Stereo Matching

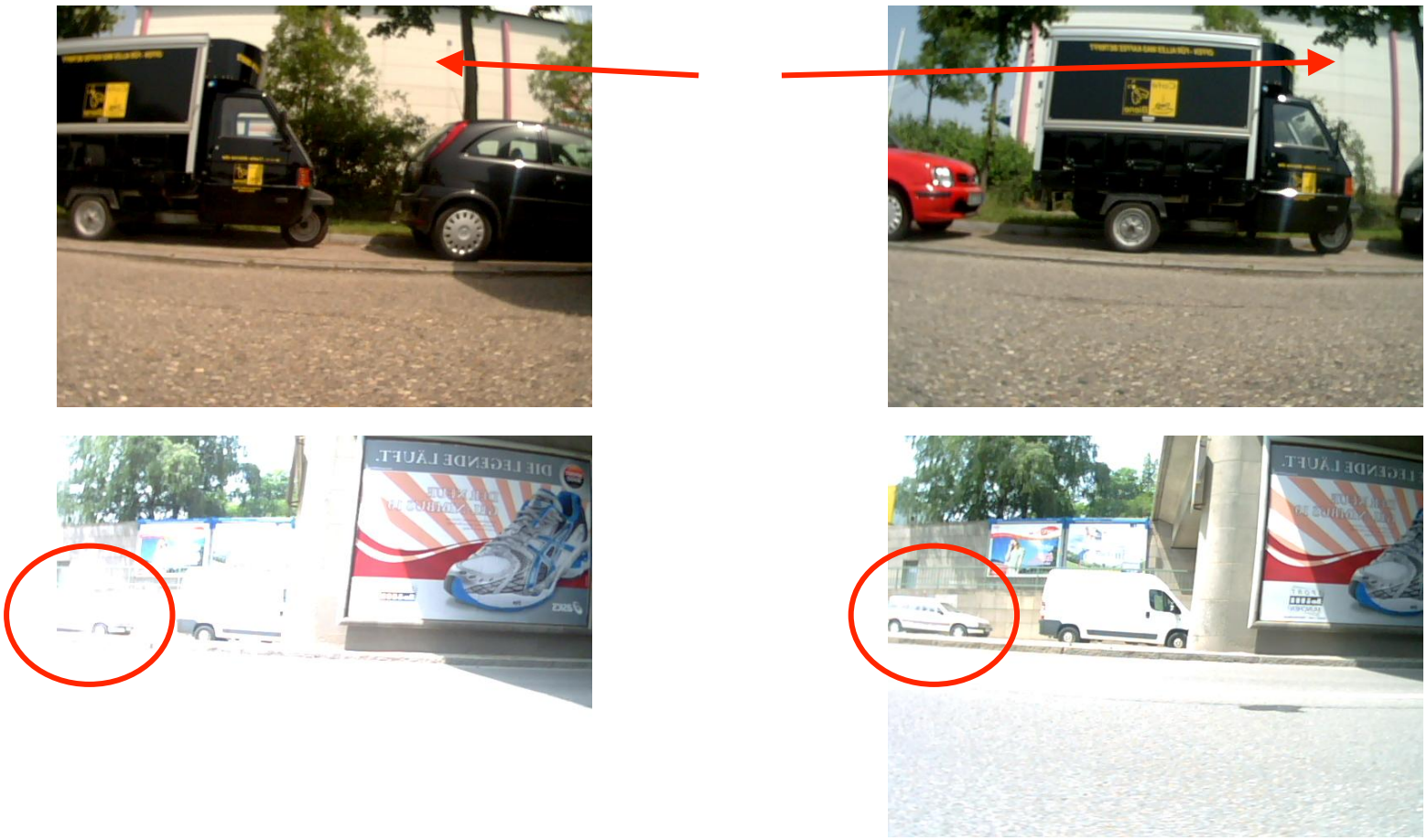
Nassir Navab

Slides prepared by Christian Unger

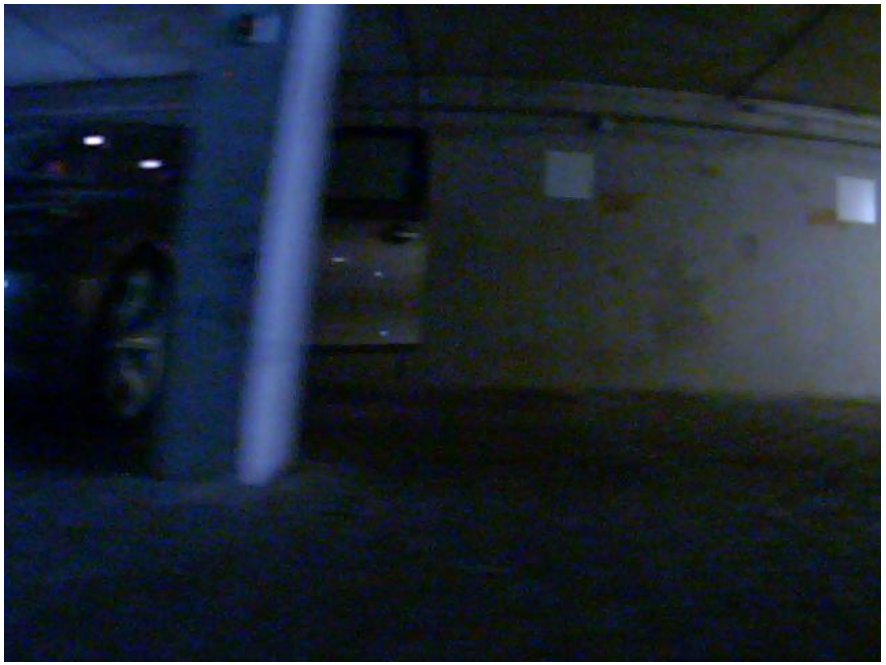
Hardware Architectures.

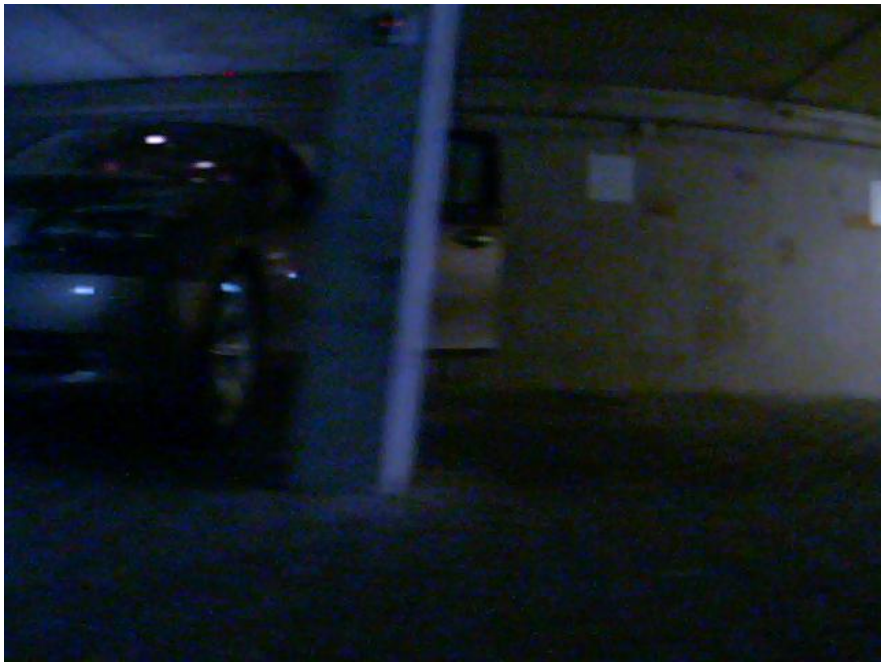
- Microprocessors
  - Pros: floating-point, SIMD, C/C++
  - Cons: power-consumption, cost, size
- Low Power/Low Cost Processors
  - Pros: low power, low cost, C/C++
  - Cons: mostly no floating-point, mostly no SIMD
- GPUs
  - Pros: processing power
  - Cons: difficult programming, power-consumption, cost, size
- FPGA
  - Pros: efficient, low power, low cost
  - Cons: code is very specifc (VHDL/Verilog)

Challenges: Photometric Variations.



Challenges: Image Sensor Noise.



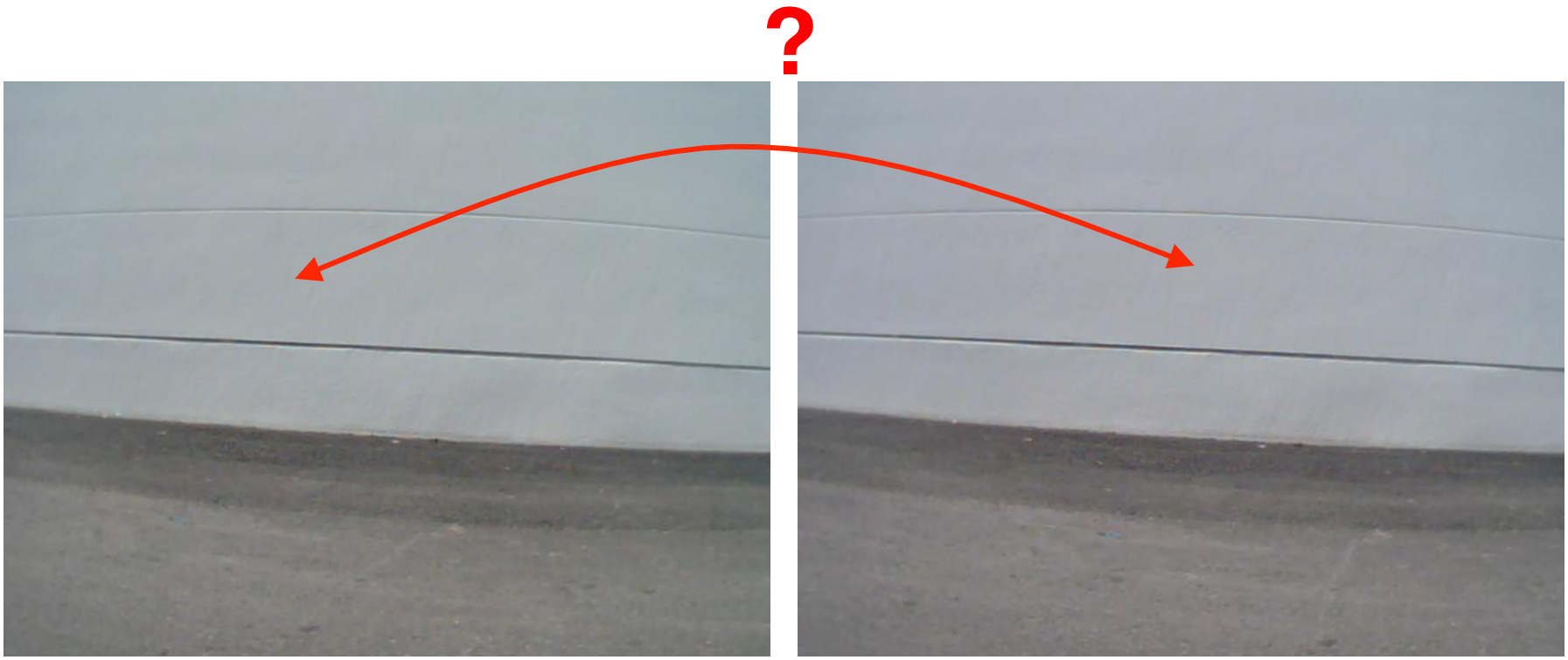


Challenges: Specularities.

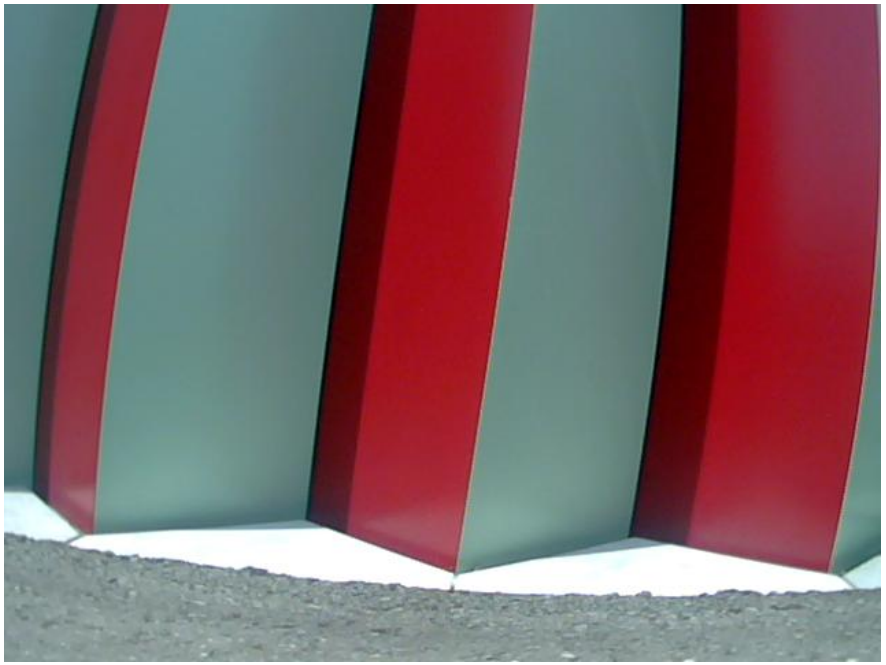
Challenges: Foreshortening and the Uniqueness Constraint.

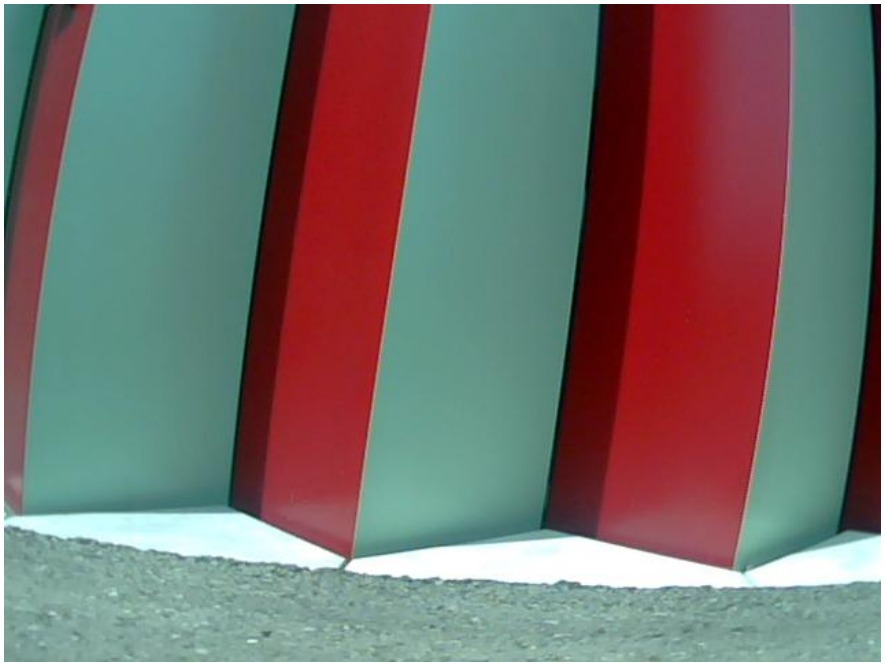
Challenges: Perspective Distortions.

Challenges: Textureless Regions.



Challenges: Repetitive Structures and Textures.



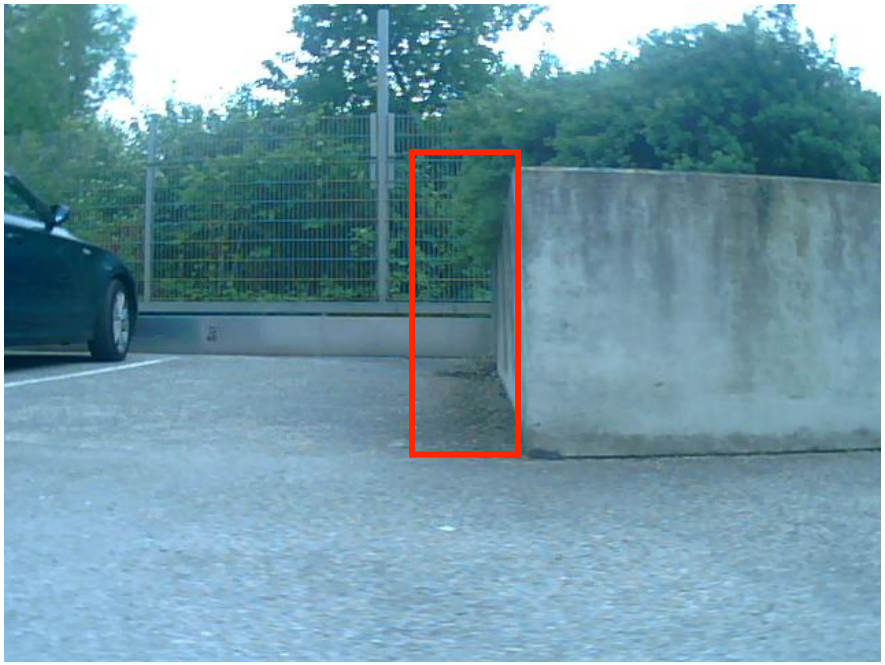


Challenges: Reflections.

Challenges: Reflections.

Challenges: Transparency.

Challenges: Occlusions.



Challenges: Occlusions.

Taxonomy of Stereo Matching.

Most stereo algorithms are composed of the following steps:

1. Preprocessing
2. Matching cost computation
3. Cost aggregation
4. Disparity computation
5. Disparity refinemen

There are mainly two classes of algorithms:

1. Local methods (“simple search”): perform 2, 3, 4
2. Global methods (e.g. minimizing a global energy functional): perform 2, (3,) 4

Preprocessing.

To alleviate sensor noise and photometric distortions, typically the following methods are employed:

- Laplacian of Gaussian (LoG) filtering (Kanade et al. Development of a Video-Rate Stereo Machine. IROS 1995.)
- Histogram Equalization/Matching
- Subtraction of mean values computed in the neighbours of each pixel (Faugeras et al. Real-Time correlation-based stereo: Algorithm, Implementation and Applications, INRIA TR 2013, 1993.)
- Bilateral filtering (Ansar et al. Enhanced real-time stereo using bilateral filtering. CVPR 2004.)

Matching Cost.

A **matching cost** measures the **similarity** of pixels.
Simple examples:

- Absolute intensity difference (AD)

$$| I_L(x,y) - I_R(x,y) |$$

- Squared intensity difference (SD)

$$(I_L(x,y) - I_R(x,y))^2$$

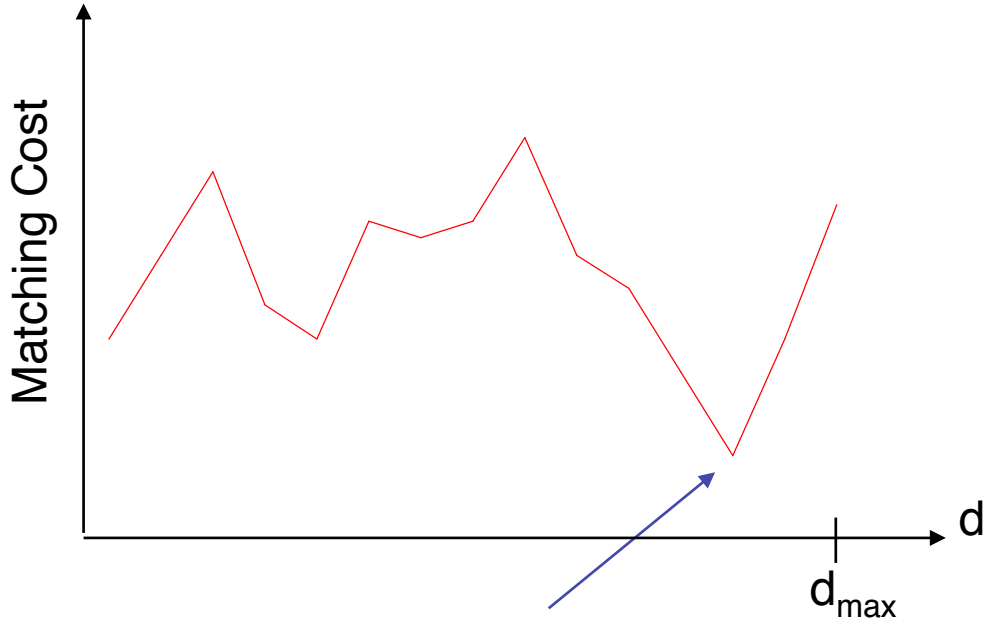
Disparity Computation.

The corresponding pixel is chosen in a way such that the similarity between the pixels is high (“dissimilarity” = cost).

Simple “Winner Takes All”-Algorithm:

For every pixel select the disparity with lowest cost

$$|I_L(x,y) - I_R(x+d,y)|$$



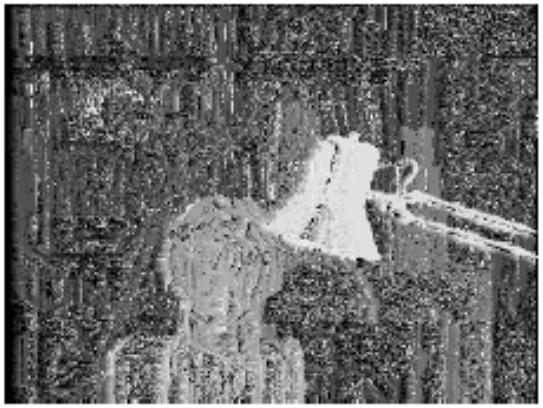
Example Algorithm.

Using this simple algorithm the disparity map looks like this:

Left Camera Image

Optimal Result

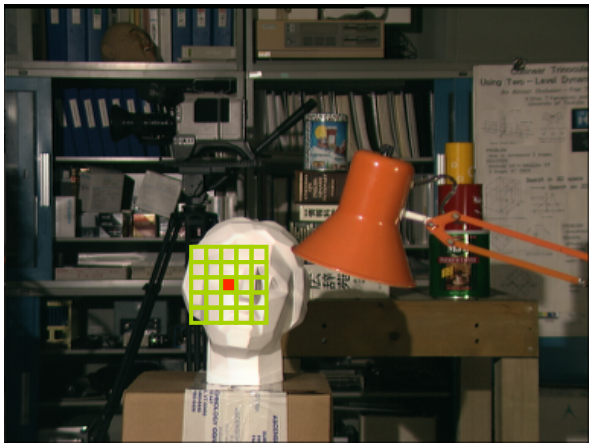
Actual Result (Bad!)

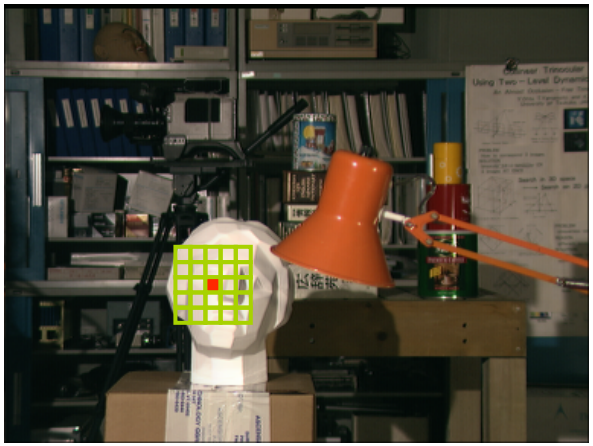


The disparity map is very noisy, due to a low signal-to-noise ratio (SNR).

Remedy: **Cost Aggregation**: Do not compare single pixels, but small patches.

Cost Aggregation.





Use a “matching window” around the pixel of interest.

Sum of **absolute** intensity **differences** (SAD):

$$\sum_{(x,y)\in W} | I_R(x,y) - I_L(x+d,y) |$$

Cost Aggregation.

Examples for such **area-based matching costs**:

- Sum of absolute differences (SAD)

$$\sum_{(x,y)\in W} | I_R(x,y) - I_L(x+d,y) |$$

- Sum of squared differences (SSD)

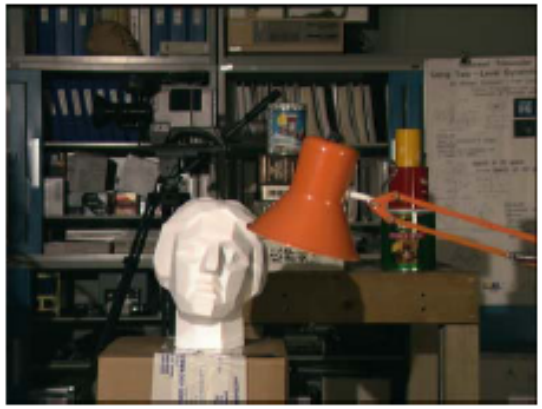
$$\sum_{(x,y)\in W} (I_R(x,y) - I_L(x+d,y))^2$$

- Normalized Cross Correlation (NCC)
- Mutual Information

Cross Correlation (WTA with SAD).

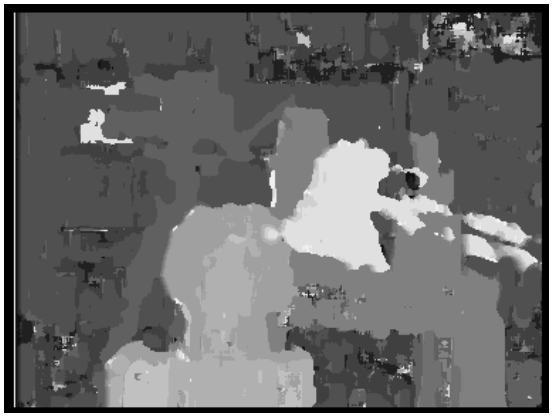
Again we use the “Winner-Takes-All”-algorithm as described before, but now with an area-based matching cost (SAD).

Left Camera Image



Optimal Result

Actual Result (Still many errors)



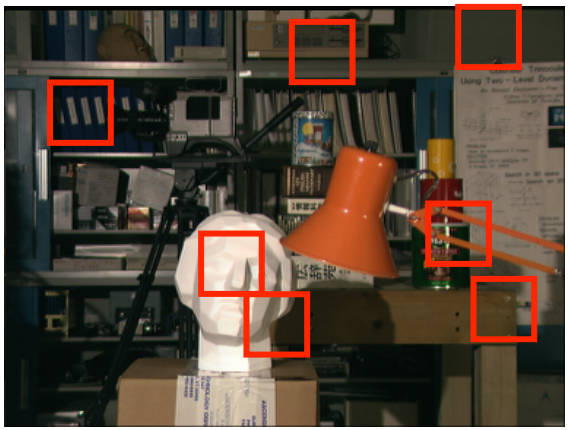
Better than before, but still not optimal.

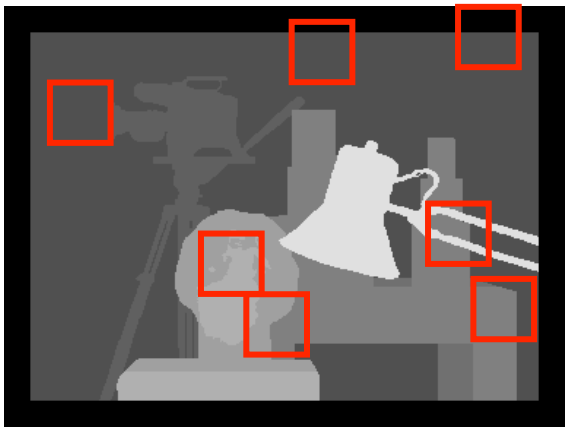
Why?

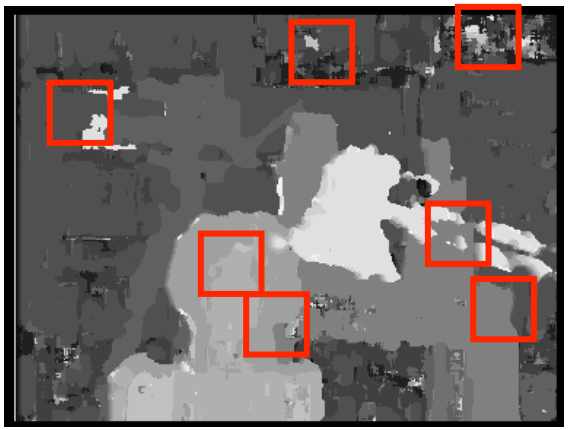
Problems with Fixed Windows.

The area-based approach has other problems:

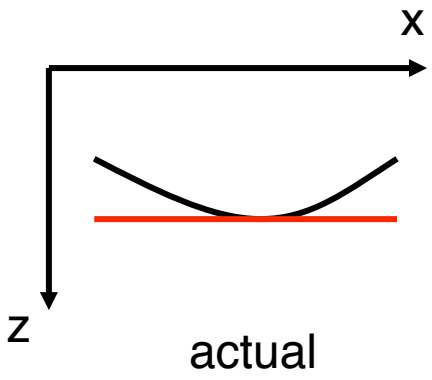
- Assumes constant depth within the window – this implicit assumption is violated at
  - Depth discontinuities
  - Slanted/non-planar surfaces
- Repetitive textures
- Uniform areas
- Thin structures (window larger than the structure)

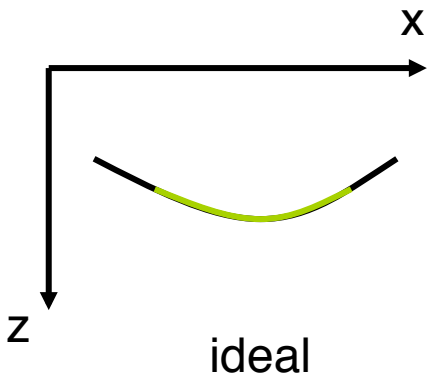






Problems with Slanted Surfaces in Detail.





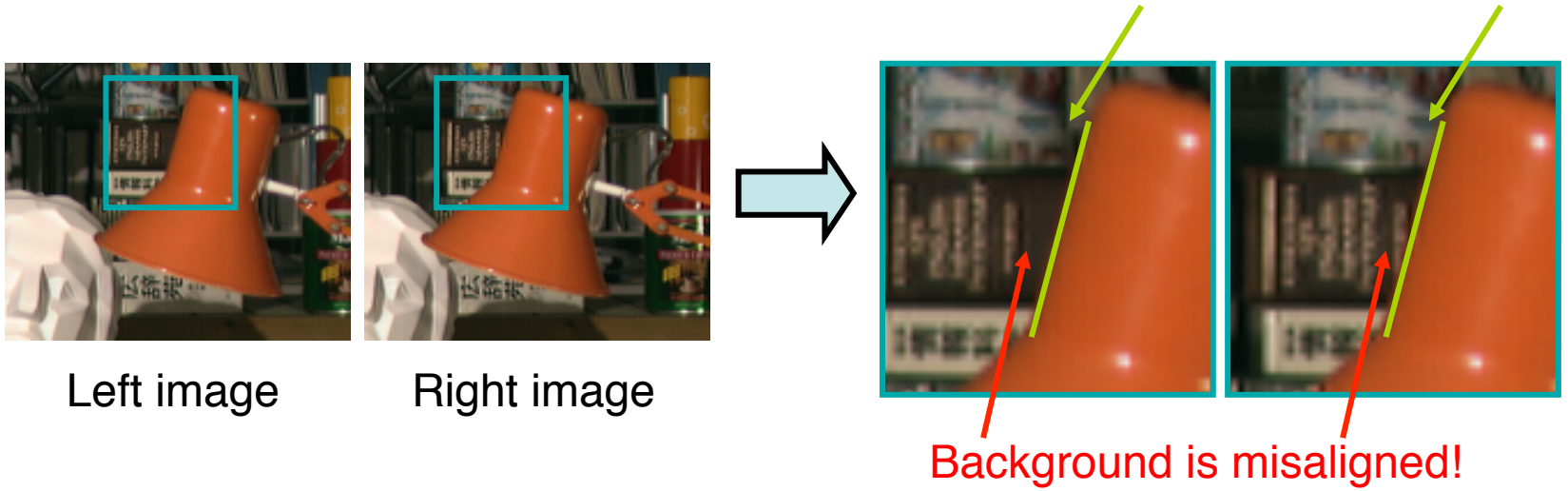
Implicitly, flat fronto-parallel surfaces are assumed in most area-based approaches.

Ideally, the matching window should take the perspective distortion into account, too.

But this is a “chicken and egg problem”:

We would need the depth for that, but depth is actually what we want to determine!

Problems with Discontinuities in Detail.



The problem is that the matching windows cover pixels which lie at different depths.

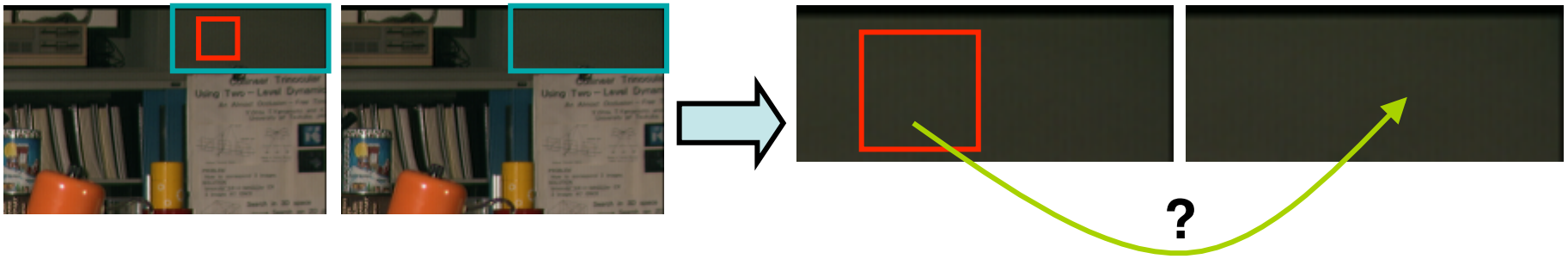
=> This leads to wrong values in the matching costs.

=> Occluded pixels should not be compared to any other pixel.

Possible remedies:

- Split the matching window into multiple parts (“Multiple Windows”: Hirschmüller et al. Real-Time Correlation-Based Stereo Vision with Reduced Border Errors. IJCV 2002.).
- Shiftable windows (change the center pixel of the windows dynamically)

Problems with Uniform and Repetitive Areas in Detail.



In such cases, there are usually many “weak” minima of the matching cost and image sensor noise easily leads to wrong matches.

Possible remedies:

- Use larger matching windows adaptively (Veksler. Fast variable windows using integral images. CVPR 2003.).
- Measure the significance of the minimum (are there other minima with similar costs) and detect weak minima.

Summary: Cross Correlation.

However – despite of the drawbacks of area-based approaches, cross correlation (WTA with SAD) is often adopted in practice.

It is

- simple,
- fast (Real-Time on standard hardware),
- and has low memory requirements.

Memory requirement is low, because we need no additional information execpt the disparity for every pixel.

How about execution time?

Currently: 1400 ms for 320x200 pixels!!

Improvement: Integral Images.

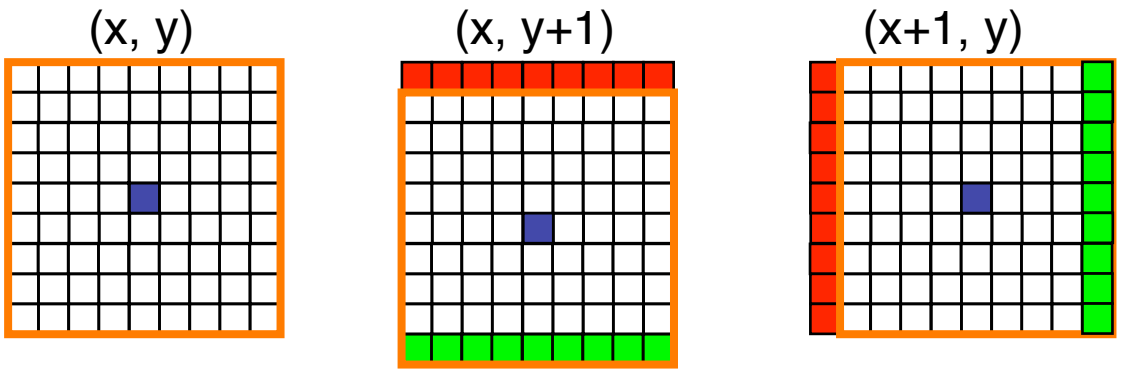
TODO: include or exclude Integral Imges?

1 Slide: Integral Images

Improvement: Box Filtering.

Based on a mathematical observation that many matching costs can be separated, such that they can be computed incrementally.

If we know the SAD-value of a window at a certain position – a nearby window can be computed incrementally.



Improvement: Box Filtering.

The mathematics in detail:

$$SAD(x,y,d) = \sum_{j=-W}^{W} \sum_{i=-W}^{W} | IL(x+i, y+j) - IR(x+i+d, y+j) |$$

$$SAD(x,y+1,d) = \sum_{j=-W}^{W} \sum_{i=-W}^{W} | IL(x+i, y+1+j) - IR(x+i+d, y+1+j) |$$

$$SAD(x,y+1,d) = \sum_{j=-W+1}^{W+1} \underbrace{\sum_{i=-W}^{W} | IL(x+i, y+j) - IR(x+i+d, y+j) |}_{=:U(x,y,d,j)}$$

$$SAD(x,y+1,d) = SAD(x,y,d) + U(x,y,d,W+1) - U(x,y,d,-W)$$

Improvement: SIMD.

Single Instruction, Multiple Data.

More or less a purely implementational optimization utilizing advanced assembler routines.

Allows to process data in parallel (for example, work on up to 16 pixels at a time)

Of special interest is the `PSADBW`-instruction (MMX, SSE2):

Computes the sum of absolute differences of eight/sixteen unsigned byte integers

Execution Time Revisited.

Using these optimizations, 320x200x40 disparities can be processed in 30 ms.

Can we go further?

Yes (to about 50%).

> By not processing all disparities: instead of a brute-force search, use an iterative search technique that automatically stops at a certain point (Unger et al. Efficient Disparity Computation without Maximum Disparity for Real-Time Stereo Vision. BMVC 2009.).

Example.

Example: Video-Sequence from the vehicle.

Global Methods: Overview.

The area-based approach presented falls into the category of “local methods”, since the disparity-computation is done for every single pixel alone.

However, there is another big class of methods, the global and semi-global methods. In these approaches, the task of computing disparities is cast as an energy minimization problem.

Typically, an energy functional is formulated such as:

$$E(D) = E_D(D) + \lambda E_S(D)$$

where E_D measures the pixel similarity and E_S penalizes disparity variations.

Well known methods to solve such problems are

- Dynamic Programming
- Scanline Optimization
- Graph Cuts
- Belief Propagation, …

Global Methods: Examples.

Adapting Belief Propagation

Graph Cuts

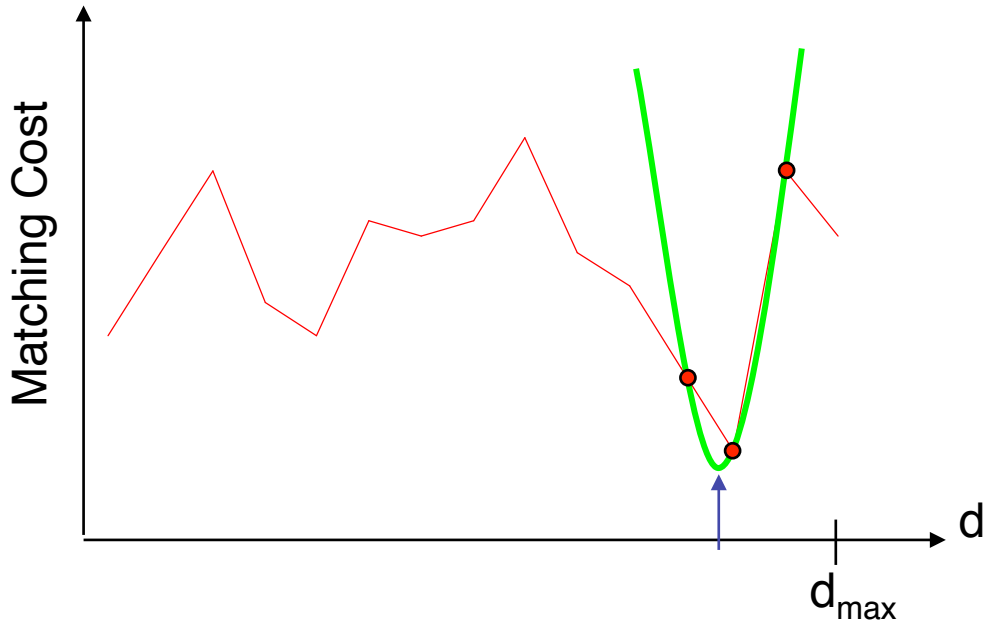
However, this quality is at the cost of execution time (e.g. Graph Cuts: 5 minutes).

Disparity Refinements: Sub-Pixel Interpolation.

We discussed only the computation of integer valued disparities, but the world is continuous.

Real valued disparities may be obtained by approximating the cost function locally using a parabola:

$$|I_L(x,y) - I_R(x+d,y)|$$



Disparity Refinements: Left-Right Consistency Check.

Outlier detection routine.

Perform the stereo matching two times:

1. By computing a disparity for every pixel of the left image (left to right)
2. And again, by computing a disparity for every pixel of the right image (right to left)

Then, if these two disparities differ an outlier has been found.