J Real-Time Image Proc
DOI 10.1007/s11554-012-0313-2

SURVEY PAPER

Review of stereo vision algorithms and their suitability for resource-limited systems

Beau Tippetts · Dah Jye Lee · Kirt Lillywhite · James Archibald

Received: 20 September 2012 / Accepted: 12 December 2012

Abstract A significant amount of research in the field of stereo vision has been published in the past decade. Considerable progress has been made in improving accuracy of results as well as achieving real-time performance in obtaining those results. This work provides a comprehensive review of stereo vision algorithms with specific emphasis on real-time performance to identify those suitable for resource-limited systems. An attempt has been made to compile and present accuracy and runtime performance data for all stereo vision algorithms developed in the past decade. Algorithms are grouped into three categories: (1) those that have published results of real-time or near real-time performance on standard processors, (2) those that have real-time performance on specialized hardware (i.e. GPU, FPGA, DSP, ASIC), and (3) those that have not been shown to obtain near real-time performance. This review is intended to aid those seeking algorithms suitable for real-time implementation on resource-limited systems, and to encourage further research and development of the same by providing a snapshot of the status quo.

1 Introduction

The field of stereo vision has received a great deal of attention over the past decade. New algorithms have been developed, and existing algorithms have been augmented and tuned in efforts to both produce more accurate results and obtain them faster. The nature of the stereo vision problem makes these two endeavors non-trivial. The accuracy of results is affected by missing information such as that caused by occlusions, slanted surfaces, and other issues relating to extracting information about three dimensions from two dimensional images. Textureless regions and non-Lambertian surfaces along with the difficulties of perfectly calibrating cameras are also cited by Zitnick and Kang [165] as reasons why stereo vision is still considered an unresolved problem. The number of pixels that each image contains increases the number of calculations required to match it with any number of possible matches, making the correspondence problem a computationally complex one that severely limits the speed at which one can obtain results. Most of the time, accuracy and speed are pitted against each other, making it more difficult to obtain more of both at the same time. In any given instance of the stereo vision problem, various questions arise; is one of these two attributes more desirable? How can the trade-off between the two be minimized? What options already exist, and how do they compare to each other?

In most circumstances, determining an acceptable trade-off between speed and accuracy is dependent upon the target application. Although many applications have existed for some time (e.g. Desouza and Kak [32], Bruch et al. [18] and Howard [55]), more and more applications are being developed that could benefit from real-time three-dimensional information. Image sensors and processing hardware are becoming more prevalent, especially because they are available as light-weight, low-power, passive devices. Basic low-quality image sensors can be found on systems from cell phones to entertainment game consoles

B. Tippetts (✉) · D. J. Lee · K. Lillywhite · J. Archibald
Brigham Young University, Provo, USA
e-mail: beau.tippetts@byu.edu

D. J. Lee
e-mail: djlee@byu.edu

K. Lillywhite
e-mail: kirt.lillywhite@gmail.com

J. Archibald
e-mail: james_archibald@byu.edu

to security systems to high-tech micro unmanned aerial vehicles. Each of these systems may have limited computational resources for several possible reasons, including constraints on weight, size, power, and cost, or perhaps a requirement that the bulk of computing resources be dedicated to a different primary task.

For many of these systems, extreme computational complexity poses a problem because of their resource limitations. When 3D information is available for resource-limited systems, tasks such as obstacle avoidance, pose identification, and landscape mapping could be implemented to realize applications like hands-free human control, autonomous vehicle control, augmented video surveillance, threat analysis, handicap assistance, and a host of other applications. Some may be satisfied simply by waiting for advancements in processing hardware technologies to make this kind of computational power available for systems with constrained resources. It can be shown, however, that algorithmic improvements can result in far greater increases in speed than result from recent advances in hardware technology.

Most recently, substantial advances have been made in increasing the accuracy of stereo vision results. In Scharstein and Szeliski [117] a method and infrastructure were provided for quantitative evaluation of the accuracy of dense stereo vision algorithms that has become a standard. Their evaluation includes a comparison of root mean squared error between the disparity map from the algorithm and ground truth. This offers a precise single value that is used to rank the quality of results from different stereo vision algorithms, that can be found at Scharstein [116]. This type of quantitative evaluation helps motivate researchers to develop more accurate stereo vision algorithms and it helps them know when they have done so. Although similar tools for such precise quantitative comparisons of algorithm runtimes are currently not available, the successful development of many applications depend on such comparisons. The data that is available is presented here to aid in understanding how existing stereo vision algorithms compare in terms of both accuracy and speed.

This work provides a review of published stereo vision algorithms, categorizing them according to their suitability for real-time implementation on resource-limited platforms. It also addresses the trade-off in accuracy that typically must be made to achieve real-time performance. First, other works providing reviews or other comparisons of stereo vision algorithms are summarized in Sect. 2. Details of how accuracy and runtime performance of the algorithms are compared for this work are given in Sect. 3. The next two sections contain summaries of existing algorithms; those that have not been shown to achieve 1 Hz performance rates on current standard CPUs are described in Sect. 4, while those with higher levels of performance are detailed in Sect. 7. Aside from a review of current stereo vision algorithms, the bulk of the contributions of this paper are encapsulated in Figs. 10, 11, and 12 in Sect. 7. Some algorithms have been implemented on higher resource devices such as GPUs, ASICs, FPGAs, etc. to obtain real-time performance and are presented in Sects. 5 and 6. We conclude with a summary of our review in Sect. 8.

This paper contains references to 184 algorithms and 166 publications. In order to clearly distinguish between multiple algorithms in a single publication and to maintain clarity in the graphs, figures, and tables each algorithm has been given a custom label. The custom label for each algorithm along with the citation of the publication in which it was presented is listed in Table 1.

2 Background

There are several informative publications that review and compare available stereo vision algorithms, but none of them discuss suitability for real-time performance on resource-limited systems. The review by Scharstein and Szeliski [117] is a comprehensive treatise on stereo vision algorithms as of 2002, while other publications focus on specific types or key components of stereo vision algorithms.

Scharstein and Szeliski [117] discuss different types of algorithms and provide an accuracy metric to compare and evaluate these stereo vision algorithms, with an online tool still available to submit newly developed algorithms (Scharstein [116]). Brown et al. [17] reviewed advances in stereo vision regarding correspondence methods, occlusion handling methods, and real-time implementations. They discuss how the advancement of technology such as higher processor clock speeds and SIMD instructions have allowed implementations of existing stereo algorithms on desktop computers to reach real-time performance. A comparison of matching costs for local, semi-global, and global stereo algorithms is given in Hirschmuller and Scharstein [52]. Gong et al. [44] compare six different cost aggregation approaches on a GPU. Tombari et al. [131] compared and evaluated multiple methods of variable support cost aggregation for stereo vision. In one of the most recent reviews available, Nalpantidis et al. [99] discuss some of the attributes of new local and global algorithms developed since Scharstein and Szeliski [117]. Szeliski et al. [126] more recently compared global energy minimization techniques. Humenberger et al. [58] compared several existing stereo algorithms to their own census-based algorithm for real-time implementations. They cite the growing fields of mobile robotics and embedded autonomous systems as motivation to develop stereo

Table 1 Every algorithm cited in this work uses the labels included in this table

Algorithm	Reference	Label	Algorithm	Reference	Label
SAD-IGMCT	Ambrosch and Kubinger [1]	aa	ADCensus	Mei et al. [87]	me
OpenCVSAD	Ambrosch et al. [2]	ab	ADCensus (GPU)	Mei et al. [87]	mf
SAD(FPGA)	Ambrosch et al. [2]	ac	HistoAggr	Min et al. [89]	mg
SparseCensConf	Ambrosch et al. [3]	ad	Trinocular	Mingxiang and Yunde [90]	mh
BilateralSAD	Ansar et al. [4]	ae	SADleft-right	Miyajima and Maruyama [91]	mi
BP+DirectedDiff	Banno and Ikeuchi [5]	ba	H-Cut	Miyazaki et al. [92]	mj
VarMSOH	Ben-Ari and Sochen [6]	bb	MVSegBP	Montserrat et al. [93]	mk
InteriorPtLP	Bhusnurmath and Taylor [7]	bc	TensorVoting	Mordohai and Medioni [94]	ml
Segm + visib	Bleyer and Gelautz [8]	bd	MeanSAD	Muhlmann et al. [95]	mm
WarpMat	Bleyer et al. [9]	be	CurveletSupWgt	Mukherjee et al. [96]	mn
SurfaceStereo	Bleyer et al. [10]	bf	BioPsyASW	Nalpantidis and Gasteratos [97]	na
ObjectStereo	Bleyer et al. [12]	bg	LCDM + AdpWgt	Nalpantidis and Gasteratos [98]	nb
PatchMatch	Bleyer et al. [12]	bh	MeanCensus	Naoulou et al. [100]	nc
DP	Bobick and Intille [13]	bi	ConnectSlant	Ogale and Aloimonos [101]	oa
GC (Exp)	Boykov et al. [14]	bj	Infection	Olague et al. [102]	ob
CostRelaxAW	Brockers [15]	bk	MultiResGC	Papadakis and Caselles [104]	pa
CostRelax	Brockers et al. [16]	bl	Trellis	Park and Jeong [105]	pb
SimAnneal	Cassisa [19]	ca	BP	Perez et al. [107]	pc
FLTG-ICM	Cassisa [19]	cb	7 × 7SAD	Perri et al. [108]	pd
FLTG-DDE	Cassisa [19]	cc	ConvexTV	Pock et al. [109]	pe
4 × 5JigsawSAD	Chang et al. [22]	cd	ConvexTV(GPU)	Pock et al. [109]	pf
MCADSW	Chang et al. [23]	ce	IterAdpWgt	Psota et al. [110]	pg
RT-ColorAW	Chang et al. [24]	cf	CostFilter	Rhemann et al. [111]	ra
SemiGlobGC	Chen et al. [25]	cg	DCBGrid	Richardt et al. [112]	rb
RegionTreeDP	Lei et al. [76]	ch	FasttrackDPML	Sabihuddin and MacLean [113]	sa
ConnectCons	Cornells and Van Gool [26]	ci	OptimizedDP	Salmen et al. [114]	sb
RTSVP	Cuadrado et al. [27]	cj	GC	Scharstein and Szeliski [117]	sc
RecursiveAdaptive	Chan et al. [21]	ck	SSD+MF	Scharstein and Szeliski [117]	sd
GradAdpWgt	De-Maeztu et al. [29]	da	DP	Scharstein and Szeliski [117]	se
P-LinearS	De-Maeztu et al. [29]	db	SO	Scharstein and Szeliski [117]	sf
G-LinearS	De-Maeztu et al. [29]	dc	BP + MLH	Stankiewicz and Wegner [118]	sg
SegTreeDP	Deng and Lin [31]	dd	PUTv3	Stankiewicz and Wegner [119]	sh
CostAggr + occ	Min and Sohn [88]	de	DistinctSAD	Stefano et al. [120]	si
MultipleAdaptive	Demoulin and Droogenbroeck [30]	df	GenModel	Strecha et al. [121]	sj
SNCC	Einecke and Eggert [33]	ea	SymBP + occ	Sun et al. [123]	sk
PhaseDiff	El-Etriby et al. [34]	eb	RSR/TSDP	Sun [122]	sl
PhaseBased	El-Etriby et al. [35]	ec	RDP	Sun et al. [124]	sm
EfficientBP	Felzenszwalb and Huttenlocher [36]	fa	ICM	Szeliski et al. [126]	sn
RTDP	Forstmann et al. [37]	fb	BP-S	Szeliski et al. [126]	so
RandomVote	Gales et al. [38]	ga	BP-M	Szeliski et al. [126]	sp
ImproveSubPix	Gehrig and Franke [39]	gb	GC (Swap)	Szeliski et al. [126]	sq
SegBasedOutlier	Gerrits and Bekaert [40]	gc	GC (Exp)	Szeliski et al. [126]	sr
7 × 7SADSubpix	Goldberg and Matthies [41]	gd	TRW-S	Szeliski et al. [126]	ss
ReliabilityDP	Gong and Yang [42]	ge	AdpOvrSegBP	Taguchi et al. [127]	ta
Square-window	Gong et al. [44]	gf	ProfileShape	Tippetts et al. [128]	tb
Shiftable-window	Gong et al. [44]	gg	FastAggreg	Tombari et al. [131]	tc
Oriented-rod	Gong et al. [44]	gh	SegmentSupport	Tombari et al. [129]	td
Adaptive-win	Gong et al. [44]	gi	Unsupervised	Trinh and McAllester [132]	te

Table 1 continued

Algorithm	Reference	Label	Algorithm	Reference	Label
Boundary-guided	Gong et al. [44]	gj	3 × 3SAD	Tippetts et al. [128]	tf
Adaptive-wght	Gong et al. [44]	gk	SADL	van der Mark and Gavrila [82]	va
RDP + 3D-AdpWgt	Gong et al. [45]	gl	SADRec	van der Mark and Gavrila [82]	vb
HBpStereoGpu	Grauer-Gray and Kambhamettu [46]	gm	SADLR	van der Mark and Gavrila [82]	vc
AdpDispCalib	Gu et al. [47]	gn	SADMW5L	van der Mark and Gavrila [82]	vd
TwoWin	Gupta and Cho [49]	go	SADMW5Rec	van der Mark and Gavrila [82]	ve
RealTimeABW	Gupta and Cho [49]	gp	SADMW5LR	van der Mark and Gavrila [82]	vf
GradientGuided	Gong and Yang [43]	gq	SADDP	van der Mark and Gavrila [82]	vg
SemiGlob	Hirschmuller [50]	ha	OpenCVSGBM	van der Mark and Gavrila [82]	vh
C-SemiGlob	Hirschmuller [51]	hb	StereoSONN	Vanetti et al. [133]	vi
SAD-MW	Hirschmuller et al. [53]	hc	TreeDP	Veksler [135]	vj
GeoSup	Hosni et al. [54]	hd	MSOM	Venkatesh et al. [136]	vk
VSW	Hu et al. [56]	he	VariableWindows	Veksler [134]	vl
PlaneFitSGM	Humenberger et al. [58, 59]	hf	RealtimeGPU	Wang et al. [139]	wa
SAD	Humenberger et al. [58, 59]	hg	2passApp	Wang et al. [139]	wb
Census	Humenberger et al. [58, 59]	hh	CoopReg	Wang and Zheng [140]	wc
RTCensus	Humenberger et al. [58, 59]	hi	GlobalGCP	Wang and Yang [138]	wd
RTCensus(DSP)	Humenberger et al. [58, 59]	hj	ESAW	Yu et al. [158]	we
RTCensus(GPU)	Humenberger et al. [58, 59]	hk	PARTS	Woodfill and Von Herzen [141]	wf
SparseCensus	Humenberger et al. [57]	hl	Tyzx	Woodfill et al. [142]	wg
MaxFlowGC	Ishikawa and Geiger [61]	ia	2OP+occ	Woodford et al. [143]	wh
FusionMoveGC	Ishikawa [60]	ib	OutlierConf	Xu and Jia [144]	xa
11 × 11Census	Jin et al. [63]	ja	RadialAdaptive	Xu et al. [145]	xb
SymDP(FPGA)	Kalarot and Morris [64]	ka	PlaneFitBP	Yang et al. [148]	ya
SymDP(GPU)	Kalarot and Morris [64]	kb	RealtimeBP	Yang et al. [146]	yb
ShiftableWin	Kang et al. [65]	kc	SubPixDoubleBP	Yang et al. [147]	yc
3 × 3Census	Khaleghi et al. [66]	kd	DoubleBP	Yang et al. [149]	yd
AdaptingBP	Klaus et al. [69]	ke	MultiResSSD	Yang and Pollefeys [151]	ye
GC + occ	Kolmogorov and Zabih [70]	kf	AdaptiveWin	Yang et al. [152]	yf
MultiCamGC	Kolmogorov and Zabih [71]	kg	CSBP (GPU)	Yang et al. [150]	yg
SVM2.0	Konolige [72]	kh	CSBP	Yang et al. [150]	yh
RealtimeVar	Kosov et al. [73]	ki	AdaptWeight	Yoon and Kweon [154]	yi
SSDCensus	Kuhn et al. [74]	kj	DistinctSM	Yoon and Kweon [155]	yj
MaxConnected	Kim et al. [68]	kk	AdpWgtColorProx	Yoon and Kweon [153]	yk
EnhancedBP	Larsen et al. [75]	la	DSImatrix/SAD	Yoon et al. [156]	yl
TilebasedBP	Liang et al. [77]	lb	BPcompressed	Yu et al. [157]	ym
CCH + SegAggr	Liu et al. [78]	lc	VariableCross	Zhang et al. [160]	za
AdpPolygon	Lu et al. [78]	ld	RealtimeBFV	Zhang et al. [159]	zb
WinSepLap	Lu et al. [79, 80]	le	CensusVarCross	Zhang et al. [161]	zc
TrunSepLap	Lu et al. [79, 80]	lf	MultiResAdpWin	Zhao and Taubin [162]	zd
LWPC	Masrani and MacLean [83]	ma	DistSparsCens	Zinner and Humenberger [163]	ze
FastBilateral	Mattoccia et al. [86]	mb	Layered	Zitnick et al. [166]	zf
LocallyConsist	Mattoccia [84]	mc	OverSegmBP	Zitnick and Kang [165]	zg
SO + borders	Mattoccia et al. [85]	md	CoopOptim	Zitnick and Kanade [164]	zh

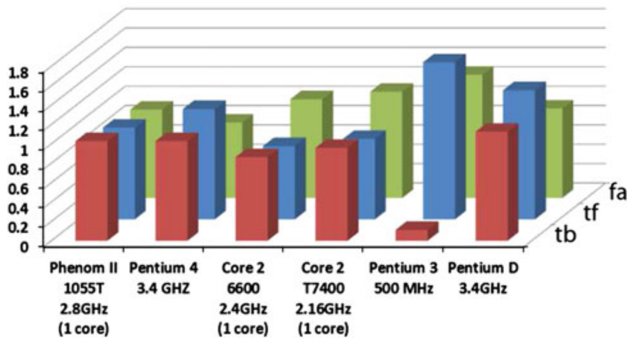

Fig. 1 Ratio of normalized runtimes to the average normalized runtime across processors of three stereo vision algorithms. Runtimes normalized using PassMark values PassMark Software (2012). A correct normalization factor would result in all ratios being 1.0

algorithms that are able to provide dense disparity maps using limited resources.

3 Algorithm comparison methods

From Scharstein and Szeliski [117], the average percentage of bad pixels measurement has become a standard for comparing the accuracy of stereo vision algorithms. On their website (Scharstein [116]), they provide three error measurements of how many pixels were assigned an incorrect disparity for each algorithm on each of four different image data sets (Tsukuba, Venus, Teddy, Cones). The three error measures are for all pixels (all), non-occluded pixels (nonocc), and pixels near discontinuities (disc). The accuracy measure used in this paper is the complement of the all pixels error measure (100 %—all), or in other words, the average percentage of correct disparities for all pixels in the four image data sets Tsukuba, Venus, Teddy, and Cones. There are also many algorithms that have published their performance using one or more of these error measures, but have not submitted their measurements to be included on the website (Scharstein [116]). Error measurements that have been published, but are not available on the website, are included in Table 8.

In addition to accuracy, runtime performance is also presented in this work. In all graphs and tables, runtime performance is given in millions of disparity evaluations per second (Mde/s), which is calculated from the time to compute the disparity map for one frame (Eq. 1). Many of the algorithms were evaluated on images of different sizes and resulted in different runtimes. Values in all tables include the highest performance each algorithm achieved along with the image size on which it was achieved.

$$\text{Mde/s} = \frac{W \times H \times D}{t} \times \frac{1}{1{,}000{,}000} \tag{1}$$

Table 2 Comparison of computations per pixel for multiple accurate stereo vision algorithms

Alg	t (s)	W × H (disp)	Mde/s	Hardware
za	1.6	450 × 375 (60)	6.3281	P4 3.0 GHz
we	2.57	450 × 375 (60)	3.9382	Core 2 Duo E6750 2.66 GHz
gp	0.46	384 × 288 (16)	3.8467	Core 2 Duo 2.80 GHz
gq[1]	3	450 × 375 (60)	3.3750	Core Duo 2.16 GHz
fa[2]	4.5	450 × 375 (60)	2.2500	Xeon 2.8 GHz
cg	0.81	384 × 288 (16)	2.1845	P4 2.8 GHz
sd	1.7	434 × 383 (20)	1.9556	n/a
sc	1.79	434 × 383 (20)	1.8572	n/a
oa	1	384 × 288 (16)	1.7695	P4 2.4 GHz
se	1.9	434 × 383 (20)	1.7497	n/a
gb	7	450 × 375 (60)	1.4464	n/a
sn[3]	7	450 × 375 (60)	1.4464	P4 1.73 GHz
sf	2.3	434 × 383 (20)	1.4454	n/a
gc	1.26	384 × 288 (16)	1.4043	P4 3.0 GHz
sm	8.7	450 × 375 (60)	1.1638	Core2 Duo 3.0 GHz
cb	10	450 × 375 (60)	1.0125	P4 1.73 GHz
dc	10	450 × 375 (60)	1.0125	Core 2 6420
bi[1]	15	450 × 375 (60)	0.6750	Core Duo 2.16 GHz
me	15	450 × 375 (60)	0.6750	Core 2 Duo 2.2 GHz
hf	15.2	450 × 375 (60)	0.6659	n/a
kc[4]	4.1	320 × 240 (30)	0.5620	n/a
ch	25	450 × 375 (60)	0.4050	1.4 GHz Centrino
ke	25	450 × 375 (60)	0.4050	2.21 GHz Athlon 64
vl[1]	25	450 × 375 (60)	0.4050	Core Duo 2.16 GHz
wb[4]	6.2	320 × 240 (30)	0.3716	n/a
cc	29	450 × 375 (60)	0.3491	P4 1.73 GHz
so[3]	30	450 × 375 (60)	0.3375	P4 1.73 GHz
de[4]	9.8	320 × 240 (30)	0.2351	n/a
sq[3]	45	450 × 375 (60)	0.2250	P4 1.73 GHz
zg	50	450 × 375 (60)	0.2025	2.8 GHz PC
kf[2]	55	450 × 375 (60)	0.1841	Xeon 2.8 GHz

If runtimes were given for multiple images in the original publication, then the one that resulted in the maximum Mde/s is included here

Citations for performance measurements if not from original publication [1] Tombari et al. [130], [2] Hirschmuller [50], [3] Cassisa [19], [4] Min and Sohn [88]

With respect to comparing runtime performance of stereo vision algorithms, we acknowledge the caution given by Einecke and Eggert [33] that although quite common, these types of comparisons are very rough. There are multiple factors that influence runtime measurements besides the computational power of a CPU such as programming language, the skill and effort of the programmer in optimizing the implementation, and parallelization techniques that can make significant differences in stereo vision algorithms due to their parallelizable nature. We also acknowledge that it is often necessary to make such a comparison when deciding which algorithm to implement for a given application. We include all published runtimes achieved by the stereo vision algorithms presented here, as well as all available hardware details to allow the reader to make such comparisons.

In the absence of standard methods to perform this comparison, some may make the mistake of scaling runtime measurements directly by a processor clock speed (ignoring other factors such as processor pipelining, cache design, memory subsystem, etc.). In an effort to aid readers in making more accurate comparisons than scaling runtimes by processor clock speeds, processor benchmarks have been used to generate a normalization factor. Benchmark values for each of the specific CPUs were gathered from PassMark Software [106] to normalize runtimes. These benchmark values are collected empirically by averaging the performance on thousands of different speed benchmarks on each processor in different configurations.

These benchmarks are comprised of many types of algorithms and so we performed our own various tests using stereo vision algorithms to see how accurate this normalization factor would be. Figure 1 shows the results of these tests. Six different processor architectures were used to run three different stereo vision algorithms: a belief propagation algorithm (fa)[1], a local 3×3 sum of absolute differences (SAD) block matching algorithm (tf), and the Profile Shape Matching algorithm (tb). The normalized runtime, K, of an algorithm was measured for each processor, i. If the normalization factor for each processor, $\mathrm{PassMark}_i$, was 100 % accurate, then the normalized runtimes for an algorithm would all be equal. For clarity in the graph in Fig. 1, ratios of the normalized runtimes to the average of the normalized runtimes (Eq. 2) are shown. If the PassMark value were an exactly accurate normalization value, then all the bars in Table 2 would have a ratio of one.

$$\mathrm{ratio}_i = \frac{K_i}{\mathrm{avg}}, \quad K_i = \frac{\mathrm{Mde/s}}{\mathrm{PassMark}_i}, \quad \mathrm{avg} = \frac{\sum_{i=0}^{N} K_i}{N} \qquad (2)$$

With the one exception of ProfileShape (tb) running on the Pentium 3 500 MHz processor, the PassMark normalization factor brings the runtimes across all processors to within 1.5× of each other. This exception shows that there are cases where this normalization factor fails, but any normalization factor is going to have cases where it fails because of the complexity of the factors that impact runtime. Despite these concerns we have chosen to include normalized runtimes, because they allow much better comparisons of algorithm runtimes than would be provided by unnormalized runtimes or runtimes normalized by processor clock speed. Where sufficient details of the hardware were published, normalized runtime values are also given here.

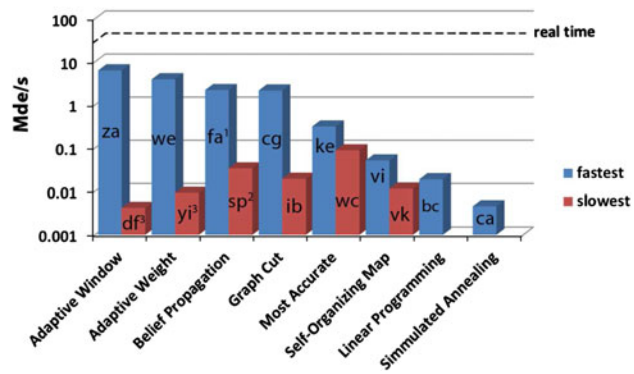

Fig. 2 A comparison of published runtimes for various accurate algorithms and their relationship to real-time performance. The two algorithms labeled most accurate are based on ranking on the Middlebury site Scharstein [116] of those with published runtimes. Citations for performance measurements if not from original publication: [1] Hirschmuller [50], [2] Cassisa [19], [3] Min and Sohn [88]

4 Accurate algorithms with high-resource requirements

When categorizing stereo vision algorithms, one of the most obvious divisions in current literature is that of global versus local methods. Gupta and Cho [48] describe local algorithms as statistical methods usually based on correlation, and global algorithms as being based on explicit smoothness assumptions that are solved through various optimization techniques. They also state that the computational complexity of global algorithms makes them impractical for real-time systems. The smoothness assumptions are defined through an energy function and the optimization techniques minimize this function. The correlation process of local methods involves finding matching pixels in left and right images of a stereo pair by aggregating costs [e.g. sum of absolute differences (SAD), sum of squared differences (SSD), normalized cross correlation (NCC)] within a region or block. For several years, the most accurate disparity maps were produced by global algorithms. Currently, eight of the ten most accurate stereo vision algorithms ranked by Middlebury evaluation criterion are global energy minimization algorithms. Recently, however, many local algorithms have been developed that are competitive with respect to accuracy. As mentioned, there is always a trade-off between accuracy and speed for stereo vision

[1] For all custom algorithm labellings and corresponding citations see Table 1.

Table 3 Comparison of computations per pixel for multiple accurate stereo vision algorithms

Alg	*t* (s)	W × H (disp)	Mde/s	Hardware
pa	60	450 × 375 (60)	0.1688	Standard PC
sr[3]	68	450 × 375 (60)	0.1489	P4 1.73 GHz
mc	37	450 × 375 (32)	0.1459	2.5 GHz Core Duo
db	94	450 × 375 (60)	0.1077	Core 2 6420
wd	96	450 × 375 (60)	0.1055	2.83 GHz dual core
wc	20	384 × 288 (16)	0.0885	1.6 GHz P Mobile
bk	20	384 × 288 (16)	0.0885	Core2 Duo T7700 2.4 GHz
pe	25	384 × 288 (16)	0.0708	Quadcore 2.66 GHz
vi	28	384 × 288 (16)	0.0632	AMD 1,800 MHz Mobile
sk	45	384 × 288 (16)	0.0393	2.8 GHz P4
sp[3]	300	450 × 375 (60)	0.0338	P4 1.73 GHz
zh[5]	40	257 × 256 (20)	0.0329	P4 1.4 GHz
hd	60	384 × 288 (16)	0.0295	Standard PC
yk	60	384 × 288 (16)	0.0295	AMD 2700+
da	60	384 × 288 (16)	0.0295	Core Duo 6420
yi[4]	87.5	320 × 240 (30)	0.0263	n/a
bj[5]	55	257 × 256 (20)	0.0239	P4 1.4 GHz
ib	524	450 × 375 (60)	0.0193	2.33 GHz Xeon E5345
bc	536	450 × 375 (60)	0.0189	Core 2 Duo
ss[3]	810	450 × 375 (60)	0.0125	P4 1.73 GHz
vk	115	258 × 256 (20)	0.0115	P4 1.4 GHz
ck[1]	25	450 × 375 (60)	0.4050	Core Duo 2.16 GHz
bg	1,200	450 × 375 (60)	0.0084	n/a
bb	1,800	450 × 375 (60)	0.0056	1.86 GHz
td[1]	2,014	450 × 375 (60)	0.0050	Core Duo 2.16 GHz
kg	369	384 × 288 (16)	0.0048	SPARC II 450 MHz
ca	2,211	450 × 375 (60)	0.0046	P4 1.73 GHz
df[1]	2,387	450 × 375 (60)	0.0042	Core Duo 2.16 GHz
ia	3,600	512 × 512 (48)	0.0035	266 MHz PII
xb[1]	3,532	450 × 375 (60)	0.0028	Core Duo 2.16 GHz
kk[1]	7,200	450 × 375 (60)	0.0014	Core Duo 2.16 GHz

If runtimes were given for multiple images in the original publication, then the one that resulted in the maximum Mde/s is included here

Citations for performance measurements if not from original publication: [1] Tombari et al [131],[3] Cassisa [19], [4] Min and Sohn [88], [5] Venkatesh et al [136]

algorithms and these accurate local algorithms are no exception. Since the focus of this review is to present stereo vision algorithms with respect to their suitability for real-time implementation on resource-limited systems, they have been divided into two groups: those that are capable of real-time or near real-time performance on common CPUs and those that require much greater computational power to approach real-time performance. For this discussion, near real-time is considered to be a rate of 1 fps or greater on images of size and disparity range similar to Teddy or Cones images from the Middlebury dataset (>10 Mde/s), and real-time is a rate of 30 fps (>53 Mde/s for Tsukuba images).

Section 4.1 provides an overview of those accurate algorithms with high-resource requirements within the context of realtime performance. The algorithms presented in this section have been grouped into common global methods (Sec. 4.2) and local methods (Sect. 4.3), with Sect. 4.4 consisting of other unique methods as well as other computationally complex techniques such as segmentation and refinement algorithms. Discussion of the trade-off between accuracy and speed is found in Sect. 4.5 for all algorithms presented in Sect. 4.

4.1 Context of accurate high-resource algorithms to realtime

For each of the different types of most accurate algorithms (e.g. Adaptive Weights, Belief Propagation, Graph Cut), there are various implementations that have been developed with varying accuracy and runtimes. To give an idea of the range of runtimes of each type of algorithm and how far they are from obtaining realtime performance, the slowest and fastest implementations of each type of algorithm are shown in Fig. 2. For example, the algorithm AdpWeight (yi) has the slowest published runtime of adaptive weight algorithms at 0.009 Mde/s, while ESAW (we) has the fastest at 3.94 Mde/s. In addition to these ranges of published runtimes for a few of the most common accurate global and local methods, the two most accurate algorithms that provide runtime measurements are included as well. The claim of being the two most accurate algorithms with published runtimes is based on ranking on the Middlebury site (Scharstein [116]). The logarithmic scale of the graph helps illustrate how far these algorithms are from achieving real-time performance. The fastest of the algorithms shown are still an order of magnitude from achieving real-time. The linear programming algorithm InteriorPtLP (bc) and simulated annealing (ca) are the only implementations of those types with published runtimes. In several stereo vision implementations simulated annealing is shown as the slowest of the optimization techniques (Cassisa [19], Szeliski and Zabih [125]).

Tables 2 and 3 show runtimes for all algorithms slower than 1 fps that publish enough information to calculate Mde/s. Many algorithms, e.g. global, do not require a disparity range to be defined, and some have additional parameters such as number of iterations that, along with the size of the image, can influence its runtime. For each of these, the disparity range that the image data set requires is used to calculate the Mde/s of the algorithm, and any other parameters that can allow for a trade-off in accuracy for faster performance are included in the discussion of those algorithms.

4.2 Common high-resource global techniques

A closer look at the algorithms in Tables 2 and 3 can help identify several common algorithms and techniques that diminish suitability for real-time implementation on resource-limited systems. For example, Bleyer has partnered with several others to develop several stereo algorithms Segm+visib (bd), WarpMat (be), SurfaceStereo (bf), ObjectStereo (bg), and PatchMatch (bh), some of which are among the most accurate, but when accounting for hardware, are also among the slowest (Table 3). All of the algorithms by Bleyer et al. minimize global energy functions. The PatchMatch (bh) algorithm also incorporates an adaptive slanted support window that performs well on large untextured regions. They implement a fusion-move graph cuts optimization to make surface assignments in SurfaceStereo (bf) and to recover the depth of regions occluded in one input image in ObjectStereo (bg).

Other graph cuts based implementations are also among the most accurate yet slowest methods applied to stereo vision. Varieties of graph cuts methods include swap-move, expansion-move [GC (bj)], fusion-move [2OP+occ (wh), FusionMoveGC (ib)], and max-flow [MaxFlowGC (ia)]. Chen et al. implement what they call a semi-global graph cuts method SemiGlobGC (cg) to extract disparity assignments for scanline segments instead of directly solving the final disparity. Graph cuts implementations developed by Kolmogorov and Zabih, GC+occ (kf) and MultiCamGC (kg), have been used frequently for comparisons and starting points for other improvements like that made by Miyazaki using a hierarchical method in H-Cut (mj). More recent implementations include MultiResGC (pa) and GlobalGCP (wd). Another attribute of graph cuts methods that should be considered is the memory requirement. Miyazaki et al. reported memory usage of H-Cut (mj) at 150 times the size of an input image.

Belief Propagation, often referred to as BP, is another popular accurate energy minimization technique used in WarpMat (be), AdaptingBP (ke), BP+MLH (sg), PUTv3 (sh), AdpOvrSegBP (ta), Unsupervised (te), OutlierConf (xa), SubPixDoubleBP (yc), and DoubleBP (yd). In general, BP requires large amounts of computations and also memory to store and execute its message passing system. Several modified versions have been published to reduce computation and/or memory resources, but the resulting algorithms are still not suitable for low-resource real-time implementation. These include EfficientBP (fa), MVSegBP (mk), SymBP+occ (sk), CSBP (yh), and BPcompressed (ym), with the most efficient being EfficientBP (fa), which reduced computation and memory requirements by a factor of 2×. Tree reweighted message passing (ss) or TRW is a variation of BP and has a factor of 2× Tree reweighted message passing (ss) or TRW is a variation of BP and has similar restrictions.

Global energy minimization has also been formulated as linear equations in InteriorPtLP (bc) and CostRelax (bl). In InteriorPtLP (bc), linear programming is used to solve the equations. It must calculate the inverse of a large sparse matrix which is computationally intensive. CostRelax (bl) uses a gradient descent method to minimize the energy function.

Both modified self-organizing maps, MSOM (vk), and self-organizing neural networks, StereoSONN (vi), are types of neural networks that can reduce higher dimensional problems to a two dimensional grid. When applied to stereo vision, Venkatesh showed that MSOM (vk) was slower than a graph cut implementation on the same hardware. It is global in nature, as it does not require predefining a disparity range nor a search window, or assuming an epipolar constraint. However, predefining a disparity range or search window decreases the runtime. These two algorithms are described as techniques that can estimate disparities of fine objects, detect discontinuities in disparities effectively, preserve the shape of objects, and compute smooth disparity maps.

4.3 Common high-resource local techniques

Many of the most accurate local algorithms include methods to adapt the size or shape of the aggregation window or vary the weighting of costs within that window. These techniques are referred to as adaptive support, variable support, adaptive window, adaptive correlation window, or adaptive weights. Multiple variable support algorithms were run on the same hardware and compared according to accuracy and runtime by Tombari et al. [130] including DP (bi), Recursive Adaptive (ck), MultipleAdaptive (df),

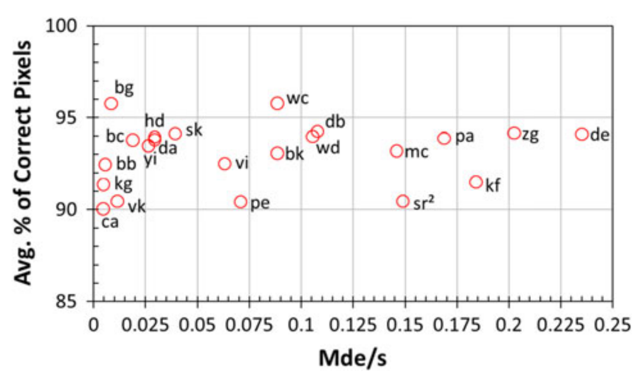

Fig. 3 A comparison of runtimes and accuracy for algorithms not able to achieve near real-time runtime performance that also provide accuracy measurements on all four Middlebury images Tsukuba, Venus, Teddy, and Cones. *Notes* Citations for performance measurements if not from original publication: [1] Hirschmuller [50], [2] Cassisa [19], [3] Min and Sohn [88]

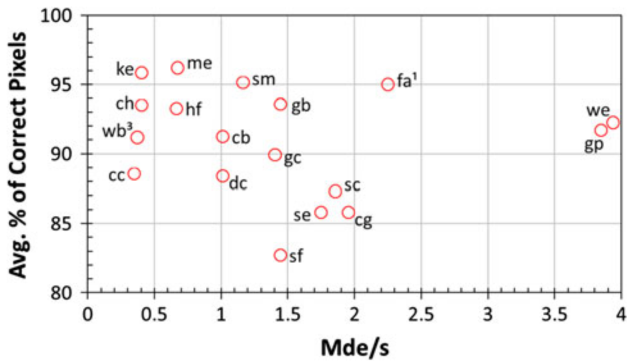

Fig. 4 (continued from Fig. 3) A comparison of runtimes and accuracy for algorithms not able to achieve near real-time runtime performance that also provide accuracy measurements on all four Middlebury images Tsukuba, Venus, Teddy, and Cones. Citations for performance measurements if not from original publication: [1] Hirschmuller [50], [2] Cassisa [19], [3] Min and Sohn [88]

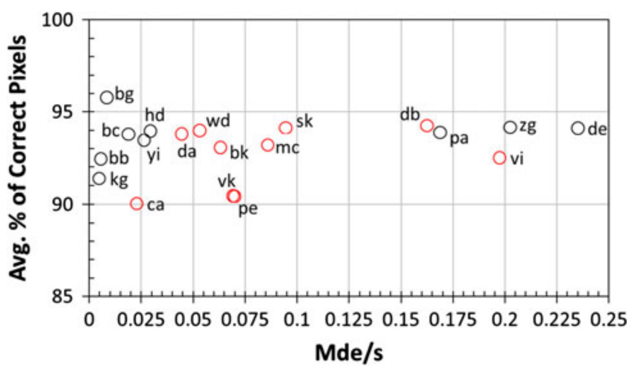

Fig. 5 A comparison of runtimes normalized for hardware and accuracy for all real-time stereo vision algorithms that provide accuracy measurements on all four Middlebury images Tsukuba, Venus, Teddy, and Cones (*red* markers). Algorithms lacking hardware details were not normalized (*black* markers)

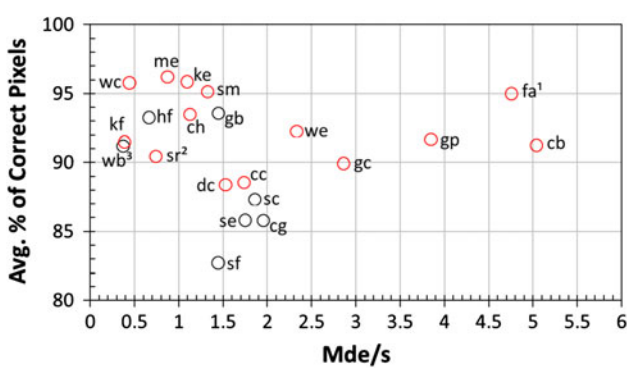

Fig. 6 (continued from Fig. 5) A comparison of runtimes normalized for hardware and accuracy for all real-time stereo vision algorithms that provide accuracy measurements on all four Middlebury images Tsukuba, Venus, Teddy, and Cones. Citations for performance measurements if not from original publication: [1] Hirschmuller [50], [2] Cassisa [19], [3] Min and Sohn [88]

SegBasedOutlier (gc), Gradient Guided (gq), ShiftableWin (kc), Max Connected (kk), SegmentSupport (td), VariableWindows (vl), Radial Adaptive (xb), and AdpWeight (yi). Their runtime results are included in Tables 2 and 3. None of the variable support algorithms they compared was able to come close to near real-time performance.

One of the fastest algorithms presented in this section is an adaptive correlation window approach by Zhang et al. VariableCross (za), which uses a cross-based support window. As seen in Table 2, they show that it is able to achieve 6.328 Mde/s on a Pentium 4 3.0 GHz processor. Other slower adaptive correlation window algorithms include TwoWin (go), AdaptPolygon (ld), and TensorVoting (ml), and adaptive support weight algorithms include CostRelaxAW (bk), GeoSup (hd), AdpDispCalib (gn), CurveletSupWgt (mn), FastBilateral (mb), BioPsyASW (na), LCDM+AdpWgt (nb), IterAdpWgt (pg), DistinctSM (yj), and AdpWeight (yi).

4.4 Other common high-resource techniques

The mean shift algorithm is one of a few algorithms that are used as part of many stereo techniques, but that are quite computationally expensive even before the energy minimization or correspondence parts of the algorithms are considered. Mean shift is an iterative algorithm that has become popular for its accuracy of segmentation in stereo vision applications Segm+visib (bd), WarpMat (be), SurfaceStereo (bf), RandomVote (ga), C-SemiGlob (hb), CCH+SegAggr (lc), SO+borders (md), AdpOvrSegBP (ta), FastAggreg (tc), SegmentSupport (td), and CoopReg (wc). Klaus et al. report that it requires more computation time than belief propagation when used together in the same implementation AdaptingBP (ke).

Anisotropic diffusion allows smoothing without losing edges, making it a useful technique to improve image segmentation as in Layered (zf) and OverSegmBP (zg) or to smooth disparity map results as in BP+DirectedDiff (ba). It is memory intensive because it stores multiple difference images, which, depending on the stage in which it is used, can require a difference image for each disparity value, as in CostAggr+occ (de).

El-Etriby et al. introduce a novel approach where images are converted to the frequency domain and represented as scalograms which involves a convolution with Gabor wavelets for PhaseBased (ec) and PhaseDiff (eb). Although quite unique, the technique also suffers from long runtimes.

Some algorithms improve accuracy by taking advantage of multiple views whether obtained with spatial differences or temporal differences, e.g. MultiCamGC (kg), EnhancedBP (la), and GenModel (sj). Including more than the standard two stereo images will obviously increase computations and consequently runtimes.

4.5 Accuracy versus speed trade-off

Figures 3 and 4 show the accuracy and runtime speed trade-off for all algorithms in Tables 2 and 3 that also have made accuracy measurements available for the four Middlebury image datasets Tsukuba, Venus, Teddy, and Cones. Since each algorithm runtime is on different hardware, normalized versions of these graphs are provided in Figs. 5 and 6 (see Sect. 3 for normalization method details). Those algorithms that were not published with sufficient details of the hardware to calculate a normalized runtime are represented with black markers versus the red markers of normalized values.

There are some interesting insights gained from these graphs, like the benefits of choosing an algorithm like the adaptive support algorithm from Mei et al. ADCensus (me) (Fig. 4). In terms of accuracy and runtime, it is pareto optimal to all algorithms in Fig. 3, i.e. there is no advantage to selecting any of those algorithms over ADCensus (me). In fact, ADCensus (me) is pareto optimal to all algorithms found to the left of it in Fig. 4 as well, since it has both better accuracy and a faster runtime than the others. On the other hand, one can trade off about 1.5 percentage points of accuracy for approximately a 5× speedup by selecting EfficientBP (fa) by Felzenszwalb and Huttenlocher over ADCensus (me). The graphs show that EfficientBP (fa) is pareto optimal for all algorithms in both Figures except ObjectStereo (bg), CoopReg (wc), ADCensus (me), AdaptingBP (ke), and RDP (sm).

5 Algorithms with real-time performance on a GPU

Kim et al. [67] discuss how vision algorithms in general are well suited for implementation on graphics processing units (GPU) because they are easily parallelizable. Even though GPUs are not resource-limited systems, some massively parallelizable algorithms are still not able to reach real-time performance, such as some of the first

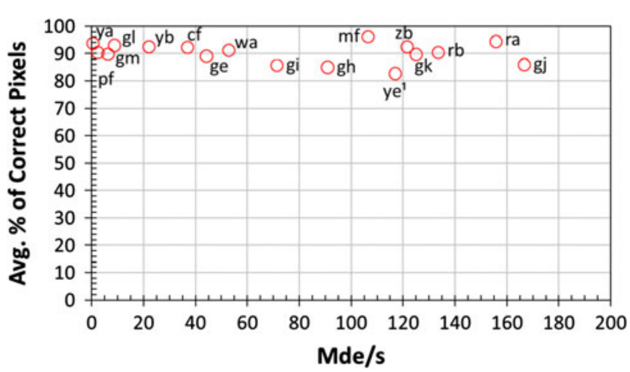

Fig. 7 A closeup of Fig. 9 showing a comparison of runtimes and accuracy for algorithms implemented on a GPU with runtimes between 0 and 200 Mde/s

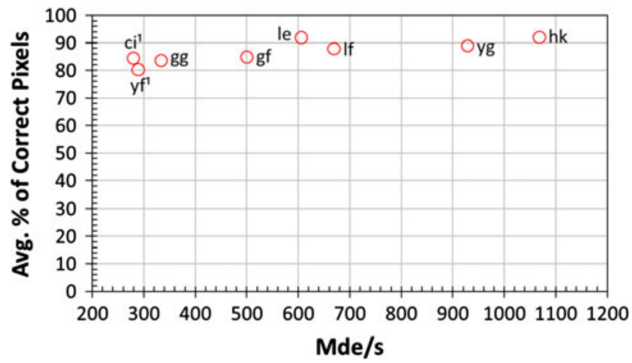

Fig. 8 A closeup of Fig. 9 showing a comparison of runtimes and accuracy for algorithms implemented on a GPU with runtimes between 200 and 1,200 Mde/s. [1] Performance measurements taken from Zhao and Taubin [162]

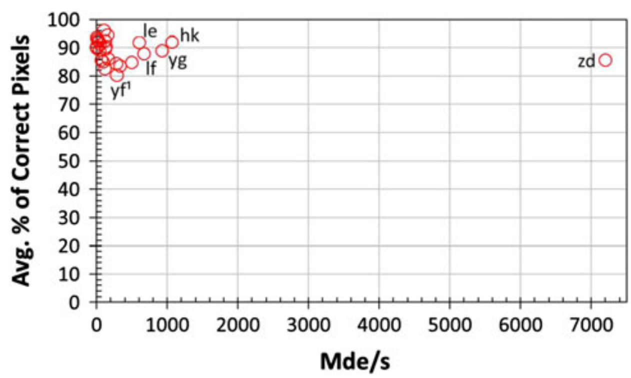

Fig. 9 A comparison of runtimes and accuracy for algorithms implemented on a GPU

implementations of belief propagation such as PlaneFitBP (ya), HBpStereoGpu (gm) and other energy minimization algorithms like ConvexTV (pf). Several researchers have touted the ability to reach real-time performance for accurate and previously slow stereo vision algorithms by optimizing them for GPUs. For example, Gong et al. implemented several local cost aggregation methods in real-time on a GPU; ReliabilityDP (ge), Square-window (gf), Shiftable-window (gg), Oriented-rod (gh), Boundary-guided (gj), RDP+3D-AdpWgt (gl) including Adaptive-win (gi) and Adaptive-weight (gk) methods. Table 4 lists the runtimes of several GPU implementations of stereo vision algorithms. Even with high resource hardware, trade-offs between accuracy and speed are made. For example, Yang et al. developed a BP algorithm Realtime BP (yb) that, by reducing iterations, allowed the runtime performance to increase at the cost of accuracy. Figures 7, 8 and 9 show the tradoffs available between GPU implementations of stereo vision algorithms that published both accuracy and runtime measurements. Figures 7 and 8 are close-up views of Fig. 9. The graphs in these figures show that varying amounts of accuracy can be traded for significant increases in runtime performance. The greatest example is that of Zhao and Taubin's multiple resolution adaptive window algorithm MultiResAdpWin (zd) that achieves over 7,200 Mde/s if an accuracy of an average of 86 % of pixels having the correct disparity is sufficient.

Table 4 Comparison of computations per pixel for stereo vision algorithms implement on a GPU

Alg	*t* (ms)	W × H (disp)	Mde/s	GPU
zd	27.8	1024 × 768 (256)	7247.8	Geforce GTX280
kb	50.0	1024 × 768 (128)	2013.3	Geforce GPX280
hk	9.5	450 × 375 (60)	1068.0	Geforce GTX280
yg	155.0	800 × 600 (300)	929.0	Geforce 8800GTX
lf	37.6	512 × 512 (96)	668.4	Geforce 7900
le	41.6	512 × 512 (96)	605.5	Geforce 7900
gf	20.3	450 × 375 (60)	500.0	Radeon X800
gg	30.3	450 × 375 (60)	333.3	Radeon X800
yf[1]	n/a	n/a	289	Radeon 9800
ci[1]	n/a	n/a	280	Geforce 6800GT
gj	60.7	450 × 375 (60)	166.7	Radeon X800
ra	65.0	450 × 375 (60)	155.8	Geforce GTX480
rb	75.8	450 × 375 (60)	133.6	Quadro FX5800
zb	83.3	450 × 375 (60)	129.6	Geforce 8800GTX
gk	81.0	450 × 375 (60)	125.0	Radeon X800
ye	224	512 × 512 (100)	117	Geforce4
mf	95.0	450 × 375 (60)	106.6	Geforce GTX480
gh	111.3	450 × 375 (60)	90.9	Radeon X800
gi	141.7	450 × 375 (60)	71.4	Radeon X800
wb	23.3	320 × 240 (16)	52.8	GPU
ge	40.0	384 × 288 (16)	44.2	Radeon 9800
cf	50	320 × 240 (24)	36.9	Geforce GTX285
yb	80.0	384 × 288 (16)	22.1	Geforce 7900GTX
lb	594.5	450 × 375 (60)	17.0	Geforce 8800GTS
gl	200	384 × 288 (16)	8.85	Geforce GTX480
gm	281.7	384 × 288 (16)	6.28	Geforce 8600M GT
pf	750	384 × 288 (16)	2.36	Geforce GTX280
ya	18,000.0	512 × 384 (48)	0.5	Geforce 8800

If runtimes were given for multiple images in the original publication, then the one that resulted in the maximum Mde/s is included here

Citations for performance measurements if not from original publication: [1] Zhao and Taubin [162]

6 Algorithms with real-time performance on a DSP, FPGA, or ASIC

FPGAs and DSPs in many ways could be considered low-resource devices. Both require little power and are many times built into physically small systems with less memory than general CPUs and GPUs. Both have the ability to massively parallelize computations, to which most stereo vision algorithms lend themselves very well. However, because they are high performance parallel processing devices, FPGAs and DSPs tend to be limited in terms of design time and availability of off-the-shelf stereo vision platforms. Development of novel algorithms is almost always performed in software for CPUs, because of the long development times for complex parallel designs on programmable or specifically designed hardware, e.g. FPGA, DSP, ASIC. Samarawickrama [115] offers a detailed discussion about the advantages and disadvantages between these technologies with respect to real-time implementations of vision algorithms. Published performance results of stereo vision algorithms on FPGA and DSP-based platforms are provided here in an effort to understand the performance increases that are available when algorithms are optimized for parallel implementations on such hardware.

Goldberg and Matthies [41] compared DSP stereo vision implementations by using Mde/s per GHz. Although the implementation SVM 2.0 (kh) by Konolige was not included in the comparison, if the same metric were applied to the values in Table 5, then SVM 2.0 (kh) would rank at the top with 372.5 Mde/s per GHz over 275.4 of Goldberg and Matthies' algorithm 7 × 7 SADSubpix (gd). Of those that provide accuracy measures for the Middlebury dataset, the 8 × 8 census algorithm RTCensus (hj) from Humenberger et al. ranks highest with 91.2 % average correct pixels, along with the distributed version of the same algorithm from Zinner and Humenberger DistSparsCens (ze) with the same accuracy, followed by the sparse census SparseCensConf (ad) from Ambrosch et al. with 88.9 %. The least accurate DSP implementation that provides such information is the 4 × 5 Jigsaw SAD 4 × 5 Jigsaw SAD (cd) by Chang et al. with 73.63 %. In terms of speed and accuracy, SparseCensConf (ad) has a decided advantage over 4 × 5 Jigsaw SAD (cd), and the accuracy trade-off is quite small to gain the nearly 4× speedup of SparseCensConf (ad) over RTCensus (hj).

Only three of the FPGA stereo vision implementations in Table 6 provided accuracy measures for all four

Table 5 Comparison of computations per pixel for stereo vision algorithms implemented on a DSP

Alg	*t* (ms)	W × H (disp)	Mde/s	DSP
gd	70	512 × 384 (51)	143.2	520 MHz TI
ad	84	450 × 375 (60)	120.2	1 GHz
cd	109.9	450 × 375 (60)	92.1	1 GHz TI
ze	129	450 × 375 (60)	78.5	1 GHz TI
kh	33	320 × 240 (32)	74.5	200 MHz TI
hj	37.9	320 × 240 (15)	30.4	1 GHz
kd	50	160 × 120 (30)	11.5	600 MHz

If runtimes were given for multiple images in the original publication, then the one that resulted in the maximum Mde/s is included here

Middlebury images. The census variable cross algorithm CensusVarCross (zc) from Zhang et al. obtained an accuracy of 92.6 % average correct pixels, while Ambrosch and Kubinger's SAD and gradient census algorithm SAD-IGMCT (aa) achieved 90.6 %, and the census-based algorithm 11 × 11 Census (ja) by Jin et al. achieved 86 % accuracy. Since CensusVarCross (zc) and SAD-IGMCT (aa) have the same runtime, then the extra accuracy of CensusVarCross (zc) gives it an edge, while a trade-off of about 6 percentage points of accuracy can be made for nearly a 4× speed up by going to 11 × 11 Census (ja).

7 Real-time or near real-time algorithms for limited resource systems

A great amount of research and development has gone into optimizing stereo vision algorithms to obtain real-time performance. Many attempts have relied on the parallel processing capabilities of hardware such as multiple cores and SIMD instructions to exploit the parallelizable nature of vision algorithms to achieve this. Some embedded applications may have such hardware available to them in the form of FGPAs, DSPs, ASICs, or even low-power multi-core processors, but other applications may be limited with regard to these resources because of development time, cost, or availability of off-the-shelf systems among other things.

Table 6 Comparison of computations per pixel for stereo vision algorithms implemented on an FPGA

Alg	*t* (ms)	W × H (disp)	Mde/s	FPGA
ac	1.7	450 × 375 (60)	6,062.9	Quartus II
cj	3.3	640 × 480 (60)	5,585.5	Stratix II EP2S60
ja	4.3	640 × 480 (60)	4,239.4	Virtex4 XC4VLX200
ka	33	1024 × 768 (128)	3,050.4	Stratix III
wg	5	512 × 480 (52)	2,555.9	DSP, FPGA, Tyzx
nc	7.7	640 × 480 (60)	2,396.9	Stratix 1S40
pd	39	512 × 512 (255)	1,714.0	Virtex4 XC4VLX15
zc	16	640 × 480 (60)	1,152.0	Stratix III EP3SL150
aa	16	750 × 400 (60)	1,152.0	n/a
mi	53	640 × 480 (80)	491.5	n/a
ma	33	640 × 480 (36 avg)	335.1	4 x Stratix S80
pb	33	320 × 240 (128)	297.9	Virtex II pro-100
ce	23.8	352 × 288 (60)	255.5	ASIC
pc	380	1280 × 720 (96)	232.8	Virtex 5 330VLX
sa	11.6	384 × 288 (16)	152.4	n/a
wf	23.8	320 × 240 (24)	77.4	PARTS
mh	33	320 × 240 (32)	74.5	Virtex II XC2VP40
kj	20.0	256 × 192 (25)	61.4	ASIC

If runtimes were given for multiple images in the original publication, then the one that resulted in the maximum Mde/s is included here

For those resource-limited applications, a comprehensive comparison of accuracy and runtime performance of available stereo algorithms is valuable when trying to select an algorithm that is most likely to meet the application requirements. This section presents data for algorithms that have a published or normalized speed of 10 Mde/s or faster, which is 1 frame per second using an image of the size and disparity range of Teddy or Cones, all of which can be seen in Table 7. In compiling the data presented here, it was observed that for many of the algorithms the Mde/s measurement is higher when using larger images than smaller images on the same hardware. For most algorithms, it is a small difference, but for others, it is quite significant, as in the case of CSBP (yh) by Yang et al. who point it out explicitly as a feature.

Of those algorithms that obtain real-time or near real-time performance, many of them employ local SAD cost aggregation, including OpenCV SAD (ab), SAD-MW (hc), SAD (hg), MeanSAD (mm), DistinctSAD (si), SADL (va), SADRec (vb), SADLR (vc), SADMW5 L (vd), SADMW5 Rec (ve), SADMW5LR (vf), SADDP (vg), OpenCV SGBM (vh), and DSI matrix/SAD (yl). Some of them use it in a winner take all (WTA) strategy where pixels with the minimum cost difference are matched—including SADL (va), SADRec (vb), SADLR (vc), SADMW5 L (vd), SADMW5 Rec (ve), and SADMW5LR (vf)—while the rest incorporate it into a more complex cost function that is then reduced until disparities corresponding to optimal values are found. Both OpenCV SAD (ab) and OpenCV SGBM (vh) use code provided in the OpenCV Library [103]. The SAD cost is used to generate disparity space images in DSI matrix/SAD (yl) and MeanSAD (mm), which requires memory of size (width) × (height) × (disparity range), from which to calculate the best disparity values for each pixel. There is a disparity space image for each disparity value and they are used to store the cost to match each pixel at that disparity. Two other algorithms RTCensus (hi) and SparseCensus (hl) also use disparity space images, but in conjunction with the census transform and hamming distance cost aggregation method instead of a SAD cost. The census transform is also used in (hh).

Another popular method applied to stereo vision that has both real-time and near real-time implementations is dynamic programming also referred to as DP. The fastest of which was developed by Forstmann et al. RTDP (fb). Other implementations include SegTreeDP (dd), OptimizedDP (sb), and RSR/TSDP (sl). Deng and Lin implement DP on a unique tree structure in SegTreeDP (dd), while Sun created a sub-regioning technique and applied a two-stage version of dynamic programming on the subregions in RSR/TSDP (sl).

Table 7 Comparison of disparity computations per pixel for real-time algorithms

Alg	*t* (ms)	W × H (disp)	Mde/s	Hardware
va	87.9	512 × 512 (48)	143.2	P4 3.2 GHz
hi	77.6	450 × 375 (60)	130.5	Core 2 T7200 2.0 GHz
vb	97.6	512 × 512 (48)	128.9	P4 3.2 GHz
vc	109.6	512 × 512 (48)	114.8	P4 3.2 GHz
tb	16	384 × 288 (16)	110.5	Phenom II 2.8 GHz
hl	100	450 × 375 (60)	108	Core2 2.5 GHz
fb	18	384 × 288 (16)	98.3	AMD Athlon 2800
ea	140	450 × 375 (60)	77.1	Std PC 3.0 GHz
vd	166.5	512 × 512 (48)	75.5	P4 3.2 GHz
ve	169.8	512 × 512 (48)	74.1	P4 3.2 GHz
vf	169.8	512 × 512 (48)	74.1	P4 3.2 GHz
vg	169.8	512 × 512 (48)	74.1	P4 3.2 GHz
si	25	320 × 240 (16)	49.1	PIII 800 MHz
yh	3,550	800 × 600 (300)	40.5	Core2 2.5 GHz
ki	285	450 × 375 (60)	37.8	P4 2.83 GHz
vh	459.4	512 × 512 (48)	27.4	P4 3.2 GHz
hg	505	450 × 375 (60)	21.3	n/a
hh	582	450 × 375 (60)	18.5	n/a
tc	600	450 × 375 (60)	18	Core Duo 2.14 GHz
ab	673	450 × 375 (60)	16	P4 3.0 GHz
sb	800	450 × 375 (60)	13.5	Std PC 1.8 GHz
ae	100	320 × 240 (16)	12.2	P4 2 GHz
dd	863	450 × 375 (60)	11.7	P4 2.4 GHz
hc	213	320 × 240 (32)	11.5	PII 450 MHz
yl	161	384 × 288 (16)	10.9	P4 2.66 GHz
mg	980	450 × 375 (60)	10.3	n/a
mm	218	384 × 288 (20)	10.1	PIII 800 MHz
ha	1,000	450 × 375 (60)	10.1	Xeon 2.8 GHz
sl	320	256 × 256 (30)	6.1	PIII 500 MHz

If runtimes were given for multiple images in the original publication, then the one that resulted in the maximum Mde/s is included here

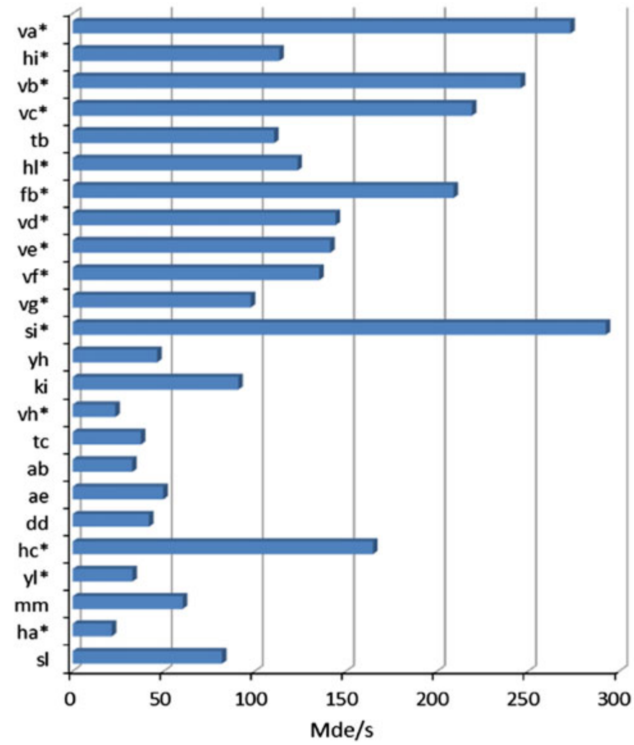

Fig. 10 Normalized runtimes of all real-time algorithms that provide sufficient hardware details. *Asterisks* Implementation optimized using SIMD instructions

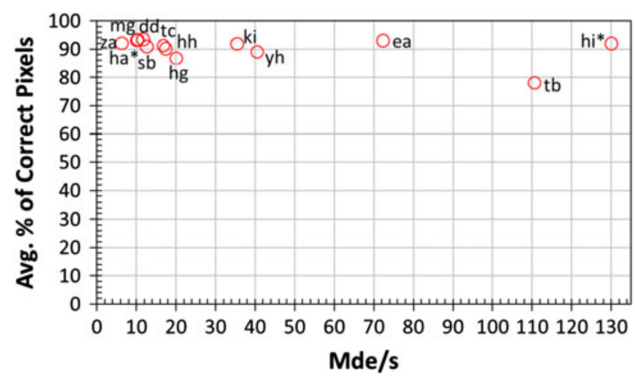

Fig. 11 A comparison of runtimes and accuracy for all real-time stereo vision algorithms that provide accuracy measurements on all four Middlebury images Tsukuba, Venus, Teddy, and Cones. *Asterisks* Implementation optimized using SIMD instructions

Other less common approaches include that of Min et al. HistoAggr (mg), where the cost aggregation is formulated as a relaxed joint histogram, with a likelihood-based disparity hypothesis. Einecke and Eggert employed a two stage summed normalized cross correlation method for their cost function in SNCC (ea). Ansar et al. based their approach on a bilateral filter BilateralSAD (ae). Kosov et al. apply a full approximation scheme to a multi-level adaption technique, which is a hierarchical approach to minimizing an energy function in RealtimeVar (ki). Tippetts et al. introduce a shape profile matching algorithm ProfileShape (tb) that does not aggregate costs in the traditional fashion, but uses intensity gradients to group and match shapes for each disparity level.

A few researchers have developed runtime-optimized versions of traditionally slow methods to achieve near real-time performance on standard PCs. Yang et al. developed a hierarchical belief propagation algorithm CSBP (yh) that is less accurate than other BP algorithms but can obtain up to 30× speedup over other BP algorithms depending on disparity range, and over a 70× reduction in memory usage. Tombari et al. implemented a variational support algorithm FastAggreg (tc) that adapts weights according to the difference in color channels (RGB) between pixels and distance from center pixel. Hirschmüller introduced a semi-global matching algorithm SemiGlob (ha) which reduces computational complexity compared to global methods by approximating a global energy minimization. Hirschm“uller also extended SemiGlob (ha) to improve accuracy by including mean shift segmentation in C-SemiGlob (hb). Gehrig later built upon SemiGlob (ha) for his algorithm ImproveSubPix (gb).

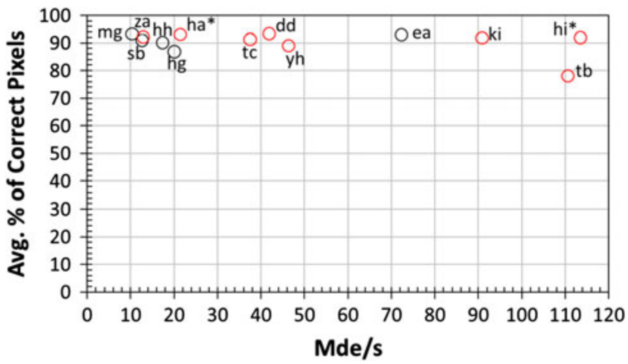

Fig. 12 A comparison of runtimes normalized for hardware and accuracy for all real-time stereo vision algorithms that provide accuracy measurements on all four Middlebury images Tsukuba, Venus, Teddy, and Cones (*red* markers). Algorithms lacking hardware details were not normalized (*black* markers). *Asterisks* Implementation optimized using SIMD instructions

Since many of these algorithm runtimes were measured on different hardware, Fig. 10 includes normalized measurements calculated according to the criteria in Sect. 3 Fig. 10 also shows which of these algorithms made use of SIMD instructions to achieve the reported runtime. Gong and Yang [42] and also Konolige [72] report that stereo vision algorithm implementations that use SIMD instruction optimizations achieve a 3–4× speedup. Humenberger et al. [58] was able to achieve close to a 22× speedup from his unoptimized code to his optimized code of RTCensus (hi). He calculated a 1.9× speedup from the use of both cores of the dual core processor, but cited heavy use of SIMD instructions as providing the biggest contribution of that optimization. If resource limitations prohibited the use of SIMD optimizations, then the algorithms in Fig. 10 marked with an asterisk would need to be reduced by at least a factor of 3–4×. This would result in ProfileShape (tb), RealtimeVar (ki), and RSR/TSDP (sl) being among the fastest alternatives, with RealtimeVar (ki) being more accurate than ProfileShape (tb) as seen in both Figs. 12 and 13. Approximately 20 Mde/s can be gained at the cost of roughly 14 percentage points of accuracy in going from RealtimeVar (ki) to ProfileShape (tb). Figure 13 shows that this loss of accuracy is due to lower edge fidelity or

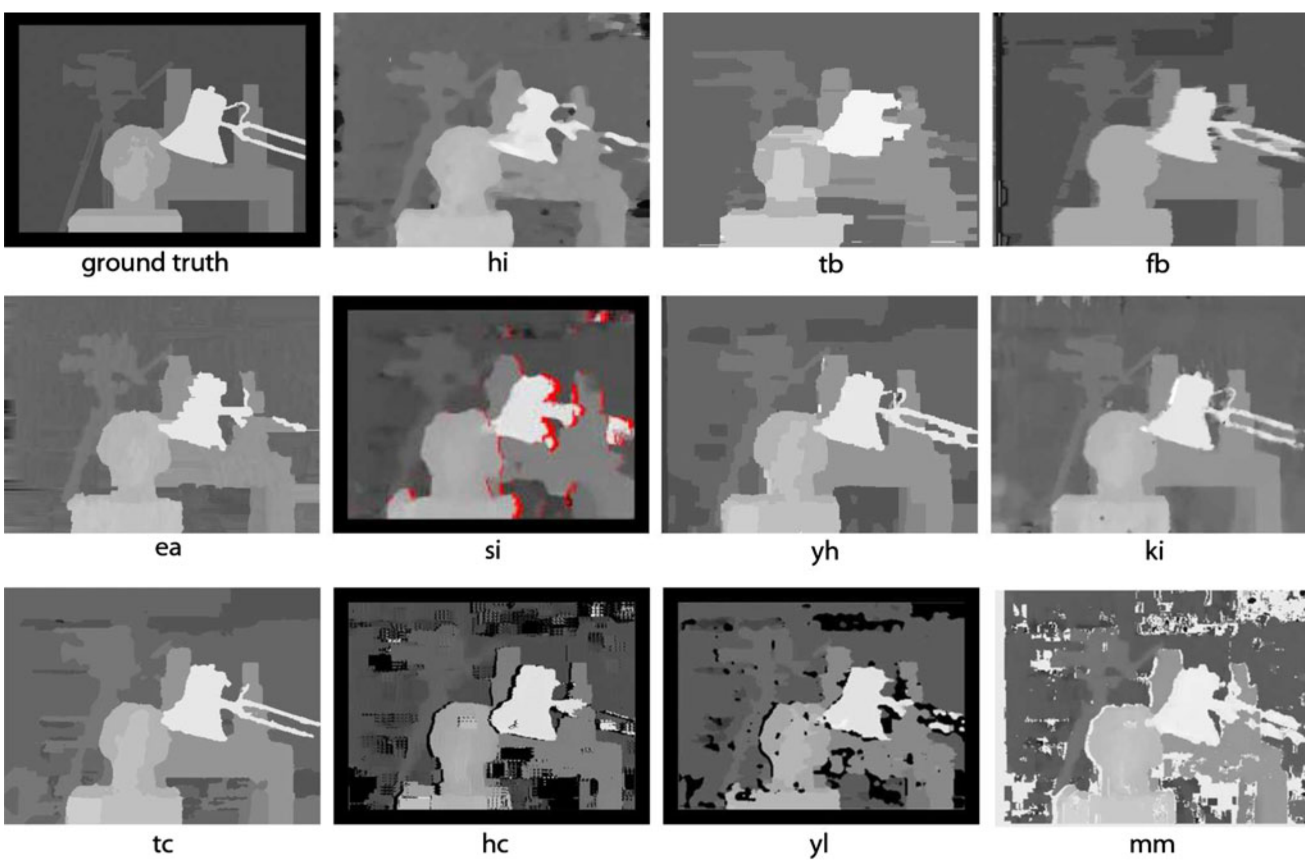

Fig. 13 A comparison of disparity map results for several real-time stereo vision algorithms of the Tsukuba dataset

Table 8 Published Middlebury performance measurements for algorithms not included on the Middlebury websiteScharstein (2012)

Alg	Tsukuba			Venus			Teddy			Cones			Avg % bad Pixels	Avg % correct "All" pixels	Hardware
	Nonocc	All	Disc	Nonocc	All	Disc	Nonocc	All	Disc	Nonocc	All	Disc			
me	1.07	1.48	5.73	0.09	0.25	1.15	4.10	6.22	10.90	2.42	7.25	6.95	3.97	96.20	CPU, GP
cb	2.04	3.92	10.60	1.60	3.00	16.00	7.33	15.20	18.70	3.35	12.90	9.47	8.68	91.25	CPU
vk	–	6.57	–	–	8.30	–	–	12.16	–	–	11.16	–	9.54	90.46	CPU
sr[1]	1.53	3.23	7.79	0.58	1.97	5.46	8.05	17.10	15.70	5.95	15.90	10.90	7.85	90.45	CPU
ca[1]	1.63	3.43	8.23	1.19	2.59	9.10	7.05	16.10	13.00	8.41	17.70	13.20	8.47	90.05	CPU
gc	–	2.27	–	–	1.22	–	–	19.40	–	–	17.40	–	10.07	89.93	CPU
dc	3.93	5.74	15.40	3.23	4.78	22.10	12.20	21.10	25.50	4.35	14.80	12.00	12.09	88.40	CPU
tb	7.78	9.54	31.10	10.50	12.00	34.70	34.00	40.80	51.60	16.20	25.50	35.20	25.74	78.04	CPU
sn[1]	42.10	42.90	29.90	40.60	41.40	30.70	51.40	56.10	–	53.40	58.00	44.50	45.08	50.40	CPU
ce	–	2.80	–	–	0.64	–	–	13.70	–	–	10.10	–	6.81	93.19	ASIC
zc	3.84	4.34	14.20	1.20	1.68	5.62	7.17	12.60	17.40	5.41	11.00	13.90	8.20	92.60	FPGA
ja	9.79	11.56	20.29	3.59	5.27	36.82	12.50	21.50	30.57	7.34	17.58	21.01	16.49	86.02	FPGA
ad	–	14.06	–	–	9.11	–	–	12.81	–	–	8.45	–	11.11	88.89	DSP
cd	21.50	21.70	48.70	16.50	17.80	29.90	26.30	33.60	35.10	24.20	32.40	31.00	28.23	73.63	DSP
gl	3.04	3.59	11.30	1.56	2.00	8.21	6.90	12.30	16.00	4.87	10.30	10.70	7.56	92.95	GPU
gk	2.27	3.61	11.20	3.57	4.61	19.80	10.90	18.80	23.20	5.92	14.30	13.80	11.00	89.67	GPU
zd	8.29	10.30	22.40	6.20	7.51	26.60	12.50	21.20	25.70	5.83	14.50	10.30	14.28	86.62	GPU
gj	3.88	5.88	15.00	7.12	8.34	26.60	13.70	21.40	25.40	11.70	20.40	20.90	15.03	86.00	GPU
gi	4.01	5.90	13.00	8.80	10.10	12.40	15.40	23.30	24.90	9.07	18.00	15.30	13.35	85.68	GPU
gh	3.29	5.09	10.50	9.82	11.00	17.90	15.20	22.30	22.90	14.10	21.70	21.30	14.59	84.98	GPU
gf	5.13	7.11	23.20	9.18	10.30	35.40	16.90	24.50	34.00	9.94	18.90	20.80	17.95	84.80	GPU
ci[2]	–	–	–	–	–	–	–	–	–	–	–	–	15.62	84.38	GPU
gg	4.46	6.03	15.70	10.90	11.90	11.00	17.90	24.60	27.10	15.80	22.90	24.30	16.05	83.64	GPU
ye[2]	–	–	–	–	–	–	–	–	–	–	–	–	17.32	82.68	GPU
yf[2]	–	–	–	–	–	–	–	–	–	–	–	–	19.69	80.31	GPU

Notes-Citations for performance measurements if not from original publication: [1] Cassisa [19] [2] Zhao and Taubin [162]

"streaky" results similar to many dynamic programming and scanline optimization methods.

Figures 11 and 12 show that algorithms SegTreeDP (dd), SemiGlob (ha), SNCC (ea), HistoAggr (mg), RealtimeVar (ki), and RTCensus (hi) are all very similar in accuracy, all within 1.5 percentage points of each other at 92 % average correct pixels or greater. Although Humenberger's census algorithm RTCensus (hi) appears to be near pareto optimal in both graphs in terms of speed and accuracy, it is the only algorithm included here with a reported runtime on a processor using more than one core. Humenberger reported the runtime of RTCensus (hi) without SIMD optimization and two threads as being 5.82 Mde/s, which were among the slowest of the algorithms in this section.

To aid the reader in calibrating the significance of the accuracy measurements in Figs. 11 and 12, the disparity maps of the Tsukuba stereo images produced by several of the real-time algorithms are included in Fig. 13. These disparity results allow the comparison of those algorithms for which information to calculate the average percentage of correct pixels is not available, e.g. DistinctSAD (si), FastAggreg (tc), SAD-MW (hc), DSI matrix/SAD (yl), and MeanSAD (mm). Although ProfileShape (tb) has the lowest accuracy of the algorithms shown in Fig. 12, it has similar quality of results to DistinctSAD (si), SAD-MW (hc), DSI matrix/SAD (yl), and MeanSAD (mm) on the Tsukuba images. Both percentages of bad pixel error rates and disparity map results are available on the Middlebury site [116] for all algorithms that have submitted results there. Many publications have included error measurements on the Middlebury data sets, but have not submitted them to the Middlebury site. We included those published results here in Table 8.

8 Conclusion

A review of available stereo vision algorithms has been given, categorizing them according to their suitability for real-time implementation on resource-limited platforms. Questions of what options exist in real-time algorithms for resource-limited systems have been addressed. Discussion was given of the trade-off in accuracy that is made in various cases to achieve real-time performance. In most circumstances, determining an acceptable trade-off between speed and accuracy is dependent upon the target application. Various applications require different refresh rates, image size, and sensor accuracy (Desouza and Kak [32], Bruch et al. [18], Howard [55]). An attempt has been made to include all available data for the current state of stereo vision so that the different criteria for a wide range of applications can be evaluated. Several algorithms have been presented that can obtain useful disparity results and achieve real-time performance, even when implemented on resource-limited systems.

9 Resources

For those applications where development time is one of the major limitations, available source code may be a determining factor in algorithm selection. Those resources where source code for stereo vision algorithms was available at time of publication include the following: SGM (hb) through the OpenCV Library [26]. Sum of Squared Differences (sd), Dynamic Programming (se), and Scanline Optimization (sf) along with a Belief Propagation implementation, and Yoon and Kweon's adaptive support-weight algorithm (yi) at Middlebury website (Scharstein [116]). A Max-Flow Graph Cut algorithm by Boykov and Kolmogorov is available at IST Austria [62]. A Graph Cuts implementation for GPUs called CudaCuts by Vineet and Narayanan [137] is available at Center for Visual Information Technology [20].

References

1. Ambrosch, K., Kubinger, W.: Accurate hardware-based stereo vision. Comput. Vision Image Underst. **114**, 1303–1316 (2010) (aCM ID: 1866603)
2. Ambrosch, K., Humenberger, M., Kubinger, W., Steininger, A.: Sad-based stereo matching using fpgas. In: Kisaanin, B., Bhattacharyya, S.S., Chai, S., (eds) Embedded, Computer Vision, pp. 121–138. Springer, London (2009) (Advances in Pattern Recognition)
3. Ambrosch, K., Zinner, C., Leopold, H.: A miniature embedded stereo vision system for automotive applications. In: Proceedings of IEEE 26th Convention of Electrical and Electronics Engineers in Israel (IEEEI), pp. 000786–000789 (2010)
4. Ansar, A., Castano, A., Matthies, L.: Enhanced real-time stereo using bilateral filtering. In: Proceedings of 2nd International Symposium. 3D Data Processing, Visualization and Transmission 3DPVT 2004, pp. 455–462 (2004)
5. Banno, A., Ikeuchi, K.: Disparity map refinement and 3d surface smoothing via directed anisotropic diffusion. In: 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 1870–1877 (2009)
6. Ben-Ari, R., Sochen, N.: Stereo matching with Mumford–Shah regularization and occlusion handling. IEEE Transact. Pattern Anal. Mach. Intell. **32**(11), 2071–2084 (2010)
7. Bhusnurmath, A., Taylor, C.J.: Solving stereo matching problems using interior point methods. In: Fourth International Symposium on 3D Data Processing, Visualization and Transmission, pp. 321–329 (2008)
8. Bleyer, M., Gelautz, M.: A layered stereo matching algorithm using image segmentation and global visibility constraints. ISPRS J Photogramm Remote Sens **59**(3), 128–150 (2005)
9. Bleyer, M., Gelautz, M., Rother, C., Rhemann, C.: A stereo approach that handles the matting problem via image warping. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 501–508 (2009)
10. Bleyer, M., Rother, C., Kohli, P.: Surface stereo with soft segmentation. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 1570–1577 (2010)
11. Bleyer, M., Rhemann, C., Rother, C.: Patchmatch stereo—stereo matching with slanted support windows. In: British Machine Vision Conference (BMVC) (2011a)
12. Bleyer, M., Rother, C., Kohli, P., Scharstein, D., Sinha, S.: Object stereo—joint stereo matching and object segmentation. In: IEEE Computer Vision and Pattern Recognition (CVPR) (2011b)
13. Bobick, A.F., Intille, S.S.: Large occlusion stereo. Int. J. Comput. Vision **33**, 181–200, (1999). doi:10.1023/A:1008150329890
14. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transact. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
15. Brockers, R.: Cooperative stereo matching with color-based adaptive local support. In: Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns, pp. 1019–1027. Springer, Berlin, (2009) aCM ID: 1618054
16. Brockers, R., Hund, M., Mertsching, B.: Stereo vision using cost-relaxation with 3d support regions. Image and Vision Computing New Zealand (IVCNZ) (2005)
17. Brown, M.Z., Burschka, D., Hager, G.D.: Advances in computational stereo. IEEE Transact. Pattern Anal. Mach. Intell. **25**(8), 993–1008 (2003)
18. Bruch, M., Lum, J., Yee, S., Tran, N.: Advances in autonomy for small ugvs. SPIE Proc 5804: Unmanned Ground Vehicle Technology VII, Orlando (2005)
19. Cassisa, C.: Local vs global energy minimization methods: Application to stereo matching. In: Proceedings of IEEE International Progress in Informatics and Computing (PIC) Conference, vol. 2, pp. 678–683 (2010)
20. Center for Visual Information Technology: CUDA Cuts. http://cvit.iiit.ac.in/index.php?page=resources (2011)
21. Chan, S.O.Y., Wong, Y.P., Daniel, J.K.: Dense stereo correspondence based on recursive adaptive size multi-windowing. In: Proceedings of Image and Vision Computing, vol. 1, pp. 256–260. New Zealand (2003)
22. Chang, N., Lin, T.M., Tsai, T.H., Tseng, Y.C., Chang, T.S.: Real-time dsp implementation on local stereo matching. In: Proceedings of IEEE International Multimedia and Expo Conference, pp. 2090–2093 (2007)
23. Chang, N.Y.C., Tsai, T.H., Hsu, B.H., Chen, Y.C., Chang, T.S.: Algorithm and architecture of disparity estimation with mini-

census adaptive support weight. IEEE Transact. Circuits Syst. Video Technol. **20**(6), 792–805 (2010)
24. Chang, X., Zhou, Z., Wang, L., Shi, Y., Zhao, Q.: Real-time accurate stereo matching using modified two-pass aggregation and winner-take-all guided dynamic programming. In: Proceedings of International 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT) Conference, pp. 73–79 (2011)
25. Chen, W., Zhang, M.J., Xiong, Z.H.: Fast semi-global stereo matching via extracting disparity candidates from region boundaries. IET Comput. Vision **5**(2), 143–150 (2011)
26. Cornells, N., Van Gool, L.: Real-time connectivity constrained depth map computation using programmable graphics hardware. In: Proceedings of IEEE Computer Society Conference Computer Vision and Pattern Recognition CVPR 2005, vol. 1, pp. 1099–1104 (2005)
27. Cuadrado, C., Zuloaga, A., Martin, J.L., Lazaro, J., Jimenez, J.: Real-time stereo vision processing system in a fpga. In: Proceedings of IECON 2006—32nd Annual Conference. IEEE Industrial Electronics, pp. 3455–3460 (2006)
28. De-Maeztu, L., Mattoccia, S., Villanueva, A., Cabeza, R.: Linear stereo matching. In: A13th International Conference on Computer Vision (ICCV2011) (2011a)
29. De-Maeztu, L., Villanueva, A., Cabeza, R.: Stereo matching using gradient similarity and locally adaptive support-weight. Pattern Recognit. Lett. **32**(13), 1643–1651 (2011)
30. Demoulin, C., Droogenbroeck, M.V.: A method based on multiple adaptive windows to improve the determination of disparity maps. In: Proceedings of IEEE Workshop on Circuit, Systems and Signal Processing (2005)
31. Deng, Y., Lin, X.: A fast line segment based dense stereo algorithm using tree dynamic programming. In: Computer Vision—ECCV 2006, Lecture Notes in Computer Science, vol. 3953, Springer, Heidelberg, pp. 201–212. (2006). doi:10.1007/11744078_16
32. Desouza, G.N., Kak, A.C.: Vision for mobile robot navigation: a survey. IEEE Transact. Pattern Anal. Mach. Intell. **24**(2), 237–267 (2002)
33. Einecke, N., Eggert, J.: A two-stage correlation method for stereoscopic depth estimation. In: Digital Image Computing: Techniques and Applications, IEEE Computer Society, Los Alamitos, CA, vol. 0, pp. 227–234 (2010)
34. El-Etriby, S., Al-Hamadi, A.K., Michaelis, B.: Dense depth map reconstruction by phase difference-based algorithm under influence of perspective distortion. Mach. Grap. Vision Int. J. **15**(3), 349–361 (2006)
35. El-Etriby, S., Al-Hamadi, A., Michaelis, B.: Dense stereo correspondence with slanted surface using phase-based algorithm. In: Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on, pp. 1807–1813 (2007)
36. Felzenszwalb, P., Huttenlocher, D.: Efficient belief propagation for early vision. Int. J. Comput. Vision **70**(1), 41–54 (2006)
37. Forstmann, S., Kanou, Y., Ohya, J., Thuering, S., Schmitt, A.: Real-time stereo by using dynamic programming. In: Proceedings of Conference Computer Vision and Pattern Recognition Workshop CVPRW '04 (2004)
38. Gales, G., Crouzil, A., Chambon, S.: A region-based randomized voting scheme for stereo matching. In: Advances in Visual Computing, Lecture Notes in Computer Science, vol. 6454, Springer, Berlin, pp. 182–191 (2010) doi:10.1007/978-3-642-17274-8_18
39. Gehrig, S., Franke, U.: Improving sub-pixel accuracy for long range stereo. In: ICCV VRML workshop (2007)
40. Gerrits, M., Bekaert, P.: Local stereo matching with segmentation-based outlier rejection. In: Proceedings of 3rd Canadian Conference Computer and Robot Vision (2006)
41. Goldberg, S.B., Matthies, L.: Stereo and imu assisted visual odometry on an omap3530 for small robots. In: Proceedings IEEE Computer Society Conference Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 169–176 (2011)
42. Gong, M., Yang, R.: Image-gradient-guided real-time stereo on graphics hardware. In: Proceedings of Fifth International Conference 3-D Digital Imaging and Modeling 3DIM 2005, pp. 548–555 (2005a)
43. Gong, M., Yang, Y.H.: Near real-time reliable stereo matching using programmable graphics hardware. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005. vol. 1, pp. 924–931 (2005b)
44. Gong, M., Yang, R., Wang, L., Gong, M.: A performance study on different cost aggregation approaches used in real-time stereo matching. Int. J. Comput. Vision (IJCV) (2007)
45. Gong, M., Zhang, Y., Yang, Y.H.: Near-real-time stereo matching with slanted surface modeling and sub-pixel accuracy. Pattern Recognit. **44**(10–11), 2701–2710 (semi-Supervised Learning for Visual Content Analysis and Understanding)
46. Grauer-Gray, S., Kambhamettu, C.: Hierarchical belief propagation to reduce search space using cuda for stereo and motion estimation. In: 2009 Workshop on Applications of Computer Vision (WACV), pp. 1–8 (2009)
47. Gu, Z., Su, X., Liu, Y., Zhang, Q.: Local stereo matching with adaptive support-weight, rank transform and disparity calibration. Pattern Recognit. Lett. **29**(9), 1230–1235 (2008)
48. Gupta, R., Cho, S.Y.: A correlation-based approach for real-time stereo matching. In: Advances in Visual Computing, Lecture Notes in Computer Science, Springer Berlin, vol. 6454, pp. 129–138 (2010a). doi:10.1007/978-3-642-17274-8_13
49. Gupta, R.K., Cho, S.Y.: Real-time stereo matching using adaptive binary window. In: Proceedings of the 3DPVT, Paris (2010b)
50. Hirschmuller, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005. vol. 2, pp 807–814 (2005)
51. Hirschmuller, H. Stereo vision in structured environments by consistent semi-global matching. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2386–2393 (2006)
52. Hirschmuller, H., Scharstein, D.: Evaluation of stereo matching costs on images with radiometric differences. IEEE Transact. Pattern Anal. Mach. Intell. **31**(9), 1582–1599 (2009)
53. Hirschmuller, H., Innocent, P.R., Garibaldi, J.: Real-time correlation-based stereo vision with reduced border errors. Int. J. Comput. Vision **47**, 229–246 (2002)
54. Hosni, A., Bleyer, M., Gelautz, M., Rhemann, C.: Local stereo matching using geodesic support weights. In: 2009 16th IEEE International Conference on Image Processing (ICIP), pp. 2093–2096 (2009)
55. Howard, A.: Real-time stereo visual odometry for autonomous ground vehicles. In: Proceedings of IEEE/RSJ International Conference Intelligent Robots and Systems IROS 2008, pp. 3946–3952 (2008)
56. Hu. W., Zhang, K., Sun, L., Li, J., Li, Y., Yang, S.: Virtual support window for adaptive-weight stereo matching. In: Visual Communications and Image Processing (VCIP) (2011)
57. Humenberger, M., Zinner, C., Kubinger, W.: Performance evaluation of a census-based stereo matching algorithm on embedded and multi-core hardware. In: Proceedings of 6th International Symposium Image and Signal Processing and Analysis ISPA 2009, pp. 388–393 (2009)
58. Humenberger, M., Engelke, T., Kubinger, W.: A census-based stereo vision algorithm using modified semi-global matching and plane fitting to improve matching quality. In: 2010 IEEE

Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 77–84 (2010a)
59. Humenberger, M., Zinner, C., Weber, M., Kubinger, W., Vincze, M.: A fast stereo matching algorithm suitable for embedded real-time systems. Comput. Vision Image Underst. **114**(11), 1180–1202 (2010b)
60. Ishikawa, H.: Higher-order gradient descent by fusion-move graph cut. In: Proceedings of IEEE 12th International Computer Vision Conference, pp. 568–574 (2009)
61. Ishikawa, H., Geiger, D.: Occlusions, discontinuities, and epipolar lines in stereo. In: In European Conference on Computer Vision, pp. 232–248 (1998)
62. IST Austria (2009) Maxflow. http://pub.ist.ac.at/~vnk/software.html
63. Jin, S., Cho, J., Pham, X.D., Lee, K.M., Park, S.K., Kim, M., Jeon, J.W.: Fpga design and implementation of a real-time stereo vision system. IEEE Transact. Circuits Syst. Video Technol. **20**(1), 15–26 (2010)
64. Kalarot, R., Morris, J.: Comparison of fpga and gpu implementations of real-time stereo vision. In: Proceedings of IEEE Computer Society Conference Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 9–15 (2010)
65. Kang, S.B., Szeliski, R., Chai, J.: Handling occlusions in dense multi-view stereo. In: Proceedings of IEEE Computer Society Conference Computer Vision and Pattern Recognition CVPR 2001, vol. 1 (2001)
66. Khaleghi, B., Ahuja, S., Wu, Q.: An improved real-time miniaturized embedded stereo vision system (mesvs-ii). In: Proceedings of IEEE Computer Society Conference Computer Vision and Pattern Recognition Workshops CVPRW '08, pp 1–8 (2008)
67. Kim, J., Hwangbo, M., Kanade, T.: Parallel algorithms to a parallel hardware: Designing vision algorithms for a gpu. In: Workshop on Embedded Computer Vision (ECV), 2009 (held in conjunction with ICCV) (2009)
68. Kim, J.C., Lee, K.M., Choi, B.T., Lee, S.U.: A dense stereo matching using two-pass dynamic programming with generalized ground control points. In: Proceedings of IEEE Computer Society Conference Computer Vision and Pattern Recognition CVPR 2005, vol. 2, pp. 1075–1082 (2005)
69. Klaus, A., Sormann, M., Karner, K.: Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In: 18th International Conference on Pattern Recognition. ICPR 2006. vol. 3, pp. 15–18 (2006)
70. Kolmogorov, V., Zabih, R.: Computing visual correspondence with occlusions using graph cuts. In: Proceedings of Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001. vol. 2, pp. 508–515 (2001)
71. Kolmogorov, V., Zabih, R.: Multi-camera scene reconstruction via graph cuts. Eur. Conf. Comput. Vision **3**, 82–96 (2002)
72. Konolige, K.: Small vision systems: hardware and implementation. In: 8th International Symposium on Robotics Research, pp. 111–116 (1997)
73. Kosov, S., Thormahlen, T., Seidel, H.P.: Accurate real-time disparity estimation with variational methods. In: ISVC '09 Proceedings of the 5th International Symposium on Advances in Visual Computing: Part I (2009)
74. Kuhn, M., Moser, S., Isler, O., Gurkaynak, F.K., Burg, A., Felber, N., Kaeslin, H., Fichtner, W.: Efficient asic implementation of a real-time depth mapping stereo vision system. In: Proceedings of IEEE 46th Midwest Symposium Circuits and Systems, vol. 3, pp. 1478–1481 (2003)
75. Larsen, E., Mordohai, P., Pollefeys, M., Fuchs, H.: Temporally consistent reconstruction from multiple video streams using enhanced belief propagation. In: IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007. pp. 1–8 (2007)
76. Lei, C., Selzer, J., Yang, Y.H.: Region-tree based stereo using dynamic programming optimization. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2378–2385 (2006)
77. Liang, C.K., Cheng, C.C., Lai, Y.C., Chen, L.G., Chen, H.H.: Hardware-efficient belief propagation. IEEE Transact. Circuits Syst. Video Technol. **21**(5), 525–537 (2011)
78. Liu, T., Zhang, P., Luo, L.: Dense stereo correspondence with contrast context histogram, segmentation-based two-pass aggregation and occlusion handling. In: Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology, Springer, Berlin, PSIVT '09, pp. 449–461 (2008) (aCM ID: 1505991)
79. Lu, J., Lafruit, G., Catthoor, F.: Fast variable center-biased windowing for high-speed stereo on programmable graphics hardware. In: Proceedings of IEEE International Conference Image Processing ICIP 2007, vol. 6 (2007a)
80. Lu, J., Rogmans, S., Lafruit, G., Catthoor, F.: Real-time stereo correspondence using a truncated separable Laplacian kernel approximation on graphics hardware. In: Proceedings of IEEE Int Multimedia and Expo Conference, pp. 1946–1949 (2007b)
81. Lu, J., Lafruit, G., Catthoor, F.: Anisotropic local high-confidence voting for accurate stereo correspondence. In: Proceedings of SPIE, San Jose, pp. 68,120J–68,120J–12 (2008)
82. van der Mark, W., Gavrila, D.M.: Real-time dense stereo for intelligent vehicles. IEEE Transact. Intell. Transp. Syst. **7**(1), 38–50 (2006)
83. Masrani, D., MacLean, W.: A real-time large disparity range stereo-system using fpgas. In: IEEE International Conference on Computer Vision Systems, 2006 ICVS '06. p. 13 (2006)
84. Mattoccia, S.: A locally global approach to stereo correspondence. In: 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 1763–1770 (2009)
85. Mattoccia, S., Tombari, F., Stefano, L.D.: Stereo vision enabling precise border localization within a scanline optimization framework. In: Computer Vision ACCV 2007, Lecture Notes in Computer Science, Springer, Berlin, vol. 4844, pp. 517–527 (2007) doi:10.1007/978-3-540-76390-1_51
86. Mattoccia, S., Giardino, S., Gambini, A.: Accurate and efficient cost aggregation strategy for stereo correspondence based on approximated joint bilateral filtering. In: Computer Vision—ACCV 2009, Lecture Notes in Computer Science, Springer, Berlin, vol. 5995, pp. 371–380 (2010)
87. Mei, X., Sun, X., Zhou, M., Jiao, S., Wang, H., Zhang, X.: On building an accurate stereo matching system on graphics hardware Technical Report, Samsung Advanced Institute of Technology (2011)
88. Min, D., Sohn, K.: Cost aggregation and occlusion handling with wls in stereo matching. IEEE Transact. Image Process. **17**(8), 1431–1442 (2008)
89. Min, D., Luy, J., Do, M.N.: A revisit to cost aggregation in stereo matching: How far can we reduce its computational redundancy? In: International Conference on Computer Vision (2011)
90. Mingxiang, L., Yunde, J.: Stereo vision system on programmable chip (svsoc) for small robot navigation. In: Proceedings of IEEE/RSJ International Intelligent Robots and Systems Conference, pp. 1359–1365 (2006)
91. Miyajima, Y., Maruyama, T.: A real-time stereo vision system with fpga. In: Field-Programmable Logic and Applications, Lecture Notes in Computer Science, vol. 2778, Springer, Berlin, pp. 448–457 (2003)
92. Miyazaki, D., Matsushita, Y., Ikeuchi, K.: Interactive shadow removal from a single image using hierarchical graph cut. In: Computer Vision—ACCV 2009, Lecture Notes in Computer Science, Springer, Berlin, vol. 5994, pp. 234–245 (2010)

93. Montserrat, T., Civit, J., Escoda, O., Landabaso, J.L.: Depth estimation based on multiview matching with depth/color segmentation and memory efficient belief propagation. In: 2009 16th IEEE International Conference on Image Processing (ICIP), pp. 2353–2356 (2009)
94. Mordohai, P., Medioni, G.: Stereo using monocular cues within the tensor voting framework. IEEE Transact. Pattern Anal. Mach. Intell. **28**(6), 968–982 (2006) (pMID: 16724590)
95. Muhlmann, K., Maier, D., Hesser, R., Manner, R.: Calculating dense disparity maps from color stereo images, an efficient implementation. In: Proceedings of IEEE Workshop Stereo and Multi-Baseline Vision (SMBV 2001), pp. 30–36 (2001)
96. Mukherjee, D., Wang, G., Wu, Q.: Stereo matching algorithm based on curvelet decomposition and modified support weights. In: 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 758–761 (2010)
97. Nalpantidis, L., Gasteratos, A.: Biologically and psychophysically inspired adaptive support weights algorithm for stereo correspondence. Robotics Auton. Syst. **58**(5), 457–464 (2010)
98. Nalpantidis, L., Gasteratos, A.: Stereo vision for robotic applications in the presence of non-ideal lighting conditions. Image Vision Comput. **28**(6), 940–951 (2010)
99. Nalpantidis, L., Sirakoulis, G.C., Gasteratos, A.: Review of stereo vision algorithms: From software to hardware. Int. J. Optomechatron. **2**(4), 435–462 (2008)
100. Naoulou, A., Boizard, J.L., Fourniols, J.Y., Devy, M.: An alternative to sequential architectures to improve the processing time of passive stereovision algorithms. In: Proceedings of International Conference Field Programmable Logic and Applications FPL '06, pp. 1–4 (2006)
101. Ogale, A.S., Aloimonos, Y.: Shape and the stereo correspondence problem. Int. J. Comput. Vision 65 (2005)
102. Olague, G., de Vega, F.F., Prez, C.B., Lutton, E.: The infection algorithm: an artificial epidemic approach for dense stereo matching. In: Parallel Problem Solving from Nature - PPSN VIII, Lecture Notes in Computer Science, vol. 3242, Springer, Berlin, pp. 622–632 (2004)
103. OpenCV Library (2010) http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html
104. Papadakis, N., Caselles, V.: Multi-label depth estimation for graph cuts stereo problems. J. Math. Imaging Vision **38**(1), 70–82 (2010)
105. Park, S., Jeong, H.: Real-time stereo vision fpga chip with low error rate. In: Proceedings of International Conference Multimedia and Ubiquitous Engineering MUE '07, pp. 751–756 (2007)
106. PassMark Software: Cpu benchmarks. http://www.cpubenchmark.net/cpu_list.php (2012)
107. Perez, J.M., Sanchez, P., Martinez, M.: High memory throughput fpga architecture for high-definition belief-propagation stereo matching. In: Proceedings of 3rd International Signals, Circuits and Systems (SCS) Conference, pp. 1–6 (2009)
108. Perri, S., Colonna, D., Zicari, P., Corsonello, P.: Sad-based stereo matching circuit for fpgas. In: Proceedings of 13th IEEE International Conference Electronics, Circuits and Systems ICECS '06, pp. 846–849 (2006)
109. Pock, T., Schoenemann, T., Graber, G., Bischof, H., Cremers, D.: A convex formulation of continuous multi-label problems. In: Proceedings of the 10th European Conference on Computer Vision: Part III, Springer, Berlin, pp. 792–805 (2008) (aCM ID: 1478235)
110. Psota, E.T., Kowalczuk, J., Carlson, J., Perez, L.C.: A local iterative refinement method for adaptive support-weight stereo matching. In: International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV) (2011)
111. Rhemann, C., Hosni, A., Bleyer, M., Rother, C., Gelautz, M.: Fast cost-volume filtering for visual correspondence and beyond. In: IEEE Computer Vision and Pattern Recognition (CVPR) (2011)
112. Richardt, C., Orr, D., Davies, I., Criminisi, A., Dodgson, N.: Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. In: Computer Vision - ECCV 2010, Lecture Notes in Computer Science, vol. 6313, Springer, Berlin, pp. 510–523 (2010)
113. Sabihuddin, S., MacLean, W.J.: Maximum-likelihood stereo correspondence using field programmable gate arrays. In: The 5th International Conference on Computer Vision Systems (2007)
114. Salmen, J., Schlipsing, M., Edelbrunner, J., Hegemann, S., Lke, S.: Real-time stereo vision: Making more out of dynamic programming. In: Computer Analysis of Images and Patterns, Lecture Notes in Computer Science, vol. 5702, Springer, Berlin, pp. 1096–1103 (2009)
115. Samarawickrama, M.G.: Performance evaluation of vision algorithms on fpga. Master's thesis, University of Moratuwa, Sri Lanka (2010)
116. Scharstein, D.: Middlebury stereo evaluation. http://vision.middlebury.edu/stereo/eval/ (2012)
117. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vision **47**, 7–42 (2002)
118. Stankiewicz, O., Wegner, K.: Depth map estimation software version 2. Techinical report, ISO/IEC MPEG meeting M15338 (2008)
119. Stankiewicz, O., Wegner, K.: Depth map estimation software version 3. Techinical report, ISO/IEC MPEG meeting M15540 (2009)
120. Stefano, L.D., Marchionni, M., Mattoccia, S.: A fast area-based stereo matching algorithm. Image Vision Comput. **22**(12), 983–1005 (2004)
121. Strecha, C., Fransens, R., Gool, L.V.: Combined depth and outlier estimation in multi-view stereo. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2394–2401 (2006)
122. Sun, C.: Fast stereo matching using rectangular subregioning and 3d maximum-surface techniques. Int. J. Comput. Vision **47**, 99–117 (2002). doi:10.1023/A:1014585622703
123. Sun, J., Li, Y., Kang, S., Shum, H.Y.: Symmetric stereo matching for occlusion handling. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005. vol. 2, pp. 399–406 (2005)
124. Sun, X., Mei, X., Jiao, S., Zhou, M., Wang, H.: Stereo matching with reliable disparity propagation. In: Proceedings of Int 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT) Conference, pp. 132–139 (2011)
125. Szeliski, R., Zabih, R.: An experimental comparison of stereo algorithms. Vision Algorithms: Theory and Practice, pp. 1–19 (2000)
126. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. IEEE Transact. Pattern Anal. Mach. Intell. **30**(6), 1068–1080 (2008)
127. Taguchi, Y., Wilburn, B., Zitnick, C.: Stereo reconstruction with mixed pixels using adaptive over-segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008. pp. 1–8 (2008)
128. Tippetts, B.J., Lee, D.J., Archibald, J.K., Lillywhite, K.D.: Dense disparity real-time stereo vision algorithm for resource-limited systems. IEEE Transact. Circuits Syst. Video Technol. **21**(10), 1547–1555 (2011)

129. Tombari, F., Mattoccia, S., Stefano, L.D.: Segmentation-based adaptive support for accurate stereo correspondence. In: Proceedings of the 2nd Pacific Rim conference on Advances in image and video technology, Springer, Berlin, PSIVT'07, pp. 427–438 (2007)
130. Tombari, F., Mattoccia, S., Di Stefano, L., Addimanda, E.: Classification and evaluation of cost aggregation methods for stereo correspondence. In: Proceedings of IEEE Conference Computer Vision and Pattern Recognition CVPR 2008, pp. 1–8 (2008a)
131. Tombari, F., Mattoccia, S., Stefano, L.D., Addimanda, E.: Near real-time stereo based on effective cost aggregation. In: 19th International Conference on Pattern Recognition, 2008. ICPR 2008. pp. 1–4 (2008b)
132. Trinh, H., McAllester, D.: Unsupervised learning of stereo vision with monocular cues. In: British Machine Vision Conference (2009)
133. Vanetti, M., Gallo, I., Binaghi, E.: Dense two-frame stereo correspondence by self-organizing neural network. In: Proceedings of the 15th International Conference on Image Analysis and Processing, Springer, Berlin, ICIAP '09, pp. 1035–1042 (2009) (aCM ID: 1618209)
134. Veksler, O.: Fast variable window for stereo correspondence using integral images. In: Proceedings of IEEE Computer Society Conference Computer Vision and Pattern Recognition, vol. 1 (2003)
135. Veksler, O.: Stereo correspondence by dynamic programming on a tree. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005. vol. 2, pp. 384–390 (2005)
136. Venkatesh, Y.V., Raja, S.K., Kumar, A.J.: On the application of a modified self-organizing neural network to estimate stereo disparity. IEEE Transact. Image Process. **16**(11), 2822–2829 (2007)
137. Vineet, V., Narayanan, P.J.: Cuda cuts: Fast graph cuts on the gpu. In: Proceedings of IEEE Computer Society Conference Computer Vision and Pattern Recognition Workshops CVPRW '08, pp. 1–8 (2008)
138. Wang, L., Yang, R.: Global stereo matching leveraged by sparse ground control points. In: IEEE Computer Vision and Pattern Recognition (CVPR) (2011)
139. Wang, L., Liao, M., Gong, M., Yang, R., Nister, D.: High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06), IEEE Computer Society, Washington, 3DPVT '06, pp. 798–805 (2006) (aCM ID: 1249375)
140. Wang, Z.F., Zheng, Z.G.: A region based stereo matching algorithm using cooperative optimization. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recogn. **0**, 1–8 (2008)
141. Woodfill, J., Von Herzen, B.: Real-time stereo vision on the parts reconfigurable computer. In: Proceedings of 5th Annual IEEE Symp FPGAs for Custom Computing Machines, pp. 201–210 (1997)
142. Woodfill, JI., Gordon, G., Jurasek, D., Brown, T., Buck, R.: The tyzx deepsea g2 vision system, ataskable, embedded stereo camera. In: Proceedings of Conference Computer Vision and Pattern Recognition Workshop CVPRW '06 (2006)
143. Woodford, O., Torr, P., Reid, I., Fitzgibbon, A.: Global stereo reconstruction under second-order smoothness priors. IEEE Transact. Pattern Anal. Mach. Intell. **31**(12), 2115–2128 (2009)
144. Xu, L., Jia, J.: Stereo matching: An outlier confidence approach. In: Computer Vision—ECCV 2008, Lecture Notes in Computer Science, vol. 5305. Springer, Berlin, pp. 775–787 (2008)
145. Xu, Y., Wang, D., Feng, T., Shum, H.Y.: Stereo computation using radial adaptive windows. In: Proceedings of 16th International Pattern Recognition Conference, vol. 3, pp. 595–598 (2002)
146. Yang, Q., Wang, L., Yang, R., Wang, S., Liao, M., Nister, D.: Real-time global stereo matching using hierarchical belief propagation. In: The British Machine Vision Conference, pp. 989–998 (2006)
147. Yang, Q., Yang, R., Davis, J., Nister, D.: Spatial-depth super resolution for range images. In: IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07. pp. 1–8 (2007)
148. Yang, Q., Engels, C., Akbarzadeh, A.: Near real-time stereo for weakly-textured scenes. In: British Machine Vision Conference (2008)
149. Yang, Q., Wang, L., Yang, R., Stewenius, H., Nister, D.: Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. IEEE Transact. Pattern Anal. Mach. Intell. **31**(3), 492–504 (2009)
150. Yang, Q., Wang, L., Ahuja, N.: A constant-space belief propagation algorithm for stereo matching. In: Proceedings of IEEE Conference Computer Vision and Pattern Recognition (CVPR), pp. 1458–1465 (2010)
151. Yang, R., Pollefeys, M.: Multi-resolution real-time stereo on commodity graphics hardware. In: Proceedings of IEEE Computer Society Conference Computer Vision and Pattern Recognition, vol. 1 (2003)
152. Yang, R., Pollefeys, M., Li, S.: Improved real-time stereo on commodity graphics hardware. In: Proceedings of Conference Computer Vision and Pattern Recognition Workshop CVPRW '04 (2004)
153. Yoon, K.J., Kweon, I.S.: Locally adaptive support-weight approach for visual correspondence search. In: Computer Vision and Pattern Recognition, pp. 924–931 (2005)
154. Yoon, K.J., Kweon, I.S.: Adaptive support-weight approach for correspondence search. IEEE Transact. Pattern Anal. Mach. Intell. **28**(4), 650–656 (2006)
155. Yoon, K.J., Kweon, I.S.: Stereo matching with the distinctive similarity measure. In: IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007. pp. 1–7 (2007)
156. Yoon, S., Park, S.K., Kang, S., Kwak, Y.K.: Fast correlation-based stereo matching with the reduction of systematic errors. Pattern Recogn. Lett. **26**(14), 2221–2231 (2005)
157. Yu, T., Lin, R.S., Super, B., Tang, B.: Efficient message representations for belief propagation. In: IEEE International Conference on Computer Vision. vol. 0, IEEE Computer Society, Los Alamitos, pp. 1–8 (2007)
158. Yu, W., Chen, T., Franchetti, F., Hoe, J.: High performance stereo vision designed for massively data parallel platforms. IEEE Transact. Circuits Syst. Video Technol. **20**(11), 1509–1519 (2010)
159. Zhang, K., Lu, J., Lafruit, G.: Cross-based local stereo matching using orthogonal integral images. IEEE Transact. Circuits Syst. Video Technol. **19**(7), 1073–1079 (2009)
160. Zhang, K., Lu, J., Lafruit, G., Lauwereins, R., Gool, L.V.: Real-time accurate stereo with bitwise fast voting on cuda. In: 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 794–800 (2009b)
161. Zhang, L., Zhang, K., Chang, T.S., Lafruit, G., Kuzmanov, G.K., Verkest, D.: Real-time high-definition stereo matching on fpga. In: Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays, ACM, New York, FPGA '11, pp. 55–64 (2011)
162. Zhao, Y., Taubin, G.: Real-time stereo on GPGPU using progressive multi-resolution adaptive windows. Image Vision Comput. **29**(6) 420–432 (2011)
163. Zinner, C., Humenberger, M.: Distributed real-time stereo matching on smart cameras. In: Proceedings of the Fourth ACM/

IEEE International Conference on Distributed Smart Cameras, ACM, New York, ICDSC '10, pp. 182–189 (2010)
164. Zitnick, C.L., Kanade, T.: A cooperative algorithm for stereo matching and occlusion detection. IEEE Transact. Pattern Anal. Mach. Intell. **22**(7), 675–684 (2000)
165. Zitnick, C.L., Kang, S.B.: Stereo for image-based rendering using image over-segmentation. Int. J. Comput. Vision **75**(1), 49–65 (2007)
166. Zitnick, C.L., Kang, S.B., Uyttendaele, M., Winder, S., Szeliski, R.: High-quality video view interpolation using a layered representation. In: ACM Transactions on Graphics (TOG), ACM, New York, SIGGRAPH '04, pp. 600–608 (2004) (aCM ID: 1015766)

Author Biographies

Beau Tippetts received his Ph.D. from the Department of Electrical and Computer Engineering at Brigham Young University, Provo, Utah, in 2012. His research interests include real-time vision algorithm optimization for resource limited systems, in part for use on hovering micro-UAVs. Specific areas include stereo vision, feature detection, and tracking. He is currently doing vision research at Smart Vision Works.

Dah-Jye Lee received his B.S. degree from National Taiwan University of Science and Technology in 1984, M.S. and Ph.D. degrees in electrical engineering from Texas Tech University in 1987 and 1990, respectively. He also received his MBA degree from Shenandoah University, Winchester, Virginia in 1999. He is currently a Professor in the Department of Electrical and Computer Engineering at Brigham Young University. He worked in the machine vision industry for 11 years prior to joining BYU in 2001. His research work focuses on Medical informatics and imaging, shape-based pattern recognition, hardware implementation of real-time 3-D vision algorithms and machine vision applications. Dr. Lee is a senior member of IEEE.

Kirt Lillywhite recieved his Ph.D. from the Department of Electrical and Computer Engineering at Brigham Young University, Provo, Utah, in 2012. His research interests include real-time image processing, pattern recognition, parallel processing, and robotic vision. He is currently doing research at Smart Vision Works.

James Archibald received the B.S. degree in mathematics from Brigham Young University, Provo, UT, in 1981 and the M.S. and Ph.D. degrees in computer science from the University of Washington, Seattle, in 1983 and 1987, respectively. He has been with the Department of Electrical and Computer Engineering, Brigham Young University since 1987. His research interests include robotics, multiagent systems, and machine vision. Dr. Archibald is a member of the ACM and Phi Kappa Phi.