Stereo Vision II: Dense Stereo Matching

Nassir Navab

Slides prepared by Christian Unger

Outline.

- Hardware.
- Challenges.
- Taxonomy of Stereo Matching.
- Analysis of Different Problems.
- Practical Considerations.
- Global Methods.

Hardware.
Cameras.

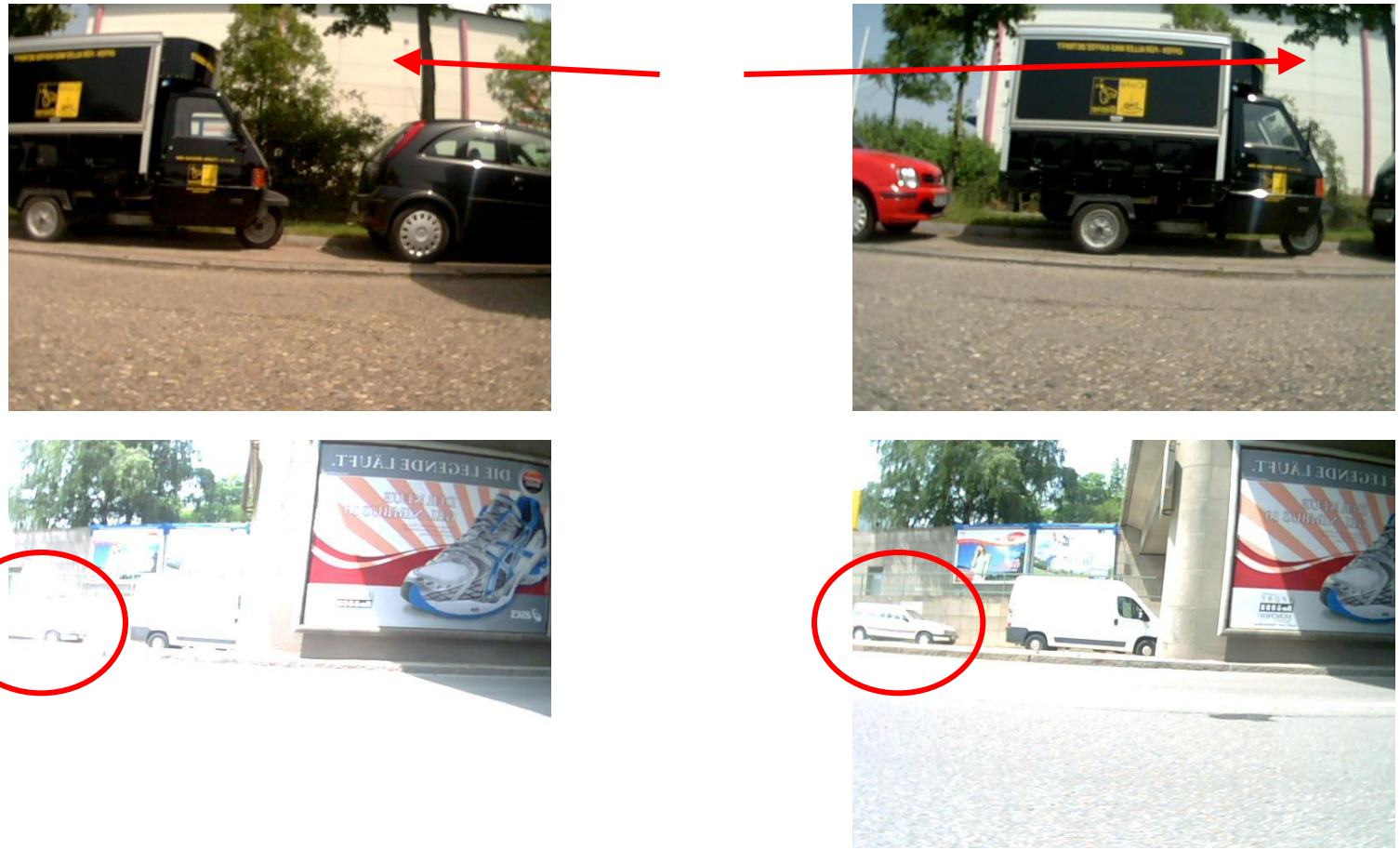
Cameras may be self-built, but some aspects need special attention:

- Imager Choice (e.g. RGGB/Monochrome/Red-Clear, IR-cut-off).
- Lens Choice.
- Temporal synchronization of left and right camera.
- Mechanical stability and baseline.
- Interface (e.g. Full-HD @ 24 FPS @ 16 Bit = 94 MB/s per camera).

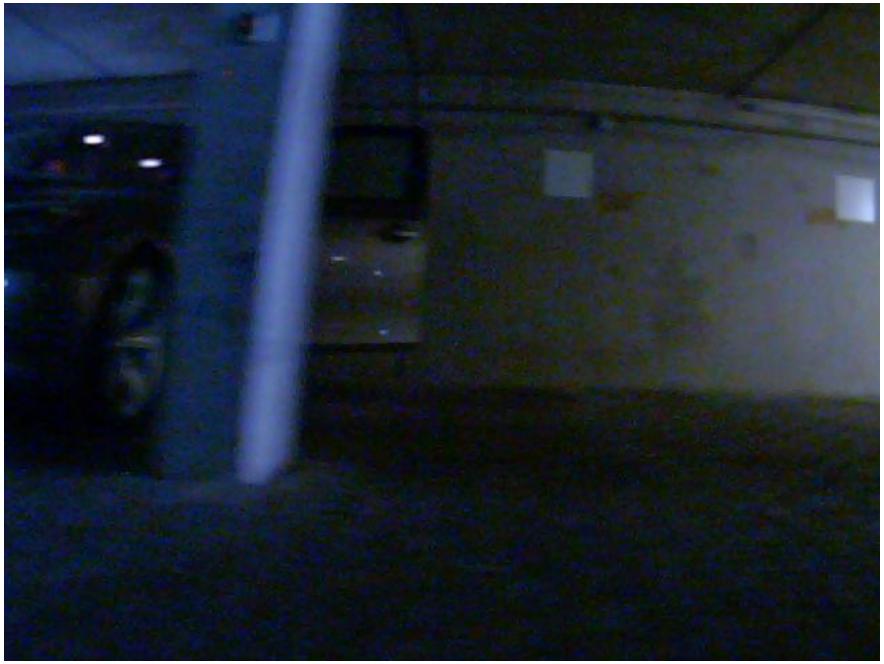
Hardware.
Processing Architectures.

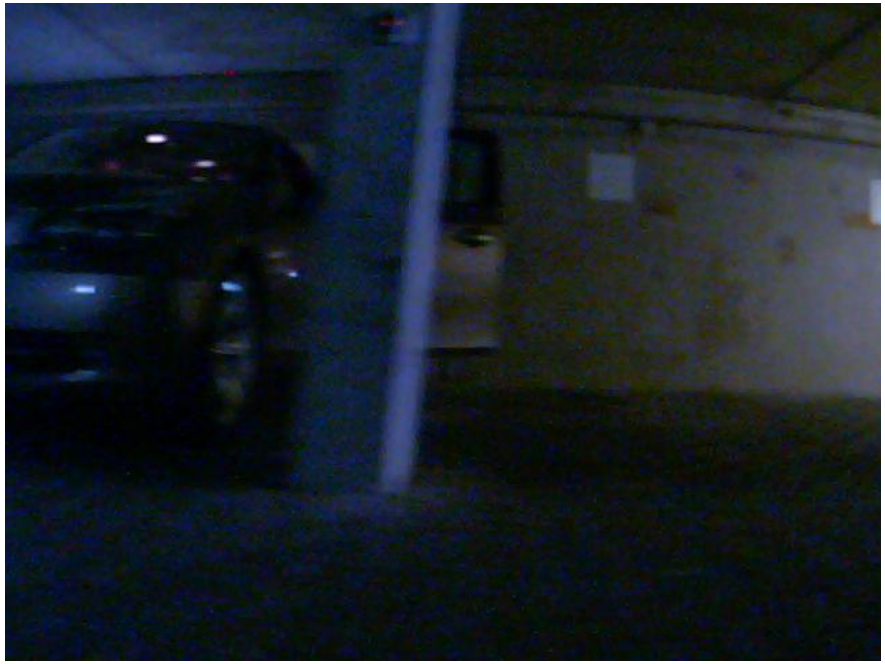
- Microprocessor, for example Intel x86 or PowerPC G3
  - Pros: Floating-point and SIMD support; Programmable using C/C++
  - Cons: High consumption; Cost
- Low Power/Low Cost Processors, for example ARM
  - Pros: Low consumption; Low cost; Programmable using C/C++
  - Cons: Often no floating-point and no SIMD support
- GPU, for example NVIDIA Tesla or Tegra
  - Pros: Processing power of non-embedded versions
  - Cons: Difficult programming; Power-consumption; Not yet comparable to FPGA/DSP
- FPGA, for example XILINX Spartan or Virtex
  - Pros: Efficient; Low consumption
  - Cons: Code is very specifc (VHDL/Verilog)
- DSP, for example Texas Instruments Vision Series
  - Pros: Efficient; Low consumption; Programming easier than FPGA
  - Cons: Performance for image processing tasks varies

Challenges: Photometric Variations.



Challenges: Image Sensor Noise.





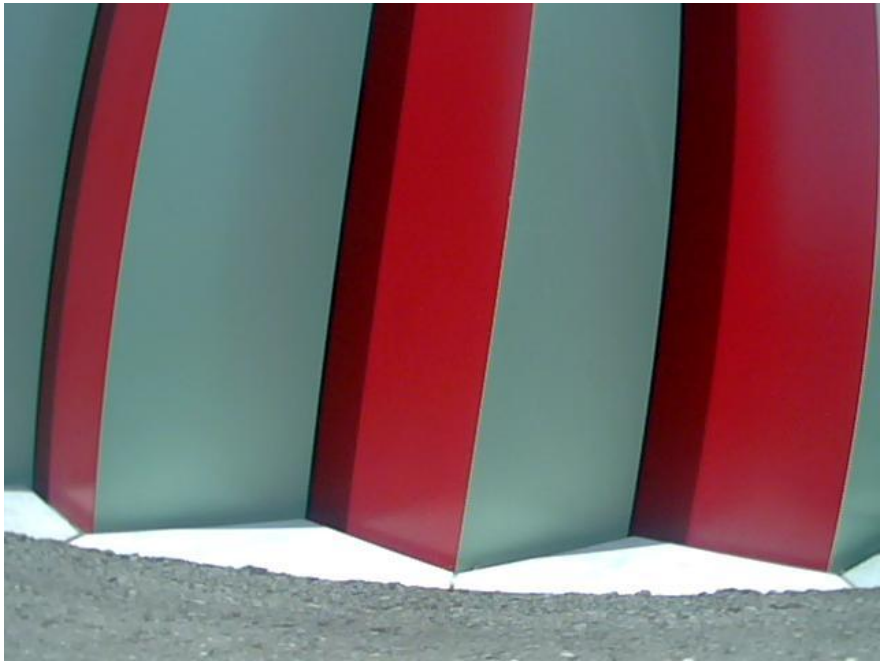
Challenges: Specularities.

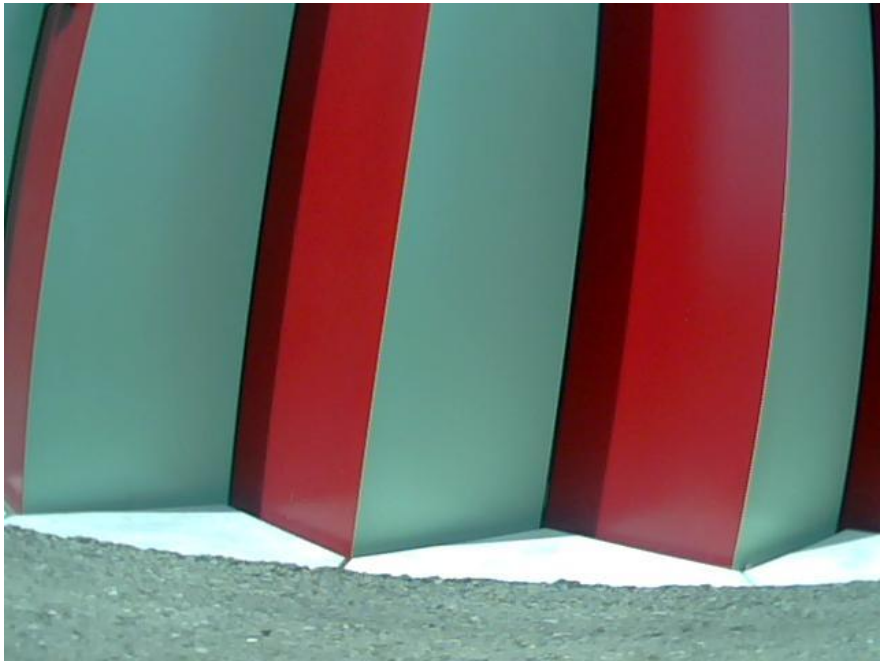
Challenges: Foreshortening and the Uniqueness Constraint.

Challenges: Perspective Distortions.

Challenges: Textureless Regions.

Challenges: Repetitive Structures and Textures.





Challenges: Reflections.

Challenges: Reflections.

Challenges: Transparency.

Challenges: Occlusions (right to left matching).

Challenges: Occlusions (left to right matching).

Taxonomy of Stereo Matching.

Almost all stereo algorithms are composed of the following steps:

1. Pre-Processing
2. Matching cost computation
3. Cost aggregation
4. Optimization
5. Disparity computation
6. Disparity refinement

There are mainly two classes of algorithms:

1. Local methods: do not perform a global optimization and use only a “simple search” at every pixel.
2. Global methods: minimize a global energy functional and often do not use cost aggregation.

Taxonomy of Stereo Matching.
Preprocessing.

To alleviate sensor noise and photometric distortions, usually 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.)

Taxonomy of Stereo Matching.
Matching Cost.

A **matching cost** measures the **similarity** of pixels and is a function of (x, y, d).
Simple examples:

- Absolute intensity difference (AD)

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

- Squared intensity difference (SD)

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

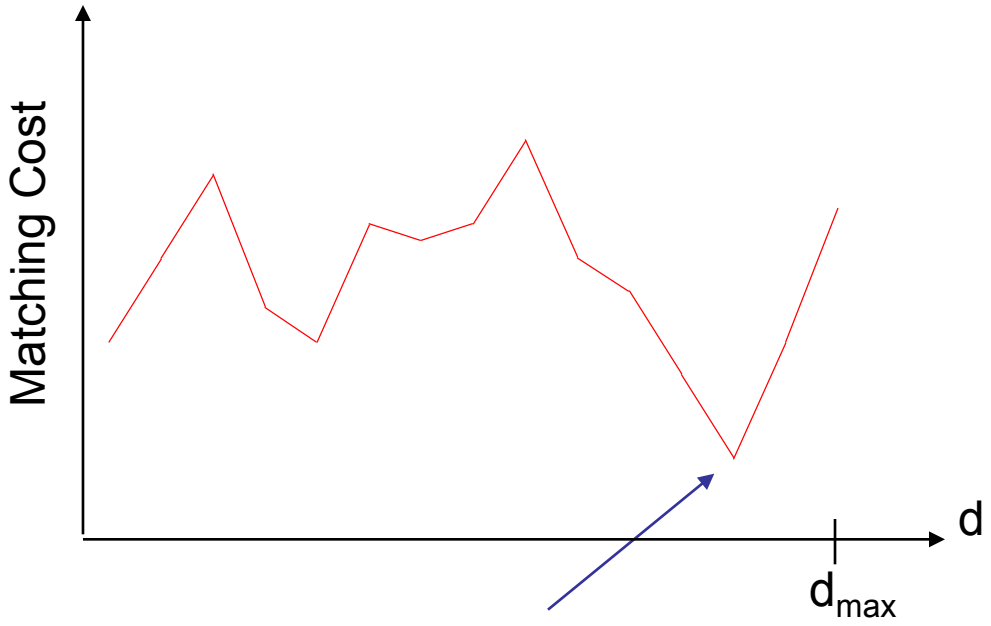
Taxonomy of Stereo Matching.
Disparity Computation in Local Methods.

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

Simple “Winner Takes All”-Algorithm (WTA):

For every pixel select the disparity with lowest cost

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

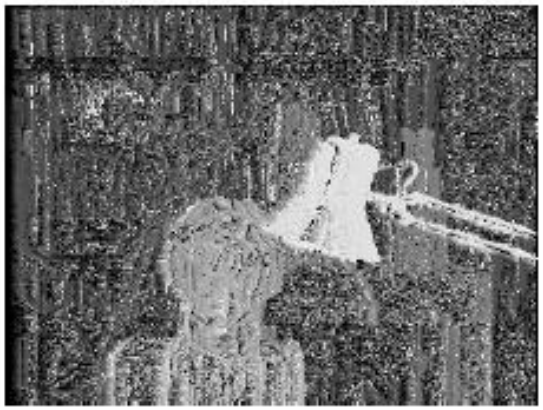


Taxonomy of Stereo Matching.
Example Algorithm: WTA with AD.

Using this simple algorithm (WTA + AD) the disparity map looks like this:

Left Camera Image | Optimal Result | Actual Result (Bad!)

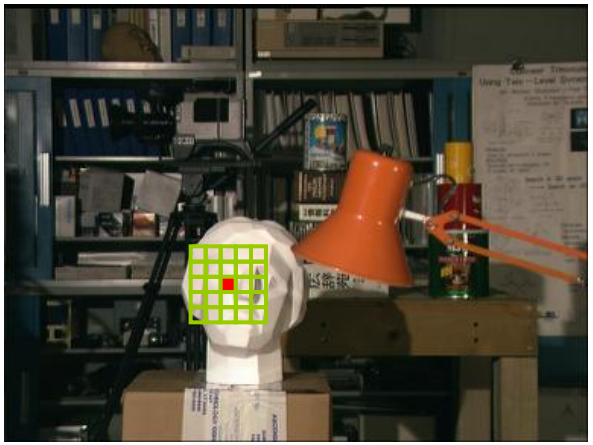


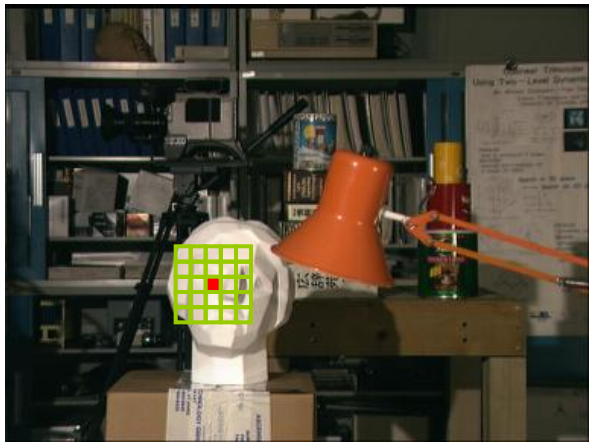


The disparity map is very noisy, due to a low signal-to-noise ratio (SNR) and due to a high ambiguity in textureless regions.

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

Taxonomy of Stereo Matching.
Cost Aggregation.





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

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

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

Taxonomy of Stereo Matching.
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

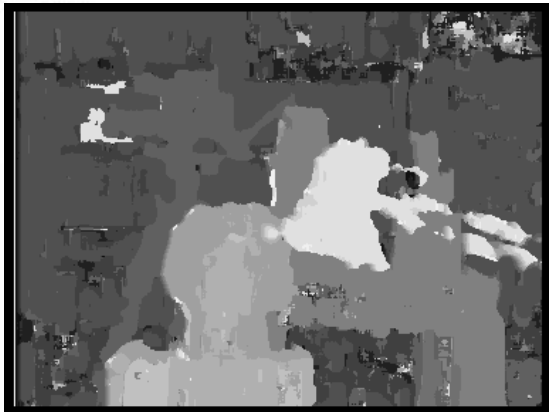
Taxonomy of Stereo Matching.
Example Algorithm: WTA with SAD.

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

Left Camera Image

Optimal Result

Actual Result (Still many errors)



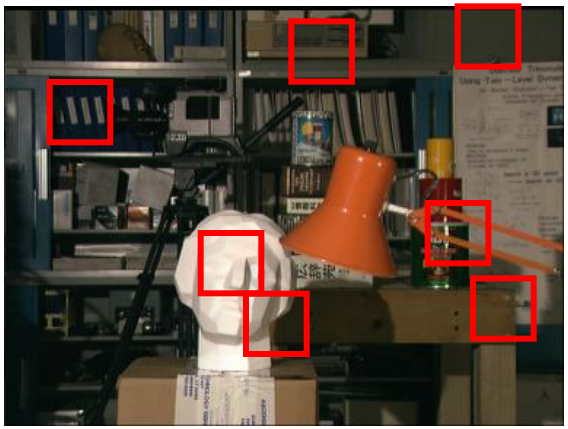
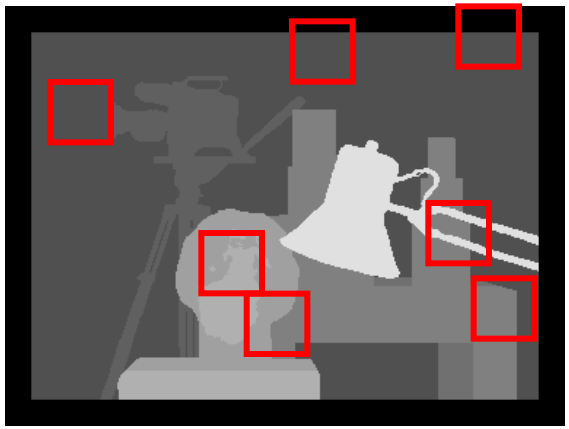
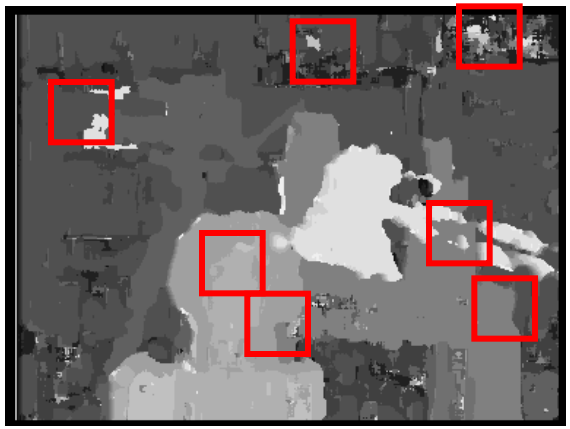
Better than before, but still not optimal.

Why?

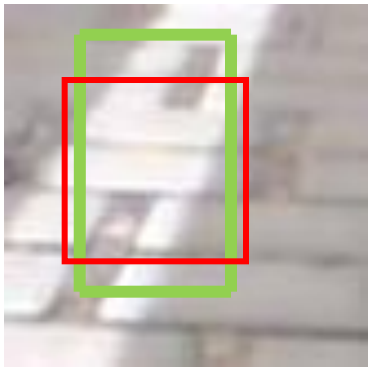
Analysis.
Problems with Fixed Windows.

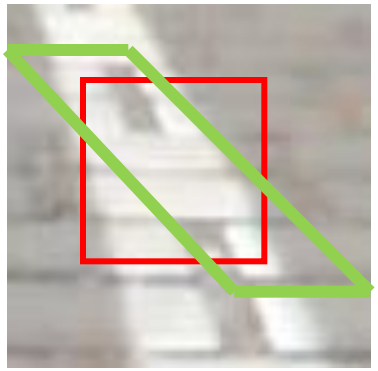
The area-based approach brings other merits:

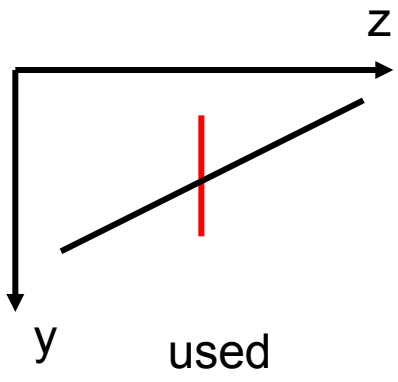
- A constant depth within the window is assumed – this implicit assumption is violated at
  - Depth discontinuities
  - Slanted/non-planar surfaces
- Matching for repetitive textures is still ambiguous.
- Matching in uniform areas is still ambiguous.
- Thin structures may be lost, if window is larger than the structure

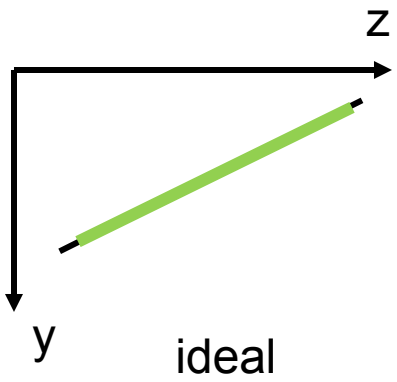




Analysis.
Problems with Slanted Surfaces in Detail.









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!

Analysis.
Problems with Discontinuities in Detail.

Left image

Right image

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:

- **Multiple Windows**: Split the matching window into multiple parts (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).

Analysis.
Problems with Discontinuities in Detail.

Multiple and shiftable windows are heuristics and only slightly improve the results!

Better solution:
Use a segmentation-prior and incorporate color information into the matching windows.

Idea: **Weighted cost aggregation**.
For every pixel of the matching window, compute a weight, based on color similarity.

$$\sum_{(x,y)\in W} | w(I_R, x_0, y_0, x, y) \cdot (I_R(x,y) - I_L(x+d, y)) |$$

Here, w is a function that computes a weight for pixel (x, y) for the window at position (x_0, y_0):
⇒ Relevant pixels receive high weights – others low weights.

Popular methods:
- Adaptive Support Weights (Yoon and Kweon, CVPR 2005).
- Geodesic Support Weights (Hosni et al., ICIP 2009).

Analysis.
Problems with Discontinuities in Detail.

Left image

Matching Window

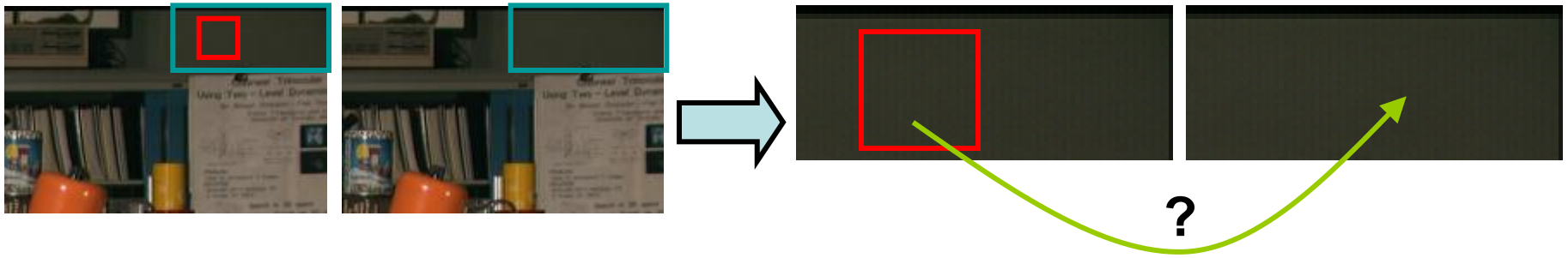
Geodesic Support Weights (Hosni 2009)

Left Camera Image

Optimal Result

WTA with Geodesic Support Weights

Analysis.
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:

- **Variable Windows**: Use larger matching windows adaptively (Veksler. Fast variable windows using integral images. CVPR 2003.).
- Measure the significance of every minimum and invalidate all weak minima.

Practical Considerations.
Summary: Local Methods.

However – despite of the drawbacks of area-based approaches they are often adopted in practice, because they are

- simple,
- fast (Real-Time on most hardware),
- have low memory requirements,
- and are relatively robust.

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

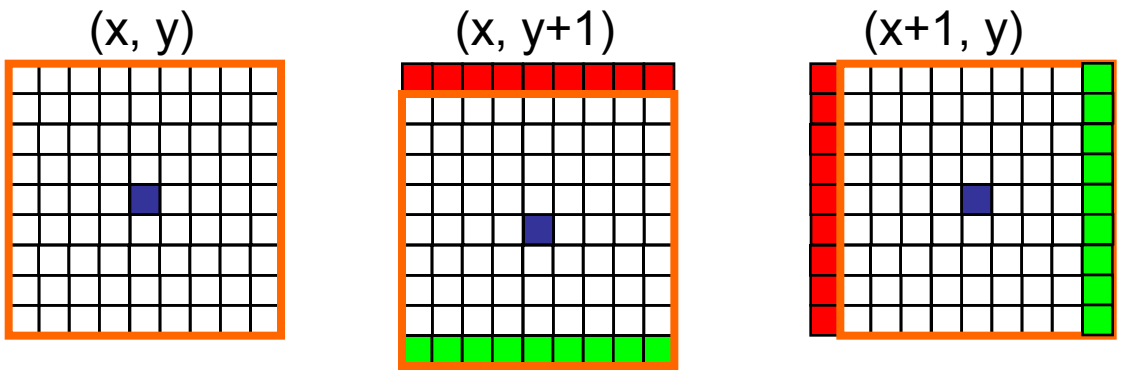
How about execution time?

Naïve implementation: 1400 ms for 320x200 pixels!!

Practical Considerations.
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.



Practical Considerations.
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

Practical Considerations.
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.).

Practical Considerations.
Example.

Global and Non-Global Optimization Methods: Overview.

The area-based approaches presented fall into the category of “local methods”, since the optimization (disparity-computation) is done for every single pixel individually.

The problems with local methods:

- Cost aggregation is required for robustness but introduces errors at depth discontinuities and slanted surfaces.
- Homogeneous areas and repetitive textures are problematic.

To improve in such situations, global optimization methods have been found. There, the stereo problem is formulated as an energy minimization problem:

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

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

Global and Non-Global Optimization Methods: Overview.

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

This is a functional, where D (the disparity map) is a function mapping from image space to the space of disparities.

Data term E_D is a unary term and may, for example, be based on AD.

Smoothness term E_S is a binary term and may be chosen as:

$$E_S(D) = \sum_{(p,q)\in N} V(D(p), D(q)) \qquad \text{with} \quad V(d_p, d_q) = \begin{cases} 0 & d_p = d_q \\ C & otherwise \end{cases}$$

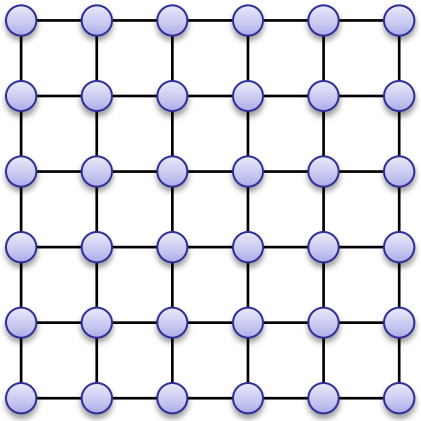
Well known standard methods to solve such problems (not only stereo!) are:

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

Global and Non-Global Optimization Methods: Overview.

Create a graph over the whole image, where every pixel is a node, connected to its four neighbors.

Stereo as a labeling problem: assign a disparity (label) to every pixel (node) which is consistent with the image intensities (data term) and such that the disparity does not vary too much in the local neighborhood (smoothness term).



Dynamic Programming.

In general, such combinatorial problems are NP-hard in computational complexity.

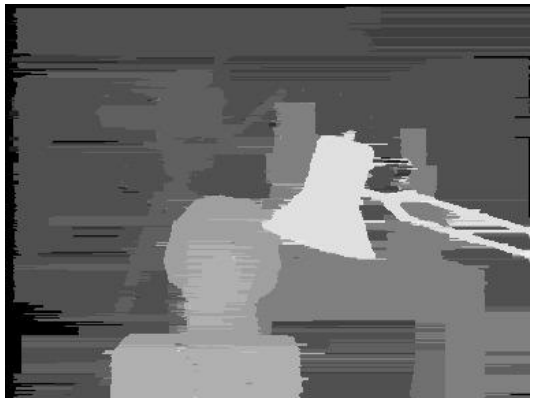
One trick is to reduce the problem to individual scanlines: using dynamic programming the global minimum can be found in polynomial time.

The idea is to construct a cost matrix that relates the left and right image scanlines. Then, the minimal cost path is determined

Left Camera Image

Optimal Result

Dynamic Programming



Global Optimization Methods: Examples.

Adapting Belief Propagation

Graph Cuts

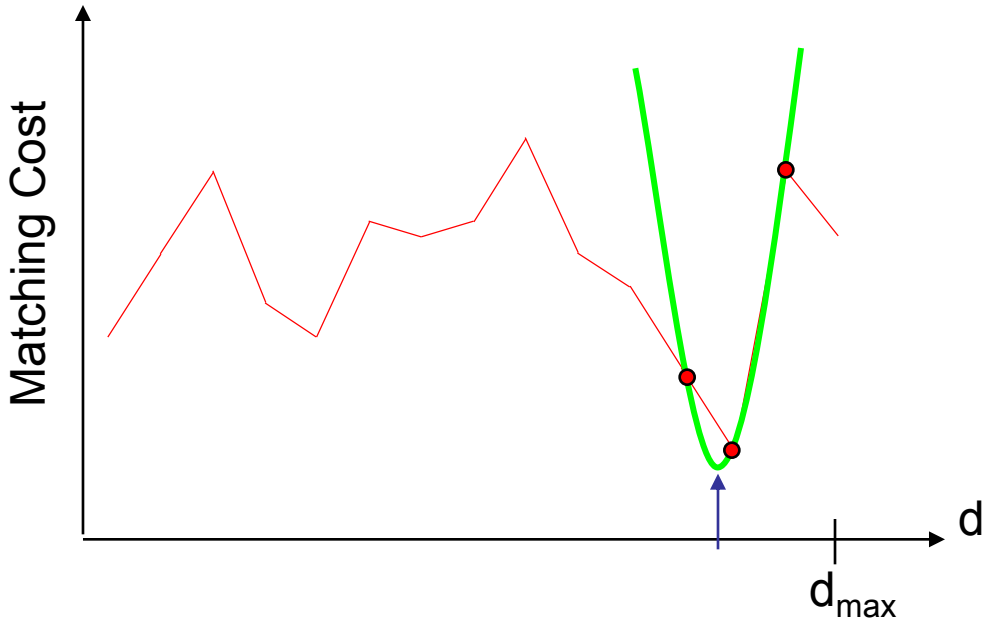
However, this quality is at the cost of execution time (e.g. Graph Cuts: 5 minutes).

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.