VIDEO SENSORS III

Mihai Bărbulescu

BOSCH

Contents
Disparity Estimation & Optical Flow

1. Recap and Problem Statement
2. Overview of Disparity Estimation Algorithms
3. Matching Cost Computation & Cost Aggregation
4. Disparity Selection & Refinement
5. Optical Flow
6. Lucas – Kanade Algorithm

RECAP AND PROBLEM STATEMENT

Recap and Problem Statement

Monocular and stereo cameras

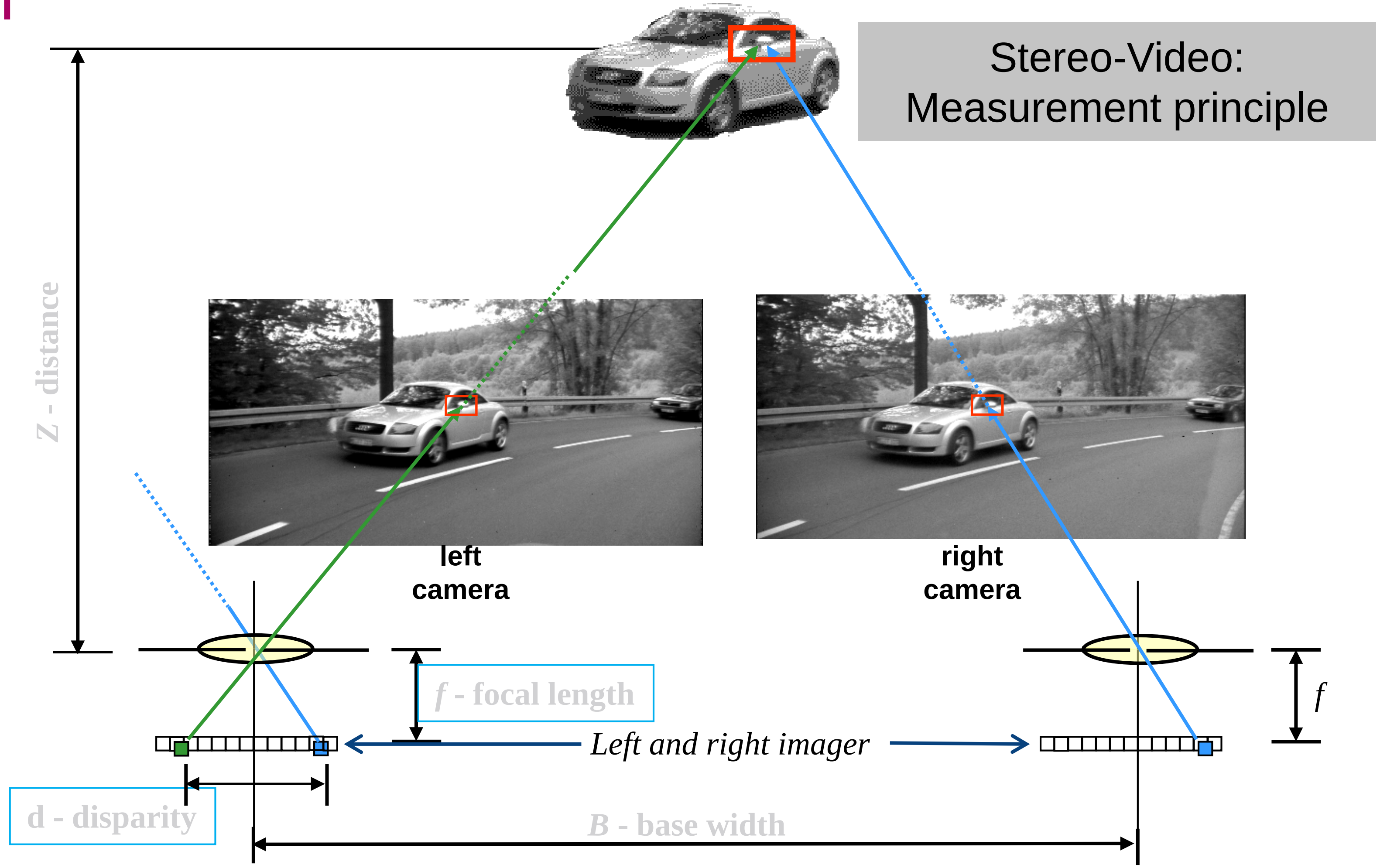
Two types of cameras are typically used in driver assistance applications:

☐ Monocular camera

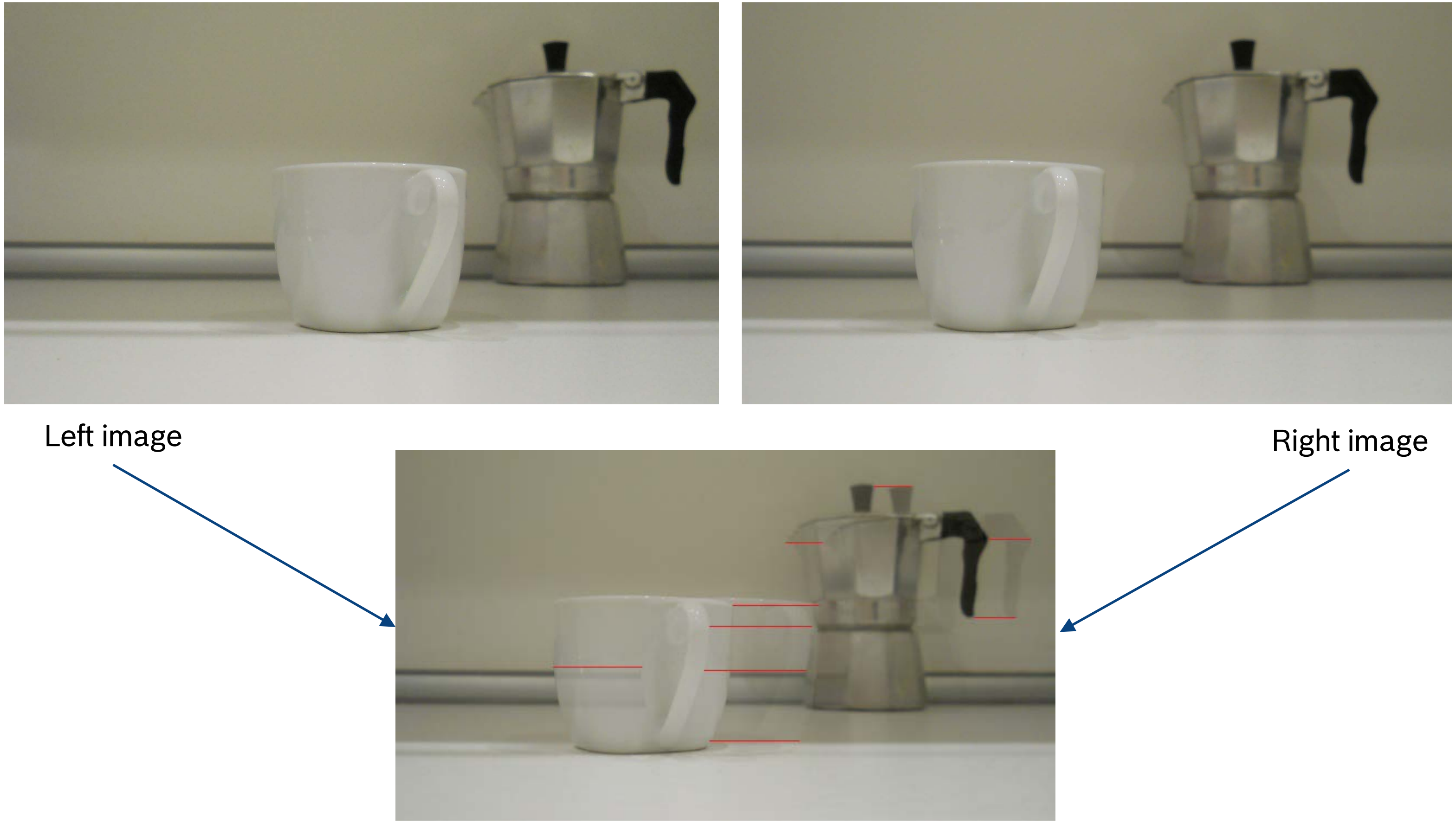
☐ Stereo camera

Recap and Problem Statement
Stereo vision

Stereo-Video:
Measurement principle



Recap and Problem Statement
Disparity



- **Stereo** cameras capture two images of a scene simultaneously.
- The displacement of pixels between the two images is called the **disparity**.
- Each pixel in each image has an associated disparity value.

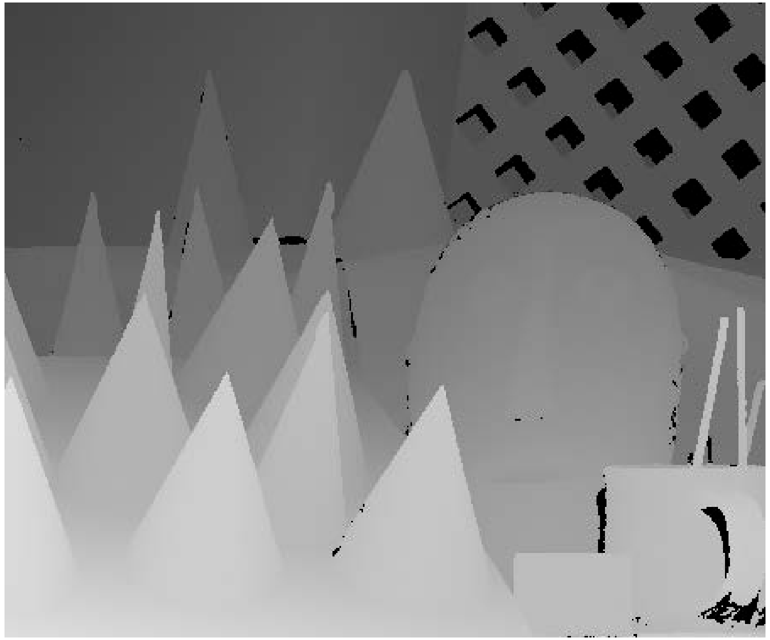
Stereo camera.

Recap and Problem Statement
Disparity map example

- □ A disparity map is a visualization of the **horizontal shift** between the left image and right image.
- □ **Higher intensity** of the disparity map is associated with **greater disparity**.

Left image

Right image



Disparity map

Source: http://vision.middlebury.edu/stereo/data/scenes2003/

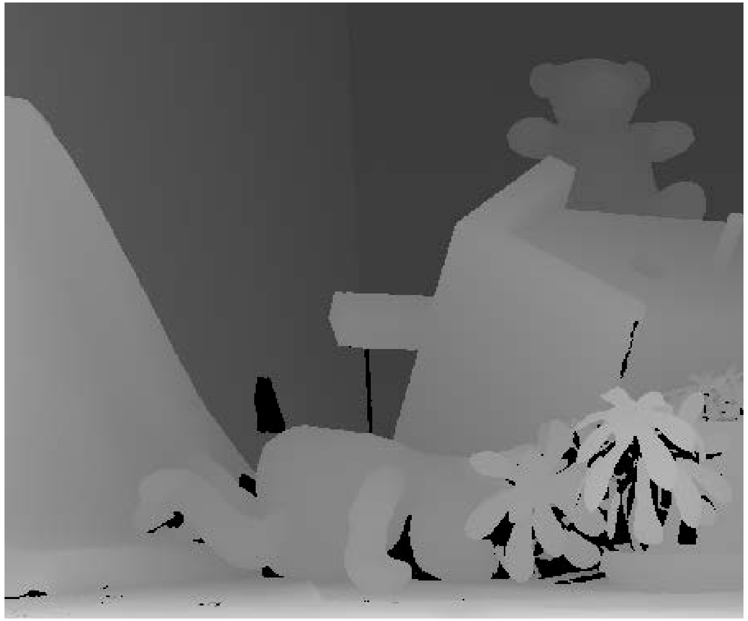
Recap and Problem Statement

Disparity map example

- ❑ A disparity map represents corresponding pixels that are horizontally shifted between the left image and right image.

Left image

Right image

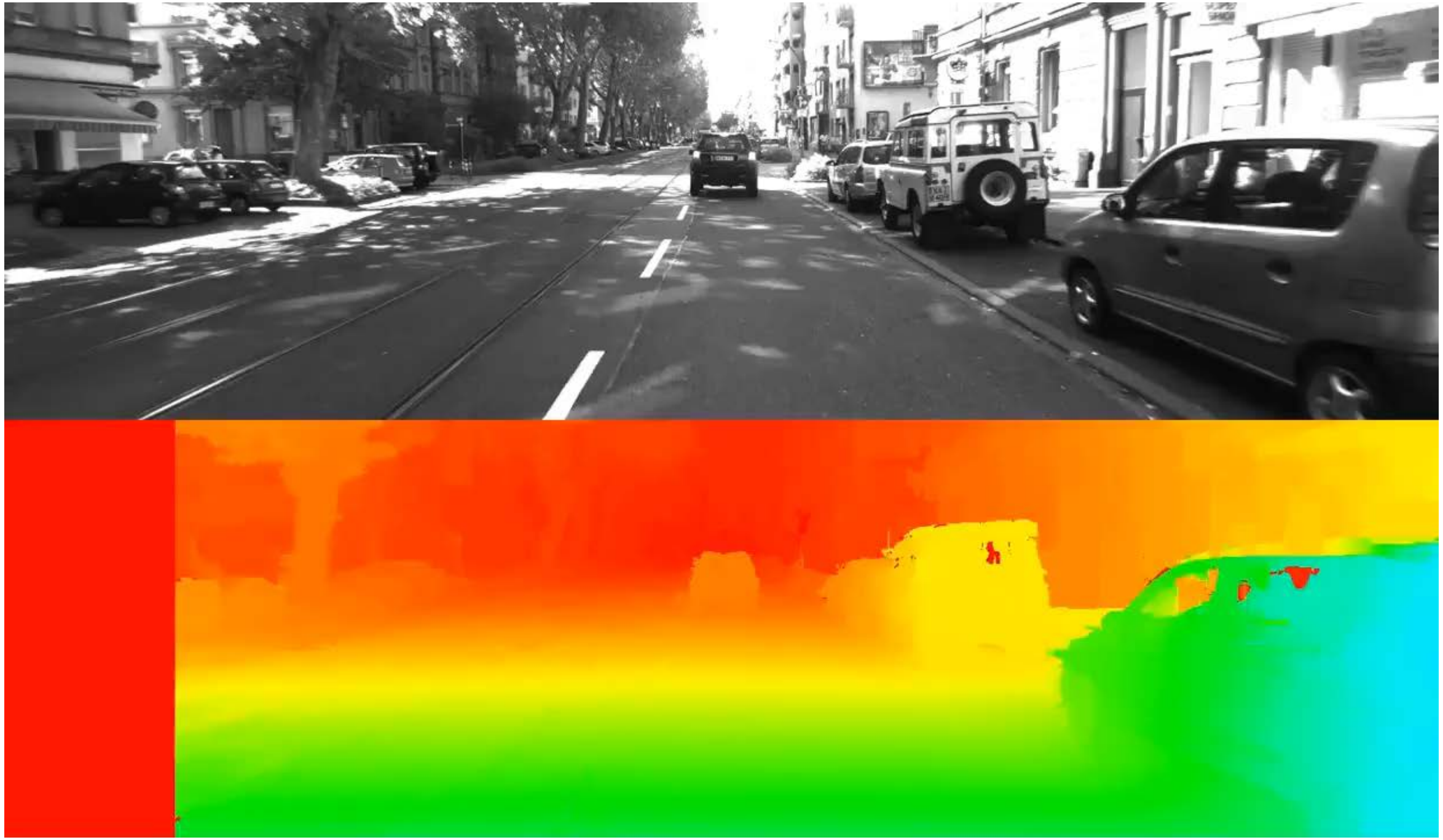


Disparity map

Source: http://vision.middlebury.edu/stereo/data/scenes2003/

Disparity Maps and Rectification

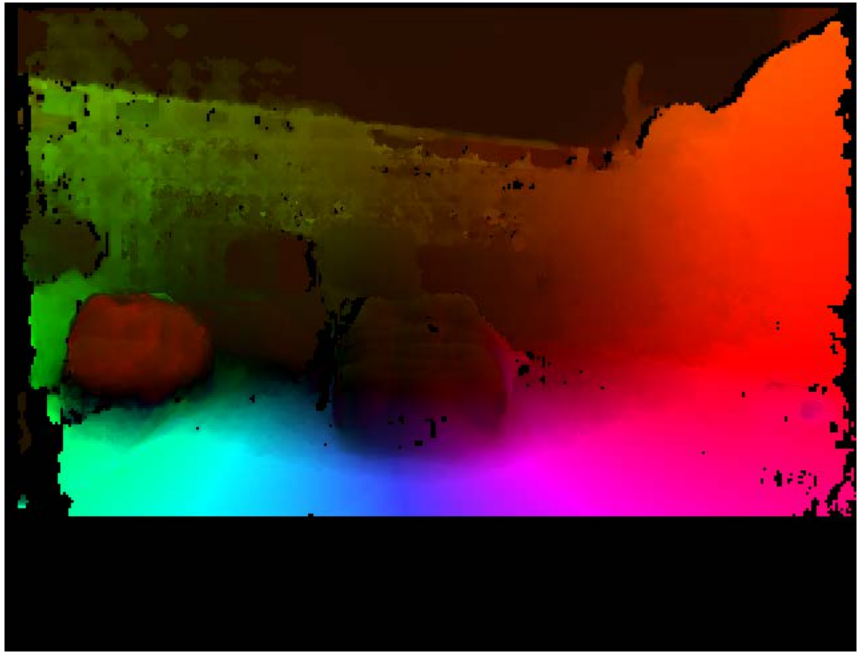
Real time disparity map visualization



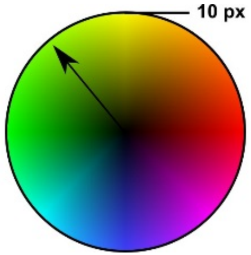
Source: https://www.youtube.com/watch?v=fEaHN8FsW_E

Recap and Problem Statement

Optical flow



Mono camera.



❑ Mono cameras do not provide the information necessary to calculate disparity.

❑ A result analogous to disparity is **optical flow.**

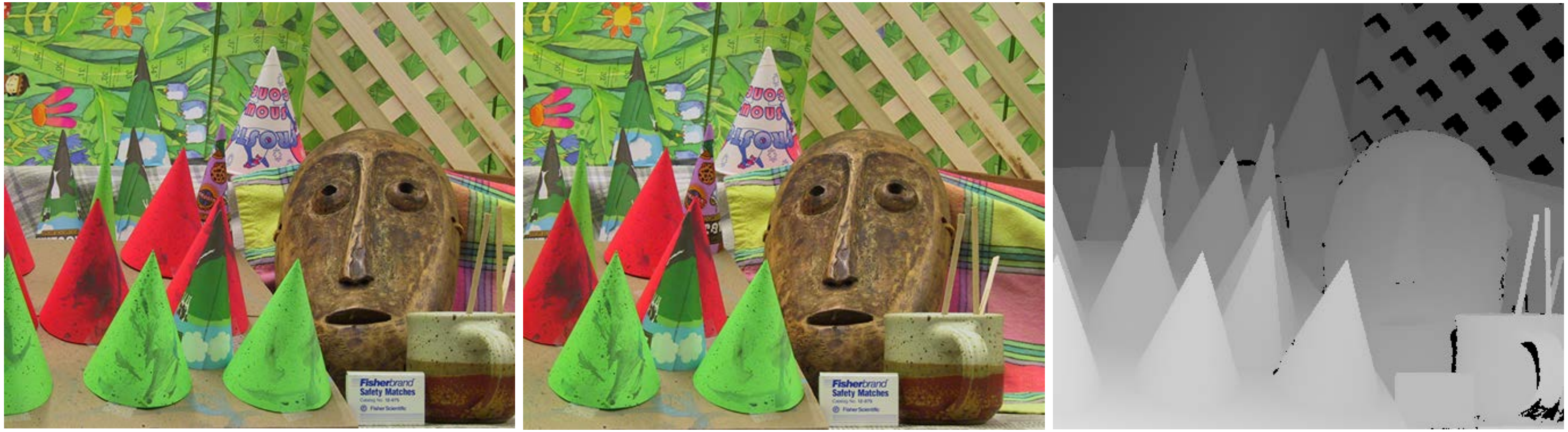
❑ Calculated from pairs of images taken **consecutively**, instead of simultaneously.

❑ Has **direction**, as well as magnitude.

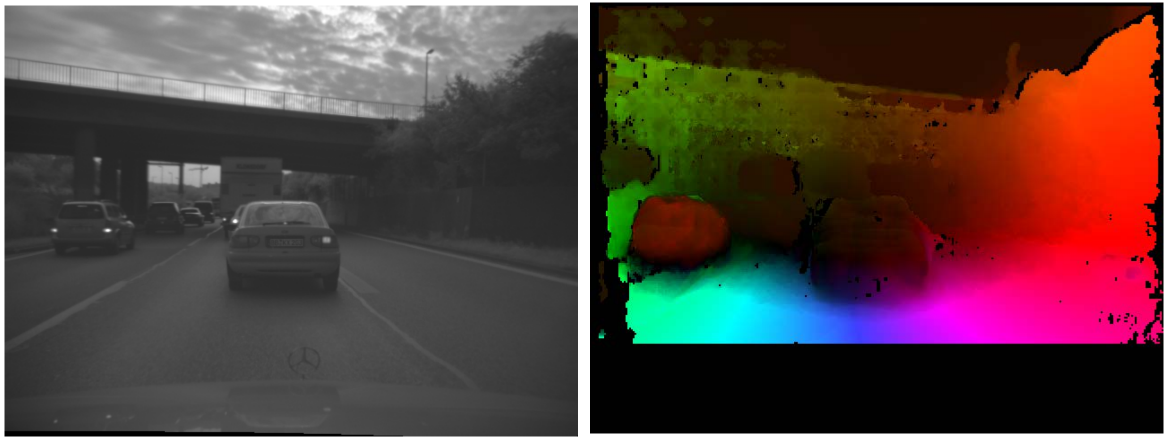
Image source: http://www.6d-vision.com/

Recap and Problem Statement

Comparison to disparity

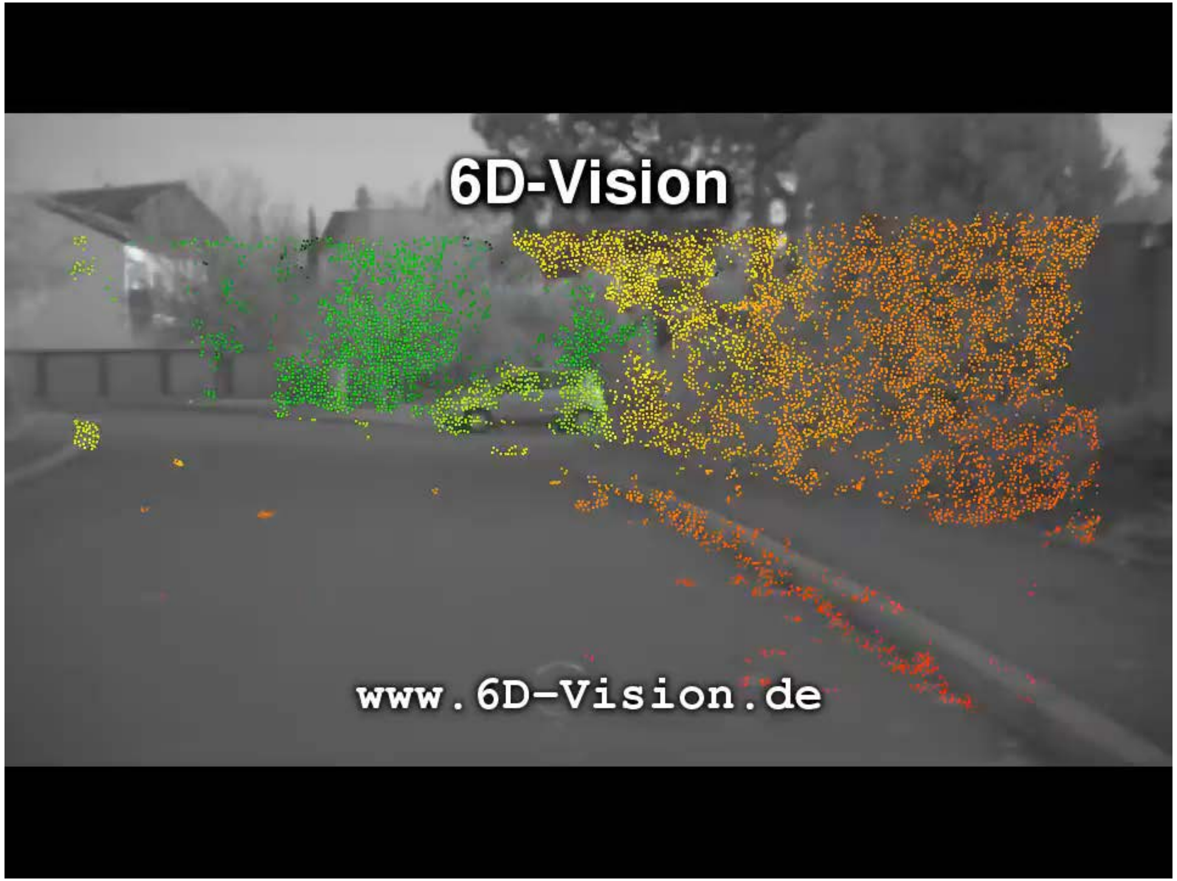


Stereo camera.



Mono camera.

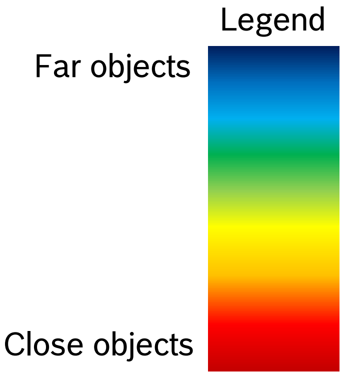
Recap and Problem Statement
Real time optical flow visualization

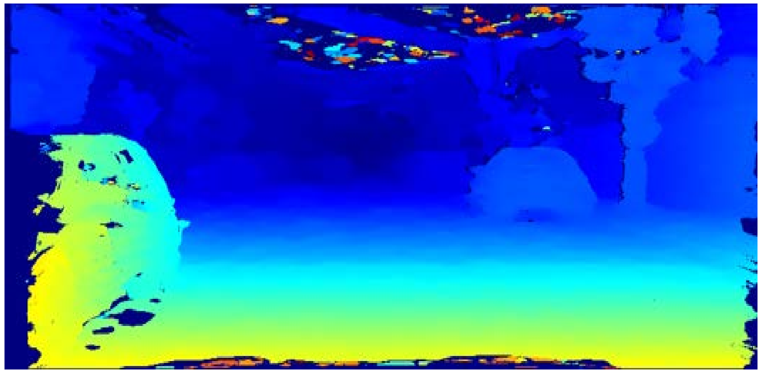


Source: http://www.6d-vision.com/

Recap and Problem Statement

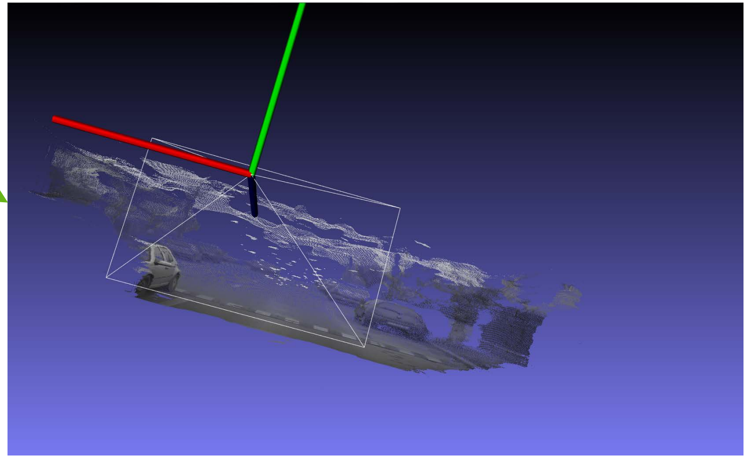
3D reconstruction from disparity





Disparity estimation.

3D reconstruction



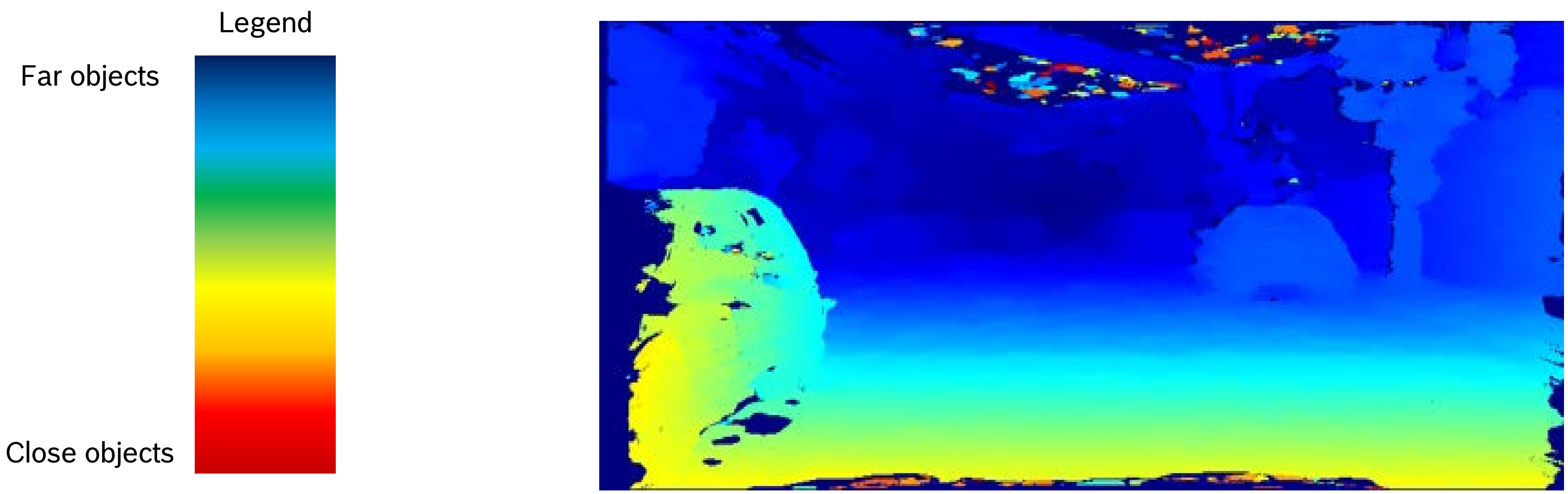
- ❑ Disparity is **small** for objects that are **far** away, and **large** for objects in **close** range.
- ❑ The formula relating the **depth**, z, to the **disparity** is

$$z = \frac{f \cdot b}{d}$$

where f is the focal length, and b is the baseline.

Recap and Problem Statement

Problem statement



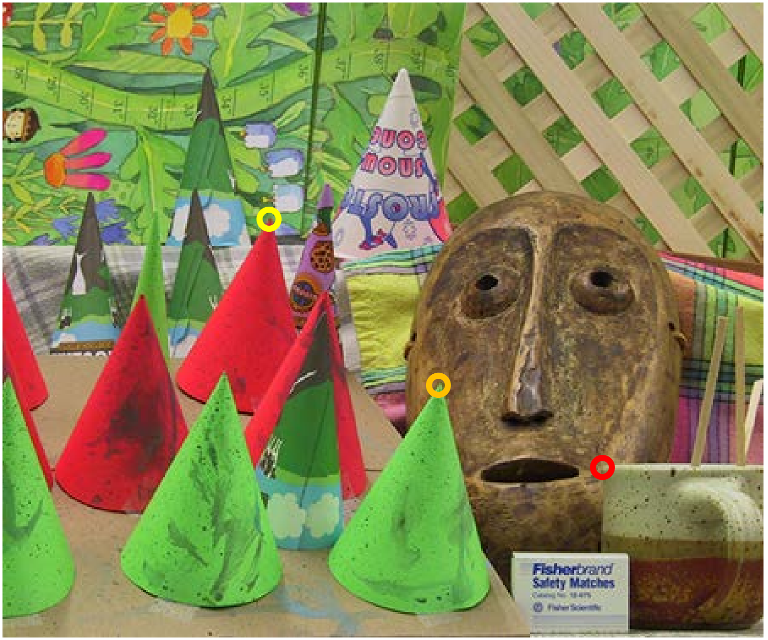
- □ Using a **stereo camera**, the 3D scene may be reconstructed via a **disparity map**.
- □ Using a **monocular camera**, the 3D scene may be reconstructed via the **optical flow**.

OVERVIEW OF DISPARITY ESTIMATION ALGORITHMS

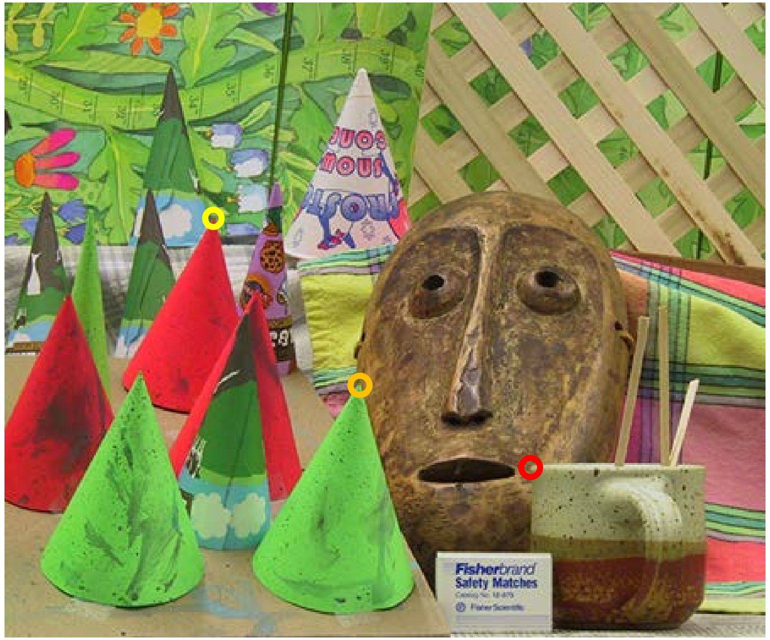
Overview of Disparity Estimation Algorithms
Problem statement

□ How can we tell that the marked points in the stereo pair represent the same point in a scene?

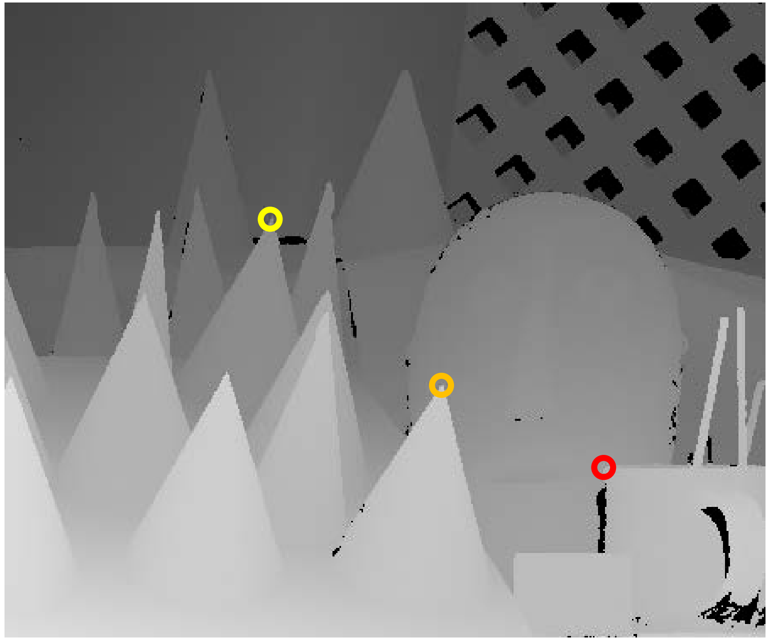
□ How can we obtain an accurate disparity?



Left image



Right image

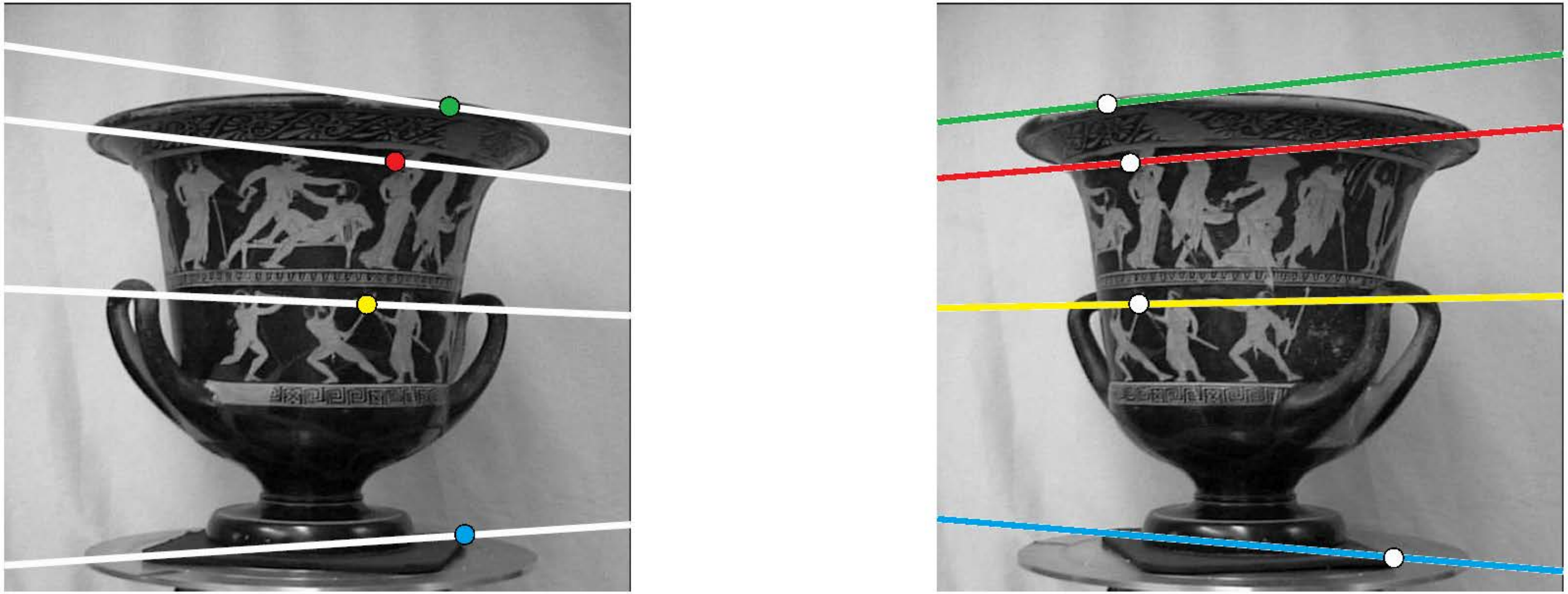


Disparity map

Source: http://vision.middlebury.edu/stereo/data/scenes2003/

Matching Cost Computation & Cost Aggregation

The fundamental matrix

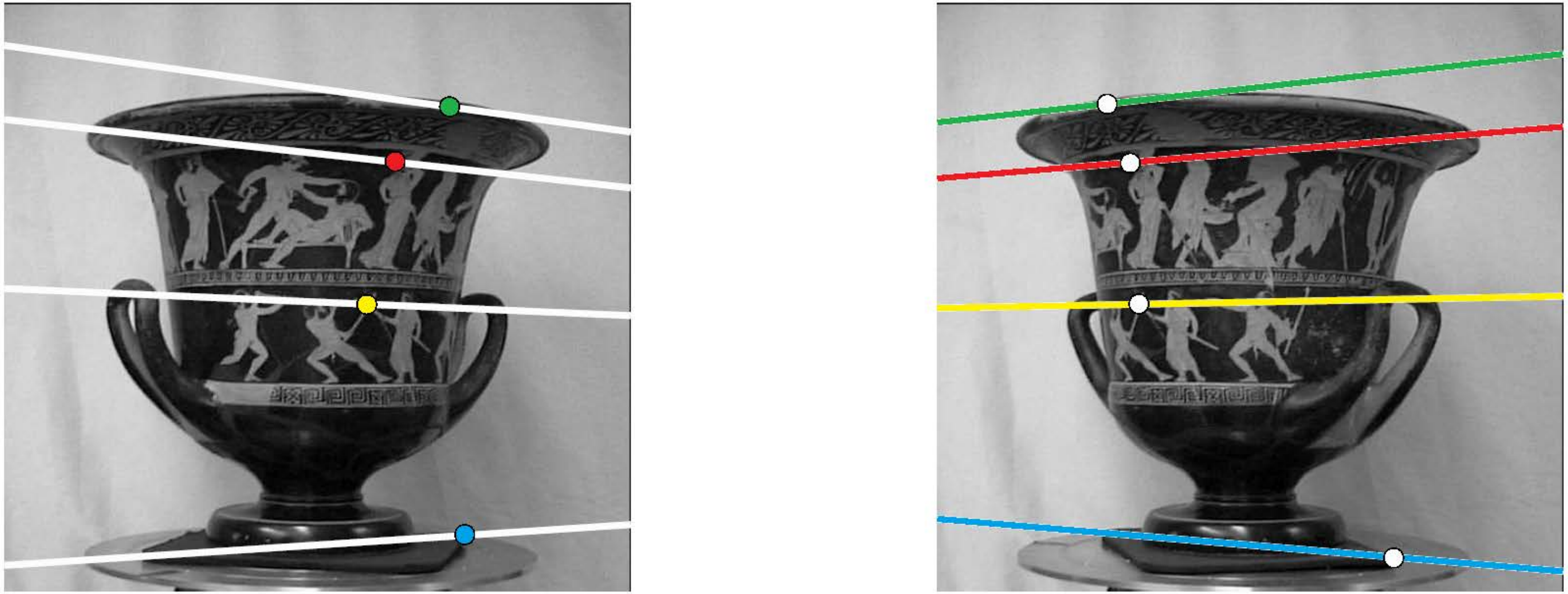


Source:https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf

- The fundamental matrix relates corresponding points in stereo images.
- Using the fundamental matrix, and a point in one of the images, one may obtain an associated line in the other image called the **epipolar line**.

Matching Cost Computation & Cost Aggregation

Epipolar lines

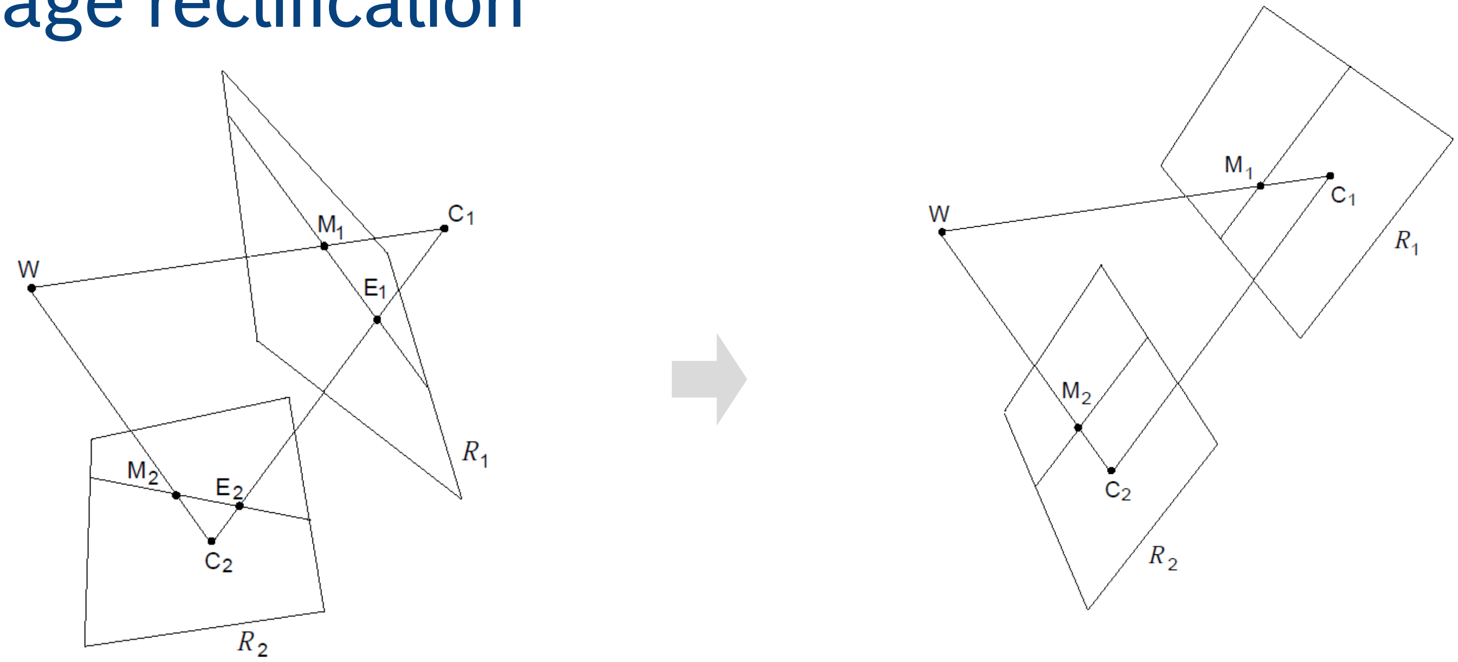


Source:https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf

❑ Each point in the right image that is associated to some point in the left image will lie on its **epipolar line**.

❑ This property is called the **epipolar constraint**.

Matching Cost Computation & Cost Aggregation
Stereo image rectification

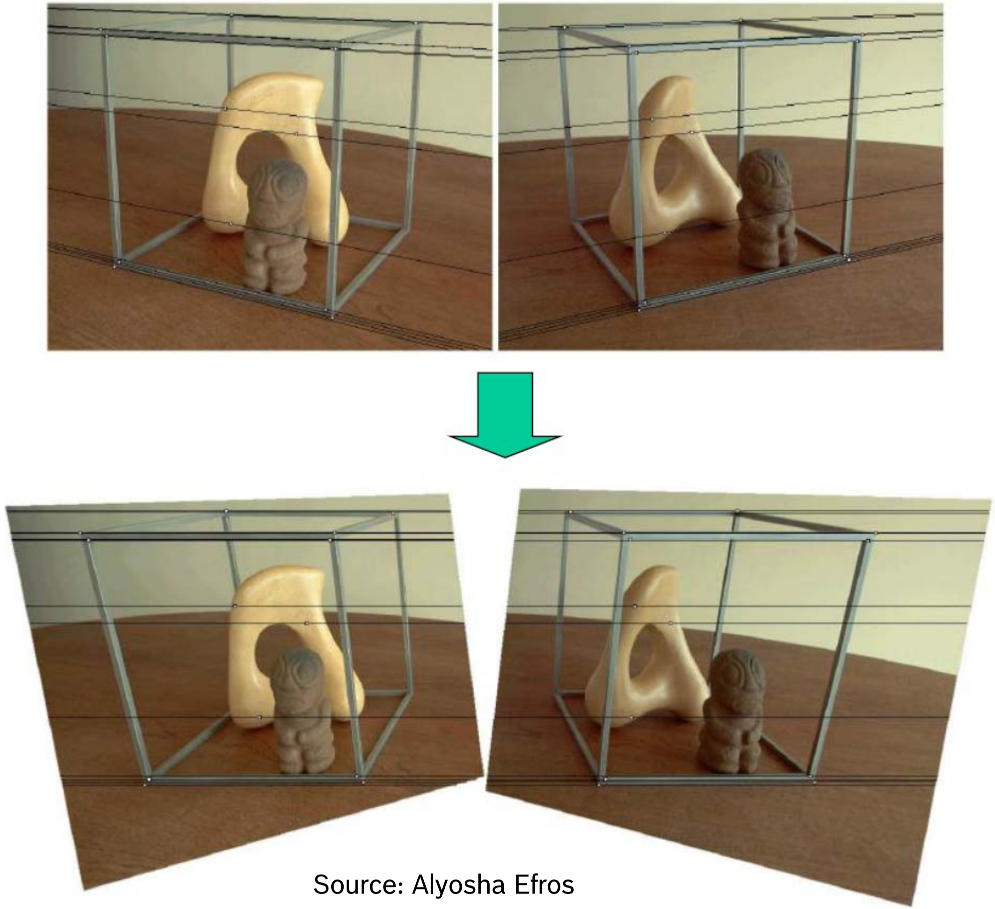


Source: Fusiello, A. et al. (2000)

❑ Stereo image rectification is the process of projecting pairs of images onto the **same image plane**.

❑ Epipolar lines become **parallel**.

Matching Cost Computation & Cost Aggregation
Rectification



Source: Alyosha Efros

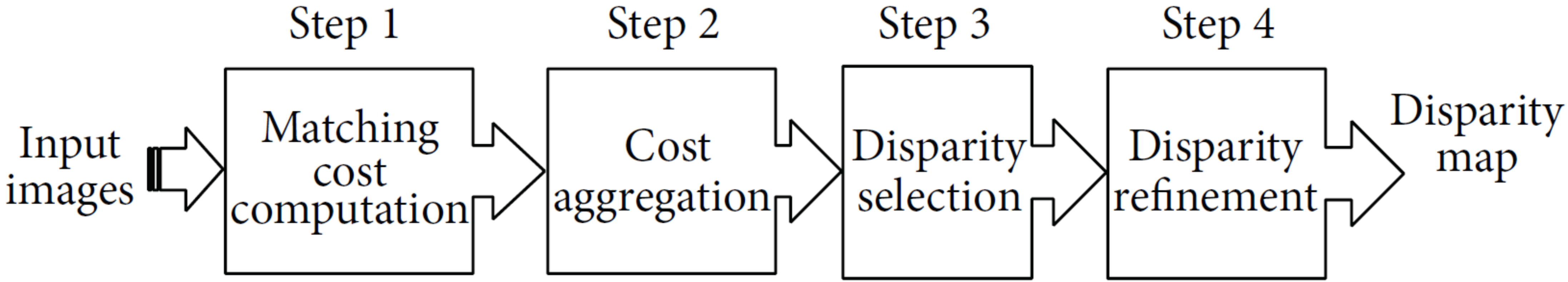
□ In rectified images, all epipolar lines are **parallel to the horizontal axis**.

□ Corresponding points have the **same vertical coordinates**.

□ Rectification simplifies the task of **matching** points in stereo image pairs and obtaining disparity maps.

Overview of Disparity Estimation Algorithms
Main steps

- ❑ There are **four** main steps in computing disparity maps.
- ❑ Ideally, the **output** should be a smooth **disparity map**.
- ❑ In steps 1 and 2, points in the stereo image pair are **matched**.
- ❑ In steps 3 and 4, the **disparity** between the points is selected and refined.



Source: Hamzah, R.A. & Ibrahim H. (2015)

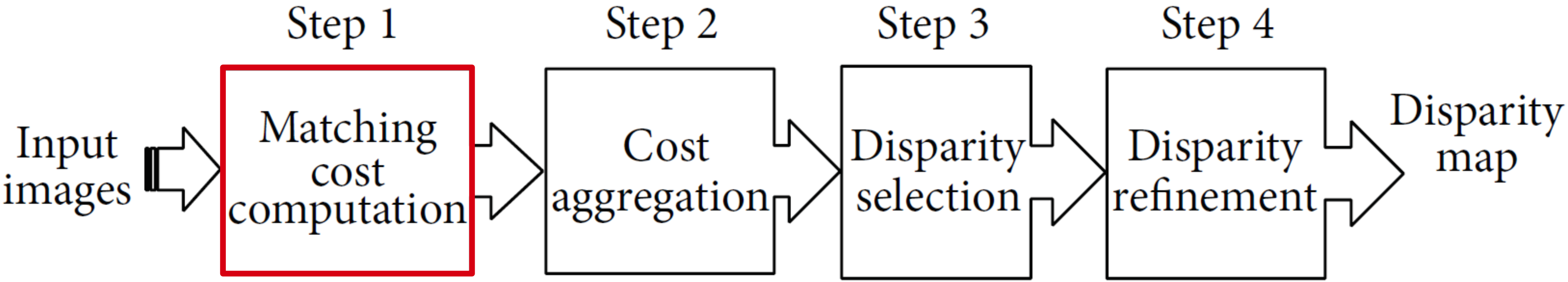
Overview of Disparity Estimation Algorithms

Step 1: Matching cost computation

❑ Requires a **cost criterion** to measure the extent of matching between two pixels.

❑ This is the stage where it is determined whether the values of two pixels correspond to the **same point in a scene**.

❑ The value is computed at each pixel.



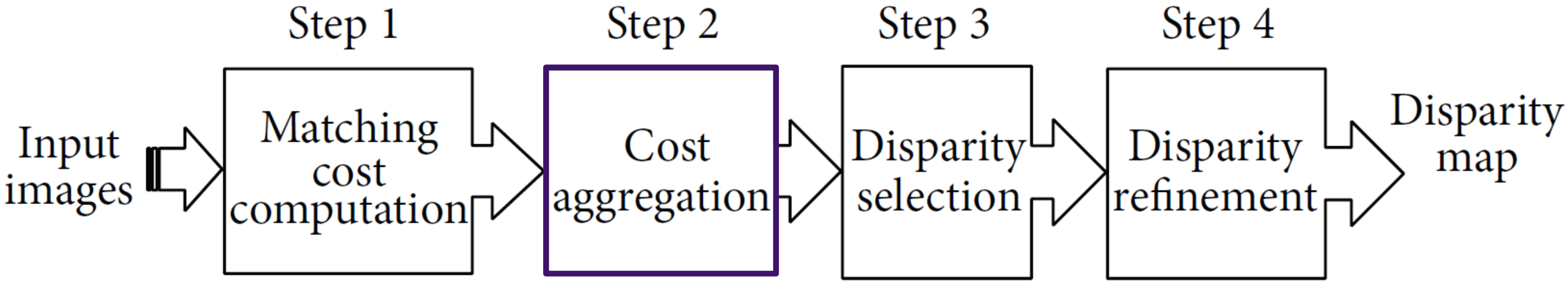
Source: Hamzah, R.A. & Ibrahim H. (2015)

Overview of Disparity Estimation Algorithms
Step 2: Matching cost aggregation

- ❑ Is done locally using a window over **multiple pixels**.

- ❑ The purpose of cost aggregation is to **minimize matching uncertainties**.

- ❑ This step is necessary since knowing the matching cost for a single pixel is not sufficient for precise matching.



Source: Hamzah, R.A. & Ibrahim H. (2015)

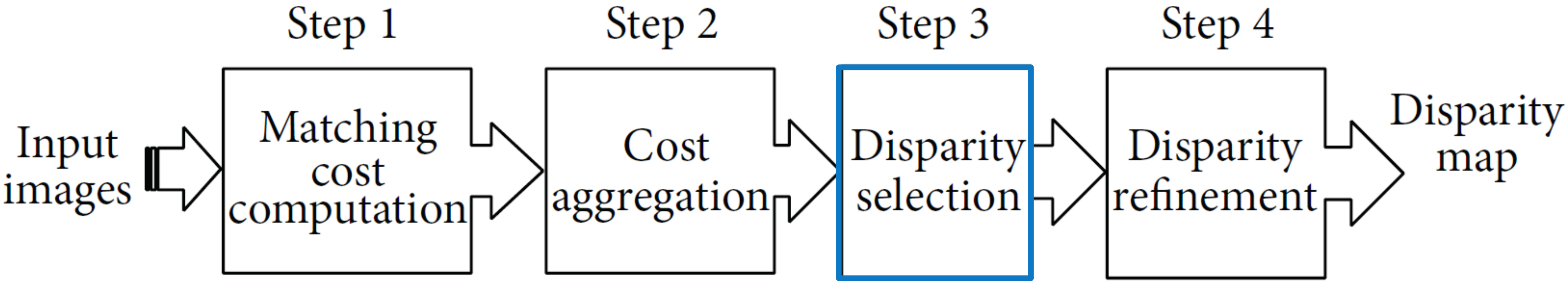
Overview of Disparity Estimation Algorithms

Step 3: Disparity selection

- ❑ In local stereo matching algorithms, after the disparities are calculated, the disparity for each pixel is selected using a local **winner-take-all** (WTA) strategy:

$$d_p = \underset{d \in D}{\operatorname{argmin}}\, C(p, d)$$

- ❑ Here, p and d represent the pixel and disparity respectively, C(p, d) is the aggregate cost, and D denotes the set of allowed disparities.
- ❑ The result, d_p, is the **minimum aggregated cost** at each pixel.

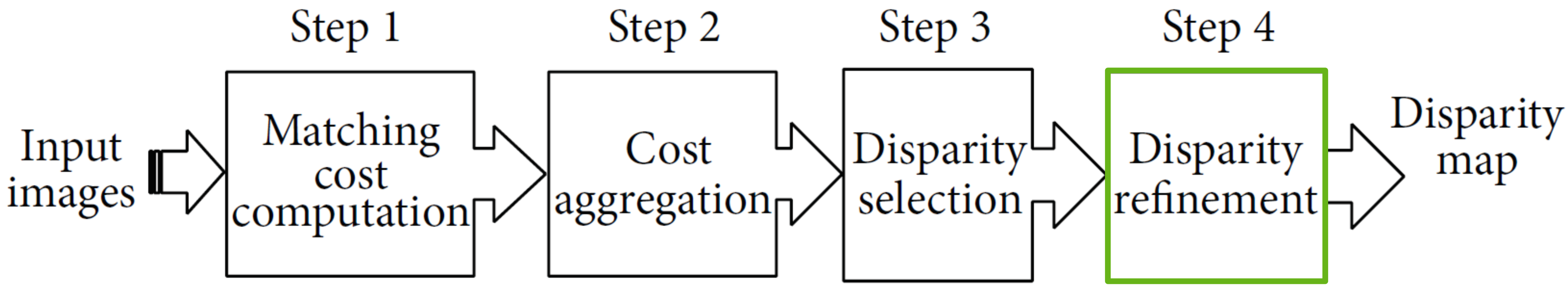


Source: Hamzah, R.A. & Ibrahim H. (2015)

Overview of Disparity Estimation Algorithms
Step 4: Disparity refinement

- □ The purpose of this step is to improve the quality of disparity maps.

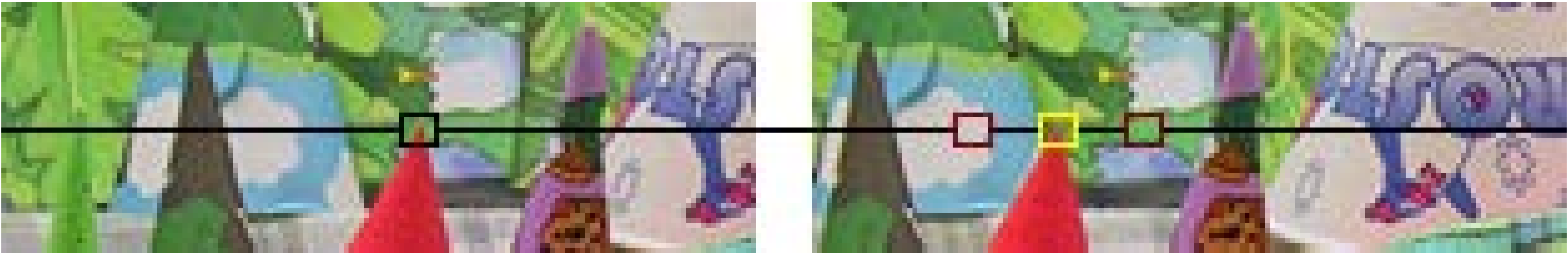
- □ This is done by **reducing noise** by filtering inconsistent pixels, and filling in gaps through **interpolation**.



Source: Hamzah, R.A. & Ibrahim H. (2015)

MATCHING COST COMPUTATION & COST AGGREGATION

Matching Cost Computation & Cost Aggregation
Block Matching



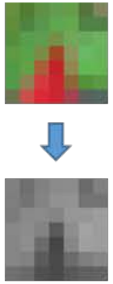
- ❑ For each pixel that we want to match, we choose a small **region of pixels** in one image and search for the closest matching region in the other.
- ❑ The search is constrained to the **epipolar line** of the point.
- ❑ In order to determine which is the closest matching region we employ a comparison metric, a.k.a a **cost function**.

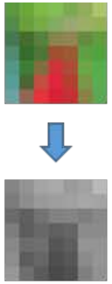
Matching Cost Computation & Cost Aggregation

Cost function: sum of absolute differences

- ❑ The simplest cost function we may use is the **sum of absolute differences** (SAD).
- ❑ This cost function computes the absolute differences between the pixel intensities in two associated blocks and sums up the results:

$$SAD(x,y,d) = \sum_{(x,y)\in w} |I_l(x,y) - I_r(x-d,y)|$$





$$\text{abs}\left(\begin{matrix} 103 & 86 & 104 & 111 & 112 & 88 & 118 \\ 99 & 92 & 105 & 113 & 116 & 95 & 94 \\ 96 & 108 & 111 & 111 & 123 & 123 & 128 \\ 105 & 108 & 111 & 91 & 128 & 130 & 125 \\ 103 & 111 & 108 & 91 & 122 & 123 & 122 \\ 103 & 119 & 111 & 96 & 101 & 95 & 105 \\ 101 & 119 & 105 & 97 & 104 & 94 & 95 \end{matrix} - \begin{matrix} 104 & 111 & 129 & 136 & 126 & 138 & 159 \\ 99 & 109 & 120 & 134 & 146 & 144 & 137 \\ 92 & 75 & 93 & 115 & 121 & 104 & 106 \\ 112 & 70 & 85 & 97 & 100 & 108 & 103 \\ 123 & 82 & 102 & 106 & 101 & 116 & 112 \\ 131 & 78 & 113 & 108 & 97 & 114 & 115 \\ 145 & 112 & 113 & 102 & 100 & 113 & 115 \end{matrix}\right) = \begin{matrix} 1 & 25 & 25 & 25 & 14 & 50 & 41 \\ 0 & 17 & 15 & 21 & 30 & 49 & 43 \\ 4 & 33 & 18 & 4 & 2 & 19 & 22 \\ 7 & 38 & 26 & 6 & 28 & 22 & 22 \\ 20 & 29 & 6 & 15 & 21 & 7 & 10 \\ 28 & 41 & 2 & 12 & 4 & 19 & 10 \\ 44 & 7 & 8 & 5 & 4 & 19 & 20 \end{matrix}$$

Source: http://vision.middlebury.edu/stereo/data/scenes2003/

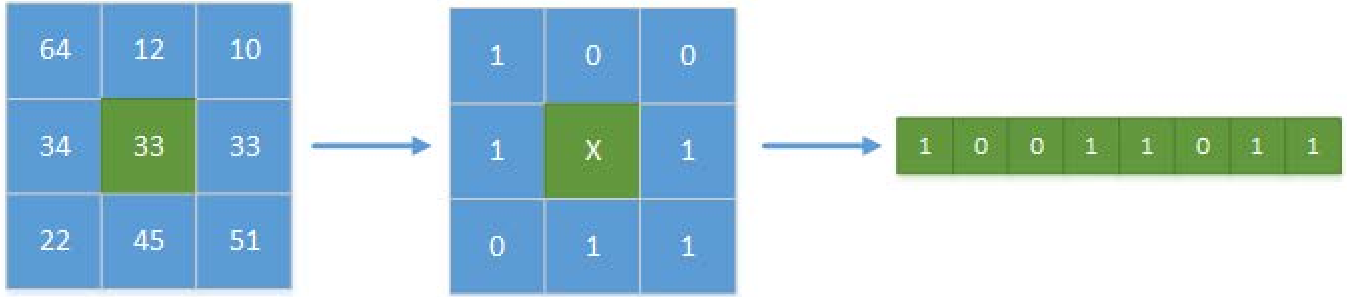
SAD = 938

Matching Cost Computation & Cost Aggregation

Cost function: census transform

- The **census transform** (CT) algorithm creates a bit string by comparing the value of a center pixel to all 8 adjacent pixels.
- The **hamming distance** is calculated between the reference image (ref) and the candidate target image (tar).

$$Bit(i,j) = \begin{cases} 0, & I(i,j) < I(x,y) \\ 1, & I(i,j) \geq I(x,y) \end{cases}$$



64	12	10
34	33	33
22	45	51

1	0	0
1	X	1
0	1	1

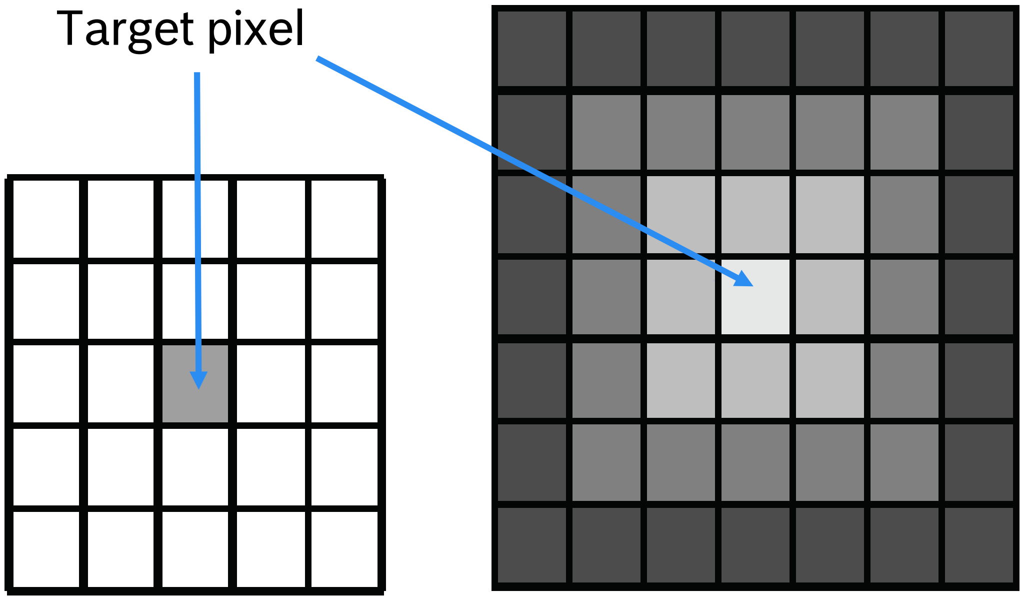
1	0	0	1	1	0	1	1

$$Census(x,y) = Bitstring_{(i,j)\in w}(Bit(i,j))$$

$$CT(x,y,d) = \sum\nolimits_{(x,y)\in w} Hamming\,(Census_{ref}(x,y) - Census_{tar}(x-d,y))$$

Matching Cost Computation & Cost Aggregation

Examples of cost aggregation windows



(a)

(b)

a) A **fixed** 5x5 square window. The shape of fixed windows may rectangular as well as square. Common sizes include 5x5, 7x7, and 7x11. Typically, larger window sizes correspond to more accurate matching for the target pixel, in exchange for greater blur at object boundaries.

b) Window with **adaptive support weights** – each pixel is assigned a support weight based on its intensity dissimilarity and distance from the center.

DISPARITY SELECTION & REFINEMENT

Disparity Selection & Refinement
Disparity Selection

- ❑ In local stereo matching algorithms, after the disparities are calculated, the disparity for each pixel is selected using a local **winner-take-all** (WTA) strategy:

$$d_p = \underset{d \in D}{\operatorname{argmin}}\, C(p, d)$$

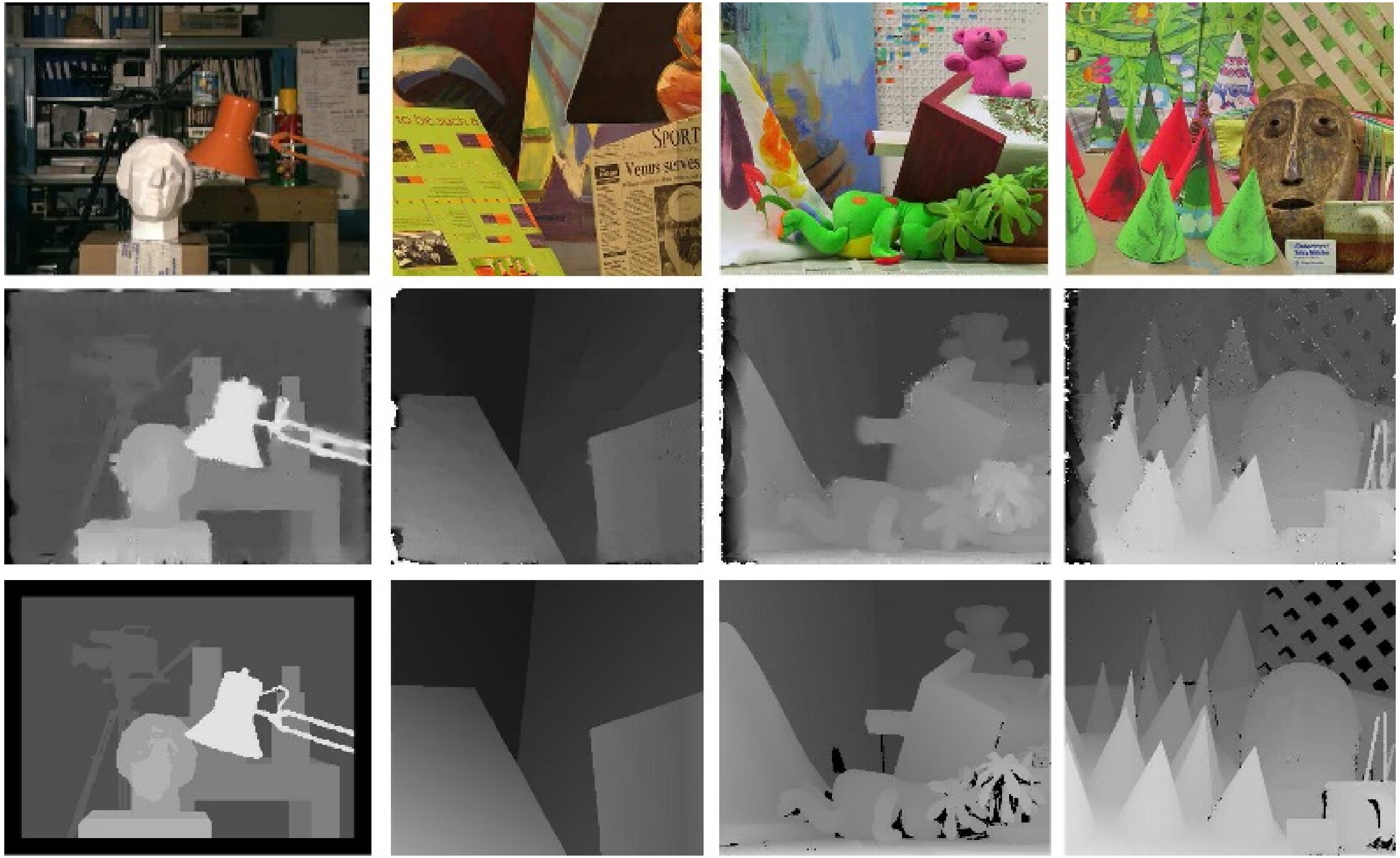
- ❑ Disparity maps obtained initially may contain **errors** and **unmatched pixels**, depending on the method used.

- ❑ The accuracy depends on the matching cost computation and cost aggregation.

- ❑ The accuracy is sensitive to **noise** and unclear regions.

Disparity Selection & Refinement

Disparity maps



Left stereo image.

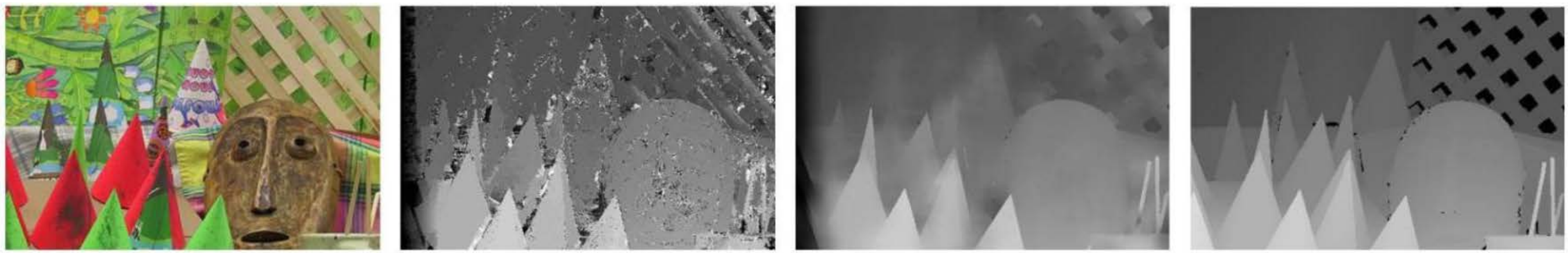
Computed disparity.

Ground truth.

Source: Banno, A. & Ikeuchi, K. (2011)

Disparity Selection & Refinement

Disparity refinement



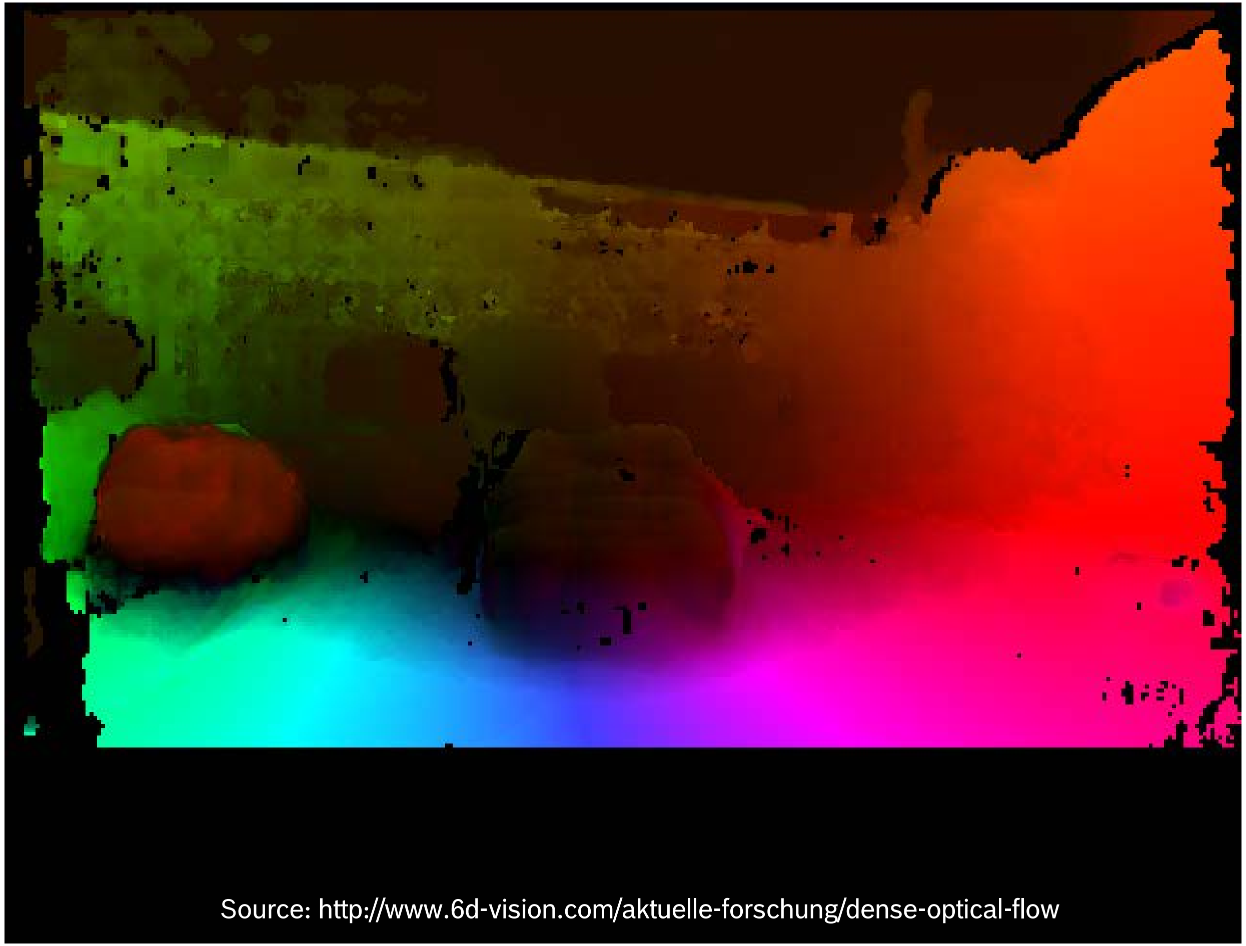
Left stereo image. Initial disparity map. Refined disparity map. Ground truth.

Source: Pei, R. et al. (2016)

- Disparity maps are refined using a variety of methods, including **regularization** and **interpolation**.
- **Regularization** (also called denoising) reduces the overall noise by filtering inconsistent pixels.
- The **interpolation** process is responsible for filling in gaps by **approximating the disparity** in areas where it is unclear or absent.
- The disparity is calculated based on values in its neighborhood.
- Common techniques for disparity refinement include Gaussian convolution and median filtering.

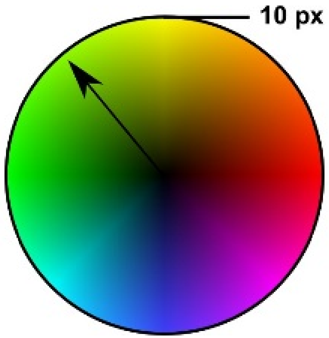
OPTICAL FLOW

Optical Flow
Definition



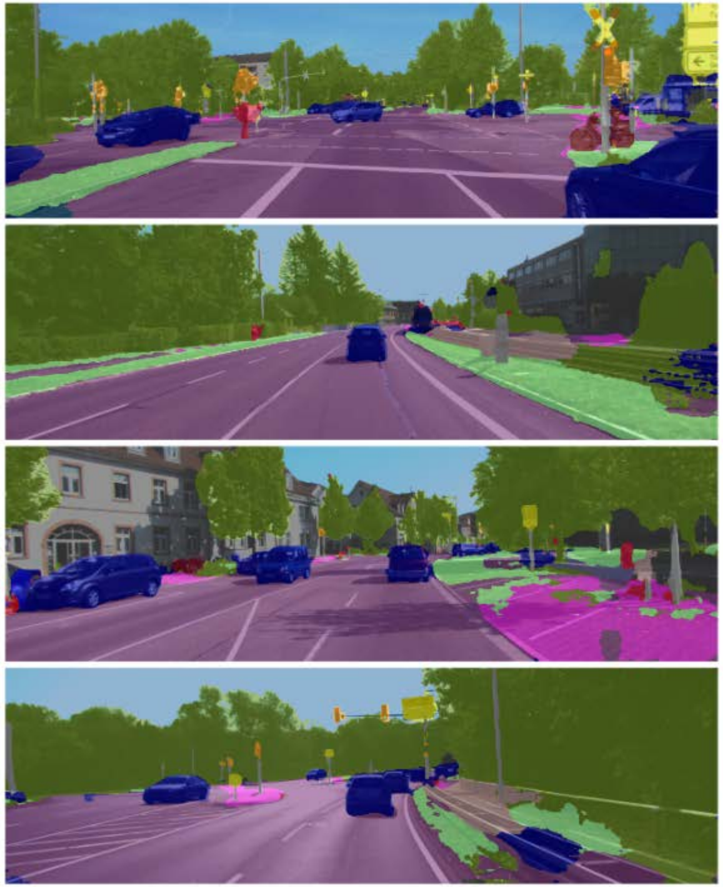
Source: http://www.6d-vision.com/aktuelle-forschung/dense-optical-flow

- ❑ **Optical flow** is the distribution of apparent **velocities** of features in **successive images** in a scene.
- ❑ The color in the image represents the **direction** of the flow vector at that pixel.
- ❑ The brightness represents the **magnitude**.

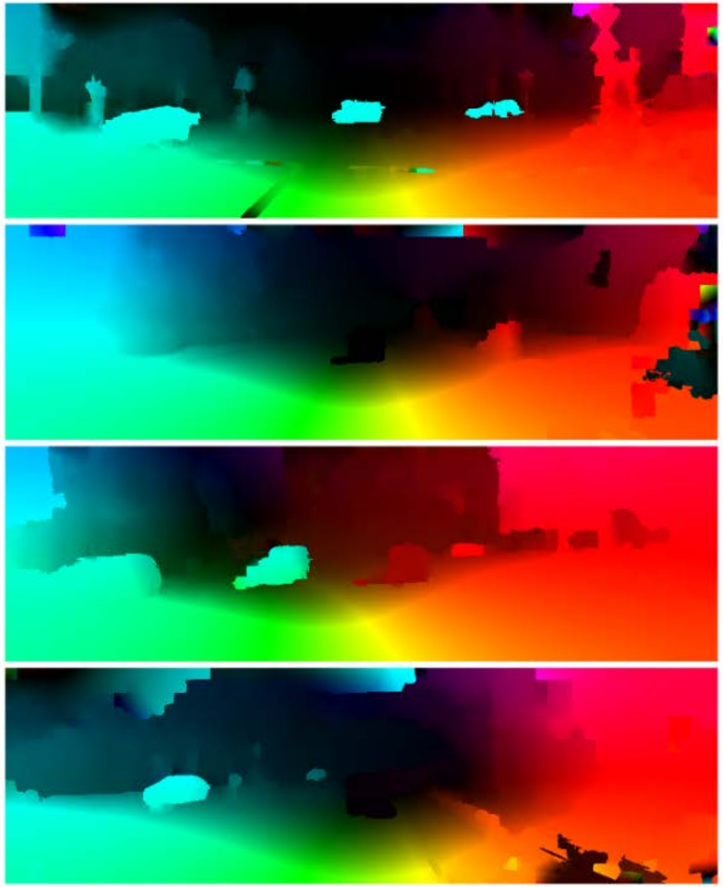


Mono camera.

Optical Flow
Further examples



Segmented images.



Overlaid optical flow.

LUCAS – KANADE ALGORITHM

The Lucas-Kanade Algorithm
Reminder: derivatives and linear approximation

- ❑ Reminder: the definition of the **derivative** in 1D is

$$f'(x) = \lim_{dx \to 0} \frac{f(x+dx) - f(x)}{dx} \tag{1}$$

- ❑ A concept which is closely related to the notion of derivative is that of the **linear approximation.**
- ❑ If dx is **small** ($dx \ll 1$), but still **finite**, the equation has the following approximate form

$$f(x+dx) \approx f(x) + f'(x)dx \tag{2}$$

- ❑ When saying that $dx \ll 1$, it should be understood that dx is typically not greater than 1/10. This condition may vary depending on the application and should be taken as a rule of thumb.
- ❑ The **gradient operator,** ∇**,** is the generalization of the derivative to multiple dimensions.

The Lucas-Kanade Algorithm
Reminder: the gradient operator and linear approximation

❑ The **gradient operator, ∇,** is the generalization of the derivative to multiple dimensions.

❑ In two dimensions, it takes the form $\nabla f(\boldsymbol{x}) = \left(\frac{df(\boldsymbol{x})}{dx}, \frac{df(\boldsymbol{x})}{dx}\right)$, where $\boldsymbol{x} = (x, y)$.

❑ The definition of the gradient operator is

$$\lim_{dx\to 0} \frac{|f(\boldsymbol{x} + \boldsymbol{dx}) - f(\boldsymbol{x}) - \nabla f(\boldsymbol{x}) \cdot \boldsymbol{dx}|}{|\boldsymbol{dx}|} = 0 \qquad (3)$$

❑ Equation (3) is analogous to Equation (1), but in multiple dimensions. This is more readily seen if

Equation (1) is written as $\lim_{dx\to 0} \frac{f(x+dx)-f(x)-f'(x)dx}{dx} = 0$.

❑ The **linear approximation** in multiple dimensions, analogous to Equation (2), may be written as

$$f(\boldsymbol{x} + \boldsymbol{dx}) \approx f(\boldsymbol{x}) + \nabla f(\boldsymbol{x}) \cdot \boldsymbol{dx} \qquad (4)$$

The Lucas-Kanade Algorithm
Motivation

❑ We wish to associate a **velocity vector,** $\boldsymbol{v} = (u, v)$, to each pixel corresponding to some object in a scene, using two consecutive images.

❑ The Lucas-Kanade method takes two images as input and outputs a local **flow vector**.

❑ The Lucas-Kanade algorithm makes two basic **assumptions**:

I. The pair of images used are separated by a small time increment, dt, such that pixel intensities don’t rapidly change between consecutive frames.

II. A neighborhood of pixels moves together smoothly, with the same velocity.

❑ The algorithm works on **grayscale** images.

The Lucas-Kanade Algorithm
Derivation I

- By the first assumption, if a point $\boldsymbol{x} = (x, y)$ moves at time t, to the point $\boldsymbol{x} + d\boldsymbol{x}$ at time $t + dt$, the **intensities** of the two points are the same. Let the intensity be I, such that:

$$I(\boldsymbol{x} + d\boldsymbol{x}, t + dt) = I(\boldsymbol{x}, t) \tag{5}$$

- If $\boldsymbol{v}$ is the **optical flow** vector, we may write $d\boldsymbol{x} = \mathbf{v}\mathrm{dt}$, such that:

$$I(\boldsymbol{x} + \boldsymbol{v}dt, t + dt) = I(\boldsymbol{x}, t) \tag{6}$$

- Again, by the first assumption, dt is a **small quantity**. For a camera recording at 30 FPS, $dt \approx 0.03$.
- Using the **linear approximation** of $I(\boldsymbol{x} + \boldsymbol{v}dt, t + dt)$, from Equations (2) and (4), we obtain

$$I(\boldsymbol{x}, t) + \nabla I(\boldsymbol{x}, t) \cdot \boldsymbol{v}dt + I_t(\boldsymbol{x}, t)dt = I(\boldsymbol{x}, t) \tag{7}$$

The Lucas-Kanade Algorithm
Derivation II

- ❑ Equation (7), i.e. the equation for the intensity, may be simplified as follows:

$$I(\boldsymbol{x},t) + \nabla I(\boldsymbol{x},t) \cdot \boldsymbol{v} dt + I_t(\boldsymbol{x},t) dt = I(\boldsymbol{x},t)$$

$$\nabla I(\boldsymbol{x},t) \cdot \boldsymbol{v} = -I_t(\boldsymbol{x},t)$$

$$I_x(\boldsymbol{x},t) u + I_y(\boldsymbol{x},t) v = -I_t(\boldsymbol{x},t)$$

- ❑ Reminder: the second assumption is that a neighborhood of pixels moves with the **same velocity**.
- ❑ Let $\boldsymbol{x}_1 \ldots \boldsymbol{x}_n$ denote n points from the local neighbourhood of $\boldsymbol{x}$.

$$\begin{pmatrix} I_x(\boldsymbol{x}_1,t) & I_y(\boldsymbol{x}_1,t) \\ \vdots & \vdots \\ I_x(\boldsymbol{x}_n,t) & I_y(\boldsymbol{x}_n,t) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} I_t(\boldsymbol{x}_1,t) \\ \vdots \\ I_t(\boldsymbol{x}_n,t) \end{pmatrix}$$

$$A\boldsymbol{v} = \boldsymbol{b}$$

The Lucas-Kanade Algorithm

Derivation III

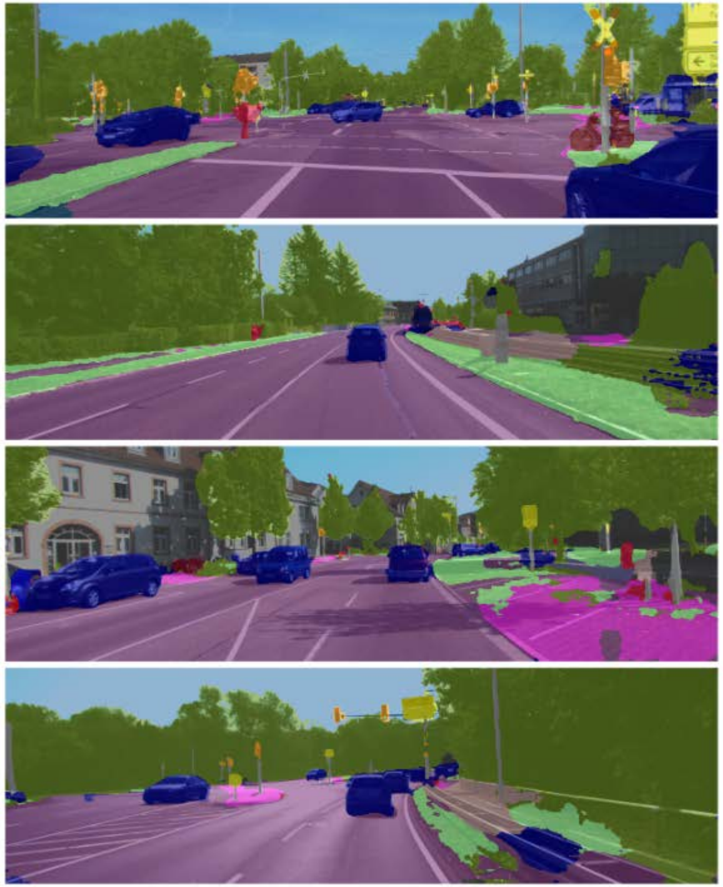
❑ If there are more than two points $(n > 2)$, the system $A\boldsymbol{v} = \boldsymbol{b}$ is overdetermined.

❑ We may solve it using the method of **least squares**:

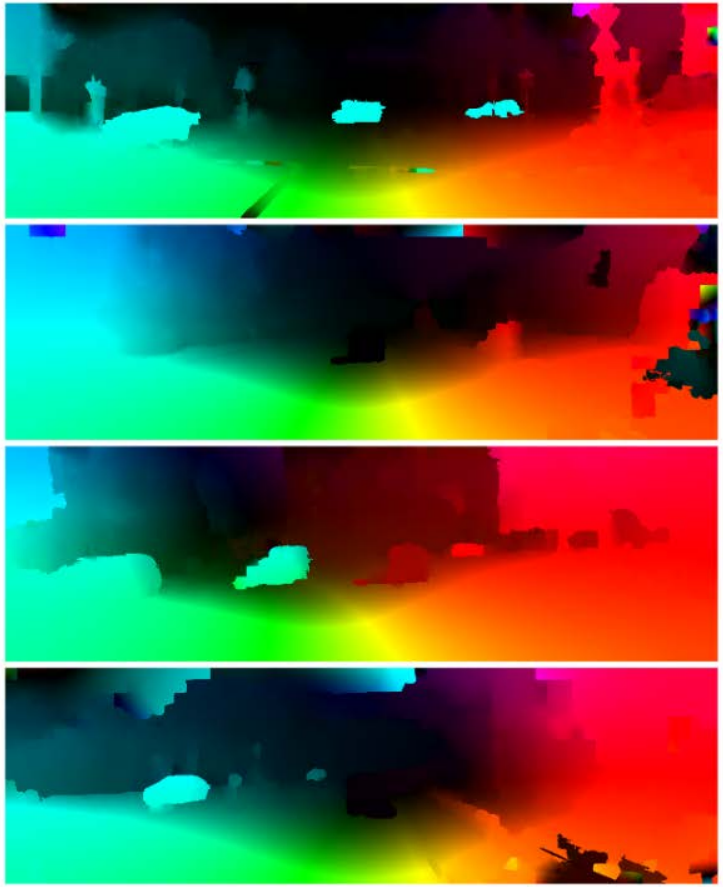
$$\boldsymbol{v} = -(A^T A)^{-1} A^T \boldsymbol{b}$$

$$\boldsymbol{v} = \begin{pmatrix} \sum_{i=1}^{n} I_x(\boldsymbol{x}_i, t)^2 & \sum_{i=1}^{n} I_x(\boldsymbol{x}_i, t) I_y(\boldsymbol{x}_i, t) \\ \sum_{i=1}^{n} I_y(\boldsymbol{x}_i, t) I_x(\boldsymbol{x}_i, t) & \sum_{i=1}^{n} I_y(\boldsymbol{x}_i, t)^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum_{i=1}^{n} I_x(\boldsymbol{x}_i, t) I_t(\boldsymbol{x}_i, t) \\ -\sum_{i=1}^{n} I_y(\boldsymbol{x}_i, t) I_t(\boldsymbol{x}_i, t) \end{pmatrix}$$

Optical Flow
Visualization (again)



Segmented images.



Overlaid optical flow.

Conclusions
Disparity Estimations and Optical Flow

- ❑ 3D scenes can be **reconstructed** from **disparity** or **optical flow**.
- ❑ Disparity algorithms must **match** points before calculating the disparity.
- ❑ Images should be **rectified** before matching is attempted.
- ❑ After calculating the disparity, it should be **refined**.
- ❑ **Optical flow** algorithms take successive pairs of images an calculate flow vectors.
- ❑ The **Lucas-Kanade algorithm** may be used to solve the optical flow equations.

Thank you for your attention!