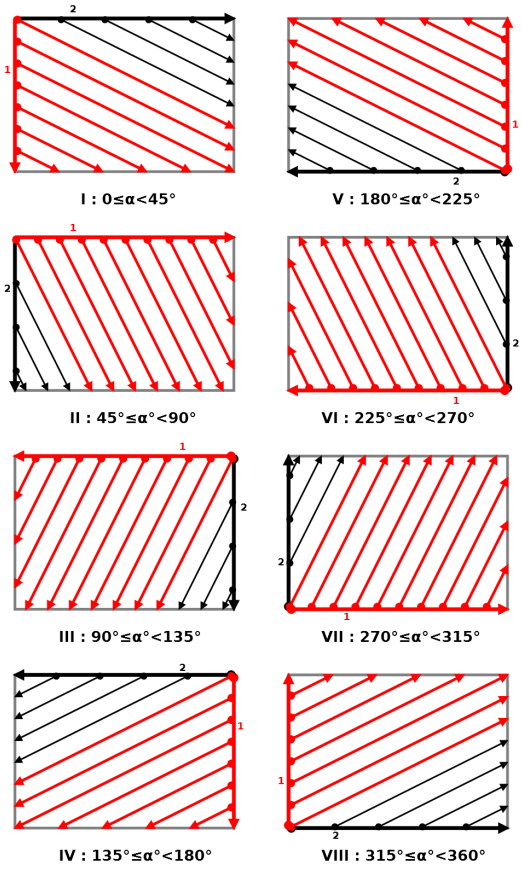


Figure 1. Primary (red) and secondary (black) sweeps over the disparity space image for paths with an arbitrary slope α

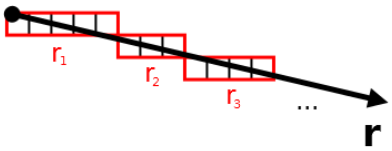


Figure 2. Runs r_1, r_2, r_3 (red) of the rasterized line $\mathbf{r}$ with a length of five, three and four pixels

altered by one pixel in the minor line direction, and the run sequence gets rewound to its start.

The primary sweep will be complete as soon as a new path of direction $\mathbf{r}$ begins outside the DSI. When this happens, at least 50% (diagonal path, square image) and at most 100% (horizontal or vertical path) of the matching cost for direction $\mathbf{r}$ have been aggregated. Because adjacent rasterized paths are guaranteed to be piece-wisely parallel and tightly aligned when the same run sequence gets repeatedly replayed as described, each DSI pixel and its counterpart in the aggregation volume are visited exactly once during the primary sweep. Therefore, redundant calculations are avoided. More importantly, no cost update will be omitted or performed multiple times for a particular $(\mathbf{p}, d)$. This ensures that the aggregation volume will remain unbiased when the minimum cost stereo disparities are chosen.

3.2 Secondary sweep

Except for perfectly horizontal and vertical path directions, a secondary sweep will be needed to aggregate the matching cost for the DSI pixels not visited during the primary sweep. Like in the first pass, the run sequence is followed to obtain the DSI pixel $\mathbf{p}$ for the calculation of $L_{\mathbf{r}}$. However, the start position of the k-th secondary path along the respective DSI boundary is computed as the accumulated length of the runs $r_1, ..., r_k$ of the rasterized line prototype. Also, the path gets stripped off its first k runs to be piecewisely adjacent to the paths of the primary sweep (see figure 3). Overlaps and gaps are avoided during cost aggregation between scanlines of the secondary sweep. The sweep scheme prevents collisions with the positions visited in the first pass over the disparity space image.

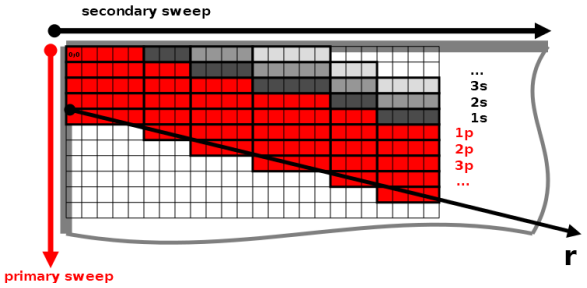


Figure 3. Truncation and shifted start of paths of the secondary sweep (shades of gray) of direction $\mathbf{r}$ ensures the proper alignment to the paths of the primary sweep (red)

4. IMPLEMENTATION AND OPTIMIZATION

As a proof of concept, the SGM algorithm has been reimplemented in platform-independent C++ based on run sequences for the cost aggregation stage. The software prototype takes two rectified RGB or graylevel TIFF input images with 8 or 16 bits per pixel and performs forward (left to right) and reverse (right to left) stereo matching. On success, this will generate a pair of floating-point TIFF bitmaps directly encoding the set of consistent disparity values. Optionally, the median filter, small segment removal and interpolation to fill up image areas with invalid disparities from neighbor pixels can be applied to the output in a postprocessing step.

Cost calculation utilizes the census function with a kernel of 7x7 pixels. The disparity space image and aggregation volume are stored as partially consecutive 16 bit unsigned integers. Assuming a penalty $P_2 = 100$ for depth discontinuities, the maximum SGM path direction count before an aggregation overflow occurs is calculated as $n_{max} = \lfloor (2^{16} - 1)/(7^2 + P_2) \rfloor = 439$ as of equation 2. Since in theory the runtime of the algorithm will increase linearly with the number of SGM path directions, the code has been applied optimizations to speed up all stages of the matching process. Aside from pointer arithmetic, this includes disparity gating, SIMD intrinsics and extensive multithreading.

4.1 Disparity gating

Like in tSGM, instead of running the matcher for a fixed predetermined range of stereo shifts, disparity gating dynamically restricts the range for the disparities d per DSI pixel $(\mathbf{p}, d)$ depending on the input image content. Therefore, the proposed SGM algorithm is run on recursively subsampled bitmaps built from the stereo pair. At the lowest resolution, the full image width is taken as the range for the disparities d producing a first estimate for the stereo shifts that occur for the pictured scene. To accommodate mismatches, as a variation to tSGM, a local histogram filter with a fixed support of 51x51 pixels is applied to the disparity maps. It sets the disparity bounds to the lower

and upper 16th percentile. The histogram can be quickly updated as described in (Faugeras et al., 1993). When the window is shifted horizontally or vertically over the disparity images, $O(max(f_w, f_h))$ time complexity is achieved for each move where f_w and f_h denote the filter dimensions.

The minimum and maximum disparities are subsequently passed up the matching hierarchy range-gating the stereo shifts for the image pair of the next higher resolution. Since the estimates tend to become more precise at upper levels, less disparities must be considered per pixel during cost aggregation along the SGM paths compared to a predefined constant interval. Also, the computed disparity range gets concentrated more accurately around the true value. This reduces matching ambiguities and improves the time efficiency of the algorithm. Runtime is further optimized with a limit on the number of SGM path orientations for the coarse tiers of the image pyramid while the maximum direction count will only be used for the full-resolution level of detail.

Congruently to the speedup, the amount of storage space for the DSI and aggregation volume will be greatly reduced when only the necessary values with respect to the disparity interval and reduced image dimensions are kept. Therefore, instead of a three-dimensional memory block accommodating the full resolution, the proposed implementation utilizes a single array of 16 bit unsigned integers sized to the sum over all per-pixel disparity ranges of the current SGM hierarchy level. Access to individual matching cost values is facilitated indirectly through an index structure. For each pixel $\mathbf{p}$, the index holds the minimum disparity considered d_0, the disparity count and the address of the first cost value assigned to $\mathbf{p}$, i.e. the offset to $(\mathbf{p}, d_0)$. Since the array is consecutive in memory, pointer arithmetic can be used to efficiently navigate adjacently stored costs, and a suitable alignment prepares the storage for vector instructions.

4.2 Multithreading

On a more technical level, the SGM implementation utilizes multiple execution threads for its submodules based on OpenMP (OpenMP Architecture Review Board, 2015). This will ideally lead to a speedup that is directly proportional to the number of available processor cores. For the cost computation, histogram filter, minimum disparity calculation and postprocessing stages, the involved local operators are run concurrently for each horizontal image scanline with static scheduling.

The cost aggregation loop over the set of paths of direction $\mathbf{r}$ along which the values $L_\mathbf{r}(\mathbf{p}, d)$ are summed up is multithreaded using dynamic scheduling. In contrast to the default scheduling policy of OpenMP using a fixed number of paths per thread, dynamic scheduling effectively assigns each path its own thread and starts processing a new scanline as soon as another path has been entirely covered. Thus, with the lines of varying length in the secondary SGM sweep, all CPU cores remain fully loaded for a maximum of efficiency. Because there are no overlaps between the paths, there will be no race condition when the costs $L_\mathbf{r}$ are stored concurrently to the cells $(\mathbf{p}, d)$ of the aggregation volume. Therefore, time-consuming thread synchronization to ensure exclusive data writes becomes unnecessary, and the absence of any memory locks will positively contribute to the scalability of the SGM implementation to a high CPU core count. As a drawback, the use of multithreading introduces duplicate data structures and hence a small memory overhead. For example, the current and previously aggregated matching costs along the path $L_\mathbf{r}(\mathbf{p}, d)$ and $L_\mathbf{r}(\mathbf{p} - \mathbf{r}, d)$ now must be kept and updated separately for each thread.

4.3 SIMD instructions

On the hardware level, census calculation and cost aggregation of the SGM implementation are further parallelized using vector or SIMD (single instruction multiple data) commands supported by current processors. In these stages, instead of looping over single data elements, blocks of input image pixels and consecutive cells of the DSI and aggregation volume are uniformly processed by a single instruction in few clock cycles. The block size which affects the achievable speedup depends on the actual CPU architecture. For the dominating Intel and ARM processors, 8 (Intel SSE2/3/4, ARM NEON), 16 (Intel AVX/AVX2) or 32 (Intel AVX512BW) unsigned integers of 16 bits as stored by both three-dimensional data structures can be processed at once (Intel Corporation, 2019)(ARM Limited, 2019).

For the census matching cost, the crucial part is to derive a bitmask that indicates which elements of the neighborhood have an intensity greater or equal to the input image pixel under the filter mask center. Using SSE2 instructions and retaining a symmetric kernel, the comparison can be performed simultaneously for a row of seven pixels as shown in figure 4, and consequently, the filter size is set to 7x7. In the C++ code, the SIMD instructions are prefixed with _mm_ and take either 128 bit SSE registers or memory pointers aligned to 16 bytes as arguments.

```
uint64_t census7_SSE2(CU16Img& f_img, const CP3D& f_pos) {

  // get pointer to upper left pixel under kernel
  CU16Img::Iterator it=f_img.getIterator(f_pos.m_x-3,
  f_pos.m_y-3, 0);
  // reset bitmask
  uint64_t bitmask=0;
  // SSE: load center pixel into xmm regs, duplicate 8x
  const __m128i centerInt=_mm_set1_epi16(f_img.getPixel(
  f_pos));

  for (data::pos_type ky=0; ky<7; ++ky) {
    // SSE: load 8 consecutive uint16_t's, left-shift
    // value to have symmetric 7-mask
    __m128i rowInt=_mm_slli_si128(_mm_loadu_si128(
    reinterpret_cast<__m128i*>(it)), 2);

    // SSE: row intensities >= center intensity
    __m128i cmp=_mm_cmple_epu16(centerInt, rowInt);
    // SSE: macro to compress MSBs of cmp into 8-bit mask
    (reinterpret_cast<uint8_t*>(&bitmask))[ky]=
    _mm_movemask_epi16eo(cmp);
    // next image row
    f_img.incY(it);
  } // ky

  return bitmask;
}
```

Figure 4. C++ code with SSE2 intrinsics for the census bitmask

For SGM cost aggregation, the loop over the disparities contained in equation 1 is rewritten. Here blocks of consecutive stereo shifts $d_n, ..., d_{n+b-1}$ with b being the SIMD block size are simultaneously processed. In the sample code for the SSE2 instruction set, the calculation of the $L_\mathbf{r}(\mathbf{p}, d)$ comprises multiple vector additions, the subtraction of the $\min_k L_\mathbf{r}(\mathbf{p} - \mathbf{r}, k)$ kept in a local variable and the 4-minimum operator (figure 5). This operator, which is not directly available, must be emulated with three calls to the binary minimum intrinsic.

5. PERFORMANCE ANALYSIS

Performance analysis of the proposed algorithm focuses on the influence of the number of path directions on the quality of the resulting disparity maps and the scalability of lock-free pathwise multithreading on modern multi-core processors. For this

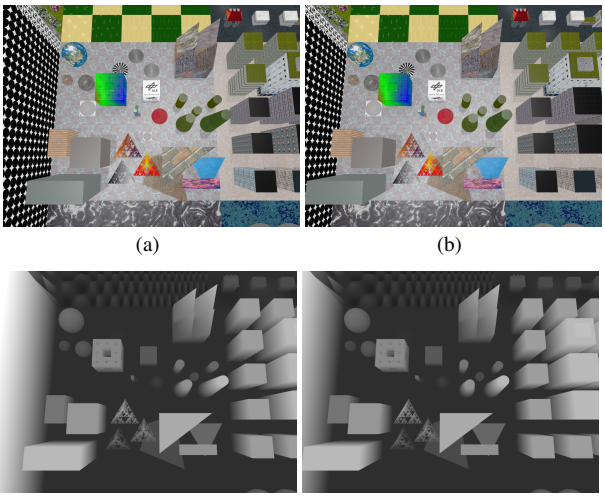
```
// cost at previous pixel at disparity d (no d change)
__m128i prevCostD=_mm_loadu_si128(reinterpret_cast<
__m128i*>(f_prevCostOnPath_p+arrD));

// cost at previous pixel at d-1 w/ penalty P1
__m128i prevCostD_m_1=_mm_loadu_si128(reinterpret_cast<
__m128i*>(f_prevCostOnPath_p+arrD-1));
prevCostD_m_1=_mm_add_epi16(prevCostD_m_1, penalty1_x8);

// cost at previous pixel at d+1 w/ penalty P1
__m128i prevCostD_p_1=_mm_loadu_si128(reinterpret_cast<
__m128i*>(f_prevCostOnPath_p+arrD+1));
prevCostD_p_1=_mm_add_epi16(prevCostD_p_1, penalty1_x8);

// min cost over all d at previous pixel w/ penalty P2
__m128i minCostAtD=_mm_add_epi16(prevCostMin_x8,
penalty2_x8);

// compute minimum cost at previous position, i.e.
// take the minimum of the four values above
minCostAtD=_mm_min_epu16(prevCostD, minCostAtD);
minCostAtD=_mm_min_epu16(prevCostD_m_1, minCostAtD);
minCostAtD=_mm_min_epu16(prevCostD_p_1, minCostAtD);

// add Census cost of current pixel from DSI to
// minimum cost at previous pixel calculated above
minCostAtD=_mm_add_epi16(curCostAtD, minCostAtD);

// subtract the minimum cost of path up to previous pixel
// to keep the aggregation sum low without changing the
// path itself
minCostAtD=_mm_sub_epi16(minCostAtD, prevCostMin_x8);

// store cost for path up to current pixel in
// aggregation volume
aggrCostAtD=_mm_add_epi16(aggrCostAtD, minCostAtD);
_mm_storeu_si128(reinterpret_cast<__m128i*>(aggrCostIt),
aggrCostAtD);
```

Figure 5. C++ snippet with SSE2 intrinsics for cost aggregation

purpose, the optimized SGM implementation is passed a set of RGB stereo images of a synthetic 3D scene (Frommholz, 2019). The input bitmaps are 8192 by 6144 pixels (50.3 megapixels) each with 8 bits per color channel. They cover an area of roughly 500 by 375 length units at a ground sampling distance (GSD) of about 0.07 length units. The images have been rendered by a raytracing software where the virtual pinhole cameras satisfy the stereo normal case. Except for inherent aliasing artifacts suppressed by excessive oversampling with 81 rays per pixel, the bitmaps are free of distortions and noise. Congruently to the stereo pair, the raytracer also provides depth ground truth that comprises the z coordinates of the intersections when the emitted rays hit the scene objects. If the disparity maps of SGM are triangulated keeping only the z coordinates, both the actual and ground truth z images will become directly comparable, and as long as the z axis and viewing vector approximately coincide, matching errors will get reflected adequately. Also, by working on the z coordinates instead of raw stereo shifts, there will be an immediate connection to the height values of the pictured synthetic 3D geometry which simplifies manual point sampling. Figure 6 depicts the reference data set.

5.1 Matching quality

For the matching quality, the SGM implementation is run on the synthetic input images with a steadily increasing number of path directions n_{dir}. Aggregation for most passes is carried out symmetrically along the $k = 0, ..., n_{dir} - 1$ lines with inclinations of $k \cdot 360^\circ / n_{dir}$. However, the initial path angle of runs 8/11 and 16/7 is 11° and 7° respectively. The prime offsets have been chosen to reduce the chance of a correlation between the direction of the scanlines and the predominant orientation of the imaged scene objects. Penalties remain fixed at $P_1 = 90$



(a) (b) (c) (d)

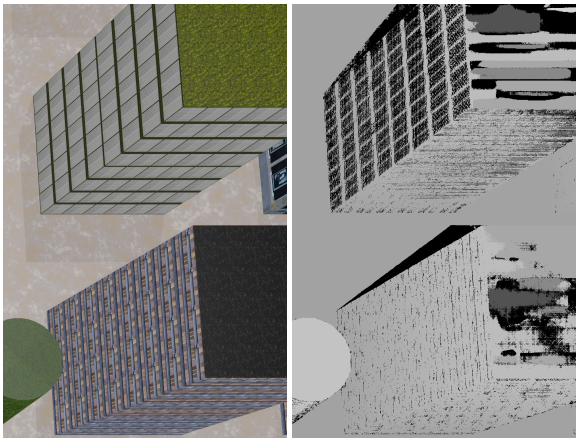
Figure 6. Synthetic reference data (a) left image, (b) right image, (c) left z bitmap, (d) right z bitmap

and $P_2 = 100$ for slowly and rapidly changing depths. The lower and upper disparity bounds estimated on all levels of the matching pyramid are widened by 20 pixels to absorb errors. Output stereo shifts for the left-to-right and right-to-left SGM passes are checked for consistency and filtered allowing a maximum deviation of one pixel. Any other postprocessing or interpolation remains disabled. The produced disparity maps are transformed into z images of which the left-to-right bitmap is compared to the ground truth pixel by pixel. Statistics are compiled for the entire scene and two subsets comprising detailed objects (i.e. Sierpinsky tetrahedra of 4, 5 and 6 iterations) and a homogeneously textured cube. These special cases tend to be prone to matching errors in practice and hence are of particular interest.

Matching quality will be quantified in terms of coverage and accuracy. For this purpose, the number of successfully reconstructed points n_{valid}, the median height deviation from the ground truth $\tilde{\Delta h}$ and the aggregated count of pixels with an absolute height error less or equal than 0.2 length units $n_{\Delta h \leq 0.2}$ are determined. The threshold has been empirically chosen as roughly three times the ground sampling distance. Table 1 lists the statistics for the whole scene where the row labeled "gt" has the reference values. It shows that the number of pixels with a valid correspondence basically stagnates between 85.5% and 85.95% when the path count increases. The median distance error to the ground truth first falls quickly, decreases by around 11% from $n_{dir} = 8$ to $n_{dir} = 96$ and stabilizes for the upper zone of directions tested. Similarly, over the medium range of orientations, the number of accurate pixels with a height deviation below three times the GSD grows by about 2%.

However, for $n_{dir} = 4, 8, 12$ and 16, spikes occur in the number of valid pixels. For these cases, the median height error remains in line, and the values $n_{\Delta h \leq 0.2}$ mostly increase disproportionately compared to the neighboring direction counts. Hence, the new consistent disparities likely have contributed to the accurate pixel class, and there must have been a move of already successfully matched stereo shifts towards smaller distance deviations. A review of the produced z images for the respective configurations indicates that the cause for this effect is local minima in the disparity space image which specifically occur perpendicularly to the direction of perspective distortions, e.g. on wall surfaces of high-rise buildings (figure 7). Due to

their low cost, these regions will be followed by aggregation paths that have a similar orientation, which is the correct solution according to equation 1. For dense and accurate surface reconstruction, it may therefore be beneficial to analyze the DSI three-dimensionally for linear structures and run SGM with a set of paths coinciding with their orientation.



(a) (b)

Figure 7. Dense reconstruction of wall surfaces with perspective distortion using SGM with eight orientations (a) RGB image, (b) height image with streaks of valid pixels

In congruence to this observation, when the direction count remains constant but the start angle of the paths is altered breaking the alignment to the DSI minima, the number of valid pixels significantly drops. Remarkably, for 8 directions/11° and 16 directions/7° respectively, using oddly oriented aggregation scanlines reduces the median distance error to the ground truth to values obtained for 24 and 32 directions. The number of accurate pixels also drops by 0.73% ($n_{dir} = 8/0°$ to $n_{dir} = 8/11°$) and 1.02% ($n_{dir} = 16/0°$ to $n_{dir} = 16/7°$) which, for eight directions, is less than the decrease in consistent disparities. Thus, if the distance error is to be optimized with SGM in general, it may be advantageous to start cost aggregation with an angular offset and choose a low direction count for speed.

For the high-detail subset of the synthetic stereo pair, table 2 contains the coverage and accuracies. The number of successfully matched pixels is about 3-4% below the values for the full scene. Except for the settings with an angular offset, it grows slowly with a rising path direction count by 1% from $n_{dir} = 8$ to $n_{dir} = 256$. The median distance deviation to the ground truth is greater by roughly 0.02 length units. Introducing angular offsets for the paths decreases the number of valid pixels and lowers the height error to a value that even outperforms the configuration with 256 directions. Hence, for detailed objects to be reconstructed, either a substantial number of orientations or oddly rotated paths is the strategy of choice for SGM. However, on the examined Sierpinsky tetrahedra, if the number of directions is increased and the final aggregated cost $S(\mathbf{p}, d)$ stabilizes, the distribution of valid pixels will also change. Isolated disparity pixels and small regions vanish in favor of larger consolidated segments. This may be disadvantageous for a subsequent interpolation step because supporting points inside the nodata areas will get lost. Figure 8 illustrates selected results.

For the homogeneously textured cube, the statistics are given in table 3. Due to the ambiguous input pixels, the number of points that survived the consistency check of the matcher is only around two thirds of the reference. It slowly rises by 3% when

n_{dir}	n_{valid}	n_{valid} (%gt)	$\tilde{\Delta h}$	$n_{\Delta h \leq 0.2}$	$n_{\Delta h \leq 0.2}$ (%gt)
2	41089737	81.64	0.1757	23014714	45.73
3	42929427	85.29	0.1450	28233346	56.09
4	43791793	87.01	0.1272	32563113	64.70
5	42891974	85.22	0.1196	32568512	64.71
7	42895249	85.23	0.1076	34648613	68.84
8	43414535	86.26	0.1069	36053523	71.63
8/11	42755256	84.95	0.0965	35687637	70.90
9	43002263	85.44	0.1016	35683318	70.90
10	43062698	85.56	0.1059	35630609	70.79
11	43034508	85.50	0.0988	36113585	71.75
12	43289729	86.01	0.1030	36715790	72.95
16	43226879	85.88	0.0990	36812149	73.14
16/7	42980344	85.39	0.0954	36301323	72.12
24	43205948	85.84	0.0964	36901649	73.32
32	43210518	85.85	0.0954	36925649	73.36
48	43224382	85.88	0.0948	36928646	73.37
64	43233481	85.90	0.0947	36932487	73.38
96	43244271	85.92	0.0946	36937123	73.39
128	43249586	85.93	0.0945	36940793	73.39
256	43257902	85.95	0.0945	36947720	73.41
384	43260465	85.95	0.0945	36950916	73.41
gt	50331648	100	0	50331648	100

Table 1. Coverage and accuracy for the synthetic stereo pair processed with different numbers of SGM path directions

n_{dir}	n_{valid}	n_{valid} (%gt)	$\tilde{\Delta h}$	$n_{\Delta h \leq 0.2}$	$n_{\Delta h \leq 0.2}$ (%gt)
4	1696562	80.79	0.1565	1098606	52.31
8	1715489	81.69	0.1268	1326036	63.14
8/11	1709570	81.41	0.1180	1367170	65.10
16	1727627	82.27	0.1211	1376957	65.57
16/7	1726522	82.22	0.1187	1383644	65.89
24	1731993	82.48	0.1195	1388886	66.14
32	1733444	82.55	0.1192	1389931	66.19
64	1735478	82.64	0.1191	1392539	66.31
128	1736358	82.68	0.1191	1393706	66.37
256	1736889	82.71	0.1191	1394281	66.39
gt	2100000	100	0	2100000	100

Table 2. Coverage and accuracy for a scene subset with detailed objects processed with different numbers of SGM path directions

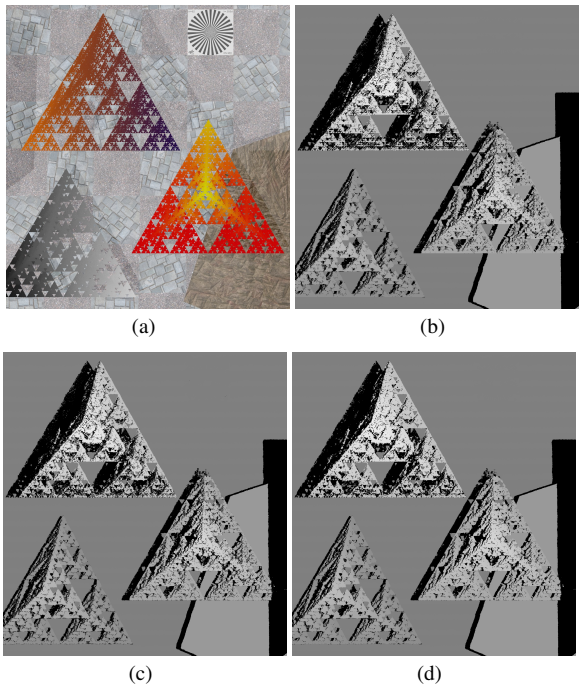
the SGM path direction count gets ramped up with slight drops on the angular offsets. Starting with an initial prime slope does not reduce the median height error which is approximately 1% higher than for the detailed objects and basically remains static beyond $n_{dir} = 32$. The increasing number of new valid samples is accompanied by a growth in accurate pixels whose number rises by roughly 6% over the entire direction range shown and by nearly 3% from $n_{dir} = 8$. Hence, although the absolute distance deviation can hardly be lowered by a vast number of path directions in semi-global matching, the point density will benefit when little information is present in the input data. In addition, when a high orientation count is selected, the characteristic streaking artifacts of SGM that were addressed by the more global matching approach will diffuse into an undirected point pattern as depicted in figure 9.

5.2 Runtime and scalability

The impact of a high number of path directions on the runtime of the optimized SGM implementation is evaluated for both the

n_{dir}	n_{valid}	n_{valid} (%gt)	$\tilde{\Delta h}$	$n_{\Delta h \leq 0.2}$	$n_{\Delta h \leq 0.2}$ (%gt)
4	253275	67.54	0.1426	173561	46.28
8	255508	68.14	0.1307	185310	49.42
8/11	247662	66.04	0.1320	176980	47.19
16	258112	68.83	0.1283	189932	50.65
16/7	256933	68.52	0.1295	186812	49.82
24	259935	69.32	0.1277	191561	51.08
32	261237	69.66	0.1278	192638	51.37
64	263520	70.27	0.1276	194786	51.94
128	264350	70.49	0.1275	195643	52.17
256	264812	70.62	0.1274	195909	52.24
gt	375000	100	0	375000	100

Table 3. Coverage and accuracy for a low-texture cube processed with different numbers of SGM path directions

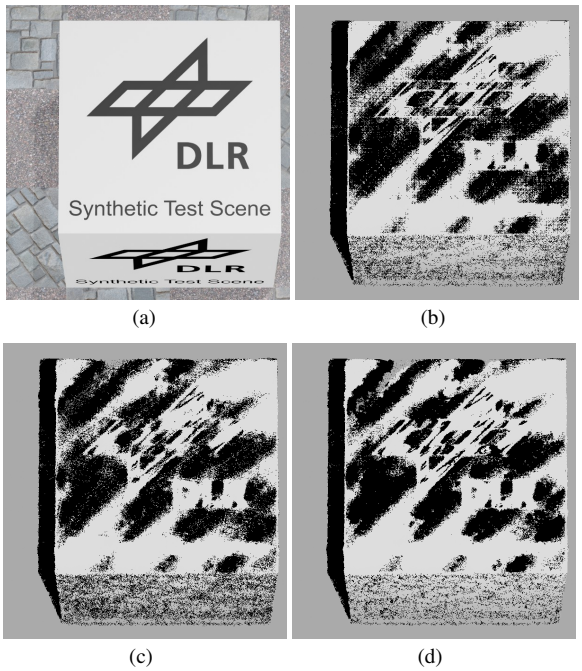


(a) (b) (c) (d)

Figure 8. SGM on the detailed scene subset (a) RGB image, (b) height image for 8 directions, (c) 8 directions offset by 11°, (d) 256 directions

Intel x86-64 and ARM v7 platforms representing server-class hardware and mobile devices. For x86-64, the test system is built around a 2019 AMD EPYC 7402p processor with 24/48 cores running at 2.8 to 3.35 GHz and 256 GiB 8-channel DDR4 RAM. On the ARM side, runtime is measured on a Hardkernel Odroid-U3 single board computer (SBC) from the year 2014. It comes with a 32-bit Cortex A9 quad core CPU nominally running at 1.7 GHz and 2 GiB LP-DDR2 RAM. On both systems, Linux with 5-series kernels is installed as the operating system.

Since the server hardware offers plenty of resources, measurements are on the full-scale synthetic images that were used for matching quality analysis. The SGM implementation is compiled to optimized binaries without platform-specific tuning. For x86-64, the executables have been built from the platform-independent C++ code and the C++ code manually enhanced with SSE2/SSE4.2 and AVX2 intrinsics for the cost calculation and aggregation steps as outlined previously. The software utilizes 24 threads of execution, i.e. one per physical processor core. Due to the lack of memory, the ARM SBC is given a re-



(a) (b) (c) (d)

Figure 9. SGM on a low-texture cube (a) RGB image, (b) height image for 8 directions, (c) 8 directions offset by 11°, (d) 256 directions

sampled version of the stereo pair of 2048 by 1536 pixels (3.15 megapixels). Two versions of SGM with and without NEON SIMD intrinsics have been executed using four threads. Table 4 shows the overall runtime in seconds including image I/O.

n_{dir}	x86-64 C++	x86-64 SSE2/4	x86-64 AVX2	ARM C++	ARM NEON
1	54	64	73	82	124
2	60	69	79	92	138
4	66	71	81	97	135
8	74	73	83	116	148
16	91	82	92	146	168
24	109	90	100	175	188
32	128	99	109	205	209
64	200	131	144	321	289
128	346	198	214	556	449
256	636	331	351	1023	772
384	925	462	491	1491	1098

Table 4. Runtime in seconds for the SGM implementation on Intel x86-64 and ARM v7 platforms

In practice, runtime grows nonlinearly with the direction count. The increase from 8 to 16 orientations is 12% on x86-64 with SSE intrinsics and 26% on ARM using plain C++ code. Quadrupling the number of directions from 8 to 32 extends the runtime by 36% and 77% respectively. Moreover, processing speed does not always benefit from SIMD optimization. On both platforms, for low to medium numbers of path directions, the binaries built from plain C++ code outperform their manually vectorized counterparts. This indicates that current C++ compilers already generate very efficient machine code. Also, when a sparse set of aggregation paths predominantly runs through DSI pixels that have been assigned narrow disparity ranges, the data transfer overhead from and to the dedicated SIMD registers may outweigh the efficiency gain from simultaneously processing blocks of stereo shifts. When n_{dir} exceeds 16 (Intel) and 32 (ARM), SIMD becomes clearly faster than unvectorized code.

On the EPYC CPU, SSE commands beat the wider and more recent AVX2 instructions. This may be due to limitations of its floating-point/vector execution pipelines (Fog, 2019).

To assess the scalability of path-wise parallelization for the cost aggregation stage, the SGM implementation is executed with a constant number of 16 scanline directions but an increasing number of processing threads. Table 5 summarizes the runtime in seconds for the fastest binaries for this configuration.

threads	1	2	4	8	16	24	32	48
x86-64 SSE2/4	997	575	340	199	102	77	71	66
ARM C++	557	281	143	-	-	-	-	-

Table 5. Runtime in seconds for different thread counts

On both platforms, doubling the number of processing threads lowers the runtime by a factor of 1.69 to 1.98 until the physical CPU cores are exhausted. Hence, the SGM implementation with path-wise multithreading almost scales linearly on the test machines. There is a marginal speedup when the logical cores are included and competition for memory bandwidth and arithmetic units of the processor starts.

6. CONCLUSION

This paper has outlined an implementation of semi-global matching that allows an arbitrary number of directions for cost aggregation. For each direction, a template sequence of line runs is constructed utilizing Bresenham's rasterizer. The runs are followed in at most two coordinated sweeps over the image to update the costs for each pixel and disparity. Since overlaps cannot occur, each path can be traced in its own thread without write locks. Runtime is further minimized through disparity gating and the use of SIMD instructions. Tests on synthetic stereo images with a high number of aggregation directions have revealed an improved point coverage and reduced height error. Future work will explore how the orientation count and path slopes can be dynamically adjusted to the image content and initial SGM cost volume in order to minimize the deviation and optimize the data throughput on real-world imagery. For mobile devices with limited CPU resources, a GPU-based implementation of the proposed algorithm is to be evaluated.

REFERENCES

ARM Limited, 2019. ARM C language extensions documentation. https://developer.arm.com/architectures/system-architectures/software-standards/acle (10 January 2020).

Boykov, Y., Veksler, O., Zabih, R., 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11), 1222-1239.

Bresenham, J. E., 1965. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1), 25–30.

Cox, I. J., Hingorani, S. L., Rao, S., Maggs, B. M., 1996. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63, 542-567.

Facciolo, G., de Franchis, C., Meinhardt, E., 2015. MGM: A significantly more global matching for stereovision. *Proceedings of the British Machine Vision Conference*, BMVA Press, 90.1–90.12.

Faugeras, O., Viéville, T., Theron, E., Vuillemin, J., Hotz, B., Zhang, Z., Moll, L., Bertin, P., Mathieu, H., Fua, P., Berry, G., Proy, C., 1993. Real-time correlation-based stereo : Algorithm, implementations and applications. Research Report RR-2013, INRIA.

Fog, A., 2019. The microarchitecture of Intel, AMD and VIA CPUs. https://www.agner.org/optimize/microarchitecture.pdf (31 January 2020).

Frommholz, D., 2019. A synthetic 3D scene for the validation of photogrammetric algorithms. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W13, 1221–1228.

Fusiello, A., Trucco, E., Verri, A., 2000. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1), 16–22.

Hirschmüller, H., 2008. Stereo processing by semi-global matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2), 328–341.

Intel Corporation, 2019. Intel 64 and IA-32 architectures software developer's manual - Combined volumes. https://software.intel.com/en-us/articles/intel-sdm (14 January 2020).

Karkalou, E., Stentoumis, C., Karras, G., 2017. Semi-global matching with self-adjusting penalties. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W3, 353–360.

Michael, M., Salmen, J., Stallkamp, J., Schlipsing, M., 2013. Real-time stereo vision: Optimizing semi-global matching. *2013 IEEE Intelligent Vehicles Symposium Proceedings*, 1197–1202.

Miclea, V., Nedevschi, S., 2017. Semantic segmentation-based stereo reconstruction with statistically improved long range accuracy. *2017 IEEE Intelligent Vehicles Symposium Proceedings*, 1795–1802.

nFrames GmbH, 2014. Parameters of the SGM module. ftp://ftp.ifp.uni-stuttgart.de/sure_public/SgmParams.pdf (28 January 2020).

OpenMP Architecture Review Board, 2015. OpenMP application programming interface version 4.5. https://www.openmp.org/specifications (12 December 2019).

Rothermel, M., 2016. Development of a SGM-based multi-view reconstruction framework for aerial imagery. PhD thesis, University of Stuttgart.

Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., Westling, P., 2014. High-resolution stereo datasets with subpixel-accurate ground truth. *German Conference on Pattern Recognition Proceedings*, 8753, 31–42.

Seki, A., Pollefeys, M., 2017. SGM-Nets: Semi-global matching with neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, IEEE, Piscataway, NJ, 6640 – 6649.

Zabih, R., Woodfill, J., 1994. Non-parametric local transforms for computing visual correspondence. *Proceedings of the Third European Conference on Computer Vision*, II, Springer-Verlag, Berlin, Heidelberg, 151–158.