SAMSUNG ADVANCED INSTITUTE OF TECHNOLOGY

Stereo Matching with Reliable Disparity Propagation

Xun Sun, Xing Mei, Shaohui Jiao, Mingcai Zhou, Haitao Wang

Samsung Advanced Institute of Technology, China Lab (Beijing)

Hangzhou
May, 2011

creation+

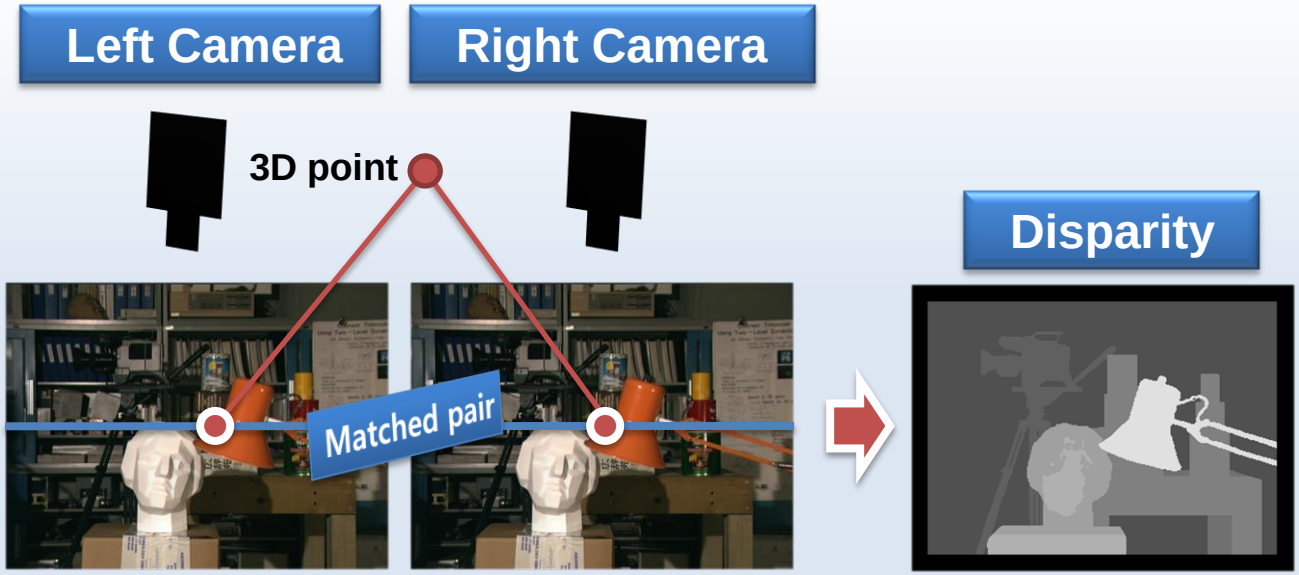
Outline

- Background && Motivation
- Algorithm
  - ✓ Framework
  - ✓ Pixelwise line segments
  - ✓ Seed detection
  - ✓ Reliable disparity propagation
  - ✓ Two-step disparity refinement
- Results
- Conclusion

- **Background && Motivation**
- Algorithm
  - ✓ Framework
  - ✓ Pixelwise line segments
  - ✓ Seed detection
  - ✓ Reliable disparity propagation
  - ✓ Two-step disparity refinement
- Results
- Conclusion

Background && Motivation

Dense two-frame stereo matching



- Compute a dense disparity map from a rectified image pair
- Broad applications: 3D reconstruction, view interpolation

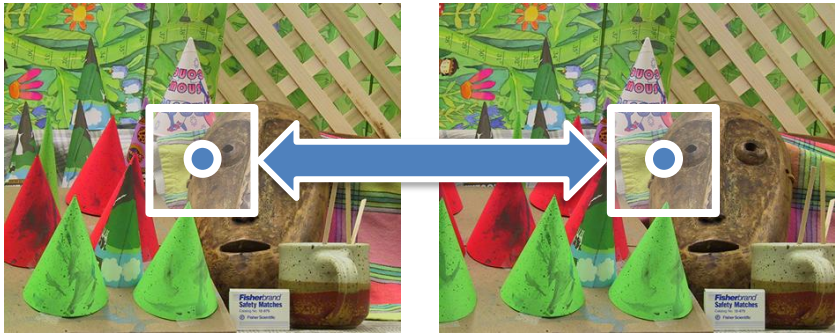
Background && Motivation

Existing methods

☐ Local methods

Compute each pixel's disparity independently over a local support region.

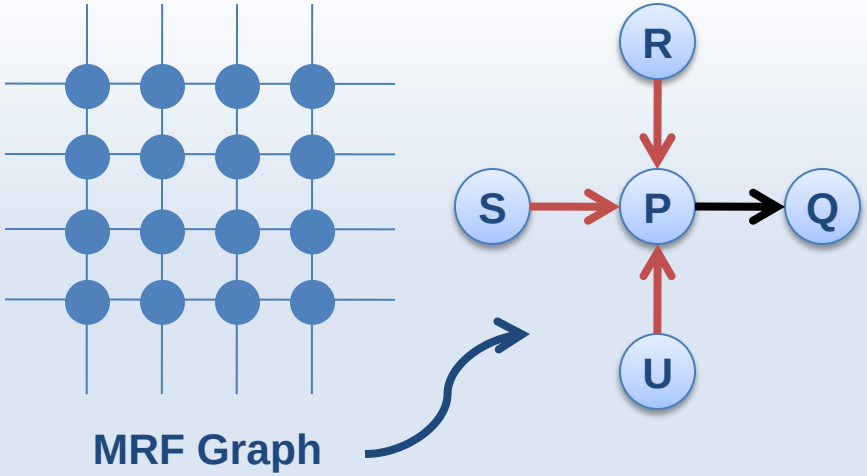
Fast but inaccurate.



☐ Global methods

Solve the stereo problem in an energy minimization process.

Accurate but slow due to time-comsuming global optimizer (GC, BP).



Background && Motivation

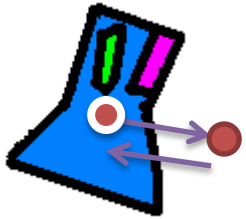
Existing methods

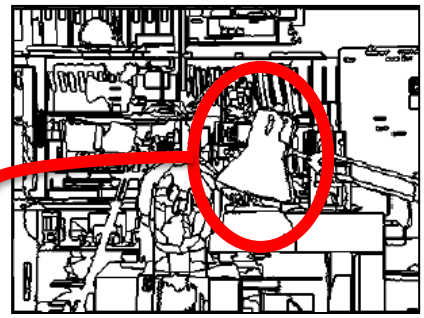
☐ Propagation-based methods?

Produce quasi-dense or dense disparity results from a set of seed pixels.
Relatively fast but sensitive to early wrong matches.

Addressing this inherent problem in *propagation-based* method, Wei and Quan[CVPR'2004] proposed to use segmented regions as guided propagation unit:

Progressively propagating between neighboring regions





Effectively alleviate the sensitivity to wrong seeds with the region-based representation at the cost of expensive image segmentation

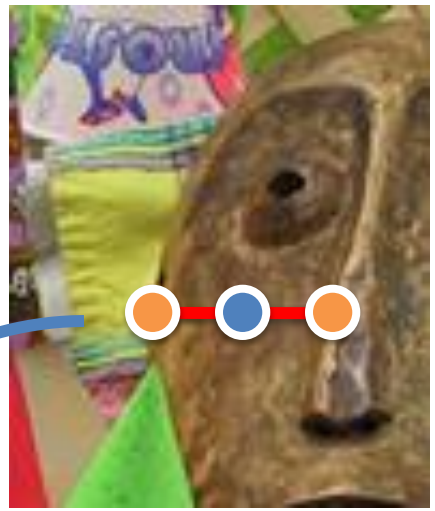
Background && Motivation

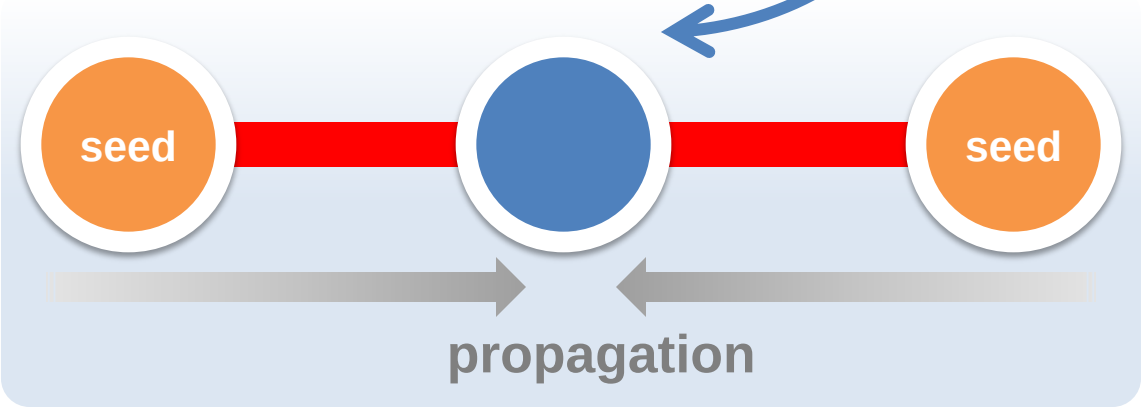
Our motivation

☐ Introduce a simple guided unit for propagation: pixelwise 1D line segments. *No image segmentation required here.*

☐ Compute initial seed points and propagate them within line segments

Simple, fast and accurate

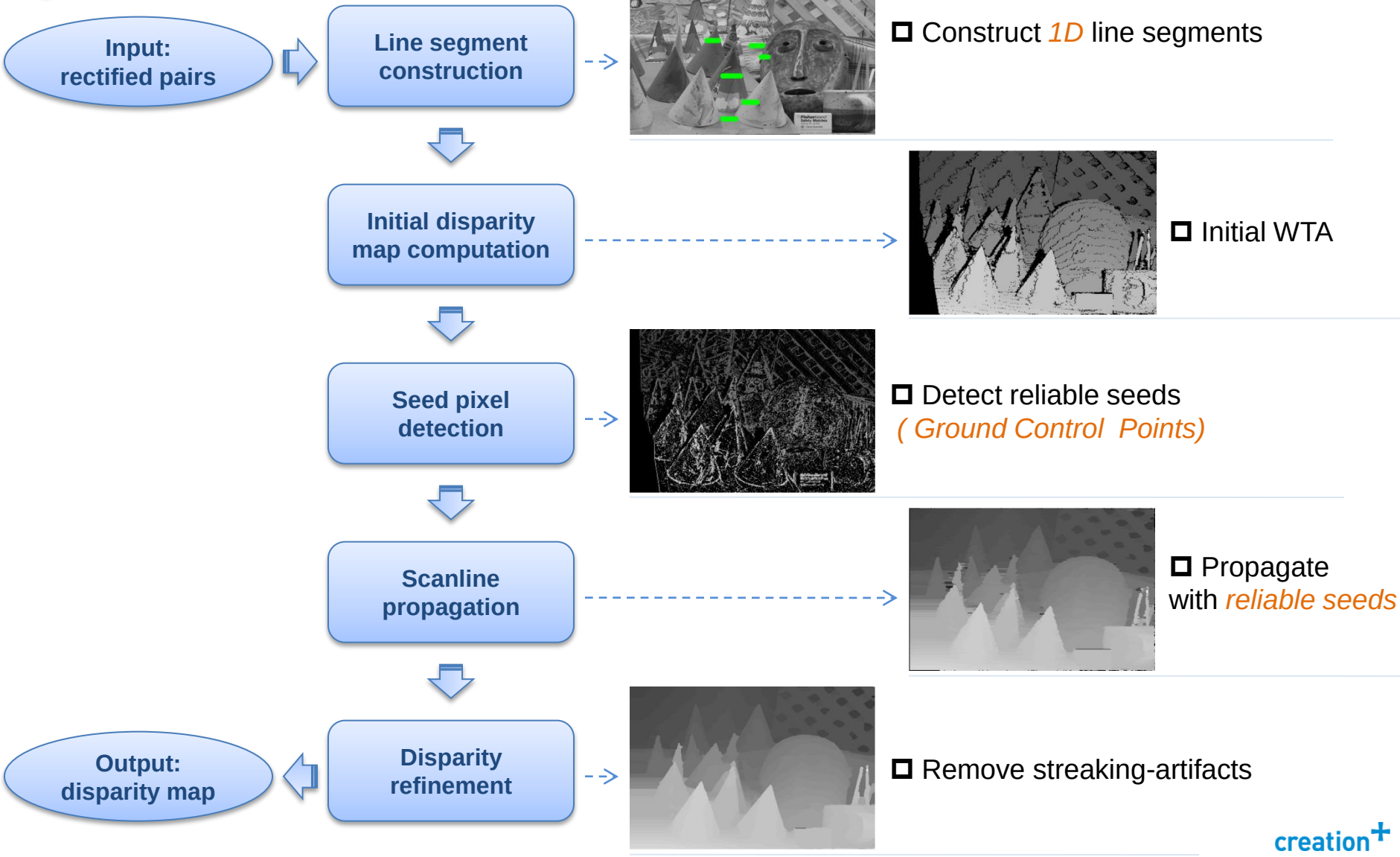




- Background && Motivation
- **Algorithm**
  - ✓ **Framework**
  - ✓ Pixelwise line segments
  - ✓ Seed detection
  - ✓ Reliable disparity propagation
  - ✓ Two-step disparity refinement
- Results
- Conclusion

Algorithm

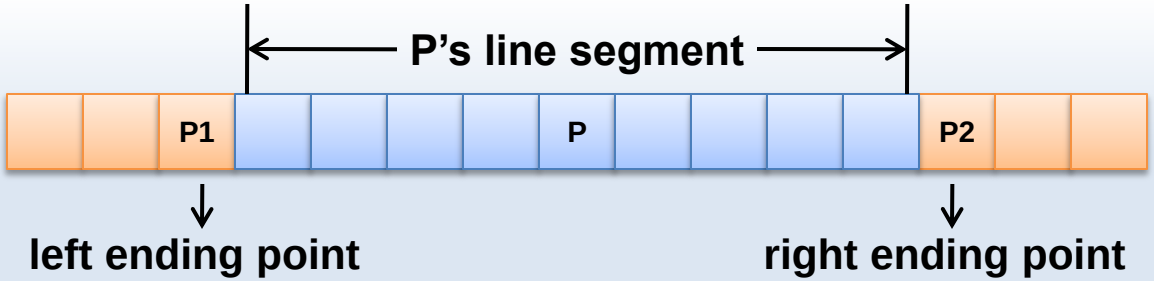
Framework



- Background && Motivation
- **Algorithm**
  - ✓ Framework
  - ✓ **Pixelwise line segments**
  - ✓ Seed detection
  - ✓ Reliable disparity propagation
  - ✓ Two-step disparity refinement
- Results
- Conclusion

Algorithm

Pixelwise line segments



Basic regions for cost initialization && disparity propagation

Line ending points P1, P2 for P are located when rule 1 or 2 are violated:

✓**Rule 1** **Color self-similarity in the line region:** ***smooth depth assumption***

$$D_c(p_1,p) = \max_{i=R,G,B} \left| I_i(p_1) - I_i(p) \right| \longrightarrow D_c(p_1,p) < \tau$$

color difference — threshold

✓**Rule 2** **Arm length limitation:** ***avoid over-smoothness***

$$D_s(p_1,p) = \left| p_1 - p \right| \longrightarrow D_s(p_1,p) < L$$

spatial distance — maximum length

Algorithm

Initial disparity map computation
-Cost aggregation with line segments

☐ We use AD-Census matching cost:

$$C_1(p,d) = \min(C_{AD}(p,d), \lambda_{AD}) + \min(C_{Census}(p,d), \lambda_{Census})$$

AD: constant color assumption **Census:** encode local image structure

Reasons:

(1) It shows improved performance by combing two kinds of constraints

(2) ***Census*** naturally encode ***inter-scanline information***

☐ The initial cost is aggregated within pixelwise line segment

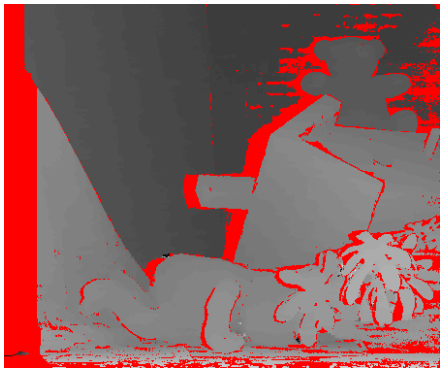
Implementation:

(1) Use integral image for acceleration

(2) To reduce noise, iterate for 2 times

$$C_2(p,d) = \frac{\sum_{q \in Line(p)} C_1(q,d)}{|Line(p)|}$$

Number of pixels



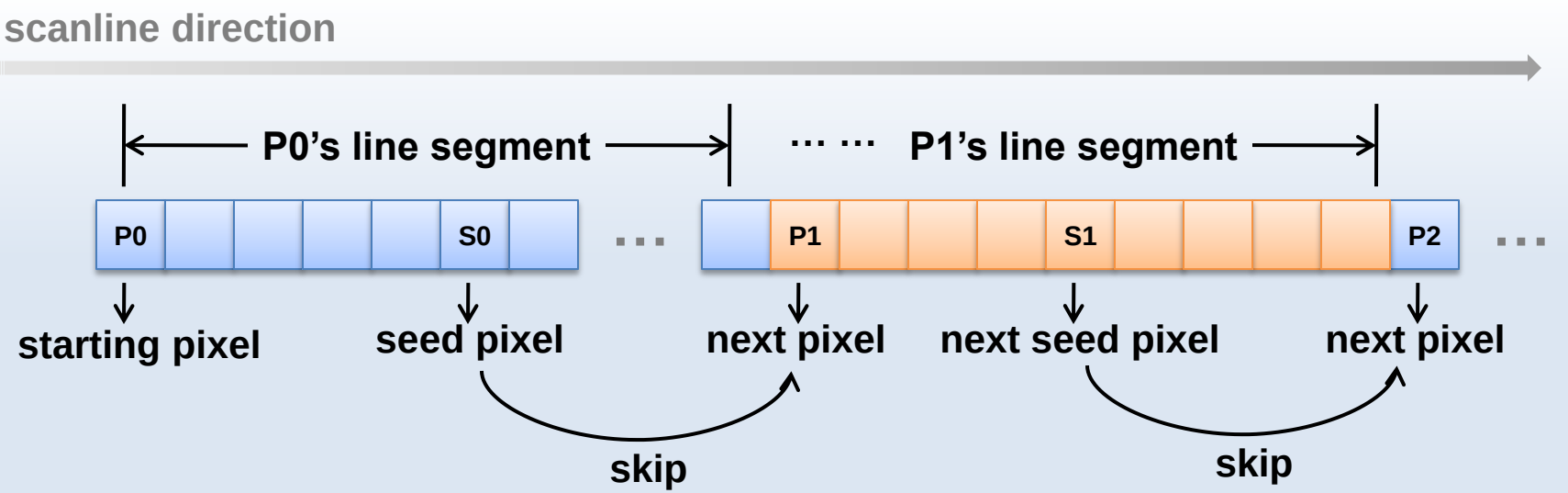
Winner-take-all result using aggregated cost.

Errors marked in red

- Background && Motivation
- **Algorithm**
  - ✓ Framework
  - ✓ Pixelwise line segments
  - ✓ **Seed detection**
  - ✓ Reliable disparity propagation
  - ✓ Two-step disparity refinement
- Results
- Conclusion

Algorithm

Reliable seed detection



- Detect highly *reliable* seed pixels (known as *Ground Control Points* in stereo literature)
- Seed pixels are detected with the following rules operating on *Winner-take-all* results :

  (1) They should pass left-right consistency check

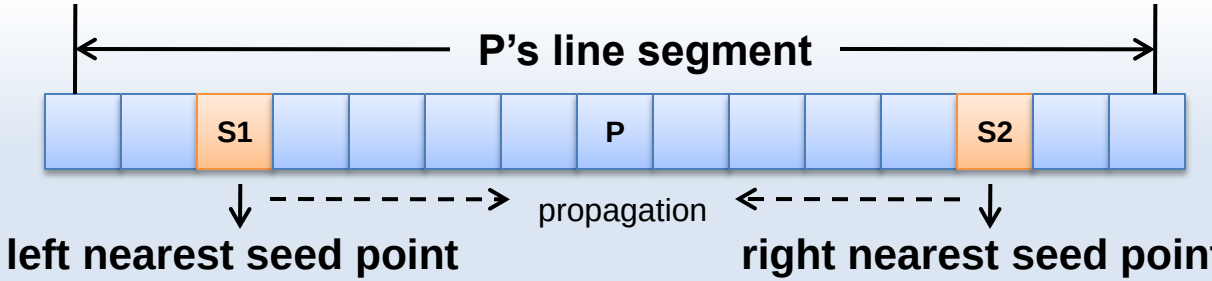
  (2) They should have high correlation confidence on cost volume

  (3) They should be well allocated near depth boaders

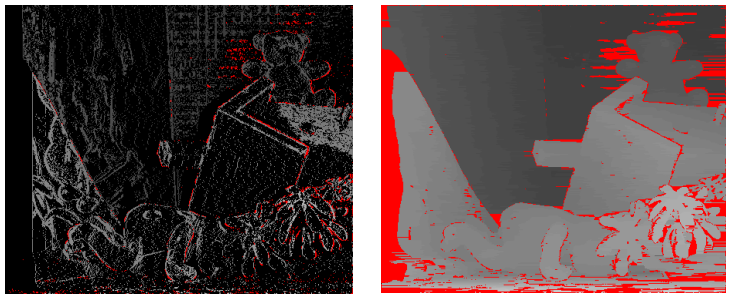
  Skip to the endpoint of previous starting point's line segment for next seed

- Background && Motivation
- **Algorithm**
  - ✓ Framework
  - ✓ Pixelwise line segments
  - ✓ Seed detection
  - ✓ **Reliable disparity propagation**
  - ✓ Two-step disparity refinement
- Results
- Conclusion

Reliable disarity propagation



Search for the nearest left and right seed pixels within its line segment, then its disparity is determined by following rules:



Propagation process

✓**Rule 1** **If only one seed is found:**

Simply replaced by the seed's disparity

✓**Rule 2** **Both seed pixels are available:**

(1) *Occluded (fail in left-right check), or **p** is near depth discontinuities*

$$D(P) = \min(D(S_1), D(S_2))$$

(2) In other cases linearly interpolated by

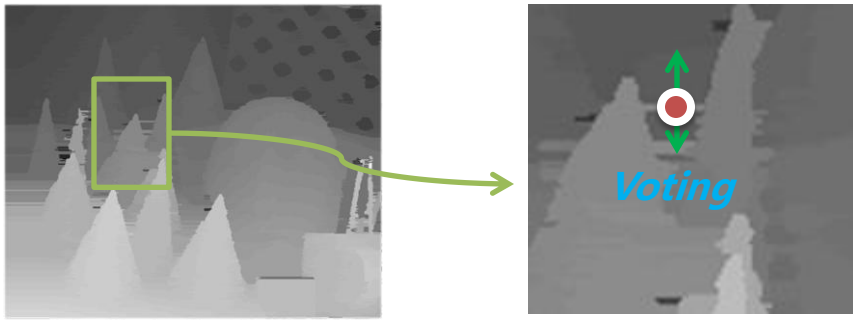
$$\frac{D(P) - D(S_1)}{D(S_2) - D(P)} = \frac{D_s(p_1, p)}{D_s(p_2, p)}$$

P's disparity

- Background && Motivation
- **Algorithm**
  - ✓ Framework
  - ✓ Pixelwise line segments
  - ✓ Seed detection
  - ✓ Reliable disparity propagation
  - ✓ **Two-step disparity refinement**
- Results
- Conclusion

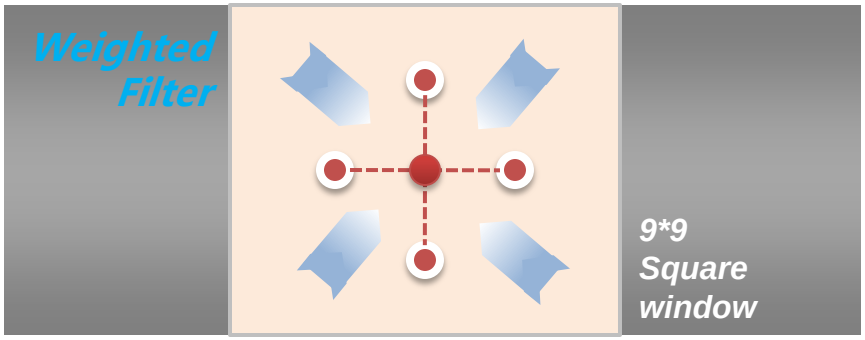
Algorithm

Two-step disparity refinement



□ Vertical voting:

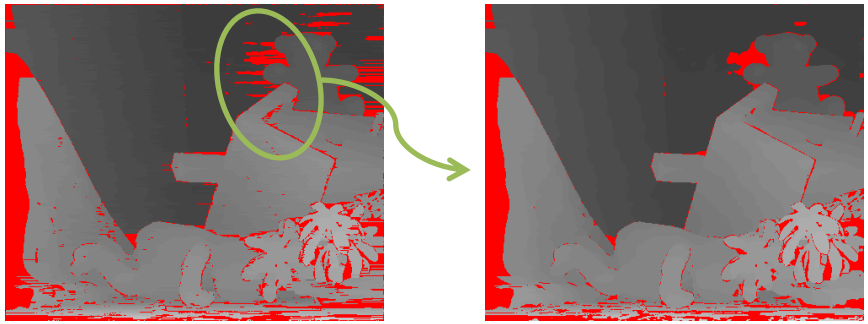
Pixelwise disparity adjustment along *vertical* direction by *voting*



□ 4-neighbor update with bilateral filtering:

(1) depth piecewise *smooth*

(2) color edges mostly *align* with depth edges



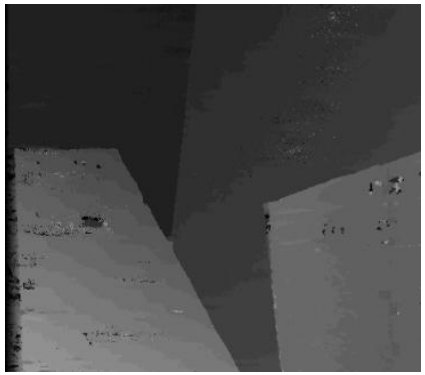
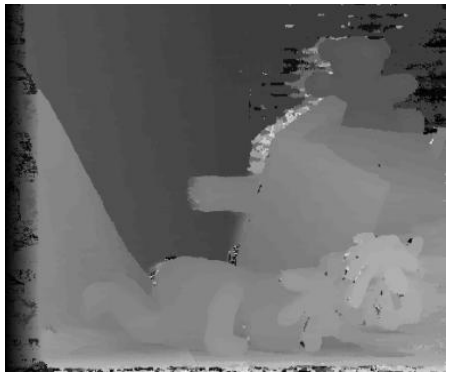
Streaking-artifact are effectively removed by this refinement process

- Background && Motivation
- Algorithm
  - ✓ Framework
  - ✓ Pixelwise line segments
  - ✓ Seed detection
  - ✓ Reliable disparity propagation
  - ✓ Two-step disparity refinement
- **Results**
- Conclusion

Results

Intermediate results: *Initial disparity map*

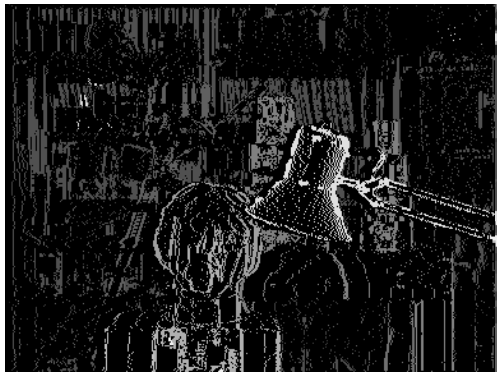


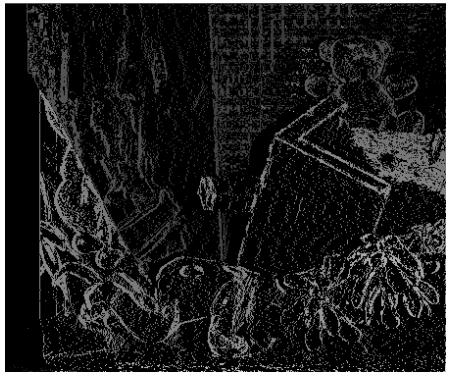





(Tsukuba) *(Venus)* *(Teddy)* *(Cones)*

Results

Intermediate results: *Seeds*



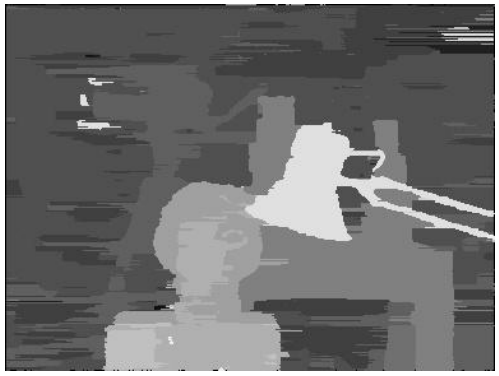
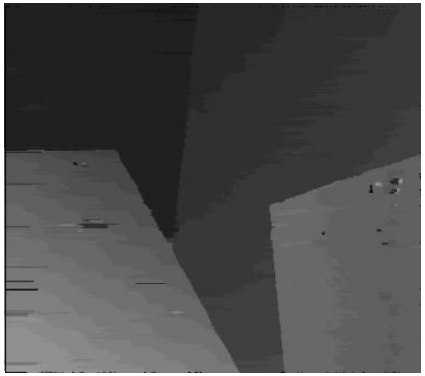
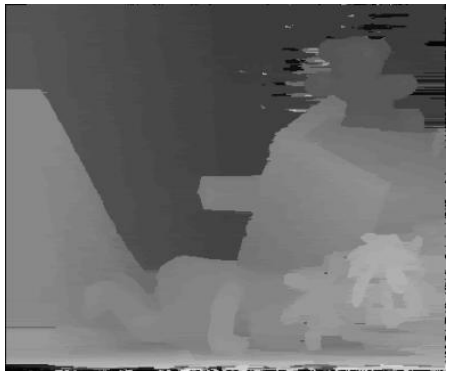
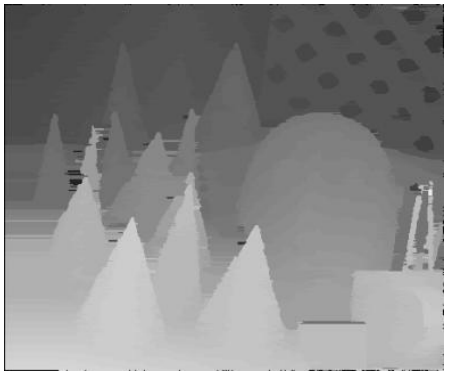


(Tsukuba) *(Venus)* *(Teddy)* *(Cones)*

Results

Intermediate results: *Propagation*



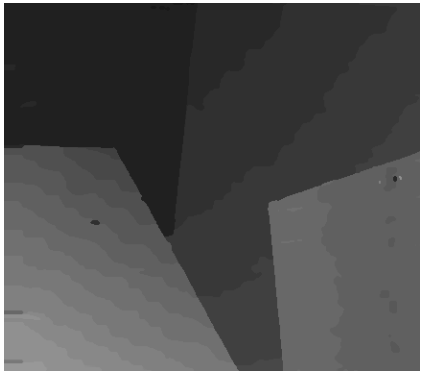
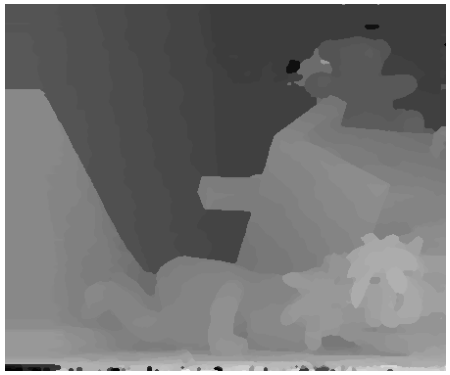
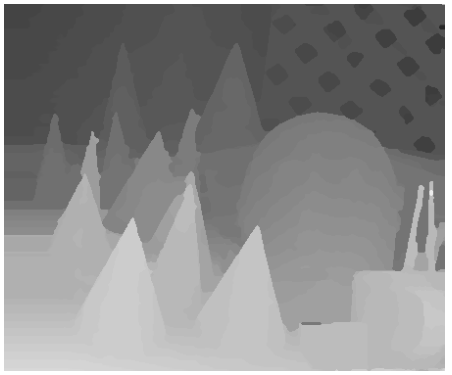





(Tsukuba) *(Venus)* *(Teddy)* *(Cones)*

Results

Intermediate results: *Refinement*

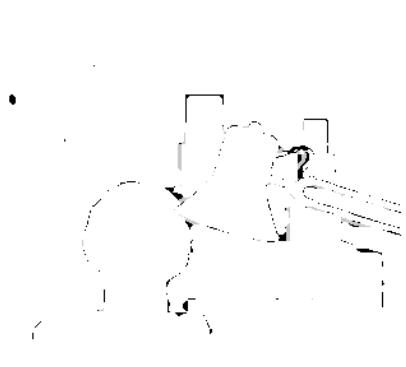


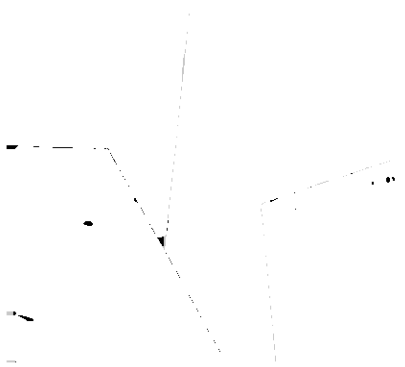





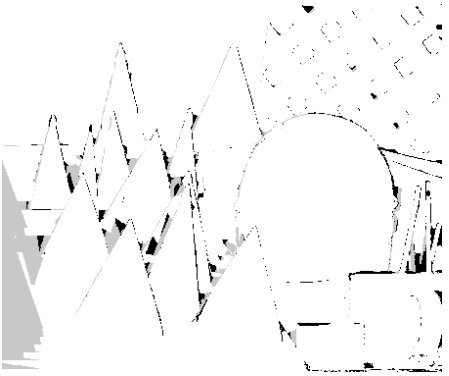
(Tsukuba) *(Venus)* *(Teddy)* *(Cones)*

Results

Intermediate results: *Errors > 1 pixel*







(Tsukuba) *(Venus)* *(Teddy)* *(Cones)*

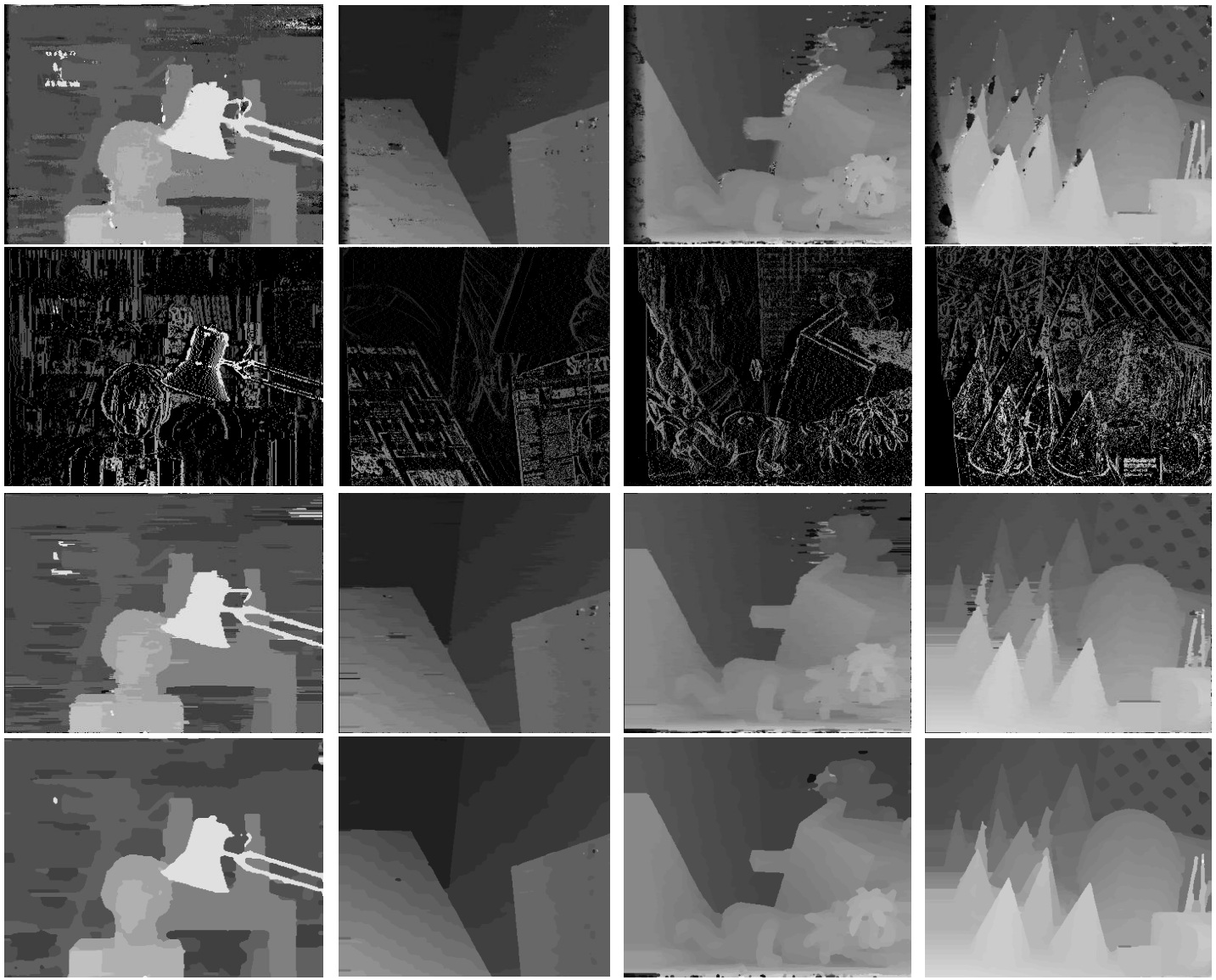
Results

Initial disparity maps

Seeds

Propagation

Refinement



Results

Quantitative Results – *Middlebury* ranking

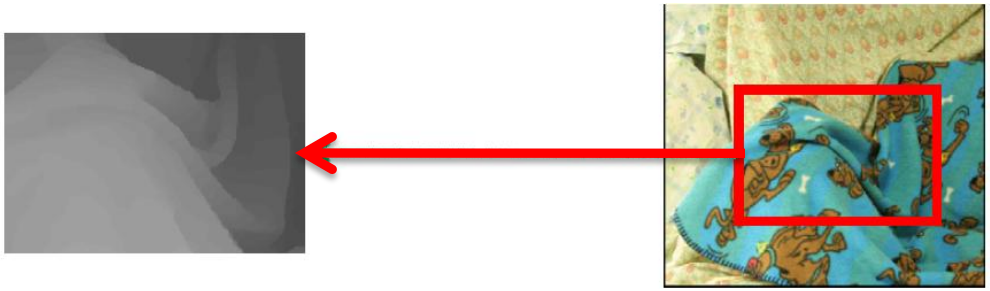
Algorithm	Avg. Rank	Tsukuba			Venus			Teddy			Cones		
		nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
CoopRegion [24]	6.6	0.87_{1}	1.16_{1}	4.61_{1}	0.11_{3}	0.21_{2}	1.54_{5}	5.16_{14}	8.31_{9}	13.0_{11}	2.79_{12}	7.18_{4}	8.01_{16}
DoubleBP [23]	8.8	0.88_{3}	1.29_{2}	4.76_{3}	0.13_{6}	0.45_{16}	1.87_{10}	3.53_{3}	8.30_{8}	9.63_{2}	2.90_{16}	8.78_{23}	7.79_{13}
Our method	9.4	0.97_{6}	1.39_{6}	5.00_{5}	0.21_{19}	0.38_{14}	1.89_{11}	4.84_{8}	9.94_{15}	12.6_{9}	2.53_{5}	7.69_{7}	7.38_{8}
OutlierConfi [25]	9.7	0.88_{2}	1.43_{8}	4.74_{2}	0.18_{13}	0.26_{8}	2.40_{17}	5.01_{10}	9.12_{12}	12.8_{10}	2.78_{11}	8.57_{19}	6.99_{4}
SurfaceStereo [26]	13.7	1.28_{24}	1.65_{15}	6.78_{28}	0.19_{15}	0.28_{9}	2.61_{24}	3.12_{1}	5.10_{1}	8.65_{1}	2.89_{15}	7.95_{11}	8.26_{20}

Our method takes the 5th rank of 106 submissions

Performance on efficiency

One of the most efficient algorithms among accurate *non-GPU* methods: time costs for *Tsukaba, Venus, Teddy* and *Cones* are *2.5 s, 3.8 s, 8.7s* and *8.6s* (PC with Core2 Duo 3.0GHz CPU, 2GB memory)

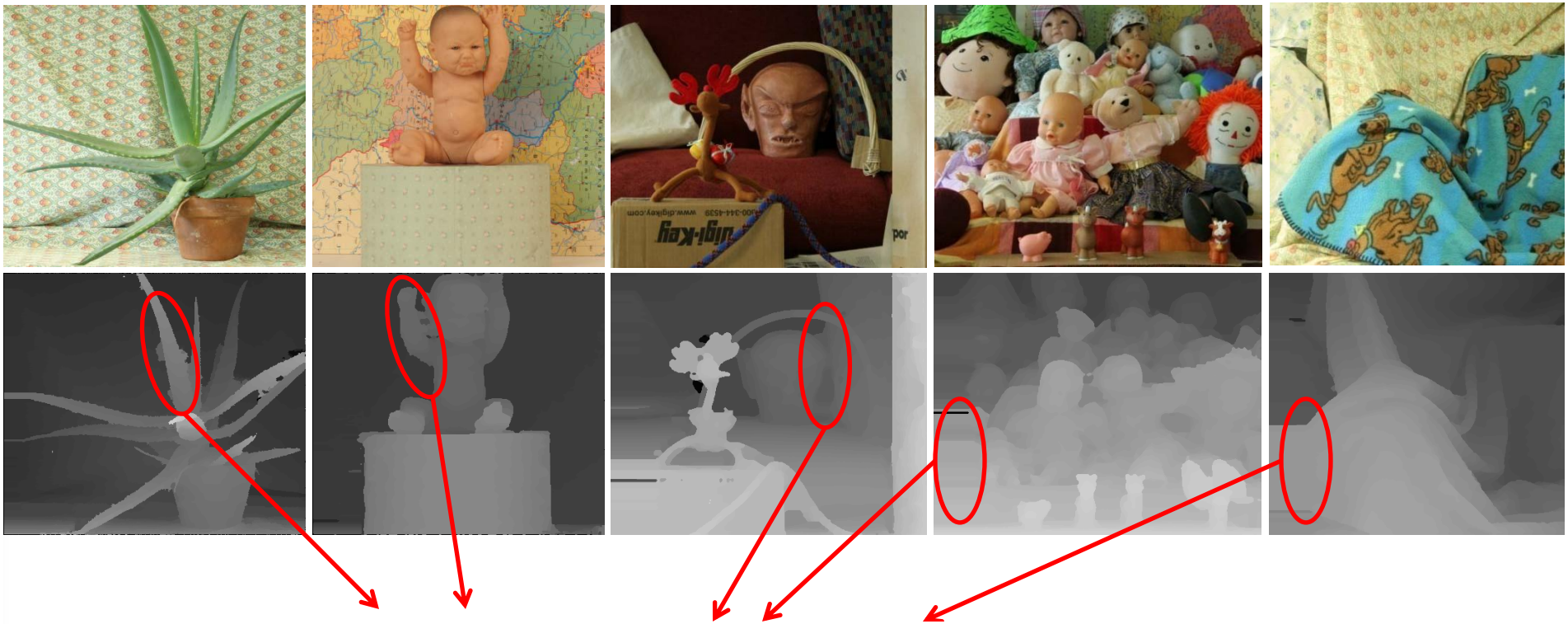
More results on *Middlebury* data sets



Fine details on curved surface

Results

More results…



Some errors are marked in *red* ellipses:
mainly due to the lack of reliable seeds in large occluded regions

- Background && Motivation
- Algorithm
  - ✓ Framework
  - ✓ Pixelwise line segments
  - ✓ Seed detection
  - ✓ Reliable disparity propagation
  - ✓ Two-step disparity refinement
- Results
- **Conclusion**

conclusions

- A simple stereo algorithm compute results comparable to current state-of-the-art on *Middlebury* benchmark

- The accuracy of the propagation can be guaranteed with the pixelwise line segments, which shows good performance in both *speed* and *accuracy*.

As a future work, we would like to apply our algorithm to real-world problems. And we plan to port the algorithm to graphics hardware for acceleration.

Thanks!

Q&A