Stereo Vision

Richard Hartley and Hongdong Li

Synonyms

Stereoscopic images, Stereopsis, Binocular vision, Stereo-photogrammetry

Definitions

Stereo vision is the process of recovering 3-dimensional geometric information of a scene from a pair of slightly different images of the scene.

1 Overview

Stereo vision is the process of extracting 3-dimensional structure of a scene from two images of the scene, taken from slightly different viewpoints. Humans and many animals use stereo vision with their two eyes to perceive 3D information of their surroundings. Figure-1 illustrates the concept of stereo vision system. Stereo vision is a fundamental problem in computer vision, and it has found wide-spread applications in many fields, including for example robot navigation, human-robot interface, virtual and augmented reality, etc.

There are two major aspects in stereo vision: (1) Stereo Geometry, which studies two-view epipolar geometry, rectification, as well as how to compute 3D depth information given pixel correspondences; and (2) Stereo Correspondence, which involves how to find a good pixel correspondences from a pair of stereo images.

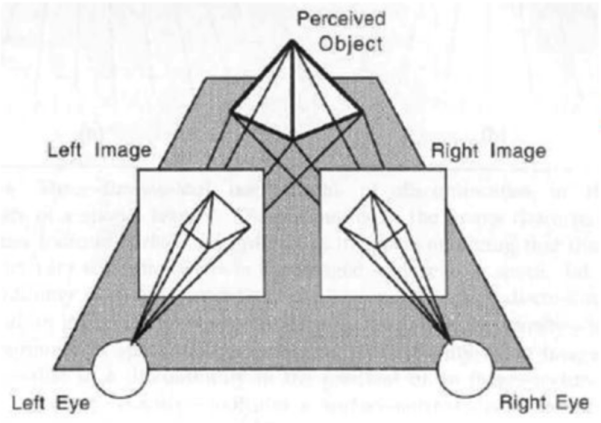

Fig. 1 *Human use two eyes to perceive 3-dimensional information of the environment. This figure illustrates the concept of binocular stereo vision perception.*

2 Stereo Geometry

Stereo vision involves the analysis of two-view epipolar geometry between the two images. It is usually assumed that the input images are accurately modelled as pinhole camera projections. In this case, a point $\mathbf{X}$ in the world is mapped to points $\mathbf{x}^R$ and $\mathbf{x}^L$ in the two images, according to the pinhole projection equations

Richard Hartley and Hongdong Li
Australian National University (ANU), Australia Centre for Robotic Vision; e-mail: firstname.lastname@anu.edu.au

$$\mathbf{x}^L = \mathtt{P}^L\mathbf{X} \;\; ; \;\; \mathbf{x}^R = \mathtt{P}^R\mathbf{X} \, . \tag{1}$$

In this equation, the world point $\mathbf{X}$ and image points $\mathbf{x}^L, \mathbf{x}^R$ are represented as *homogeneous vectors*, so $\mathbf{X}$ is a 4-vector and $\mathbf{x}^L$ and $\mathbf{x}^R$ are 3-vectors. The matrices $\mathtt{P}^L$ and $\mathtt{P}^R$ are the projection matrices for the *left* and *right* cameras respectively, and have dimension 3×4. The general form for a camera matrix is $\mathtt{P} = \mathtt{K}[\mathtt{R} \mid -\mathbf{t}]$, where $\mathtt{K}$ is an upper-triangular *calibration matrix*, and $\mathtt{R}$ is a rotation matrix, representing the orientation of the camera with respect to some world-coordinate system. The camera centre is located at the point $\mathtt{R}^{-1}\mathbf{t}$. Assuming that the *principal point* of the camera is known (often the centre of the image) and image coordinates are relative to the principal point, then the calibration matrix is a diagonal matrix $\mathtt{K}_f = \mathrm{diag}(f, f, 1)$ where f is the focal length of the camera, expressed in pixels (in other words, the focal length of the camera is equal to f times the pixel pitch). It is convenient to assume that a world coordinate frame is chosen to correspond with the centre of the left camera, oriented so that its principal axis points along the $\mathtt{Z}$-axis. In this case, $\mathtt{P}^L = \mathtt{K}_f[\mathtt{I} \mid \mathbf{0}]$.

In the ideal stereo configuration the two cameras have the same orientation, and are placed side-by-side, with the right camera located at the point $(b, 0, 0)$, hence displaced from the left camera, by a distance b along the $\mathtt{X}$-axis. In this case, the line between the two camera centres is known as the *base-line* and b is the *camera displacement*. (The length b is normally expressed in the scale of the world-coordinated – not in pixel units.) In this ideal situation, the principal axes (the pointing directions) of the two camera are parallel, and perpendicular to the base-line. A pair of cameras geometrically placed in this way are said to be *rectified*. Generally speaking, commercially available stereo cameras are rectified. The two corresponding images are known as a *rectified* stereo image pair.

The pair of camera matrices for a rectified pair of cameras (assuming equal focal lengths f) are given by

$$\begin{aligned} \mathtt{P}^L &= \mathtt{K}_f[\mathtt{I} \mid \mathbf{0}] \\ \mathtt{P}^R &= \mathtt{K}_f[\mathtt{I} \mid -\mathbf{b}] \end{aligned} \tag{2}$$

where $\mathbf{b}_R = (b, 0, 0)^\mathsf{T}$ is the centre of the right camera.
Note: for these matrices to express the correct mapping from world-to-image pixel coordinates, image coordinates must be expressed in pixels relative to their distance from the principal point.

Now consider a world point, expressed in homogeneous coordinates as $\mathbf{X} = (\mathtt{X}, \mathtt{Y}, \mathtt{Z}, 1)$. One computes that

$$\begin{aligned} \mathbf{x}^L &= \mathtt{K}_f[\mathtt{I} \mid \mathbf{0}](\mathtt{X}, \mathtt{Y}, \mathtt{Z}, 1)^\mathsf{T} = (f\mathtt{X}, f\mathtt{Y}, \mathtt{Z})^\mathsf{T} \\ \mathbf{x}^R &= \mathtt{K}_f[\mathtt{I} \mid (-b, 0, 0)^\mathsf{T}](\mathtt{X}, \mathtt{Y}, \mathtt{Z}, 1)^\mathsf{T} = (f(\mathtt{X} - b), f\mathtt{Y}, \mathtt{Z})^\mathsf{T} \end{aligned}$$

Transforming homogeneous coordinates to Euclidean coordinates (de-homogenizing) gives

$$\begin{aligned} \mathbf{x}^L &= (x^L, y^L) = (f\mathtt{X}/\mathtt{Z}, f\mathtt{Y}/\mathtt{Z}) \\ \mathbf{x}^R &= (x^R, y^R) = (f(\mathtt{X} - b)/\mathtt{Z}), f\mathtt{Y}/\mathtt{Z}) \end{aligned} \tag{3}$$

(To avoid excessive notation, the same symbols will be used for homogeneous coordinates and the corresponding de-homogenized coordinates, relying on context to disambiguate.)

Comparing the right and left image points, one observes that the y-coordinates of the two points are the same. On the other hand, the x coordinates differ by a distance

$$x^L - x^R = d = \frac{fb}{\mathtt{Z}} \, , \tag{4}$$

known as the *disparity*. Expressed the other way round

$$\mathtt{Z} = \frac{fb}{d} \, . \tag{5}$$

Hence, the depth $\mathtt{Z}$ of a point is inversely proportional to the disparity; knowing f and b, the depth of a point may be directly computed from measured disparity. The complete coordinates of the 3-D point can be computed knowing $\mathbf{x}^L$ and the disparity d according to

$$\mathbf{X} = (\mathtt{X}, \mathtt{Y}, \mathtt{Z})^\mathsf{T} = \frac{b}{d}(x^L, y^L, f)^\mathsf{T} \, . \tag{6}$$

It is possible to derive (4) geometrically, as shown in figure 2

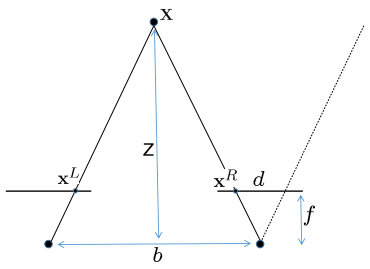

Fig. 2 *Geometric derivation of the disparity formula (5). The triangles shown are similar, so* $z/b = f/d$*, from which (5) follows.*

Accuracy of stereo depth estimation.

The equation (5) highlights a weakness of stereo depth-estimation. From $z = fb/d$ and its derivative $z' = -fb/d^2$ results an approximation for accuracy:

$$\Delta z \approx \frac{-z^2}{fb}\,\Delta d$$

where Δd represents the accuracy with which d may be estimated. In other words, error in stereo depth estimation is proportional to the square of depth.

This may be written in terms of the angular resolution of the camera. Let θ_R be the offset of the point $\mathbf{x}^R$ from the principal axis. (This calculation is done for the scan line passing through the principal point – that is, for $y^R = 0$.) Then $x^R/f = \tan(\theta_R)$. Then $d = x^L - x^R = x^L - f\tan(\theta_R)$. To first order approximation, $\Delta d = -f/\cos^2(\theta_R)\,\Delta\theta_R$. Therefore,

$$\Delta z \approx \frac{z^2}{b\cos^2(\theta_R)}\Delta\theta_R\ ,$$

where θ_R is measured in radians. This shows how depth accuracy is limited by the angular resolution of the camera.

Table 1 shows accuracy of stereo depth accuracy, if angular resolution is equal to 1 arc minute, approximately the resolution of the human eye at the fovea. This is equivalent to a camera with field of view 90 degrees, and pixel dimension 6876 in the x-direction. It is assumed disparity can be measured with an accuracy of ± 0.5 pixels.

depth (m)	disparity(degrees/minutes)	error
1 m	3.6°	$\pm$ 2.3 mm
10 m	$22'$	$\pm$ 233mm
100 m	$2.2'$	$\pm$ 24 m

Table 1 *Assuming a baseline of* 64 *milimetres (mm), approximately the distance between human eyes, and angular resolution of* $\Delta d = \pm 0.5$ *minutes of arc (approximately human eye resolution), this table shows accuracy of stereo depth resolution for points at different depths measured in metres (m). Disparity represents the angle between rays from the two eyes to a point at the given depth. Depth error is inversely proportional to focal length* f *and baseline* b*.*

Stereo rectification.

So far it has been assumed that the input stereo images are already *rectified*, in the sense that scan-lines are horizontal and matching points have the same y-coordinate in the images. For images taken with a single moving camera, or with a stereo-rig not accurately calibrated, this is not necessarily the case. A rectified pair of images correspond to cameras pointing in the same direction, perpendicular to the base line. It is possible to resample the images so as to simulate the effect of a camera rotation.

This may be done using only a small set of image correspondences between the two images by computing the essential matrix Longuet-Higgins (1981) of the camera pair. Each image must be rotated so that the epipolar lines lie at the point with homogeneous coordinates $(1, 0, 0)$, namely, the point at infinity along the x-axis. Rectification can be done without camera calibration, by computing the fundamental matrix from matched points (Hartley (1999)). See figure 3.

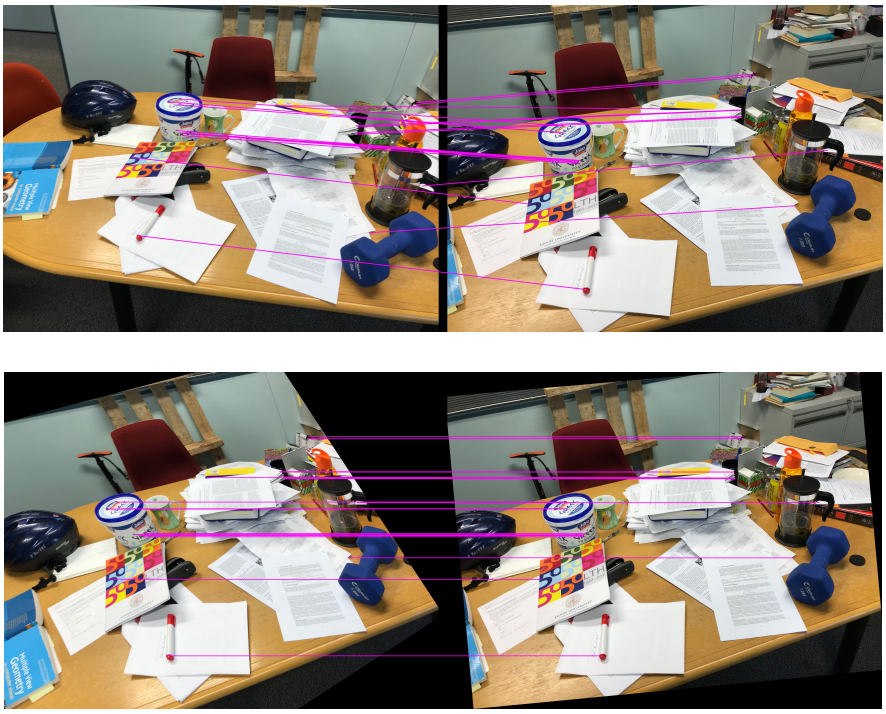

Fig. 3 *Pair of images before and after rectification. Lines indicate matching points. After rectification, matching points lie on the same horizontal scan line in the two images.*

3 Stereo Correspondence

Given a point $\mathbf{x}^L$ in the left image, its disparity d can be estimated by matching it with the corresponding point $\mathbf{x}^R$ in the right image. Then $d = x^L - x^R$. For rectified images, the point $\mathbf{x}^R$ must have the same y-coordinate as $\mathbf{x}^L$, so this matching task may be accomplished by searching along a horizontal line in the image with the given y-coordinate. Since disparity is non-negative, the matching point $\mathbf{x}^R = (x^R, y^R)$ is of the form $\mathbf{x}^R = (x^L - d, y^L)$, so the search is to be carried out over points to the left of (x^L, y^L).

The point $\mathbf{x}^R$ best matching $\mathbf{x}^L$ may be chosen as the one in the right image that looks "most like" $\mathbf{x}^L$. As shown in figure 4, for rectified images, the best matching point is found by searching along the a horizontal scan line.

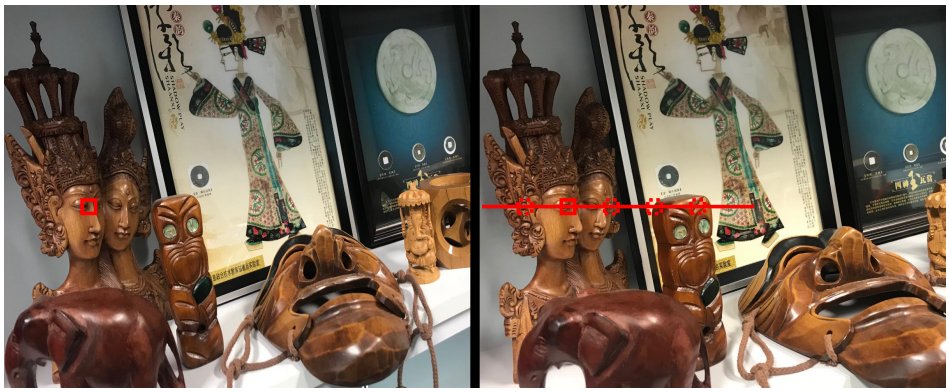

Fig. 4 *If the images are rectified, a point in the left image is matched to a point lying on the same horizontal scan line. A point may potentially match any other point on the line. (Here and elsewhere, rectified images are shown with the left-image on the right, to allow for viewing using cross-fusion – view left-image with right eye, and vice versa).*

Window based similarity.

Since a single pixel by itself rarely has enough information to allow accurate matching, image-to-image matching is commonly carried out by comparing windows around points to be compared. Let $\mathbf{x}^L$ and $\mathbf{x}^R$ be two points, and denote

by W_L and W_R windows (for example, of dimension 7×7) centred on these two points. Various measures of similarity between two windows have been proposed. It is convenient to consider vectors $\mathbf{w}^L$ and $\mathbf{w}^R$ representing all the pixels lying in the windows $\mathbf{W}^L$ and $\mathbf{W}_R$.Various similarity measures between vectors $\mathbf{w}^L$ and $\mathbf{w}^R$ are developed, some are summarized as follows. For simplicity, $\mathbf{w}^L$ will be represented as $\mathbf{x}$ and $\mathbf{w}^R$ as $\mathbf{y}$, and these all present the image information, such as RGB values but not coordinates, of the corresponding points. Different similarity measures used are

- Sum of Squares Difference (SSD), defined as

$$\mathrm{SSD}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2 = \sum_{i=1}^{n} (x_i - y_i)^2 \, .$$

  This is just the squared Euclidean (L_2) distance between the vectors.
- Sum of Absolute Difference (SAD), defined as

$$\mathrm{SAD}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1 = \sum_{i=1}^{n} |x_i - y_i|$$

  measures the L_1-norm distance. Both SSD and SAD are sensitive to lighting or intensity changes between the right and left images. If, for instance, one image is brighter than the other, then both the SAD and SSD distances may be large, even though $(\mathbf{x}, \mathbf{y})$ represents a correct match. The two windows are compared, based more on their overall intensity, than details of intensity changes in the windows. The SAD is a more robust measure of similarity than SSD, since it is less sensitive to noise. If a single pixel differs greatly between left and right, then the SSD distance will be large, because the difference is squared; the SAD distance will be less affected.
- Normalized similarity measures: If the left and right images have different intensities, one may account for this by scaling, resulting in normalized SSD and normalized SAD measures.

$$\begin{aligned} \mathrm{NSSD}(\mathbf{x}, \mathbf{y}) &= \mathrm{SSD}(\mathbf{x}/\|\mathbf{x}\|, \mathbf{y}/\|\mathbf{y}\|) = 2(1 - \cos(\theta)) \\ \mathrm{NSAD}(\mathbf{x}, \mathbf{y}) &= \mathrm{SAD}(\mathbf{x}/\|\mathbf{x}\|, \mathbf{y}/\|\mathbf{y}\|) \, , \end{aligned}$$

  where θ represents the angle between the vectors $\mathbf{x}$ and $\mathbf{y}$. NSSD is closely related to the negative Normalized Cross Correlation (NCC), defined as

$$-\mathrm{NCC}(\mathbf{x}, \mathbf{y}) = -\frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} = -\frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \, \|\mathbf{y}\|} = -\cos(\theta).$$

- ZNCC etc: If one of $\mathbf{x}$ or $\mathbf{y}$ is perturbed by adding a constant value to each component (corresponding to adding a constant to intensities in one of the images), then the value of the similarity measures above will be changed. One may address this by subtracting the mean value $\bar{\mathbf{x}}$ or $\bar{\mathbf{y}}$ from each component before computing the NCC, SSD or SAD values. For instance, a suitable similarity measure is

$$-\mathrm{ZNCC}(\mathbf{x}, \mathbf{y}) = -\frac{\langle \mathbf{x} - \mathbf{1}\bar{x}, \mathbf{y} - \mathbf{1}\bar{y} \rangle}{\|x - \mathbf{1}\bar{x}\| \, \|\mathbf{y} - \mathbf{1}\bar{y}\|} = \frac{n\bar{x}\bar{y} - \langle \mathbf{x}, \mathbf{y} \rangle}{\sigma_x \sigma_y} \, .$$

- Census transform: The Census Transform converts every pixel into a binary string by comparing its intensity value with the values in its small neighbourhood. The resulted binary string encodes whether the neighbours are smaller than the center pixel or not. For a 3×3 window, the census transform gives rise to a string of length 8. Advantages of Census Transform include that this representation is robust to light changes, photometric distortion, vignetting, image boundary and noise.
- Rank transform: The Rank Transform is a non-parametric transformation applied to image pixel intensities. It replaces the intensities of pixels within a local window with their ranking value Rs, where R is calculated as the number of neighbouring pixels whose intensity are less than the center pixel's intensity in the window.
- Dense SIFT Flow and DAISY Flow etc.: Instead of using pixel raw intensity or color as feature vector, some stereo algorithms also use dense scale invariant feature descriptors. For example, the SIFT Flow algorithm uses SIFT descriptors densely at pixel grids on the entire image. In the same spirit, dense DAISY descriptors are also used to compute stereo matching.

4 Pyramid matching

Matching based solely on local appearance and epipolar search usually gives poor performance, because there may be many points $\mathbf{x}^R$ that match a given point $\mathbf{x}^L$ closely. This is particularly true with when images have repeated structures, such as windows on a building facade, where local matching is insufficient to determine which window matches which. This is illustrated in figure 5. Moreover, natural variations between left and right images, such as slight lighting changes,

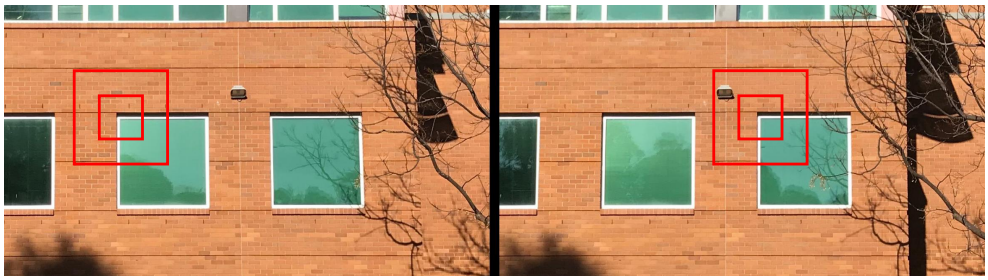

Fig. 5 *Local matching methods are not able to distinguish the correct match among repeated structures. In a small window, the two corners look very similar. Looking further in a larger window shows that they are a false match.*

or areas of low contrast can lead to noisy results.

To improve results, it is necessary to adopt a more global approach to image matching. Pyramid matching Quam (1987) is a method for addressing this problem.

The idea behind pyramid matching is to create Gaussian pyramids of the two images (Burt and Adelson (1983)) as shown in figure 6. Matching then proceeds from the lowest level (lowest-resolution) of the pyramid, and the match is refined by working up the pyramid. Note in this description, the image pyramid is envisaged as being an inverted pyramid, with lowest-resolution at the bottom.

Fig. 6 *An image pyramid (on left) is build from both images. Then, images are matched from the bottom up, propagating estimated disparities up the pyramid.*

Thus, let $I_k^L; k = 0, \ldots, n$ and $I_k^R; k = 0, \ldots, n$ represent the image pyramids for left and right images, where I_0^L and I_0^R are the original images, and each I^k is half the size (in each dimension) of the previous I^{k-1}. Successive levels I^k are obtained from I^{k-1} using a 3×3 Gaussian convolution filter, with step-size 2 (in modern Deep-learning terminology). If the maximum disparity in the original images I_0^L and I_0^R is $d_{\max}$, then the maximum disparity at level k will be $d_{\max}/2^k$. Hence, the search for a matching pixel at lower levels of the pyramid involves a search over a smaller range of disparities.

Matching starts at the bottom level of the pyramid, by matching I_n^L and I_n^R. Next, images I_{n-1}^L and I_{n-1}^R are matched, and so on. Finally, the original images I_0^L and I_0^R are matched.

Suppose that at level k, a pixel (x^k, y^k) in I_L^k matches pixel $(x^k - d, y^k)$ in I_R^k. One may denote this disparity by $d^k(x, y)$, meaning the disparity for pixel (x, y) at level k. Now, let (x^{k-1}, y^{k-1}) be a point in image I_{k-1}^L. The disparity of this point may be deduced from that of the point lying below it in the pyramid. Thus, $d^{k-1}(x, y) = 2d^k(x/2, y/2)$, where integer division is meant. This results in disparities with only even values. Because of the increase in resolution, the correct disparity may equally well be $2d^k(x/2, y/2) \pm 1$. Therefore, a refinement is made to the disparities at level $k-1$, whereby

the disparity assigned to pixel (x^{k-1}, y^{k-1}) is chosen from the set $\{2d^k(x/2, y/2) - 1,\ 2d^k(x/2, y/2),\ 2d^k(x/2, y/2) + 1\}$.

One may verify that this hierarchical scheme addresses the problem of false matches, particularly in the presence of repeated structures. At the lower level of the pyramid, matches are found at a larger scale, at lower resolution. For instance, a window a the top left of a building facade will not be mistaken for window in the middle of the facade, since their local neighbourhoods, at the lower resolution, are dissimilar. This is illustrated in figure 5.

The above description gives only an outline of the pyramid-match method, and variations are possible. In the above description, it is assumed that pixel $(x^{k-1}, y^{k-1}$ lies *above* (and hence derives its disparity from) only pixel at level k, whereas because of the Gaussian filtering, a pixel at level $k-1$ contributes to more than one pixel at the next level, so a wider choice of possible disparity choices may be explored, among those inherited from the level below.

In the paper Quam (1987), in passing up the pyramid, the right (or left) image at level $k-1$ is prewarped, according to a warping function consistent with the matching at level k, before matching takes place at level $k-1$. This is done so that image patches may be more similar in the left and right images.

A disadvantage of pyramid matching as described here is that it may miss matches between small structures. For instance, a narrow object (such as a lamppost standing out from the background) may not be visible, or significant at lower resolution. For instance, an object of width 30 pixels will almost completely disappear if images are downsampled by 5 levels of pyramids (downsampling by a factor of 32). At the bottom level, the images will be matched according to the appearance of the background behind the object, and the correct match may not be recovered as the pyramid is ascended, because of the limited range of disparity search. Nevertheless, pyramid matching remains a valuable ingredient of stereo matching, particularly when the range of disparities is large Quam (1987); Burt and Adelson (1983).

5 Dynamic programming

An occlusion occurs in a pair of stereo images if a point visible in one camera is not visible in the other camera. In this case, it is not possible to find a match. If the point is visible in the left view, but not the right view, then this is called a right-occlusion; a left-occlusion is the other way round – visible in the right view, but not the left.

Consider now two rectified cameras viewing a scene, and consider a single scan-line. It is assumed for now that the scene is connected along the scan line in question. The situation envisaged is shown in figure 7, where it is assumed for now that there are no occlusions. If a point $\mathbf{x}^L = (x^L, y)$ matches point $\mathbf{x}^R = (x^R, y)$, this may be written as $x^R = m(x^L)$, so x^R may be considered as a function of x^L.

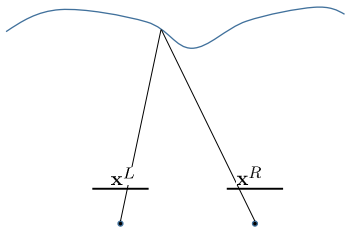

Fig. 7 *Illustration of monotonicity. As a point $\mathbf{x}^L$ moves from left to right in the left image, the corresponding point $\mathbf{x}^R = m(\mathbf{x}^L)$ also moves from left to right. The function m is monotonically increasing.*

An important property of the matching function m is the following.

Lemma 1 Monotonicity condition. *If there are no occlusions, and m is a continuous matching function for a non-transparent scene, then it is monotone-increasing.*

In stating this lemma, it is assumed that we are in the continuous domain (rather than discrete pixels), and m is defined for all x^L in some interval. Indeed, by the assumption that there are no occlusions, the point $m(x^L)$ is defined for every x^L – every point has a match. On geometric grounds, the mapping must also be one-to-one, since it is not possible for two points $\mathbf{x}_1^L$ and $\mathbf{x}_2^L$ both to match a given point $\mathbf{x}^R$. This means that the function m is invertible, and since it is continuous, it is monotonic. Once again, on geometric grounds, it cannot be decreasing.

Dynamic programming.

The monotonicity condition leads to a dynamic-programming approach to stereo-matching along a single scan line. In particular, suppose that a cost function $c(x, x')$ is defined for all pairs of pixels $x = x^L$ and $x' = x^R$. The cost $c(x, x')$

measures the degree of incompatibility of a match between points x and x'. This cost may be one of the correlation costs described previously.

Let the pixels along the left scan line be denoted $x_1, \ldots, x_m$, numbered in order from left-to-right. Let the pixels in the right scan line be similarly denoted $x'_1, \ldots, x'_n$. The notation $x'_i \leq x'_j$ is introduced to mean that x'_i lies to the left of x'_j. Suppose that a cost $c(x_i, x'_j)$ is defined for each potential match. The matching problem may then be defined as minimizing the total cost

$$C(m) = \sum_{i=1}^{m} c(x_i, m(x_i)) \,,$$

where m is a matching function, so $m(x_i) \in \{x'_1, \ldots, x'_n\}$, and m is an increasing function in the sense that $m(x_i) \geq m(x_{i-1})$ for all i. Thus, $C(m)$ represents the total cost of a matching m as the sum of costs of each individual pixel match.

The problem may be defined also in terms of disparities, defining $d(x_i) = i - j$, where $m(x_i) = x'_j$. The cost $c(x_i, x'_j)$ may also be defined in terms of disparities by $\tilde{c}(x_i, d_i) = c(x_i, x'_{i-d_i})$. If $\mathbf{d}$ represents the disparities for all pixels on the scan line, then the problem becomes one of minimizing

$$C(\mathbf{d}) = \sum_{i=1}^{m} \tilde{c}(x_i, d_i)$$

subject to $d_i \geq 0$ (assuming the images are rectified), and $d_{i+1} \leq d_i + 1$.

This problem may be efficiently solved using dynamic programming (DP). Considering a general dynamic programming problem, let $\mathtt{C}$ be a cost array, as shown in figure 8, where $\mathtt{C}_{ij}$ contains the value $c(x_i, x_j)$. The optimization task is to find the least-cost non-decreasing path through this array, as depicted in figure 8.

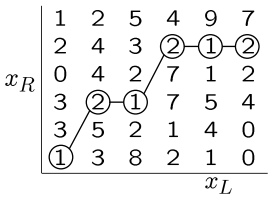

Fig. 8 *Dynamic programming to find the least-cost match. Given an array of cost values, the task is to find the monotonically increasing (or non-decreasing) path through the table. Note that there are non-monotonic paths of lesser cost.*

Algorithmically this may be solved by the following algorithm (figure 9)

1: **procedure** DYNAMIC PROGRAMMING
2: **for** $j = 1, \ldots, n$ **do** $S_{0j} = 0$
3: **for** $i = 1, \ldots, m$ **do** // forward sweep
4: **for** $j = 1, \ldots, n$ **do**
5: $S_{ij} = C_{ij} + \min_{j' \leq j} S_{i-1,j'}$
6: $j^*_n = \arg\min_j S_{nj}$
7: **for** $i = n-1, \ldots, 1$ **do** // backward sweep
8: $j^*_i = \arg\min_{j \leq j^*_{i+1}} S_{ij}$

Fig. 9 *Dynamic programming enforcing monotonicity. At termination, $j^*_i = x^R$ is the match for $i = x^L_i$.*

The algorithm can be generalized by modifying line 5 (and correspondingly modifying line 8) to

$$S_{ij} = C_{ij} + \min_{j'} \left(S_{i-1,j'} + E(i-1 : j',\, i : j) \right) \tag{7}$$

where $E(i-1 : j',\, i : j)$ measures the compatibility of matching pixels j' and j (in the right image) with neighbouring pixels $i-1$ and i in the left. The values S_{ij} represent the cost of the cheapest monotonic path through the table, from the left *to the current point* i, j.

Monotonicity with occlusions.

The monotonicity condition will hold under more general conditions than continuity of the matching function. A general condition under which the monotonicity condition will hold is where you cannot see round the back of objects, in the sense shown in figure 10.

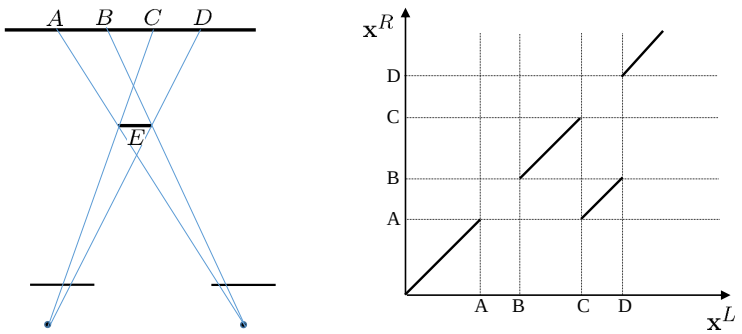

Fig. 10 *If it is possible to see "round the back" of an object, as shown in the figure, then monotonicity fails. For instance, point E is seen to the right of point B in the left view, but to the left of B in the right view.*

In the case of occlusions, the monotonicity condition will hold, as shown in figure 11. It is evident that only objects

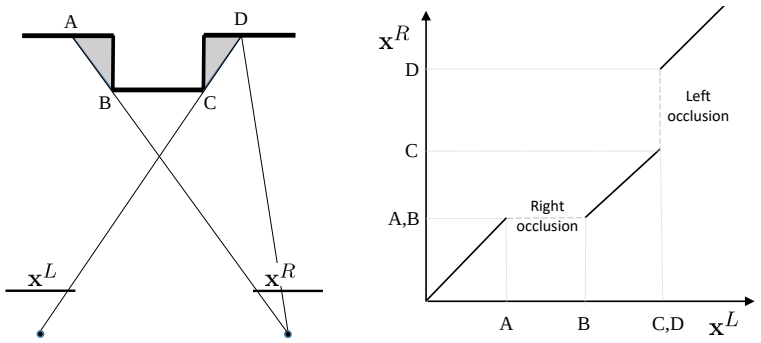

Fig. 11 *Monotonicity may still hold if there are occlusions. In this figure, the greyed areas are occluded from either the left of right views. Hence, no matching points are found, leading to gaps in the graph of the matching function (right figure). However, it is possible to fill in the gaps (with horizontal or vertical lines) such that the graph is monotonic non-decreasing. Right and left occlusions are indicated when the graph is horizontal or vertical.*

narrower than the baseline can cause a violation of monotonicity, as pointed out in Belhumeur and Mumford (1992).Even if the monotonicity condition holds, a weakness of the dynamic-programming approach described here is that it works on each line independently, and can give results that are visibly inconsistent across scan lines.

6 Markov Random Fields and Graph-cuts

The use of Markov Random Fields (MRFs) in addressing the stereo matching problem is based on the observation that neighbouring pixels (in the scan direction as well as the cross-scan (y) direction) tend to be correlated. Large differences in disparity between neighbouring pixels generally occur only at occlusion boundaries, which normally are comparatively sparse in the image.

Let $\mathbf{d}$ represent the set of disparities at all pixels in the (left) image. An MRF approach to stereo-matching involves a cost (or energy) function $E(\mathbf{d})$, assigning a cost to a given set of disparities, which in turn define a matching. Finding the best matching is then formulated as finding the assignment $\mathbf{d}$ that minimizes the cost $E(\mathbf{d})$.

Let the pixels in the image be denoted by i in a set $\mathcal{V}$, the set of all pixels in the image. Hence $i \in \mathcal{V}$ means that i is a pixel in the pixel set $\mathcal{V}$. It will later be seen that the set $\mathcal{V}$ may also be thought of as a set of vertices in a certain graph.

The disparity $\mathbf{d}$ consists of a disparity d_i assigned to each pixel i. Possible values of d_i are chosen from a finite set $\mathcal{L} = \{0, \ldots, L\}$, called the set of *labels*. (Commonly, the labels d_i represent the left-right disparity measured in pixels. However, it is possible to interpret labels as representing the disparity measured in some sub-pixel scale (such as

half-pixels). However, this approach is normally limited to choosing between a finite number of disparity values at each pixel – a continuous choice of disparity values is not supported.) Label values may be represented by Greek letters λ, μ. Consequently, $\mathbf{d}$ may be identified as the assignment of a label d_i to each pixel i, and as such may be thought of as an element of $\mathcal{L}^{\mathcal{V}}$ (the set of functions from $\mathcal{V}$ to $\mathcal{L}$). Then $E(\mathbf{d})$ is a real-valued function $E : \mathcal{L}^{\mathcal{V}} \to \mathbb{R}$.

The cost function $E(\mathbf{d})$ contains a term $E_i(d_i)$ for each i , known as a *unary term* or *vertex term*, measuring the cost of assigning label d_i to pixel i. A cost function consisting only of unary terms is of the form

$$E(\mathbf{d}) = \sum_{i \in \mathcal{V}} E_i(d_i) \ .$$

Minimizing this cost function over all choices of $\mathbf{d} \in \mathcal{L}^{\mathcal{V}}$ is equivalent to choosing d_i independently for each i as the label with minimum unary cost (the least-cost disparity for each pixel).

In the most commonly used case, however, the cost-function $E(\mathbf{d})$ is chosen to have *edge-terms* as well. Identifying the pixels i as being the vertices of a graph $\mathcal{G}$, edges can be added between the vertices to make a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is a set of edges. Thus, an edge is a pair (i, j) with $i, j \in \mathcal{V}$, and $\mathcal{E}$ should contain at most one of the pairs (i, j) and (j, i). In this case, the set of *neighbours* of a vertex i is the set

$$\mathcal{N}(i) = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E} \text{ or } (j, i) \in \mathcal{E}\} \ .$$

Corresponding to an edge $(i, j) \in \mathcal{E}$ an edge term $E_{ij}(d_i, d_j)$ may be defined. The cost function most commonly used, then, has the form

$$E(\mathbf{d}) = \sum_{i \in \mathcal{V}} E_i(d_i) + \sum_{(i,j) \in \mathcal{E}} E_{ij}(d_i, d_j) \ , \tag{8}$$

though it is also possible to define terms that depend on three or more labels.

Most commonly, the graph structure used is a 4-connected graph, in which each pixel is joined to the pixels above, below, and to left and right.

Edge terms.

The purpose of the edge terms in the cost function (8) is to prevent, or at least discourage neighbouring pixels from being assigned widely different labels (disparities). They act as regularization terms.

As a particular case, the monotonicity condition may be formulated as an MRF in which the pixels (vertices) are joined to the pixel on either side only, but not to those above and below. In this case an edge set $\mathcal{E}$ is defined in such a way that a pair $(i, j) \in \mathcal{E}$ only if the pixel j is the right-hand neighbour of pixel i. In this case, the edge term is

$$\begin{aligned} E_{ij}(d_i, d_j) &= \infty \ \text{ if } d_j - d_i > 1 \\ &= 0 \ \text{ otherwise.} \end{aligned} \tag{9}$$

Vertex terms $E_i(d_i)$ measure the cost of assigning disparity d_i to pixel i. The minimum of the cost-function defined in this way may be found efficiently, and independently on each scan-line.

In general, an edge term $E_{ij}(d_i, d_j)$ frequently depends only on the difference $d_i - d_j$, or the absolute difference $|d_i - d_j|$. This reflects the feeling that the cost of assigning disparities (labels) d_i and d_j to neighbouring pixels depends only on the difference between their disparities.

Commonly, if $d_i = d_j$, then $E_{ij}(d_i, d_j) = 0$, reflecting the observation that neighbouring points in a scene tend to lie at the same (or almost the same) depth, and hence have the same disparity. There should be no cost, therefore, in assigning neighbouring pixels the same label. The exception to this is that the two neighbouring pixels lie on opposite sides of an occluding boundary in the image, so that there is a depth discontinuity between pixels i and j. This occurs relatively sparsely, so assigning widely varying disparity should incur a larger cost. However, once a depth discontinuity occurs, extra cost should not be incurred for larger depth discontinuity. This suggests the use of the following edge function

$$E_{ij}(d_i, d_j) = w_{ij} \ \min(T^2, (d_i - d_j)^2), \tag{10}$$

known as the *truncated quadratic* cost function, with weights w_{ij}. The value T is a threshold above which all disparity differences incur the same cost.

MRFs and probability.

In a probabilistic interpretation of MRFs, the energy is interpreted as the negative log-probability of the disparity assignment $\mathbf{d}$. That is

$$P(\mathbf{d}) = \frac{\exp(-E(\mathbf{d}))}{Z} \tag{11}$$

where Z is a normalizing constant. The equivalence of energy functions and MRF probability distributions is shown by the Hammersley-Clifford theorem (Besag (1974)).

According to this formula, when $E(\mathbf{d})$ has the form given in (8),

$$P(\mathbf{d}) = (1/Z) \prod_{i \in \mathcal{V}} \exp(-E_i(d_i)) \prod_{(i,j) \in \mathcal{E}} \exp(-E_{ij}(d_i, d_j)) \; . \tag{12}$$

If this is interpreted as a product of independent probability distributions, then it allows the interpretation of $E_i(\lambda), \lambda \in \mathcal{L}$ as

$$E_i(\lambda) = -\log P_i(d_i = \lambda)$$

where $P_i(d_i = \lambda)$ means the probability that pixel i has disparity λ. This probability distribution P_i is normally taken to mean the *conditional* probability distribution of the different choices of disparity for pixel i *given* the pair of stereo images. It may then be written as $P_i(d_i = \lambda \mid D)$, where D represents the data, that is the pair of images (I^L, I^R).

The expression $E_i(\lambda) = P_i(d_i = \lambda \mid D$ represents a probability of a match between points with coordinates x_i^L and $x_i^R = x_i^L - d_i$. Under the assumption that point $\mathbf{x}^R$ is obtained from point $\mathbf{x}^L$ by corruption by Gaussian intensity noise, this probability is $\exp\big(-k\big(I(\mathbf{x}^L) - I(\mathbf{x}^R)\big)^2\big)$ (ignoring the normalizing constant). In this case,

$$E_i(\lambda) = -\log P_i(d_i = \lambda) = k\big(I(\mathbf{x}^L) - I(\mathbf{x}^R)\big)^2 \; ,$$

so, the cost of a proposed match is proportional to the squared intensity difference.

Mutual information.

The probability $P_i(d_i)$ may be written as $P(\mathbf{x}^L, \mathbf{x}^R)$, and if this probability is formulated in terms of image intensity, it becomes $P(i^L, i^R) = P(I^L(\mathbf{x}^L), I^R(\mathbf{x}^R))$. Rather than interpreting this probability in terms of the set of all possible pairs of images, a difficult thing to do, one may estimate $P(i^L, i^R)$ empirically, from the current stereo pair. Thus, if $\mathbf{d}$ is the disparity map, one defines the empirical probability of matching intensities by $P(i^L, i^R) = P(I^L(\mathbf{x}^L), I^R(\mathbf{x}^L - \mathbf{d}))$, which is computed as a $2D$ histogram of possible matched intensity pairs.

The trouble with this approach is that this empirical probability distribution cannot be computed until the disparity $\mathbf{d}$ is known – a chicken-and-egg situation. An iterative solution is proposed in Kim et al (2003) where an initial disparity map $\mathbf{d}_0$ is assumed, defining an initial match probability. On the basis of this, a new disparity match $\mathbf{d}_1$ is found by minimizing the MRF cost function based on the unary terms $-\log(P(i^L, i^R))$, allowing an improved matching probability to be computed, and so on until convergence.

The advantage of this method is that it allows for cases where the left and right images can vary significantly, perhaps by unmodelled lighting, appearance changes or specularities. In Kim et al (2003) it is interpreted in terms of mutual information. There, a further step is recommended, involving $2D$ Gaussian smoothing of the joint histogram $-\log(P(i^L, i^R))$ computed at each stage. Suppose $P(\mathbf{d})$ is a joint probability distribution for the disparities $\mathbf{d} \in \mathcal{L}^{\mathcal{V}}$, taking the form of an MRF defined on a tree, (or a single scan line), with vertices $\mathcal{V}$ and edges $\mathcal{E}$. Suppose also that marginal probabilities $P_i(d_i)$ and $P_{ij}(d_i, d_j)$ are known or can be estimated. Let vertex and edge terms be defined in terms of marginal probabilities by

$$\begin{aligned} E_i(d_i) &= -\log(P_i(d_i)) \\ E_{ij}(d_i, d_j) &= -\log\left(\frac{P_{ij}(d_i, d_j)}{P_i(d_i)\, P_j(d_j)}\right) \; . \end{aligned} \tag{13}$$

In particular, if d_i and d_j are independent, then $E_{ij}(d_i, d_j) = 0$. Then, the joint probability $P(\mathbf{d})$ is given by (12). In this case, denoting by D_i the random variable corresponding to d_i and D the joint random variable for $\mathbf{d}$, then $H(D) = \sum_{i \in \mathcal{V}} H(D_i) - \sum_{(i,j) \in \mathcal{E}} MI(D_i, D_j)$, where $H(D)$ denotes entropy and $MI(D_i, D_j)$ is the mutual information. Hence, finding the most probable set of disparities is equivalent to minimizing the cost function (8). This suggests, that if the marginal probabilities $P_i(d_i)$ and $P_{ij}(d_i, d_j)$ can be estimated, then the vertex and edge costs should be defined by (13).

Although this does not hold on an arbitrary graph, it provides a heuristic for defining the edge terms in an MRF approach to stereo-matching.

Nevertheless, it leaves the question of how marginal probabilities might be computed.

1. In one approach, P_{ij} may be interpreted as a prior probability distribution, in other words, one defined without reference to the data $D = (I^L, I^R)$. Thus, $P_{ij}(d_i, d_j)$ may represent the empirical probability, among all possible real images, (denoted W for "world"), or among all images produced by some generative model, that two neighbouring pixels have disparities d_i and d_j. This may be denoted as $P(d_i, d_j \mid W)$. Since this is a difficult thing to estimate accurately the prior is often arbitrarily set to be defined by some favourite edge function, such as a truncated quadratic, or truncated linear.
2. In another approach, the probability $P_{ij}(d_i, d_j)$ depends on the data D, as well as knowledge of the world W. Hence the probability is $P_{ij}(d_i, d_j \mid D, W)$. In this case, the edge term being dependent on the data, the resultant MRF is called a Conditional Random Field (CRF). Still, the quantity $P_{ij}(d_i, d_j \mid D, W)$ is a little obscure, or at least something that it is difficult to put a number to. A popular approach is simply to set

$$E_{ij}(d_i, d_j) = w_{ij} f(d_i, d_j) \ ,$$

where f is a favoured edge function, such as truncated quadratic, and w_{ij} is a data-dependent value, usually a decreasing function of the image intensity gradient (in the left image) between neighbouring pixels i and j. The rationale is that a large intensity gradient implies a greater likelihood that there is an occluding edge. So, the cost of predicting such an occluding edge at that point should be small when the gradient is large.

Minimizing MRF energy.

In the case of the edge-cost (9) enforcing the monotonicity condition, it was seen that minimizing the MRF energy can be done exactly, using dynamic programming. This will always be true, whatever the edge-cost, if nodes in the graph have only connections along the scan-line, in the x-direction, and the assignment of disparity-labels can be done line-by-line.

However, when energy-costs are applied to edges between scan lines (to ensure consistency between scan lines), the problem of finding a minimum-cost labelling is more complex. In the general case, finding the minimum-cost labelling of a cost-function such as (8) is an NP-hard problem. In certain cases, depending on the form of the edge-costs however, minimizing $E(\mathbf{d})$ for $\mathbf{d} \in \mathcal{L}^{\mathcal{V}}$ is possible. Here is a summary of various cases in which the cost function (8) may be minimized. In all cases, this depends on the edge-terms $E_{ij}(\cdot, \cdot)$, but not on the unary terms.

1. If the label set $\mathcal{L} = \{0, 1\}$, containing only two labels, then the minimum energy can be found exactly by solving a graph max-flow problem if each edge-term $E_{ij}(d_i, d_j)$ satisfies the condition

$$E_{ij}(0,1) + E_{ij}(1,0) \geq E_{ij}(0,0) + E_{ij}(1,1) \ .$$

This is known as the *submodularity* condition, and $E(\mathbf{d})$ is known as a *submodular function*. For details, see Kolmogorov and Zabin (2004). In this form, the condition is not immediately applicable to the stereo problem, since normally there will be more than two possible disparities at each pixel.
2. The concept of submodularity can be generalized to functions defined on large label sets $\mathcal{L}$ (Schlesinger and Flach (2006)). In particular, if $E_{ij}(d_i, d_j) = g(d_i, d_j)$, where g is a convex function, then the energy function is submodular, and can be minimized exactly by max-flow using a graph-construction due to Ishikawa (2003).
3. If $E_{ij}(d_i, d_j) = g(|d_i - d_j|)$, and g is a function satisfying a *triangle inequality*, $g(a) + g(b) \geq g(a + b)$, then the energy function E may be minimized approximately by the α-expansion algorithm, in which the energy is minimized iteratively by repeatedly solving 2-label submodular problems.
4. If none of these conditions holds, then numerous methods give approximate solutions. Chief among them are methods such as Kolmogorov (2006a); Veksler (2012); Ajanthan et al (2015); Hirschmuller (2008).

Semi-global matching (SGM).

Semi-global matching (SGM) Hirschmuller (2008) is a method for rapidly computing a good, but approximate, solution minimizing the MRF. The algorithm figure 9, with the modified forward sweep equation given by (7), will minimize the energy on an MRF consisting of a single scan-line. This algorithm can be easily modified to work on trees as well. A slightly different version of this algorithm is given in figure 12, involving identical sweeps in two different directions. The SGM algorithm extends this approach by carrying out sweeps in both horizontal and vertical direction, as well as diagonal directions across the image; it recommends sweeps in 16 directions, indexed by $r = 1, \ldots, R$. Line 13 of the algorithm in figure 12 is then modified to: $S_i^*(\lambda) = \sum_{r=1}^{R} S_i^r(\lambda) + E_i(\lambda)$.

1: **procedure** MINMARGINAL
2: **for** $\lambda \in \mathcal{L}$ **do** // Initialization
3: $S_1^R(\lambda) = 0$
4: $S_n^L(\lambda) = 0$
5: **for** $i = 2, \ldots, n$ **do** // right sweep
6: **for** $\lambda \in \mathcal{L}$ **do**
7: $S_i^R(\lambda) = \min_{\mu \in \mathcal{L}} \left(S_{i-1}^R(\mu) + E_{i-1}(\mu) + E_{i-1,i}(\mu, \lambda) \right)$
8: **for** $i = n-1, \ldots, 1$ **do** // left sweep
9: **for** $\lambda \in \mathcal{L}$ **do**
10: $S_i^L(\lambda) = \min_{\mu \in \mathcal{L}} \left(S_{i+1}^L(\mu) + E_{i+1}(\mu) + E_{i,i+1}(\lambda, \mu) \right)$
11: **for** $i = 1, \ldots, n$ **do** // Summing
12: **for** $\lambda \in \mathcal{L}$ **do**
13: $S_i^*(\lambda) = S_i^L(\lambda) + S_i^R(\lambda) + E_i(\lambda)$
14: **for** $i = 1, \ldots, n$ **do** // Finding min-marginal
15: $d_i = \operatorname{argmin}_{\lambda \in \mathcal{L}} S_i^*(\lambda)$

Fig. 12 *Sweep-algorithm for finding min-marginals. The value of $S_i^*(\lambda)$ found in this algorithm is equal to the min-marginal, $\min_{\{\mathbf{d} \mid d_i = \lambda\}} E(\mathbf{d})$, in other words, the minimum value of $E(\mathbf{d})$ among those $\mathbf{d}$ with $d_i = \lambda$. Hence, $\min_{\lambda \in \mathcal{L}} S_i^*(\lambda) = \min_{\mathbf{d}} E(\mathbf{d})$ (see line 15). The algorithm gives the optimal label d_i for each i, provided that $\operatorname{argmin}_{\lambda \in \mathcal{L}} S_i^*(\lambda)$ is unique.*

Possible edge functions have been proposed for $f(\lambda, \mu) = E_{ij}(\lambda, \mu)$:

expanded Potts-model	0 if $\lvert\lambda - \mu\rvert \leq T$ and 1 otherwise
linear	$\lvert\lambda - \mu\rvert$
truncated linear	$\min(T, \lvert\lambda - \mu\rvert)$
quadratic	$(\lambda - \mu)^2$
truncated quadratic	$\min(T, (\lambda - \mu)^2)$
Huber	$\lvert\lambda - \mu\rvert^2$ if $\lvert\lambda - \mu\rvert < T$ and $2T\lvert\lambda - \mu\rvert - T^2$ otherwise

Of these, linear, quadratic and Huber can be minimized exactly using the Ishikawa-max-flow method; linear, truncated linear and Huber can be minimized approximately using α-expansion; neither of these methods can be used for the truncated-quadratic or expanded Potts-models.

The quadratic cost function can be ruled out; it over-smooths because of excessively penalizing large disparities.

The truncated quadratic and expanded Potts-model functions have the desirable property that they assign a low penalty to small disparities, while penalizing large disparity differences (such as ones that occur at occlusion boundaries) equally. For that reason, numerous methods have been proposed for minimizing, particular, the truncated-quadratic cost function (Veksler (2012); Ajanthan et al (2015)). The tree-reweighting TRWS-algorithm (Kolmogorov (2006b)) is a good general purpose algorithm for multi-label MRF minimization. In addition, the advantage of truncated-quadratic cost over, for instance linear cost or Huber cost, both of which can be minimized exactly, has not been clearly demonstrated.

7 Deep-Learning based stereo matching

Deep Learning has had a major impact in stereo-vision in many ways. Although it is impossible to discuss all contributions in this area, some recent contributions are outlined in this section.

7.1 Learning deep features or better metrics

One idea to using Deep Learning in stereo vision is to employ a deep network to learn and compute features and cost-function weights which are subsequently fed into an MRF. Both unary and pairwise terms in an MRF (or CRF) have been learned using a deep network approach.

Although multi-layer neural networks were introduced to solve stereo-vision problems many decades ago, the first modern (i.e, post-2012) application of deep neural net for stereo was proposed in 2014 by Zbontar and Lecun Zbontar and LeCun (2015). Their network, called MC-Net, computes feature vectors from left and right patches. The feature vectors are then fed through a further "metric learning" network to generate unary matching terms for a CNN, which

is then solved using the SGM algorithm. Learning of the feature-extraction parameters and metric learning layers is accomplished using positive and negative training patches generated from ground-truth disparity maps.

The errors are back-propagated to the network to refine both feature-extraction and metric learning layers. Figure 14 shows the network architecture of the MC-net. When matching cost is computed, a standard semi-global matching (SGM) algorithm is applied to produce disparity maps. The trained feature-extraction CNN can also be inserted into any other stereo matching method by replacing pixel intensity feature with deep features.

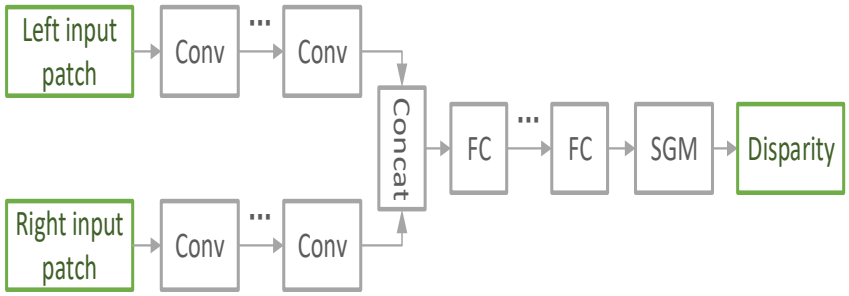

Fig. 13 *MC-CNN: features are extracted from the two images using parallel networks. These are then combined. A further network generates unary terms for a CRF, which is solved using the SGM algorithm.*

A variant of MC-CNN replaces the metric learning network with a simple dot-product layer to directly measure feature similarity, as shown in Fig-14 (Zbontar and LeCun (2015)).

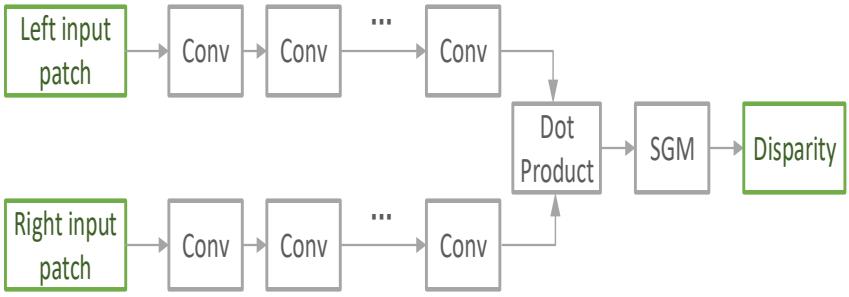

Fig. 14 *A variant of MC-CNN: extracted features from the two images are compared directly, using dot-product, to generate the unary parameters for the CRF.*

Learning both unary and edge terms.

The paper Knbelreiter et al (2017) proposes a hybrid CNN-CRF (Conditional Random Fields) architecture, which learns both unary and edge terms for the CRF in separate parallel networks. The CRF model used is more general than the simple SGM model used in MC-CNN. The overall network structure is shown in Fig-15. By back-propagating gradients of the CRF outputs (in some approximate way) to the two sub-networks, the whole system is able to learn end-to-end.

7.2 Direct disparity regression

As the depth of a deep network increases, its ability to learn and to regress complex functional mapping relationships will generally improve as well. Therefore, it is possible to train a deep network, from a large amount of training data of stereo

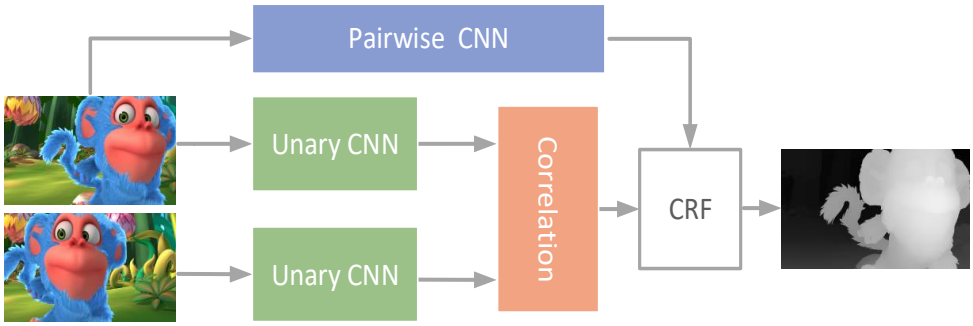

Fig. 15 *Hybrid CNN-CRF Net for stereo Knbelreiter et al (2017): The Unary-CNN computes deep features of the two images at each pixel. The features are compared using a correlation layer which produces the unary cost parameters for a CRF. The pairwise (edge) costs for the CRF are parametrized by edge weights, which are learned by the Pairwise-CNN.*

image pairs and the corresponding ground-truth disparity maps. Once the training process is completed, the network is expected to be able to directly predict (regress) a disparity map from a pair of input stereo images.

Since this class of methods are purely data-driven, they necessarily require large amount of labelled training samples covering the domain of testing images. These methods tend to handle occlusions very well, better than traditional methods, as long as there are sufficient training data exhibiting typical occlusions. Often the obtained disparity maps are visually pleasing, and quantitatively superior to those from traditional methods.

Figure 16 depicts the network architecture of a stereo regression network known as DispNet Mayer et al (2016). This is the first, and a representative deep network for stereo matching following the direct disparity regression approach. In training, the network is trained to produce a disparity field that minimizes the regression loss with respect to the ground-truth disparity map. To train this network, a large-scale computer-graphics simulated stereo dataset was created. Several following-up deep networks with different architecture or training strategies have been proposed in recent years, including Cascade residual learning (CRL) Pang et al (2017) and iResNet Liang et al (2018).

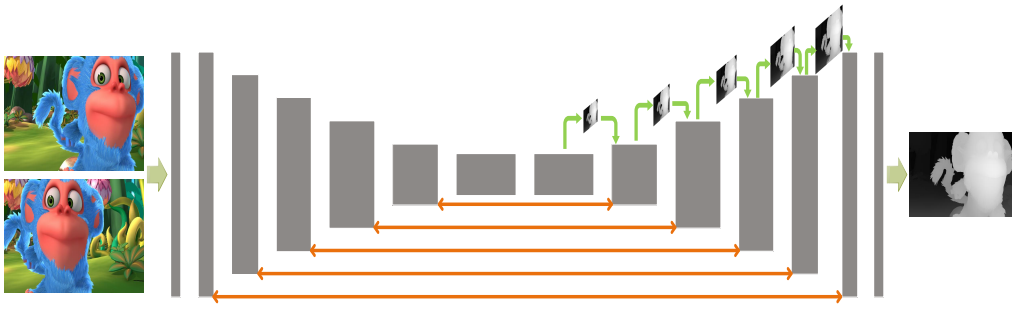

Fig. 16 *Overall network structure of DispNet Mayer et al (2016). The network learns to predict disparities directly from the stereo image pair, using a feed-forward network. Because of the large amount of training data required for effective learning, generated computer-graphic images are used for training.*

7.3 Disparity-aware feature learning

Many algorithms first evaluate all possible matches at all disparity levels and then select the ones that lead to the smallest cost. The left and right images are offset by different disparities before being passed to the CNN. This results in a 3-dimensional array ($H \times W \times D$ (image height $\times$ image width $\times$ disparity range)), called the cost-volume. This cost-volume is then fed into a CNN for cost aggregation, followed by a soft-argmin layer to select the optimal disparities that minimise the global cost.

GC-Net Kendall et al (2017), shown in Figure 17 is a successful deep net of this class though with some slight modification. Instead of forming a cost volume, this network forms a feature-pair volume by exhausting all possible disparity levels. Specifically, a pair of input stereo images are firstly fed into a Siamese (weight-sharing) neural-net to

extract deep features. Then, the obtained features are paired according to different tentative disparity levels and form a feature-pair volume (which is in the format of a 4D tensor). This 4D matrix packs all features into a 4D feature volume with dimensionality $H \times W \times D \times 2F$ for the left-to-right and right-to-left feature volume correspondingly. Here, F represents the feature dimension and the factor 2 accounts for left-to-right and right-to-left features. Next, a multi-layer 3D-CNN is applied to each feature-dimension of the feature-pair volume to learn a cost volume. Finally, the final disparity values are regressed from the cost-volume using a differentiable soft-argmin operation, which allows end-to-end training to sub-pixel accuracy.

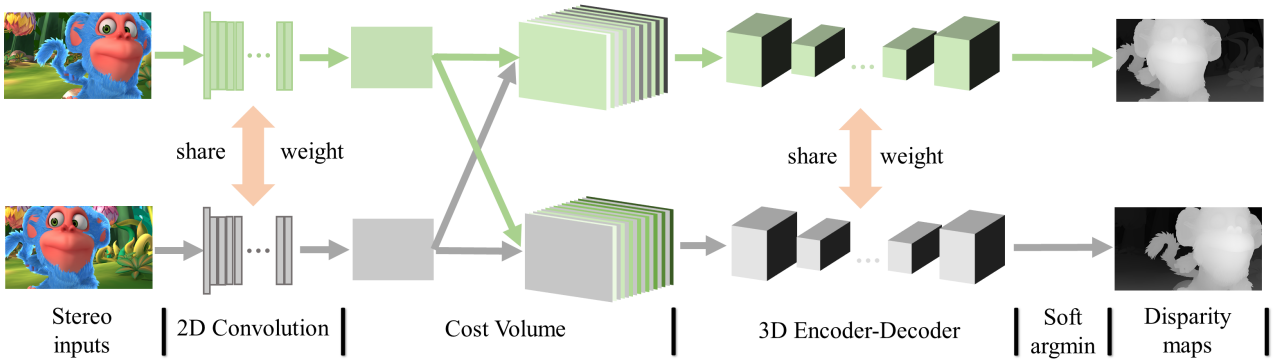

Fig. 17 *Network structure of GC-Net Kendall et al (2017). Features are extracted from both images, and are then combined at varying disparity offsets to create a 4-dimensional cost volume of features. The disparities are extracted from the cost volumes using an encoder-decoder network, followed by softmin to find the minimal-cost disparities.*

The difference between this class of method and the direct disparity regression methods is the construction of feature volume or cost-volume and the soft-argmin operation, both of which exploit the geometrical nature of the stereo problem.

PSMnet (Chang and Chen (2018)) further extended the idea of GC-Net by using a multi-scale spatial pyramid pooling module in the feature extraction stage and replacing the 3D-CNN with a stacked hourglass architecture.

7.4 Self-supervised deep stereo matching

All the aforementioned deep stereo matching methods directly use ground-truth disparity maps as supervision. Their performance relies critically on the availability of a large number of training stereo images with corresponding ground-truth disparity maps. However, getting accurate disparity maps is a laborious and expensive task (e.g., KITTI, Middlebury stereo). In contrast, most traditional stereo matching methods do not need ground-truth disparity maps in training. Instead, they find the best result simply by minimizing a predefined global energy function (such as MRF/CRF energy function). This suggests an alternative approach which simply uses the energy function as the loss function to train a deep neural network. In this context, the deep network acts as a mathematical optimizer.

In one approach (sSMNet, Zhong et al (2017)) a disparity-prediction network takes a pair of stereo images I^L and I^R as input, and outputs a disparity map $d_L = g(I^R, I^L)$ w.r.t. the left image The right image is then warped to using this resulting disparity map to the left viewpoint to obtain a warped new right-eye image $I^{L'}(u, v) = I^R(u - d_L, v)$. Similarly, a disparity map wrt. the right image is also predicted, and used to subsequently warp the generated $I^{L'}$ to obtain another version of warped left image $I^{L''}$. The photometric error between $I^{L''}$ and the original right image I^L can be used as the loss function which is minimized through training, using back-propagation. In actual implementation, the same operation is applied to both left-to-right, and right-to-left directions to encourage left-right symmetry. Note in the training stage, no external ground-truth depth map is needed, and any pair of input stereo images can be used for supervision. The network is trained in an end-to-end fashion. Once the training is complete, the disparity-prediction network (i.e. $g(I^R, I^L)$) can be applied to new (unseen) stereo images, outputting disparity maps directly. Despite not seeing any ground-truth disparity map, the network's depth estimation accuracy is on par with that of fully-supervised deep stereo methods. This approach is extended to matching stereo *video sequences*, by adding a recurrent unit (using convolutional LSTMs) to leverage temporal continuity in videos Zhong et al (2018).

References

Ajanthan T, Hartley R, Salzmann M, Li H (2015) Iteratively reweighted graph cut for multi-label MRFs with non-convex priors. CVPR

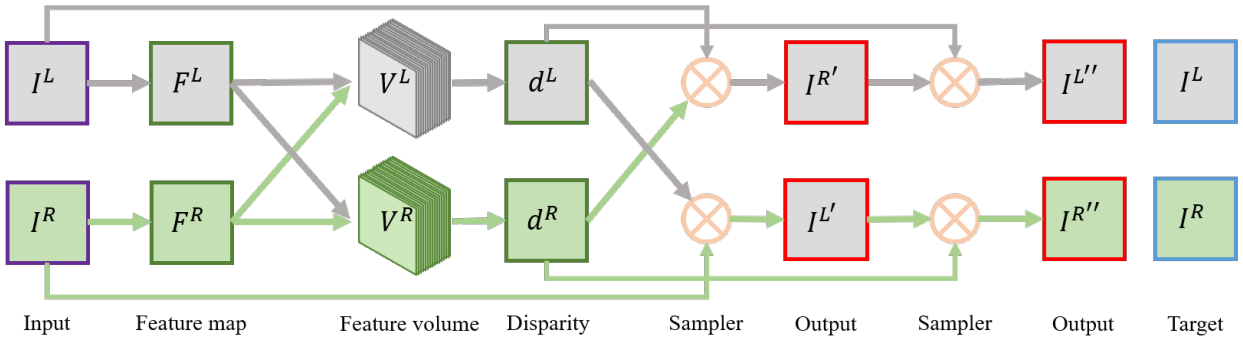

Fig. 18 *Overall network structure of SsSMnet. The network first generates disparity maps directly from the left and right input images, in a way similar to figure 17. The disparity maps are then used to warp the input images back and forth. They are then compared with the original images to compute a training loss.*

Belhumeur PN, Mumford D (1992) A bayesian treatment of the stereo correspondence problem using half-occluded regions. In: Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp 506–512, DOI 10.1109/CVPR.1992.223143

Besag J (1974) Spatial interaction and the statistical analysis of lattice systems. Journal of the Royal Statistical Society

Burt P, Adelson E (1983) The laplacian pyramid as a compact image code. IEEE Transactions on Communications 31(4):532–540

Chang JR, Chen YS (2018) Pyramid stereo matching network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 5410–5418

Hartley R (1999) Theory and practice of projective rectification. International Journal on Computer Vision 35(2):115–127, DOI doi:10.1023/A:1008115206617, URL http://dx.doi.org/10.1023/A:1008115206617

Hirschmuller H (2008) Stereo processing by semiglobal matching and mutual information. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(2):328–341, DOI 10.1109/TPAMI.2007.1166

Ishikawa H (2003) Exact optimization for Markov random fields with convex priors. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(10):1333–1336

Kendall A, Martirosyan H, Dasgupta S, Henry P, Kennedy R, Bachrach A, Bry A (2017) End-to-end learning of geometry and context for deep stereo regression. In: Proc. IEEE Int. Conf. Comp. Vis.

Kim J, Kolmogorov, Zabih (2003) Visual correspondence using energy minimization and mutual information. In: Proceedings Ninth IEEE International Conference on Computer Vision, pp 1033–1040 vol.2, DOI 10.1109/ICCV.2003.1238463

Knbelreiter P, Reinbacher C, Shekhovtsov A, Pock T (2017) End-to-End Training of Hybrid CNN-CRF Models for Stereo. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn.

Kolmogorov V (2006a) Convergent tree-reweighted message passing for energy minimization. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(10):1568 – 1583, DOI doi:10.1109/TPAMI.2006.200

Kolmogorov V (2006b) Convergent tree-reweighted message passing for energy minimization. PAMI

Kolmogorov V, Zabin R (2004) What energy functions can be minimized via graph cuts? PAMI

Liang Z, Feng Y, Guo Y, Liu H, Chen W, Qiao L, Zhou L, Zhang J (2018) Learning for disparity estimation through feature constancy. In: Computer Vision and Pattern Recognition

Longuet-Higgins HC (1981) A computer algorithm for reconstructing a scene from two projections. Nature 293:133–135

Mayer N, Ilg E, Husser P, Fischer P, Cremers D, Dosovitskiy A, Brox T (2016) A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., URL http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16/

Pang J, Sun W, Ren JS, Yang C, Yan Q (2017) Cascade residual learning: A two-stage convolutional neural network for stereo matching. In: International Conf. on Computer Vision - Workshop on Geometry Meets Deep Learning (ICCVW)

Quam LH (1987) Hierarchical warp stereo. In: Fischler MA, Firschein O (eds) Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 80–86, URL http://dl.acm.org/citation.cfm?id=33517.33526

Schlesinger D, Flach B (2006) Transforming an arbitrary minsum problem into a binary one. TU, Fak. Informatik

Veksler O (2012) Multi-label moves for MRFs with truncated convex priors. IJCV

Zbontar J, LeCun Y (2015) Computing the stereo matching cost with a convolutional neural network. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn.

Zhong Y, Dai Y, Li H (2017) Self-supervised learning for stereo matching with self-improving ability. In: arXiv:1709.00930

Zhong Y, Dai Y, Li H (2018) Open-world stereo video matching with deep rnn. In: Proc. Eur. Conf. Comp. Vis.