Bauhaus-
Universität
Weimar

Real-time Stereo Matching

CV Project Summer 2019

Gissu Valentina Naghavi, Nhu Quang Dang
Jan Frederick Eick, Huy Khan Ha
Supervised by Mariya Kaisheva, Adrian Kreskowski

Outline

- Motivation and introduction
- Approach
  - Algorithms
  - Extensions
- Evaluation
- Conclusion and future work

Motivation

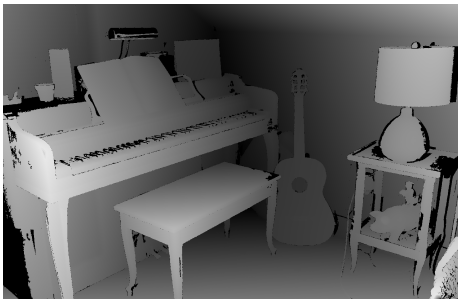
From 2D images to 3D

Left view

Right view

Motivation

From 2D images to 3D

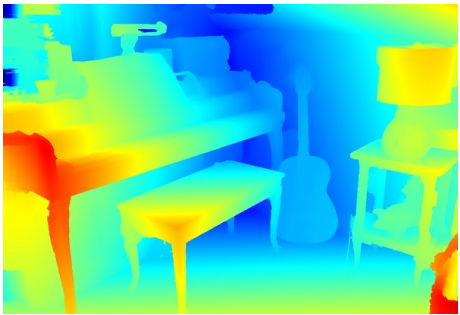


Dense disparity map

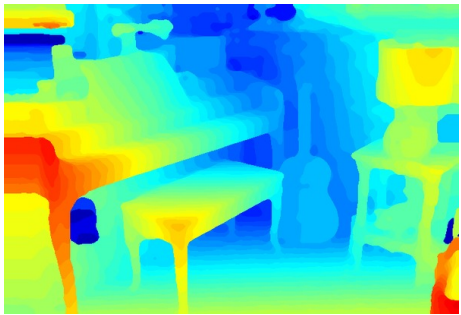
3D point cloud reconstruction

Motivation

Speed accuracy trade-off



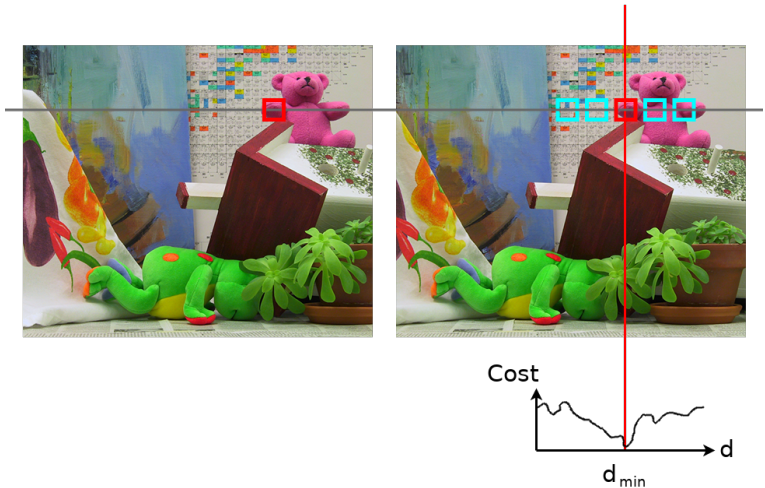
High quality disparity map (long computation)



Low quality disparity map (fast computation)

Motivation

Dense stereo matching

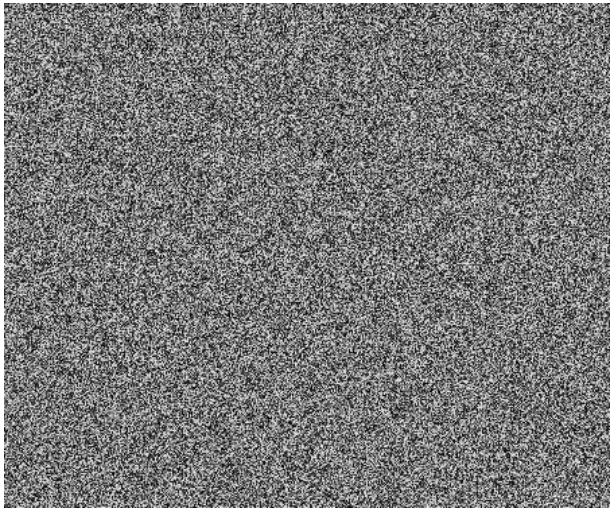


Approach

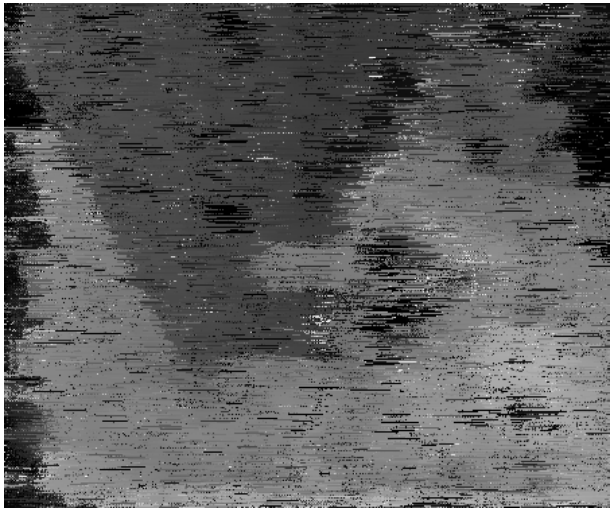
Implementation of 2 existing algorithms

- Patch Match
  Bleyer et al., *PatchMatch Stereo - Stereo Matching with Slanted Support Windows*
- AD-Census
  Mei et al., *On Building an Accurate Stereo Matching System on Graphics Hardware*

Patch Match Stereo - 0. Iteration

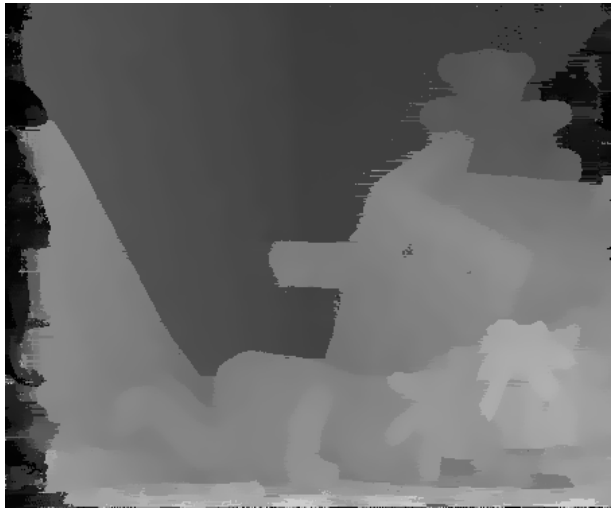


Patch Match Stereo - 1. Iteration

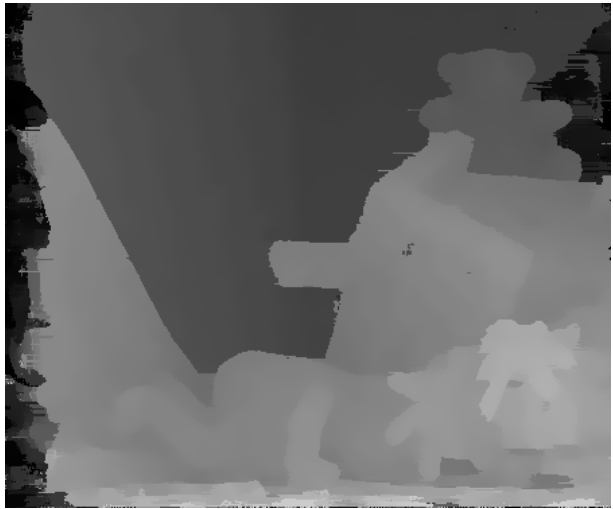


Patch Match Stereo - 2. Iteration

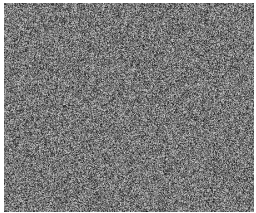
Patch Match Stereo - 3. Iteration



Patch Match Stereo - 4. Iteration

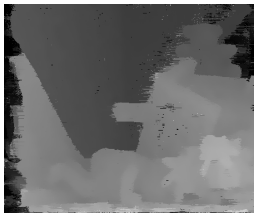


Patch Match Stereo



0. iteration

1. iteration



2. iteration

3. iteration

Patch Match Pipeline

- Random Initialization
- Iteration
  1. spatial propagation
  2. view propagation
  3. temporal propagation
  4. plane refinement
- Post Processing

Patch Match - Cost Aggregation

- Cost of a pixel:

$$m(p, f) = \sum_{q \in W_p} \omega(p, q) \cdot \rho(q, q - d))$$

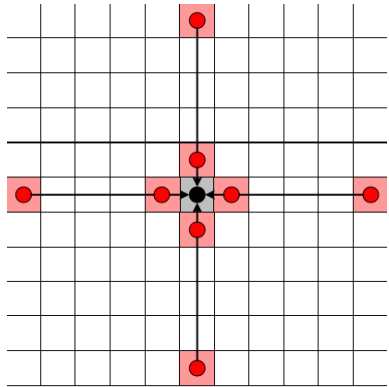
- Dissimilarity of a pixel q in W_p and its matching point:

$$\rho(q, q') = (1 - \alpha) \cdot min(||I_q - I_{q'}||, \tau_{color}) + \alpha \cdot min(||\Delta I_q - \Delta I_{q'}||, \tau_{gradient})$$

Patch Match - Iteration

SPATIAL PROPAGATION:

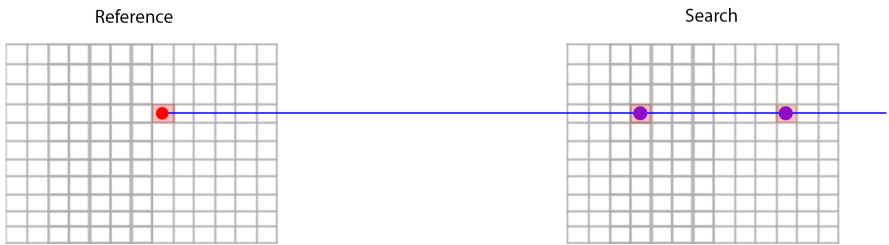
- neighbour disparities are likely to have similar disparities



source: *Multi-View Stereo with Asymmetric Checkerboard Propagation and Multi-Hypothesis Joint View Selection by Qingshan Xu, Wenbing Tao*

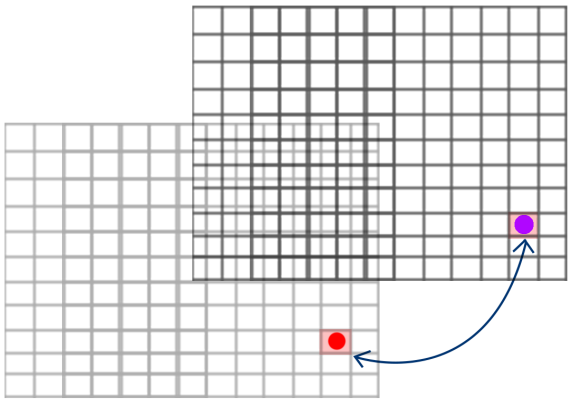
Patch Match - Iteration

VIEW PROPAGATION:



Patch Match - Iteration

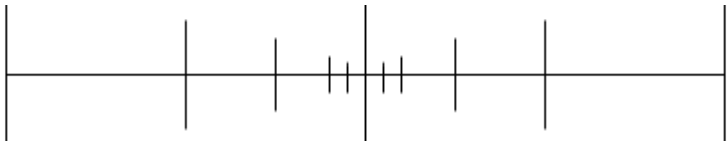
TEMPORAL PROPAGATION:



Patch Match - Iteration

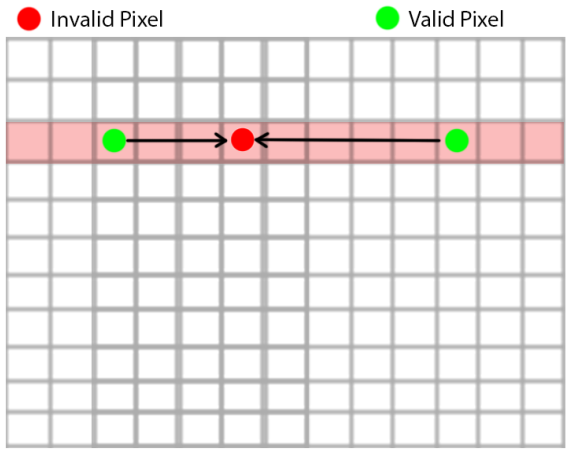
PLANE REFINEMENT:

- *Has the same pixel with other disparities a better cost?*



Patch Match - Post Processing

Fill invalid pixels:



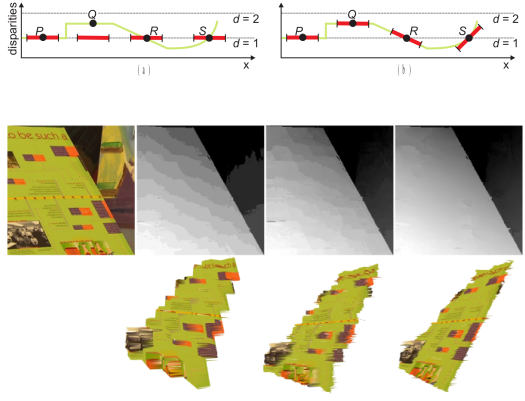
Patch Match - from a slanted plane to a disparity

- each pixel has its own local plane:
  - current disparity and normal vector

$$d = \frac{n_x x_0 + n_y y_0 + n_z d_0}{n_z} - \frac{n_x}{n_z} p_x - \frac{n_y}{n_z} p_y$$

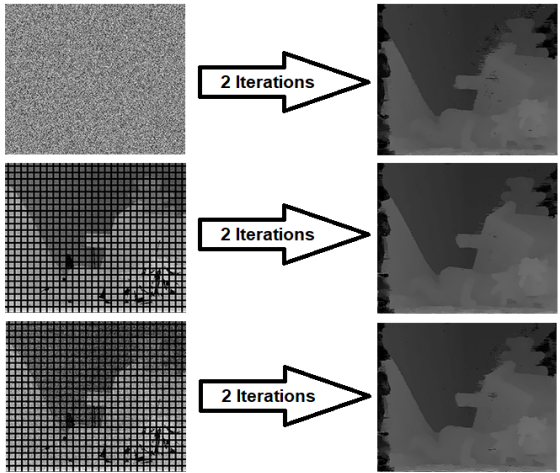
Patch Match with slanted support windows

Fronto-parallel planes vs. slanted planes with sub-pixel precision:



source: *PatchMatch Stereo - Stereo Matching with Slanted Support Windows*

Patch Match Extension



AD Census

Motivation

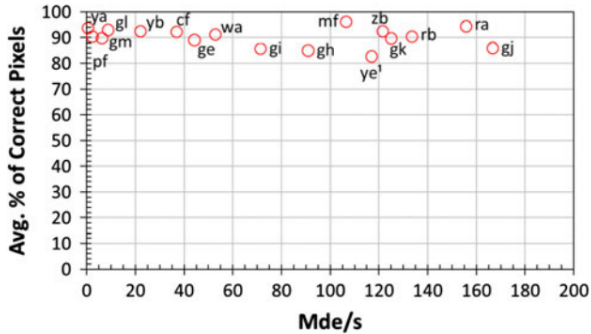


Figure: A comparison of runtimes and accuracy for algorithms implemented on a GPU with runtimes between 0 and 200 Mde/s (source: Tippets et al., Review of stereo vision algorithms and their suitability for resource-limited systems [5])

AD Census

Pipeline

- Cost Initialization
- Cost Aggregation
- Scanline Optimization
- Disparity Refinement

AD Census

Cost Initialization

- 2 cost measures combined
- Absolute Difference:

$$C_{AD}(p, d) = \frac{1}{3} \sum_{i=R,G,B} |I_i^{Left}(p) - I_i^{Right}(pd)|$$

AD Census

Cost Initialization

- Census transform records the relative intensity in a fixed path to neighbouring pixels
- Cost is calculated as Hamming distance between the 2 Bit strings

50	70	80
90	100	110
60	120	150

60	75	85
115	110	105
70	130	170

$CT_1 = 00011100$

$C_{CT} = 2$

$CT_2 = 00001101$

AD Census

Cost Initialization

- AD-census

$$C(p, d) = \rho(C_{census}(p, d), \lambda_{census}) + \rho(C_{AD}(p, d), \lambda_{AD})$$

- where $\rho(c, \lambda)$ is

$$\rho(c, \lambda) = 1 - exp(-\frac{c}{\lambda})$$

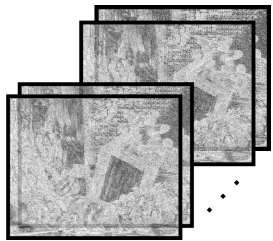
  - Maps to $[0, 1]$
  - Influence of outliers can be controlled with λ

AD Census

Cost initialization

Left image

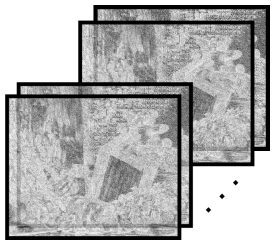
Right image



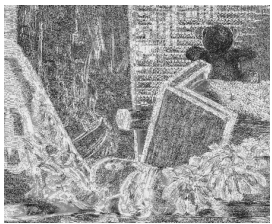
Cost volume

AD Census

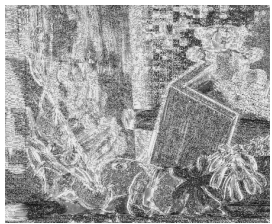
Cost Initialization



Cost volume



$d = 22$

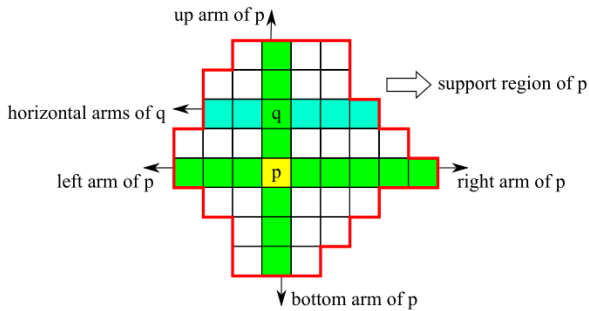


$d = 38$

AD Census

Cost Aggregation - Cross Construction

- In a pre-processing step we generate a support region for each pixel.



(a) Cross Construction

Cross Construction (source: Mei et al., On building an accurate stereo matching system on graphics hardware [4])

AD Census

Cost Aggregation - Cross Example

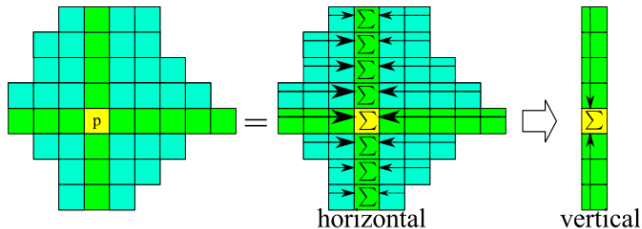
AD Census

Cost Aggregation - Cross Example

AD Census

Cost Aggregation

- For each pixel, we aggregate the costs per corresponding support region
- 4 iterations (horizontal, vertical, vertical, horizontal)
- Similar to an anisotropic diffusion process over the cost volume



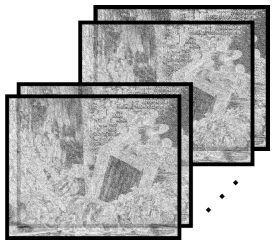
(b) Cost Aggregation

Cross Construction (source: Mei et al., On building an accurate stereo matching system on graphics hardware [4])

AD Census

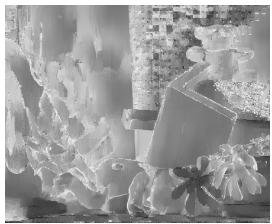
Cost Aggregation

After iteration 1



Cost volume

$d = 22$

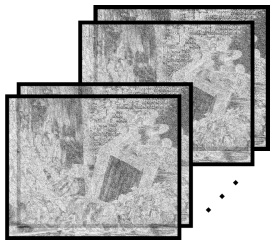


$d = 38$

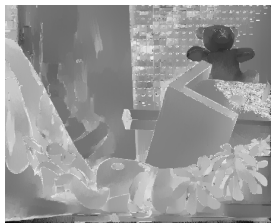
AD Census

Cost Aggregation

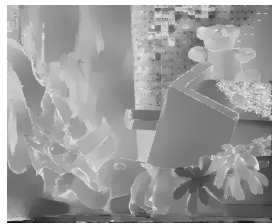
After iteration 4



Cost volume



$d = 22$

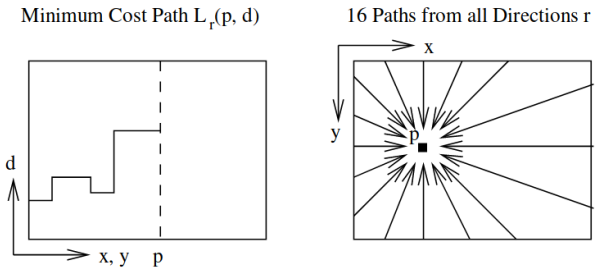


$d = 38$

AD Census

Scanline Optimization

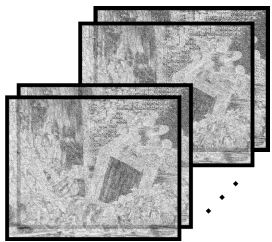
- Pixelwise matching, supported by a smoothness constrain
- Fast approximation by path-wise optimizations from several directions
- Usually performed from 8+ directions, but we omit oblique directions



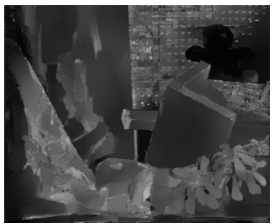
Aggregation of cost in disparity space (source: Hirschmüller, Stereo processing by semiglobal matching and mutual information [3])

AD Census

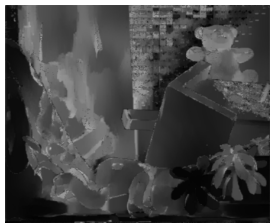
Scanline Optimization



Cost volume



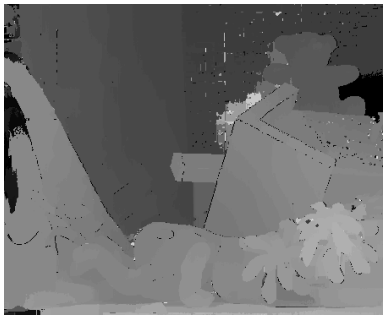
$d = 22$



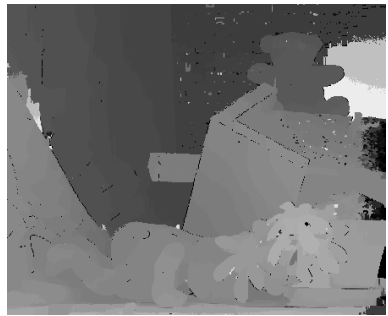
$d = 38$

AD Census

WTA (Winner Takes All)



Left disparity map after aggregation



Right disparity map after aggregation

AD Census

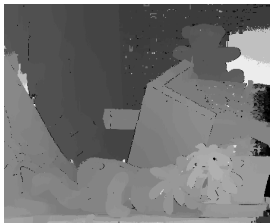
Disparity Refinement

- Perform left-right consistency check (generate outlier mask)
- Iterative Region Voting
  - For an outlier pixel build histogram of disparity values in cross region ⇒ pixel is filled with highest voted disparity
- Proper Interpolation
  - Interpolate disparity between neighbours of occlusion pixel
- Depth Discontinuity Adjustment
  - Detect edges in disparity map, replace edge pixels with smaller matching cost result
  - Reduces small errors around discontinuities
- Sub-Pixel Enhancement
  - Interpolate discrete disparity levels with quadratic polynomial

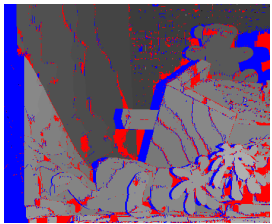
AD Census

Disparity Refinement - Left-Right-Consistency (Outlier Classification)

Left



Right



Combined disparity map for left view with outlier mask overlay

AD Census

Disparity Refinement - Region voting

Before RV

After 1st iteration RV

AD Census

Disparity Refinement - Region voting

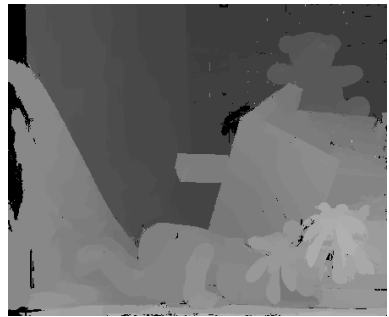
After 1st iteration RV

After 5th iteration RV

AD Census

Disparity Refinement - Proper Interpolation

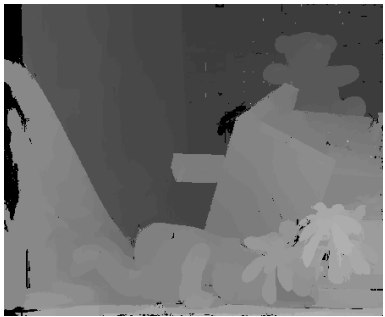
After 5th iteration RV



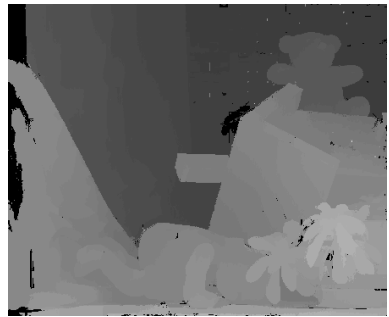
After proper interpolation

AD Census

Disparity Refinement - Discontinuity Adjustment



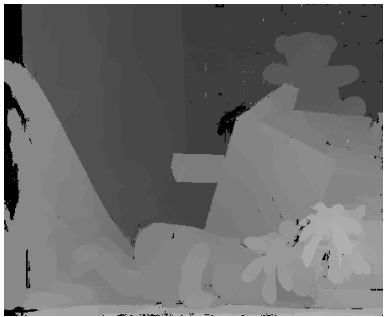
After proper interpolation



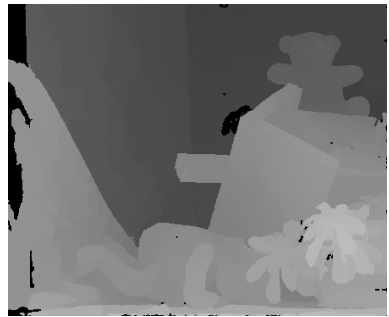
After discontinuity adjustment

AD Census

Disparity Refinement - Subpixel Enhancement



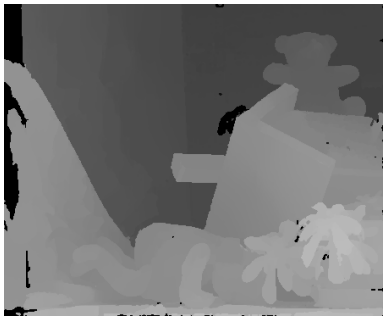
After discontinuity adjustment



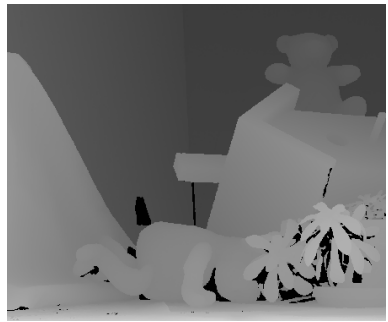
After subpixel enhancement and median filter

AD Census

Result



Result



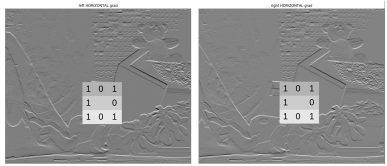
Ground truth

AdCensus Extensions

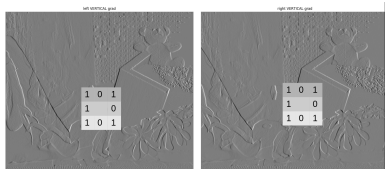
- Gradient-based cost
- Occupancy volume.
- Reduced band cost volume.

Gradient-based cost

- Horizontal gradient-based census

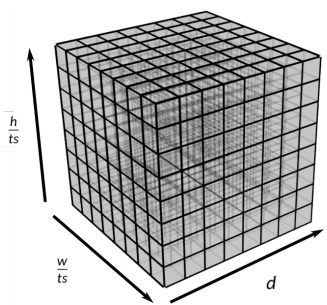


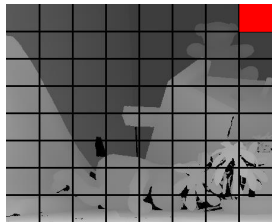
- Vertical gradient-based census

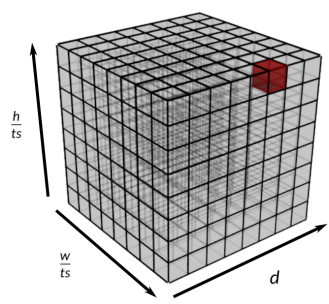


AD Census - Extension

Occupancy Volume

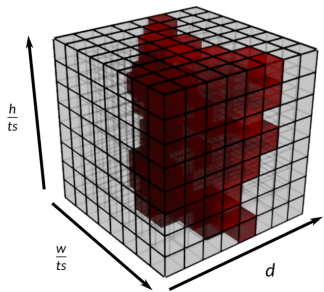


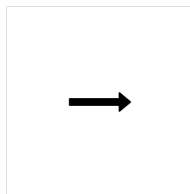


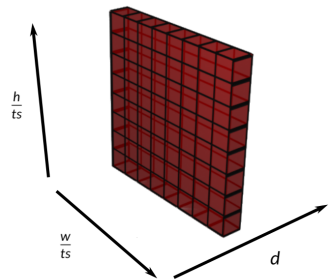


AD Census - Extension

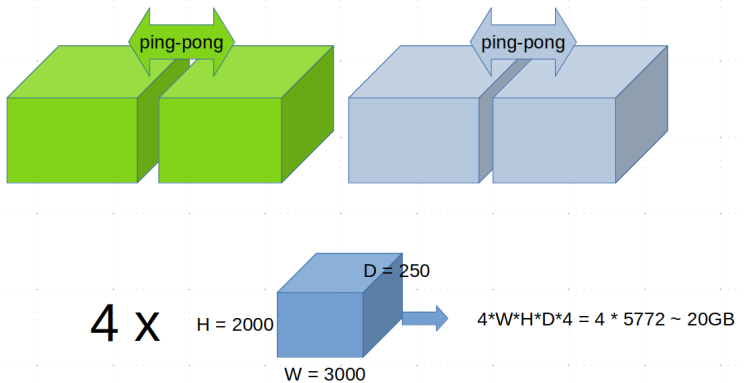
Occupancy Volume





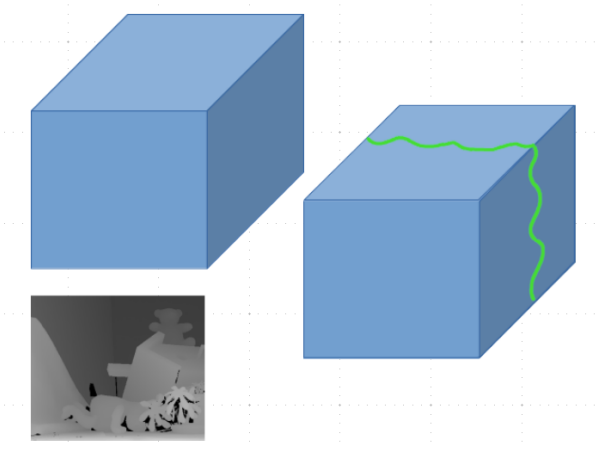


Band Cost Volume - Motivation



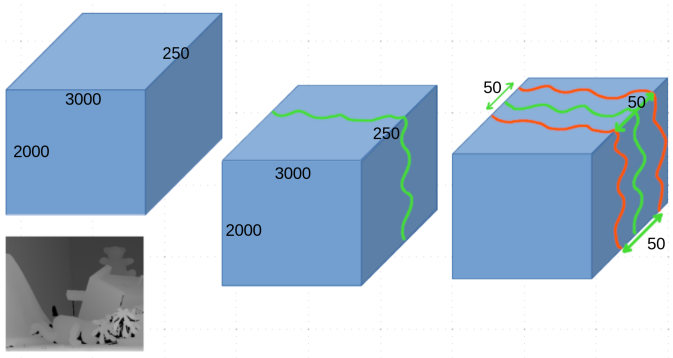
Band Cost Volume - How to reduce memory footprint?

- Disparity guess as a way to reduce search space



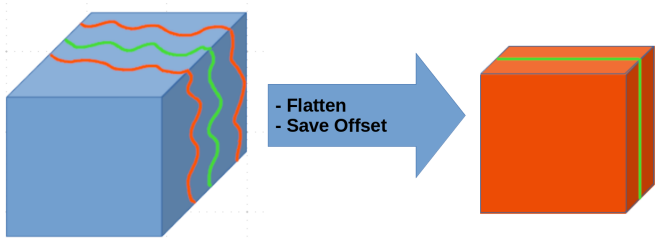
Band Cost Volume - How to reduce memory footprint?

- Disparity guess as a way to reduce search space



Band Cost Volume - How to reduce memory footprint?

- Disparity guess as a way to reduce search space



Band Cost Volume - How to get the disparity guess?

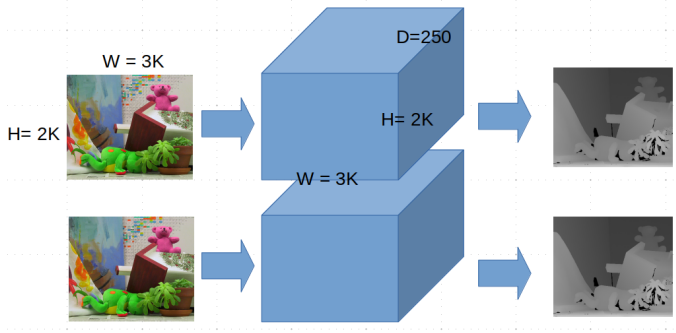
- Take disparity guess from another camera: Real-Sense.
- Calculate disparity from the down-sampled input images.
  - We focus on this approach for our comparison experiment.

Band Cost Volume - Comparison Experiment

- Run both normal cost volume vs band cost volume
  - Compare mean absolute error
  - Compare running time

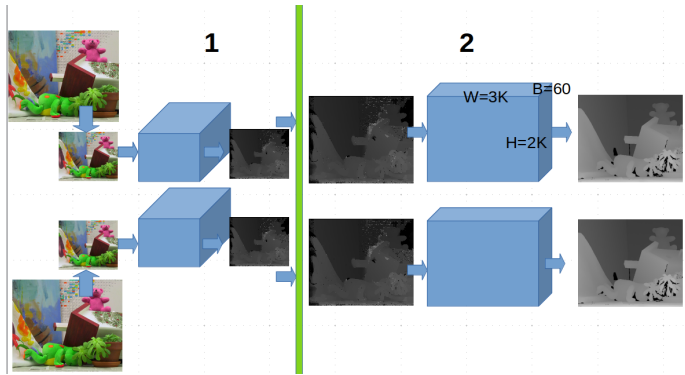
Band Cost Volume - Comparison Experiment

- Normal Cost Volume execution



Band Cost Volume - Comparison Experiment

- Band Cost Volume Execution



Evaluation - Summary

Evaluation - Summary

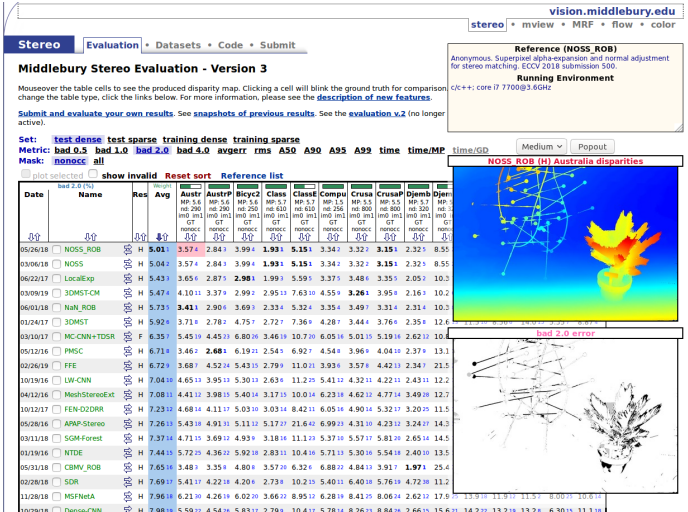
Evaluation - Summary

Evaluation - Summary

Evaluation - Summary

Evaluation

- Middlebury → http://vision.middlebury.edu



Evaluation

Key metrics:

- **bad2.0**: percentage of pixels deviating more than 2 disparity levels to ground truth
- **avg error**: average error per pixel
- **time**

Tests have been performed on the computer in the CV lab

- Nvidia Quadro P6000: 24GB RAM, 3840 CUDA cores

Evaluation

Teddy

Evaluation

Motorcycle

Evaluation

Adirondack

Evaluation

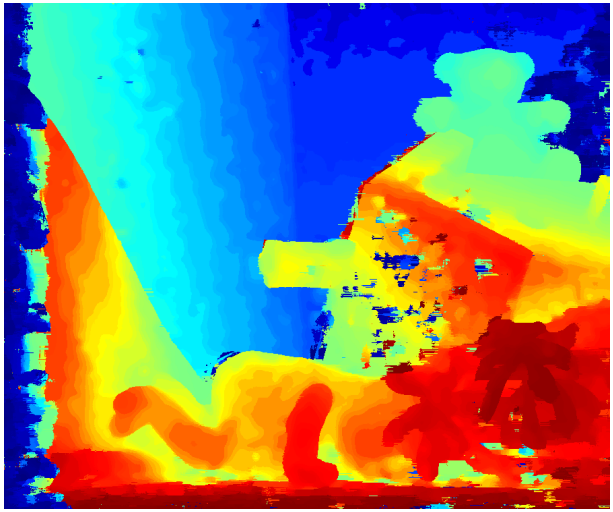
Playroom

Evaluation - Patch Match - Baseline

Baseline performance on the four test images of half resolution with optimal parameters:

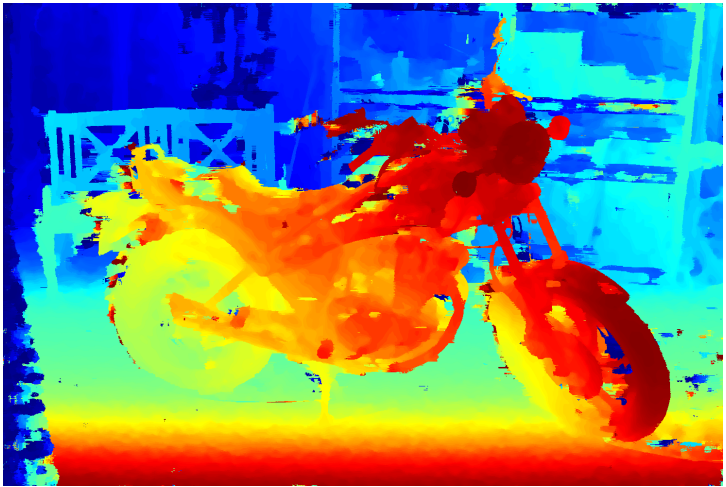
Image	Resolution	ndisp	bad2.0	avg. error	time [ms]
Teddy	900x750	128	9.78	1.74	594.929
Motorcycle	1482x994	140	11.79	2.09	1428.97
Adirondack	1436x992	145	40.85	13.57	1162.78
Playroom	1398x952	165	40.00	8.93	1274.71

Patch Match - Evaluation



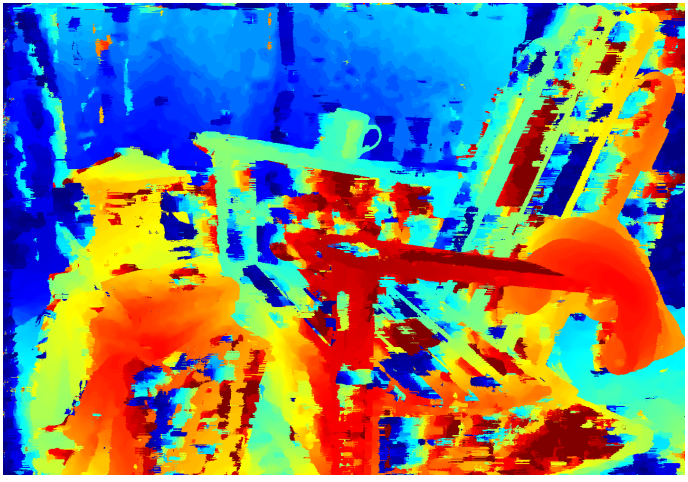
Teddy - 900x750

Patch Match - Evaluation



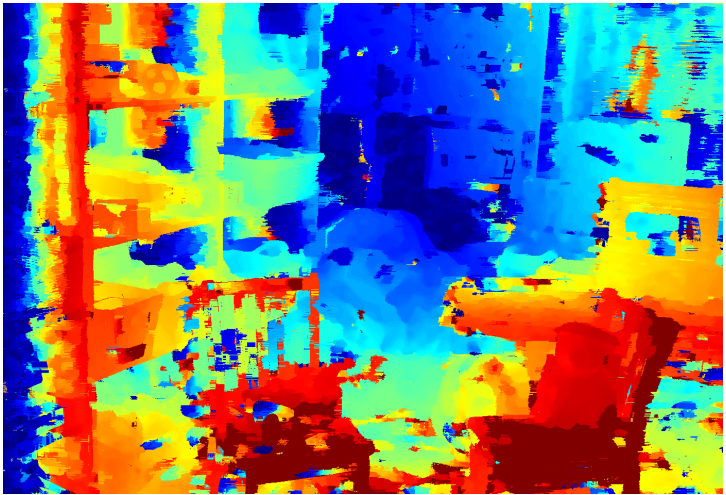
Motorcycle - 1482x994

Patch Match - Evaluation



Adirondack - 1436x992

Patch Match - Evaluation

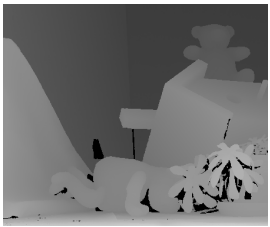


Playroom - 1398x952

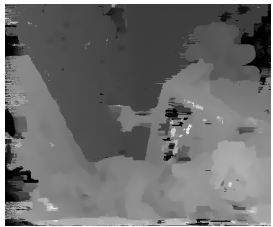
Patch Match - Evaluation

Aggregated Cost

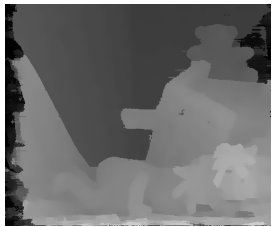
- Sum of Absolute Differences vs Adaptive Support Weights



Ground Truth



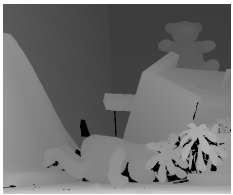
SAD



ASW

Patch Match - Evaluation

Iteration



Ground truth

2. Iteration

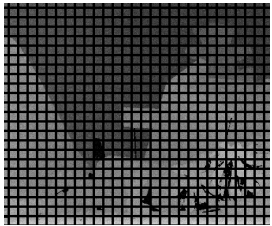
3. Iteration

4. Iteration

Iter	WinSize	bad 2.0	avg. error	time [ms]
2	9	8.91	0.98	71.4425
3	9	7.41	0.8	102.688
4	9	7.24	0.79	129.232

Patch Match - Evaluation

Initial Guess



LessNoiseWithGrid

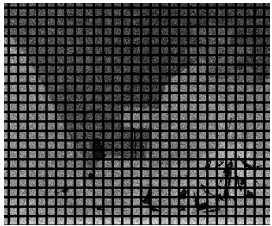
2 Iterations

3 Iterations

Iter	WinSize	Without IG			LessNoiseWithGrid			MoreNoiseWithGrid			Noisy		
		bad 2.0	avgErr	time [ms]	bad 2.0	avgErr	time [ms]	bad 2.0	avgErr	time [ms]	bad 2.0	avgErr	time [ms]
2	9	8.91	0.98	71.4425	7.37	0.79	64.215	8.22	0.89	67.617	8.12	0.82	67.2326
3	9	7.41	0.8	102.688	7.22	0.78	93.803	7.36	0.80	94.7496	7.26	0.72	94.4366
4	9	7.24	0.79	129.232	7.20	0.79	122.125	7.31	0.81	126.076	7.22	0.73	129.057

Patch Match - Evaluation

Initial Guess



MoreNoiseWithGrid

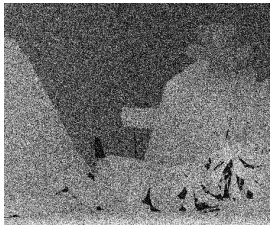
2 Iterations

3 Iterations

Iter	WinSize	Without IG			LessNoiseWithGrid			MoreNoiseWithGrid			Noisy		
		bad 2.0	avgErr	time [ms]	bad 2.0	avgErr	time [ms]	bad 2.0	avgErr	time [ms]	bad 2.0	avgErr	time [ms]
2	9	8.91	0.98	71.4425	7.37	0.79	64.215	8.22	0.89	67.617	8.12	0.82	67.2326
3	9	7.41	0.8	102.688	7.22	0.78	93.803	7.36	0.80	94.7496	7.26	0.72	94.4366
4	9	7.24	0.79	129.232	7.20	0.79	122.125	7.31	0.81	126.076	7.22	0.73	129.057

Patch Match - Evaluation

Initial Guess



Noisy

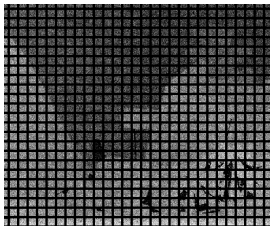
2 Iterations

3 Iterations

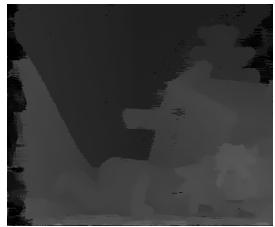
Iter	WinSize	Without IG			LessNoiseWithGrid			MoreNoiseWithGrid			Noisy		
		bad 2.0	avgErr	time [ms]	bad 2.0	avgErr	time [ms]	bad 2.0	avgErr	time [ms]	bad 2.0	avgErr	time [ms]
2	9	8.91	0.98	71.4425	7.37	0.79	64.215	8.22	0.89	67.617	8.12	0.82	67.2326
3	9	7.41	0.8	102.688	7.22	0.78	93.803	7.36	0.80	94.7496	7.26	0.72	94.4366
4	9	7.24	0.79	129.232	7.20	0.79	122.125	7.31	0.81	126.076	7.22	0.73	129.057

Patch Match - Evaluation

Initial Guess



MoreNoiseWithGrid



PlaneRef: 2

PlaneRef: 10

Iter	PlaneRefSteps	LessNoiseWithGrid			MoreNoiseWithGrid			Noisy		
		bad 2.0	avgErr	time [ms]	bad 2.0	avgErr	time [ms]	bad 2.0	avgErr	time [ms]
2	2	7.37	0.79	64.215	8.22	0.89	67.617	8.12	0.82	67.2326
2	10	7.26	0.80	44.8574	8.39	0.93	46.4501	8.62	0.90	46.7122

Patch Match - Evaluation

Temporal Propagation and Slanted Support Window

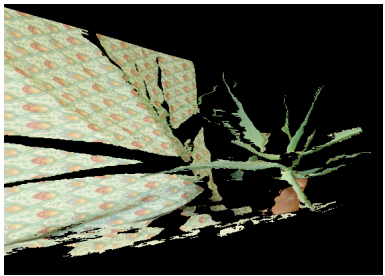
Left image

Right image

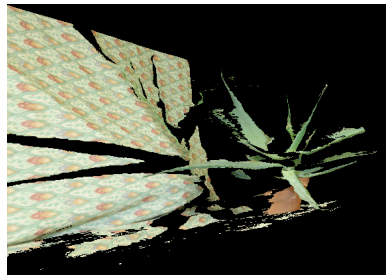
Iter	WinSize	**Without TP and SSW** time [ms per frame]	**With TP and SSW** time [ms per frame]
4	9	124.155	131

Patch Match - Evaluation

Temporal Propagation and Slanted Support Window



3D Aloe without Slanted

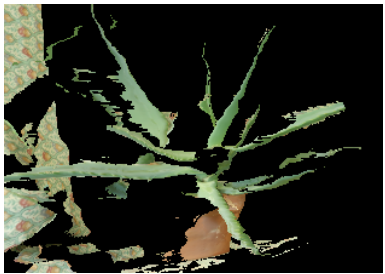


3D Aloe with Slanted

Iter	**WinSize**	**Without TP and SSW** time [ms per frame]	**With TP and SSW** time [ms per frame]
4	9	124.155	131

Patch Match - Evaluation

Temporal Propagation and Slanted Support Window



3D Aloe without Slanted

3D Aloe with Slanted

Iter	**WinSize**	**Without TP and SSW** time [ms per frame]	**With TP and SSW** time [ms per frame]
4	9	124.155	131

Evaluation - AD Census

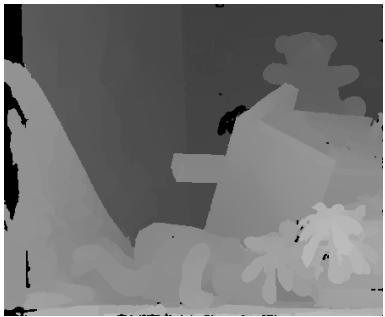
Correctness of Implementation

	Tsukuba		**Venus**		**Teddy**		**Cones**	
Implementation	**nonocc**	all	**nonocc**	all	**nonocc**	all	**nonocc**	all
Bleyer et al.	1.07	1.48	0.09	0.25	4.10	6.22	2.42	7.25
Our group	1.15	1.84	0.67	1.59	4.71	8.78	2.68	10.31

Table: Percent of pixel mismatches >1.0

Evaluation - AD Census

Correctness of Implementation



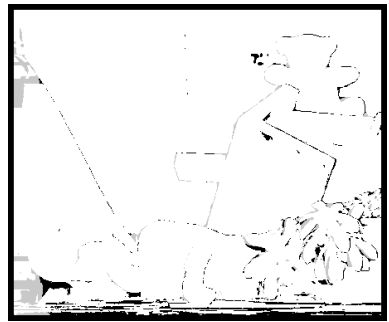
Our result

Mei et al. result

Evaluation - AD Census

Correctness of Implementation

Differences to ground truth (our result)

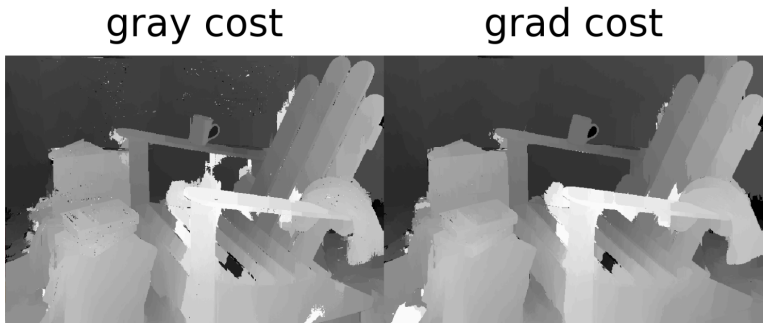


Differences to ground truth (Mei et al.)

Evaluation - AD Census

Gradient-based cost

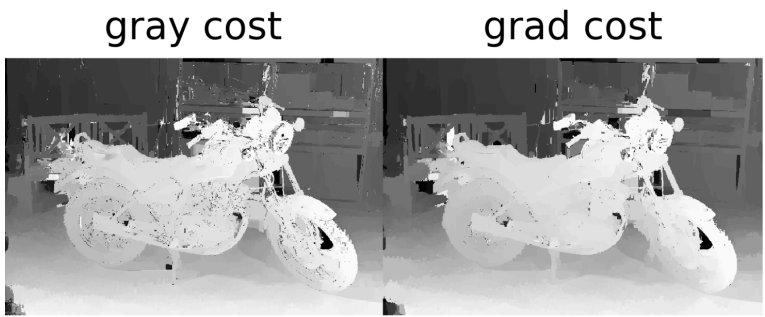
- Normal Cost vs Gradient-based Cost



Evaluation - AD Census

Gradient-based cost

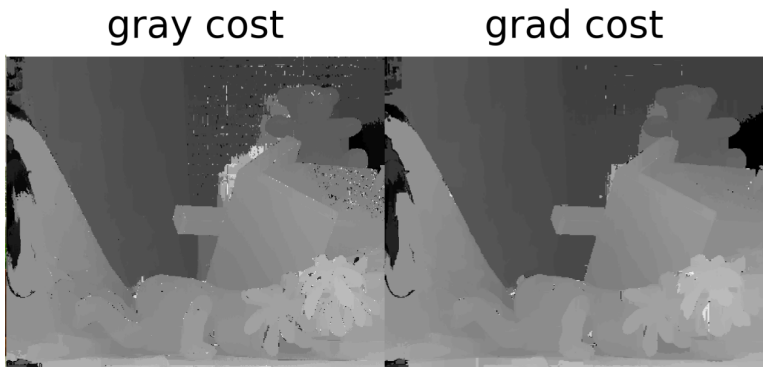
- Normal Cost vs Gradient-based Cost



Evaluation - AD Census

Gradient-based cost

- Normal Cost vs Gradient-based Cost

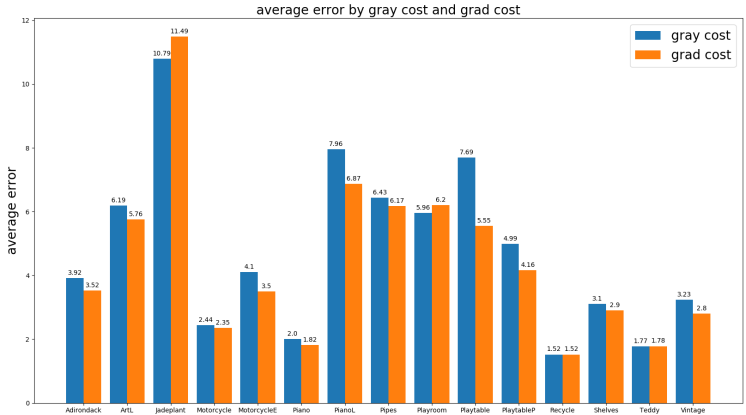


Evaluation - AD Census

Gradient-based cost

- Normal Cost vs Gradient-based Cost

Gradient-based cost



Mean Average Error Comparison

Evaluation - AD Census

The implementation of our reduced cost volume extension is incomplete.
At the moment the following stages are missing:

- Scanline Optimization
- Discontinuity Adjustment
- Subpixel Enhancement

Therefore this implementation produces integer disparity maps.

Evaluation - AD Census - Full cost volume

Performance on the four test images from trainingH:
(optimal parameters found by grid search on 13 trainingH images)

Image	Resolution	ndisp	bad2.0	avg. error	time [ms]
Teddy	900x750	128	4.76	1.54	223
Motorcycle	1482x994	140	10.28	1.85	554
Adirondack	1436x992	145	17.12	1.98	591
Playroom	1398x952	165	28.79	6.01	737

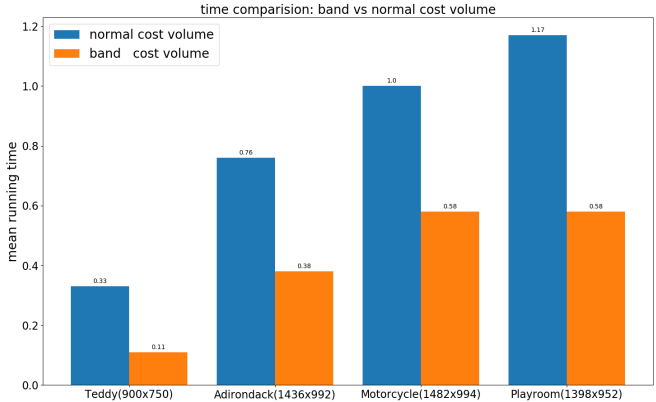
Evaluation - AD Census

Performance of reduced cost volume on selected images from trainingH with "perfect" initial guess and band size of 30:

Image	Resolution	ndisp	bad2.0	avg. error	time [ms]
Teddy	900x750	128	8.85	2.33	94
Motorcycle	1482x994	140	19.89	4.10	313
Adirondack	1436x992	145	34.62	12.83	361
Playroom	1398x952	165	41.03	8.76	417

Band Cost Volume - Comparison experiment

- Time Comparison Chart



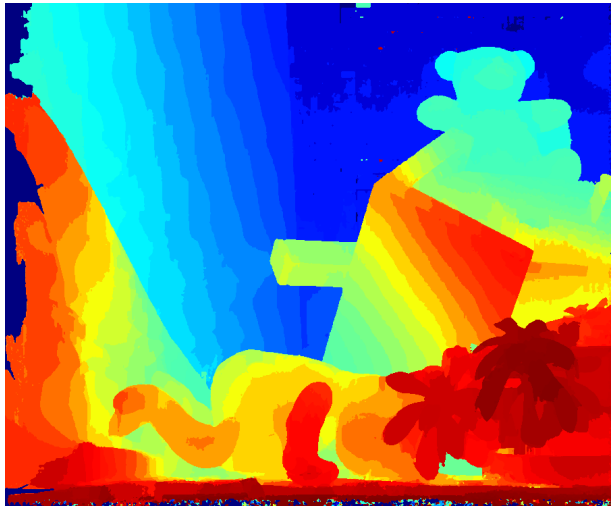
Evaluation - AD Census

Full resolution

Performance of band reduced cost volume on selected images from trainingF. Initial guess obtained by first matching on input images downscaled to 25% and using resulting left/right disparity maps.

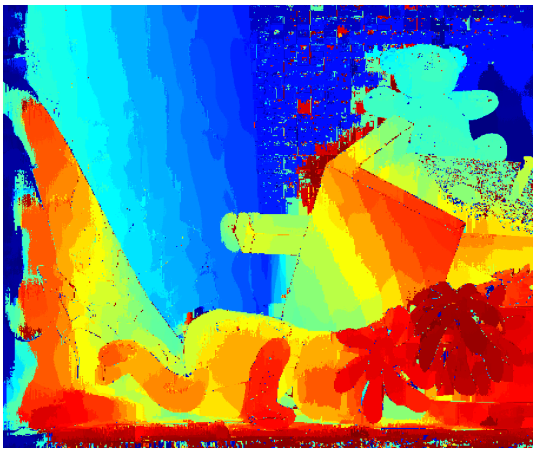
Image	Resolution	ndisp	bad2.0	avg. error
Teddy	1800x1500	256	12.83	4.81
Motorcycle	2964x2000	280	24.98	5.20
Adirondack	2880x1988	290	40.11	17.30
Playroom	2800x1908	330	48.28	16.88

Evaluation - AD Census



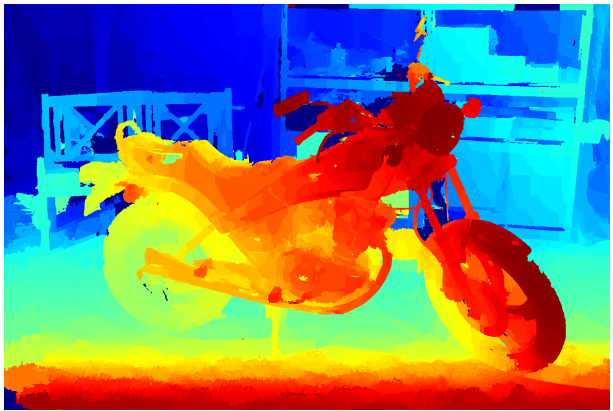
Teddy (trainingH) matched with full cost volume

Evaluation - AD Census



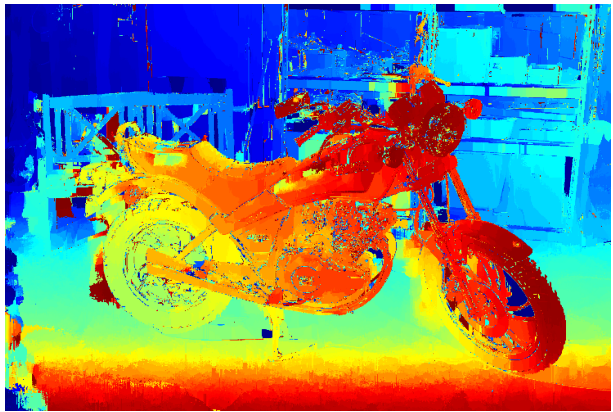
Teddy (trainingH) matched with reduced cost volume (no SO, no post processing)

Evaluation - AD Census



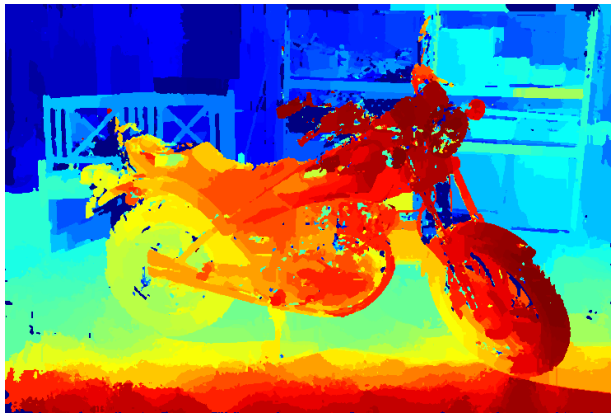
Motorcycle (trainingH) matched with full cost volume

Evaluation - AD Census



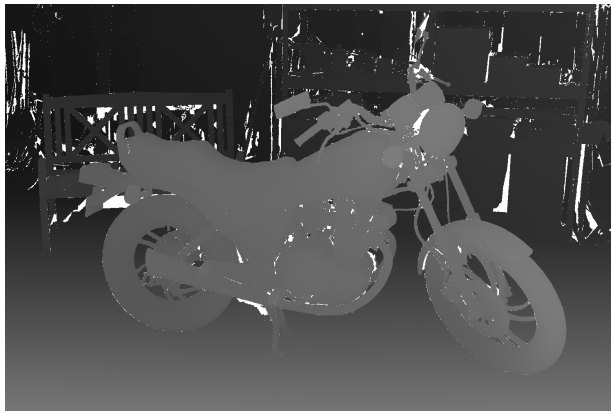
Motorcycle (trainingH) matched with reduced cost volume (no SO, no post processing)

Evaluation - AD Census



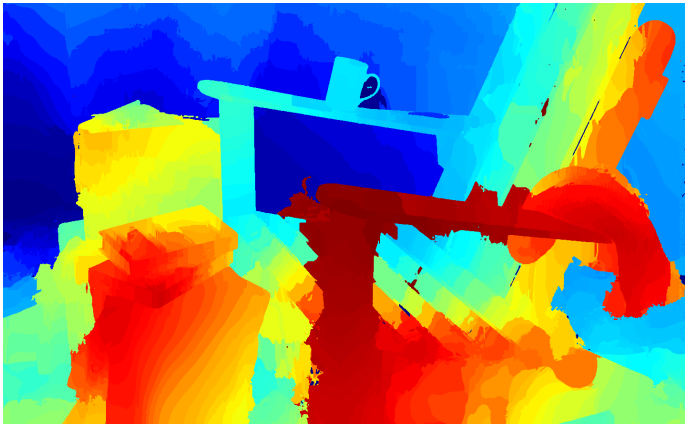
Motorcycle (trainingH) matched with reduced cost volume (no SO, with post processing)

Evaluation - AD Census



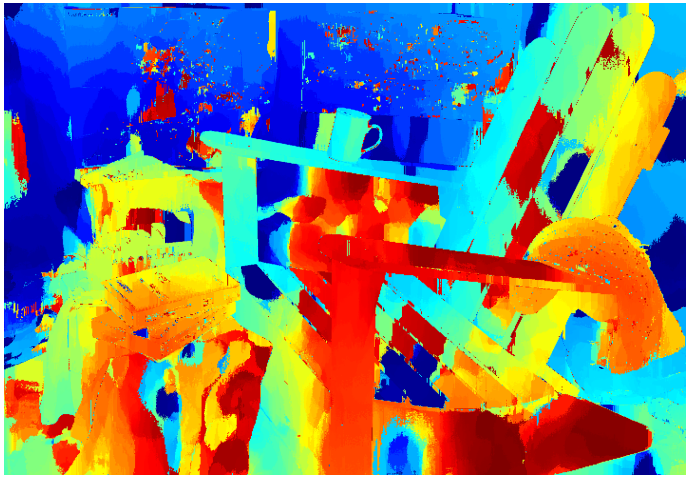
Motorcycle (trainingH) ground truth

Evaluation - AD Census



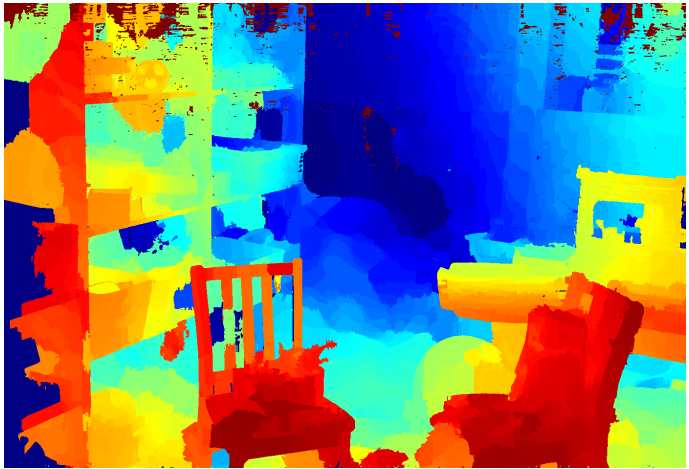
Adirondack (trainingH) matched with full cost volume

Evaluation - AD Census



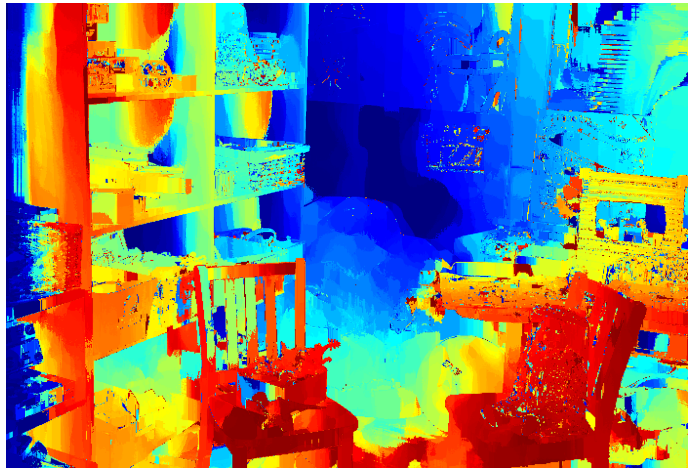
Adirondack (trainingH) matched with reduced cost volume (no SO, no post processing)

Evaluation - AD Census



Playroom (trainingH) matched with full cost volume

Evaluation - AD Census



Playroom (trainingH) matched with reduced cost volume (no SO, no post processing)

Patch Match - Conclusion

- less memory space but it depends on different resolution
- uncomplicated to work with different resolutions
- not fast because local memory is not yet implemented

Patch Match - Future Work

- local memory !
- try different (and complex) propagation pattern
- another cost aggregation could bring more precise results
- Are there other ways to integrate a better initial guess to get more performance?

AD Census - Conclusion and future work

- Complete implementation of AD Census [4] in C++/OpenCL and Python
- Several stages can be still optimized (Scanline Optimization, Region Voting)
- The proposed reduced cost volume extension shows a significant speedup and drastically reduces the memory footprint
- The extended version is very sensitive to the quality of the initial guess
- Unfortunately we could not quantify this sensitivity
- For a robust implementation a heuristic to quantify the quality of an initial guess without a priori knowledge is needed
- The cost volume could be analysed for pixels with "bad costs". But, the cost volume is noisy after cost initialization. After aggregation thresholding for "bad costs" is no longer possible, because aggregated costs are not normalized per pixel.

References I

[1] Michael Bleyer, Christoph Rhemann, and Carsten Rother.
Patchmatch stereo-stereo matching with slanted support windows.
In *Bmvc*, volume 11, pages 1–11, 2011.

[2] Jianbin Fang, Ana Lucia Varbanescu, Jie Shen, Henk Sips, Gorkem Saygili, and Laurens Van Der Maaten.
Accelerating cost aggregation for real-time stereo matching.
In *2012 IEEE 18th International Conference on Parallel and Distributed Systems*, pages 472–481. IEEE, 2012.

[3] Heiko Hirschmuller.
Stereo processing by semiglobal matching and mutual information.
IEEE Transactions on pattern analysis and machine intelligence, 30(2):328–341, 2007.

References II

[4] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang.
On building an accurate stereo matching system on graphics hardware.
In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 467–474. IEEE, 2011.

[5] Beau Tippetts, Dah Jye Lee, Kirt Lillywhite, and James Archibald.
Review of stereo vision algorithms and their suitability for resource-limited systems.
Journal of Real-Time Image Processing, 11(1):5–25, 2016.