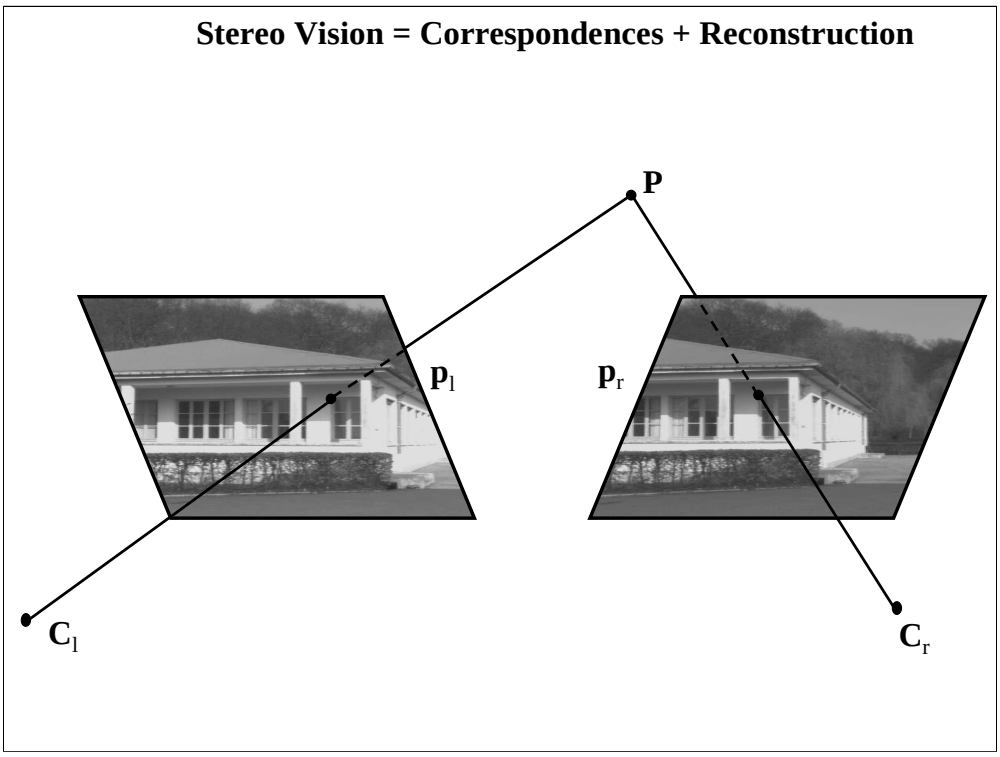
Stereo

Chap. 11



Stereo vision (or "stereopsis") is the process of recovering the three-dimensional location of points in the scene from their projections in images. More precisely, if we have two images I_l and I_r (left and right from the left and right eyes), given a pixel $\mathbf{p}_l$ in the left image and the corresponding pixel $\mathbf{p}_r$ in the right image, then the coordinates (X,Y,Z) of the corresponding point in space is computed.

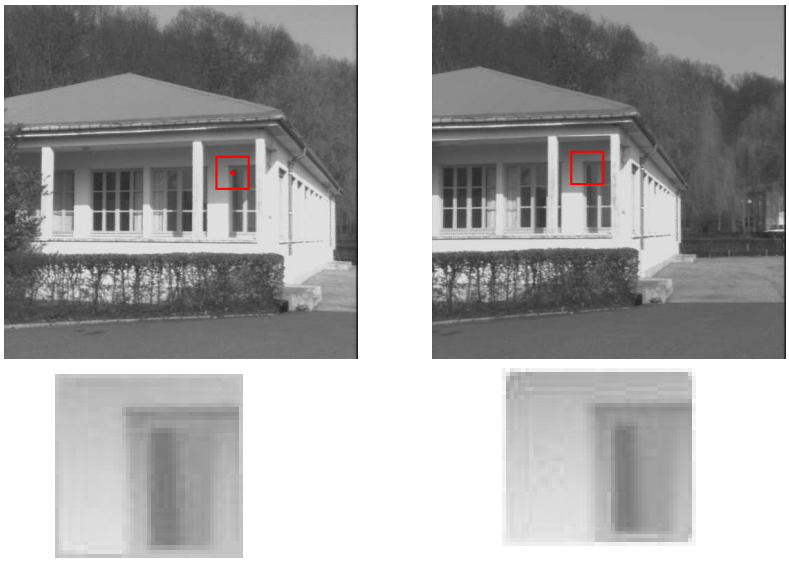
Geometrically, given $\mathbf{p}_l$, we know that the point **P** lies on the line $\mathbf{L}_l$ joining $\mathbf{p}_l$ and the left optical center $\mathbf{C}_l$ (this line is the *viewing ray*), although we don't know the distance along this line. Similarily, we know that **P** lies along a line $\mathbf{L}_r$ joining $\mathbf{p}_r$ and **P**. Assuming we know exactly the parameters of the cameras (intrinsic and extrinsic), we can explicitly compute the parameters of $\mathbf{L}_l$ and $\mathbf{L}_r$. Therefore, we can compute the intersection of the two lines, which is the point **P.** This procedure is called *triangulation*.

Stereovision therefore involves two problems:

• *Correspondence*: Given a point $\mathbf{p}_l$ in one image, find the corresponding point in the other image.

• *Reconstruction*: Given a correspondence $(\mathbf{p}_l,\mathbf{p}_r)$, compute the 3-D coordinates of the corresponding point in space, **P**.

Finding Correspondences



$W(\mathbf{p}_l)$ $W(\mathbf{p}_r)$

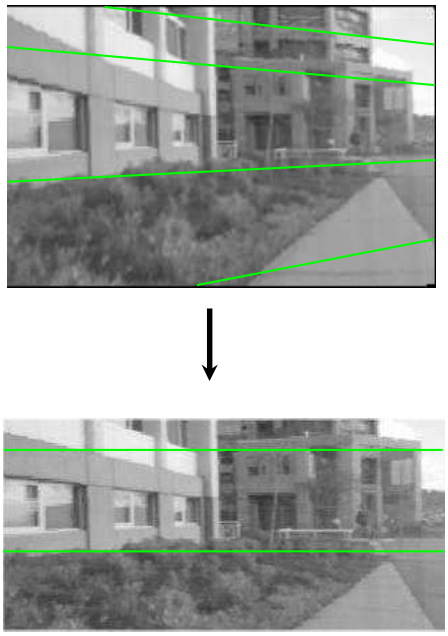
Given $\mathbf{p}_l$, finding the corresponding point $\mathbf{p}_r$ involves searching the right image for the location $\mathbf{p}_r$ such that the right image around $\mathbf{p}_r$ "looks like" the left image around $\mathbf{p}_l$. More formally, we can take a small window W around $\mathbf{p}_l$ and compare it with the right image at all possible locations. The position $\mathbf{p}_r$ that gives the best match is reported.

The fundamental operation, therefore, is to compare the pixels in a window $W(\mathbf{p}_l)$ with the pixels in a window $W(\mathbf{p}_r)$. We have seen before standard ways of comparing windows:

- Sum of Absolute Differences
- Sum of Squared Differences
- Normalized Correlation

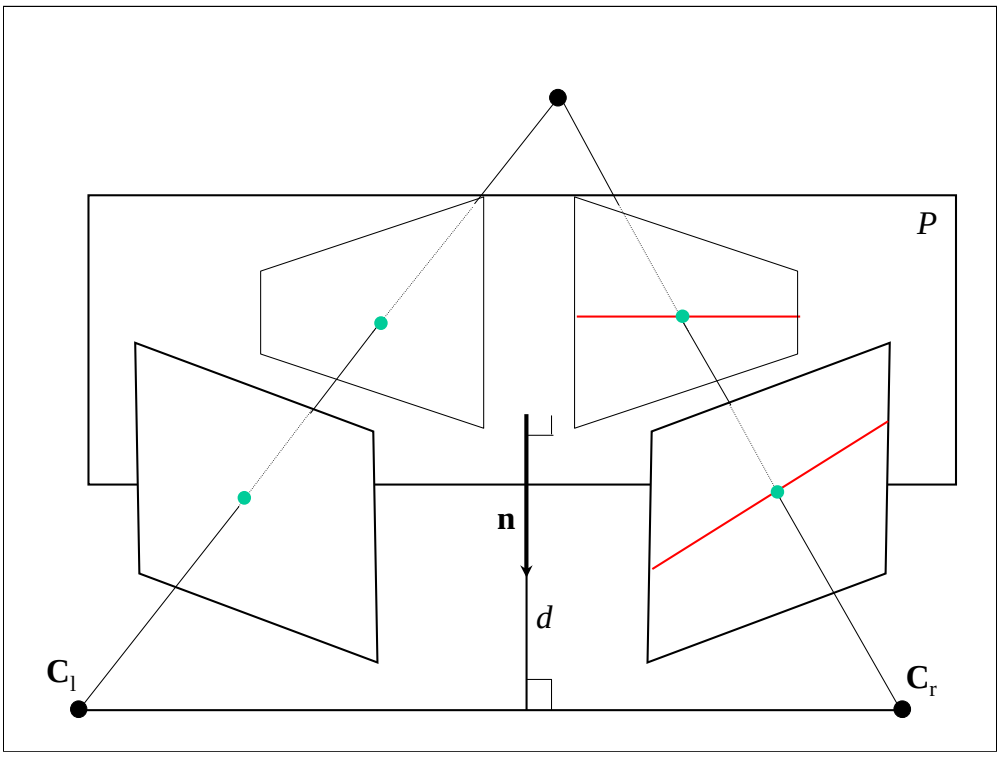
This is a simple algorithm in principle: for each $\mathbf{p}_l$, scan the right image to find the $\mathbf{p}_r$ such that $W(\mathbf{p}_l)$ is most similar to $W(\mathbf{p}_r)$ using one of the matching measures above.

Rectification



Searching along epipolar lines at arbitrary orientation is intuitively expensive. It would be nice to be able to always search along the rows of the right image. Fortunately, given the epipolar geometry of the stereo pair, there exists in general a transformation that maps the images into a pair of images with the epipolar lines parallel to the rows of the image. This transformation is called *rectification*. Images are almost always rectified before searching for correspondences in order to simplify the search.

The exception is when the epipole is inside one of the images. In that case, rectification is not possible.



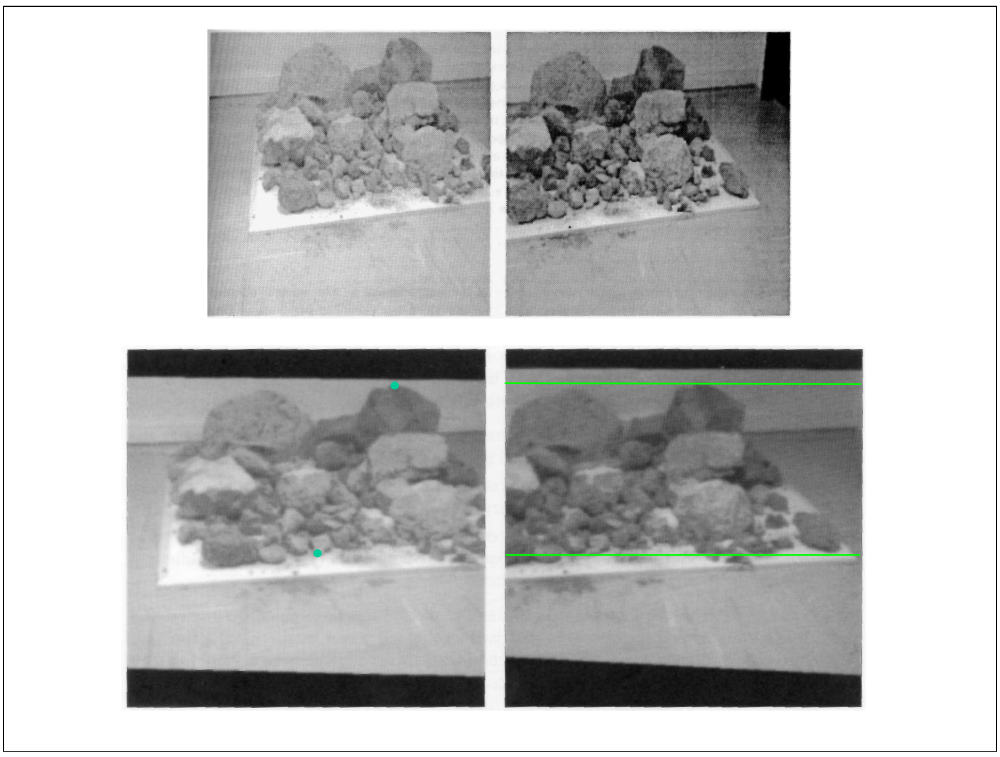
We know that, given a plane **P** in space, there exists two homographies $\mathbf{H}_l$ and $\mathbf{H}_r$ that map each image plane onto **P**. That is, if $\mathbf{p}_l$ is a point in the left image, then the corresponding point in **P** is **Hp** (in homogeneous coordinates). If we map both images to a common plane **P** such that P is parallel to the line $\mathbf{C}_l\mathbf{C}_r$, then the pair of virtual (rectified) images is such that the epipolar lines are parallel. With proper choice of the coordinate system, the epipolar lines are parallel to the rows of the image.

The algorithm for rectification is then:

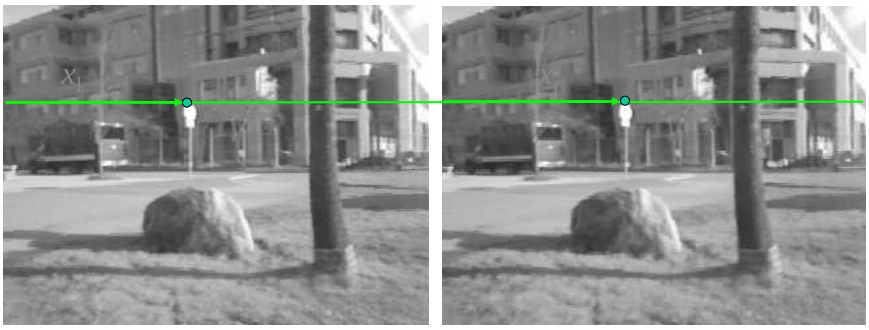
• Select a plane **P** parallel to $\mathbf{C}_r\mathbf{C}_l$

• Define the left and right image coordinate systems on **P**

• Construct the rectification matrices $\mathbf{H}_l$ and $\mathbf{H}_r$ from **P** and the virtual image's coordinate systems.



Disparity

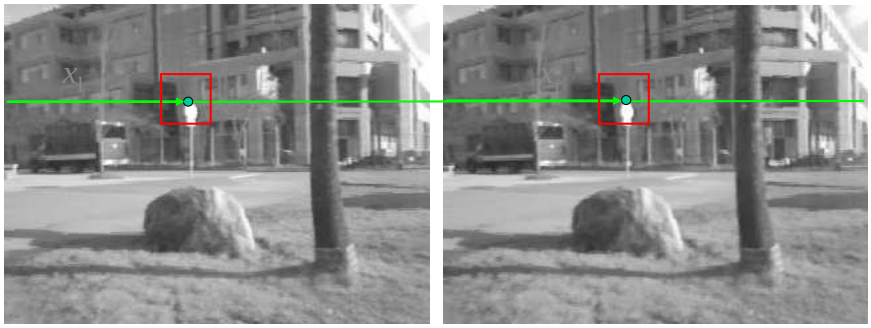


$$d = x_l - x_r$$

Assuming that images are rectified to simplify things, given two corresponding points $\mathbf{p}_l$ and $\mathbf{p}_r$, the difference of their coordinates along the epipolar line x_l-x_r is called the disparity d. The disparity is the quantity that is directly measured from the correspondence.

It turns out that the position of the corresponding 3-D point **P** can be computed from $\mathbf{p}_l$ and d, assuming that the camera parameters are known.

Disparity



$$d = x_l - x_r$$

$$\underset{d}{Min} \sum_{x,y \in W} \psi(I_l(x,y), I_r(x-d,y))$$

Stereo Matching:

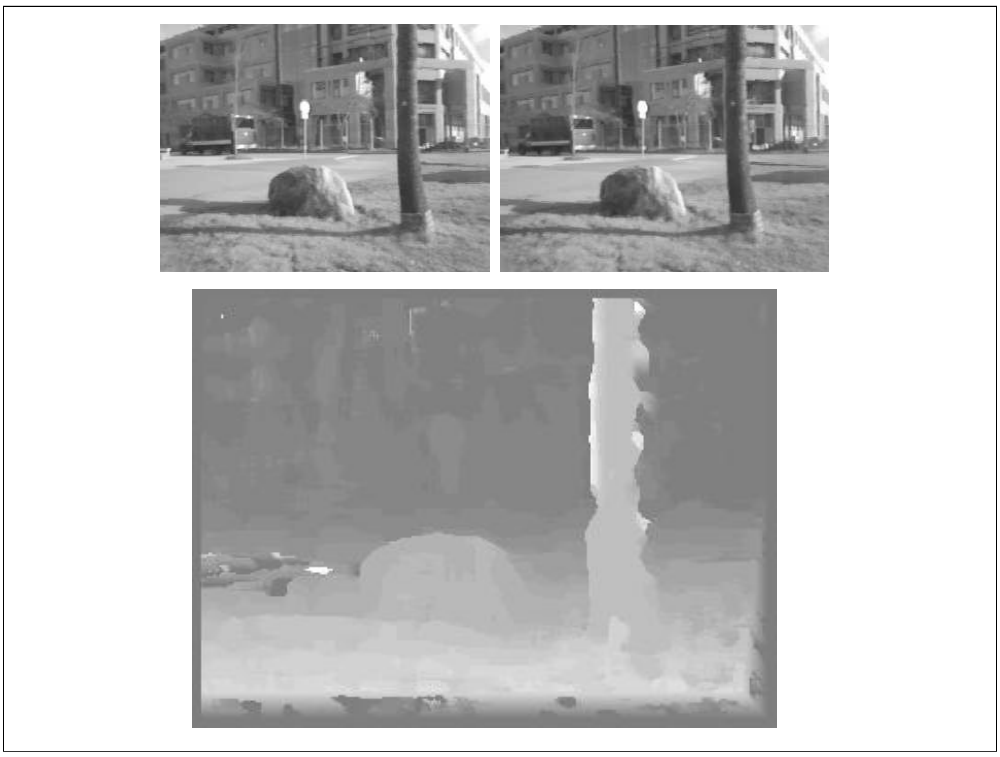
We assume from now on that the images are rectified. That is, the epipolar lines are parallel to the rows of the images. We will denote the horizontal coordinate by x and the vertical coordinate by y. With that geometry, given a pixel at coordinate x_l, the problem of stereo matching is to find the coordinate x_r of the corresponding pixel in the same row in the right image. The difference $d = x_r - x_l$ is called the disparity at that pixel.

The rest of the notes focuses on various ways of recovering d.

The basic matching approach is to take a window W centered at the left pixel, translate that window by d and compare the intensity values in W in the left image and W translated in the right image. The comparison metric typically has the form:

$$S(d) = \sum_{x,y \in W} \psi(I_l(x,y), I_r(x-d,y))$$

The function ψ measures the difference between the pixel values. Possible choices for ψ are shown next page.



The disparity is computed at every pixel in the image and for every possible disparity. The output is a disparity image. Those image can be interpreted as disparity being the inverse of the depth (larger disparity for points closer to the cameras.)

In terms of implementation, we basically have five loops:

```
for x=1:xsize
  for y=1:ysize
    for d=dmin:dmax
      Sbest = max;
      S(d) = 0;
      for u = x-w:x+w
        for v = y-w:y+w
          S(d) = S(d) + y(d)
      if(S(d) < Sbest)
        Sbest = S(d)
        dbest(x,y) = d
```

Note that the loops in x,y and d can be inverted. Also, this looks like an enormous amount of computation. It turns out that this can be done efficiently by re-using partial results.

Matching Functions

SSD:

$$\psi(I_l(x,y), I_r(x+d,y)) = (I_l(x,y) - I_r(x-d,y))^2$$

SAD:

$$\psi(I_l(x,y), I_r(x+d,y)) = \left|I_l(x,y) - I_r(x-d,y)\right|$$

Correlation:

$$\psi(I_l(x,y), I_r(x+d,y)) = I_l(x,y).I_r(x-d,y)$$

Normalized Correlation:

$$\psi(I_l(x,y), I_r(x+d,y)) = \frac{I_l(x,y).I_r(x-d,y) - \bar{I}_l\bar{I}_r}{\sigma_l\sigma_r(d)}$$

Matching Functions:

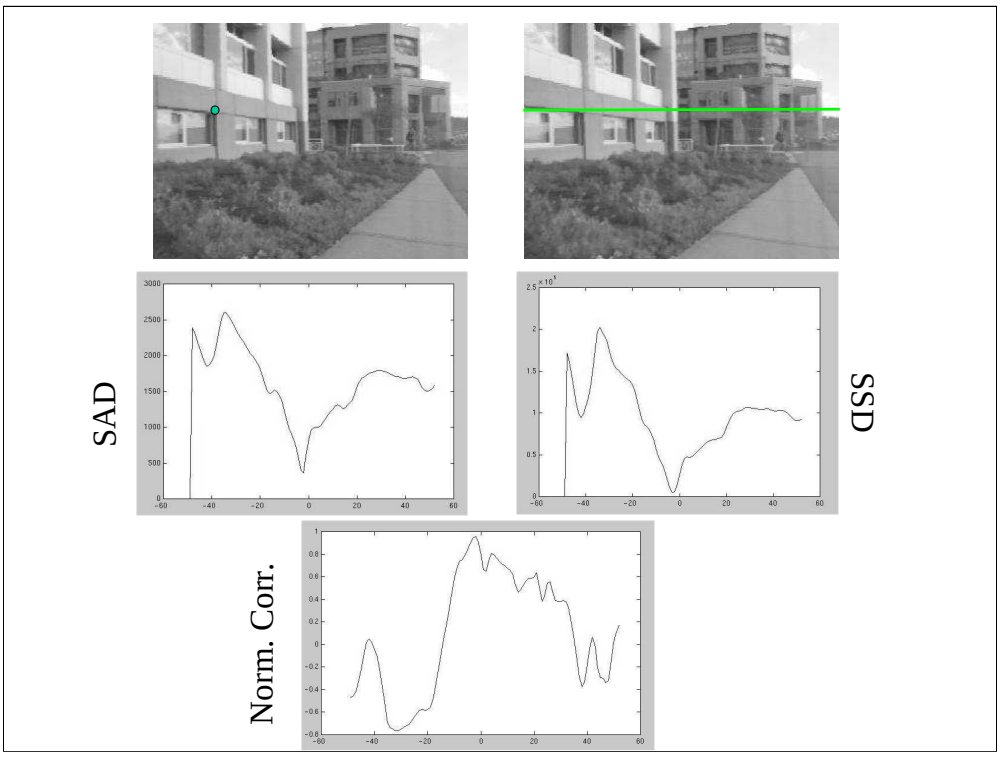
Four standard choices are used for the matching function:

SSD: ψ is the squared differences between pixel values. The SSD has nice mathematical properties and is easy to compute. The SSD tends to magnify pixel errors because of the quadratic function of the difference. This can caused problems in noisy images.

SAD: ψ is the absolute difference between pixel values. It is better behaved than the SSD because it does not increase quadratically with pixel error. Depending on the architecture it may be faster.

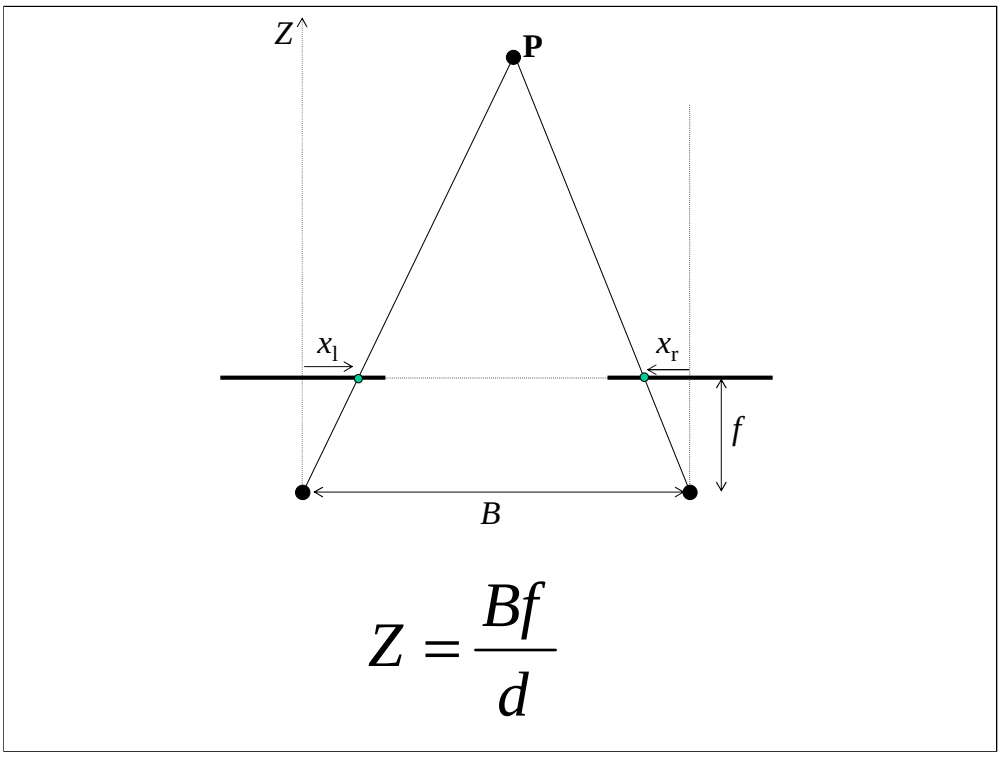
Correlation: ψ is the product of the intensities. The maximum of correlation corresponds to the best match. Correlation is roughly equivalent to SSD. It really should be used with normalization.

Normalized Correlation: Same as correlation but the intensity values are first centered by substracting the mean intensity of the correlation window (not of the image) and normalizing by the standard deviation of the intensity in the window. The normalized correlation has the advantage that it is always between -1 and +1,with +1 for a perfect match. It is obviously more expensive to compute.



The curves show the value of the matching function for one pixel in the left image. The graphs show the disparity on the horizontal axis, and the value of the matching function (SAD, SSD, or Normalized Correlation) on the vertical axis.

Those curves illustrate the ideal case in which we have a single peak in the curve which is sharp enough for the location of the optimal disparity to be determined accurately.



The next step in stereo is to recover the 3-D coordinates of the scene points corresponding to pairs of matching pixels. As a first step, we can simplify the geometry by assuming that the two cameras have parallel image planes. Consider a point **P** of coordinates *X*,*Y*, *Z*. **P** is projected to $\mathbf{p}_l$ and $\mathbf{p}_r$ in the two images. The focal length of the cameras is denoted by *f*, and the distance between the optical centers (the baseline) is denoted by *B*.

Looking at the diagram, we see that the triangles ($\mathbf{C}_r$,$\mathbf{C}_l$,**M**) and ($\mathbf{p}_r$,$\mathbf{p}_l$,**P**) are similar triangles. Therefore, the ratios of their height to base are equal:

$$\frac{Z}{B} = \frac{Z - f}{B - x_l + x_r}$$

Moving *Z* to one side of the equality, we get:

$$Z = \frac{Bf}{d}$$

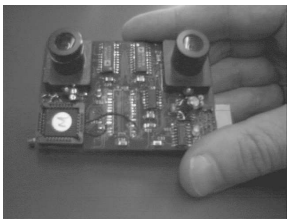
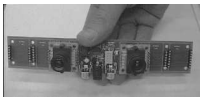
This relation is the fundamental relation of stereo. It basically states that the depth is inversely proportional to the disparity.

Once we know *Z*, the other two coordinates are derived using the standard perspective equations:

$$X = \frac{x_l Z}{f}$$

$$Y = \frac{y_l Z}{f}$$

Example SRI

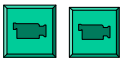




Input Image Pair

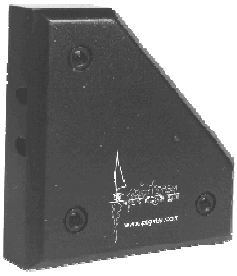
Depth Image

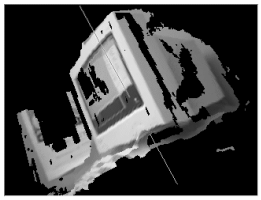
3-D Reconstruction

- Very small package
- Different configurations to meet application needs
- Computation on *conventional* PCs, or
- Embedded computing (DSP, FPGA) for very compact packaging and low power
- *Real-time* processing (e.g. 12-90 depth images/second, depending on size)

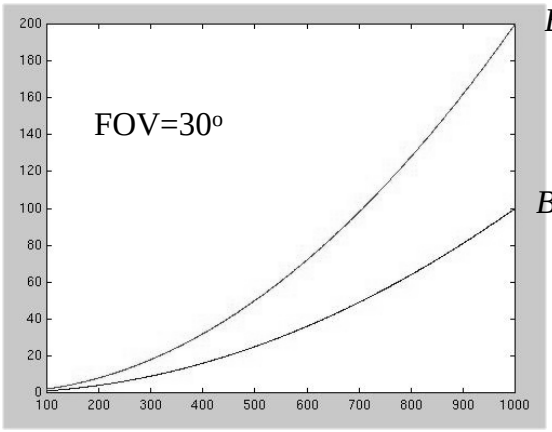


Example: Point Grey Systems





- Three cameras for increased depth precision
- Computation on standard PC
- Real-time (16 full-size depth images/second)
- Color cameras



$$\frac{Z^2}{Bf}$$

An important question about recovering 3-D coordinates is: How accurately can those coordinates be computed? We can answer that through a simple error analysis: Consider a matching pair of disparity d corresponding to a depth Z. If we make an error of one pixel (d+1 instead of d), the depth becomes Z'. We want to evaluate ΔZ, the error in depth due to the error in disparity. Taking the derivative of Z as a function of d, we get:

$$\left|\frac{\Delta Z}{\Delta d}\right| = \frac{Bf}{d^2} = \frac{Z^2}{Bf}$$

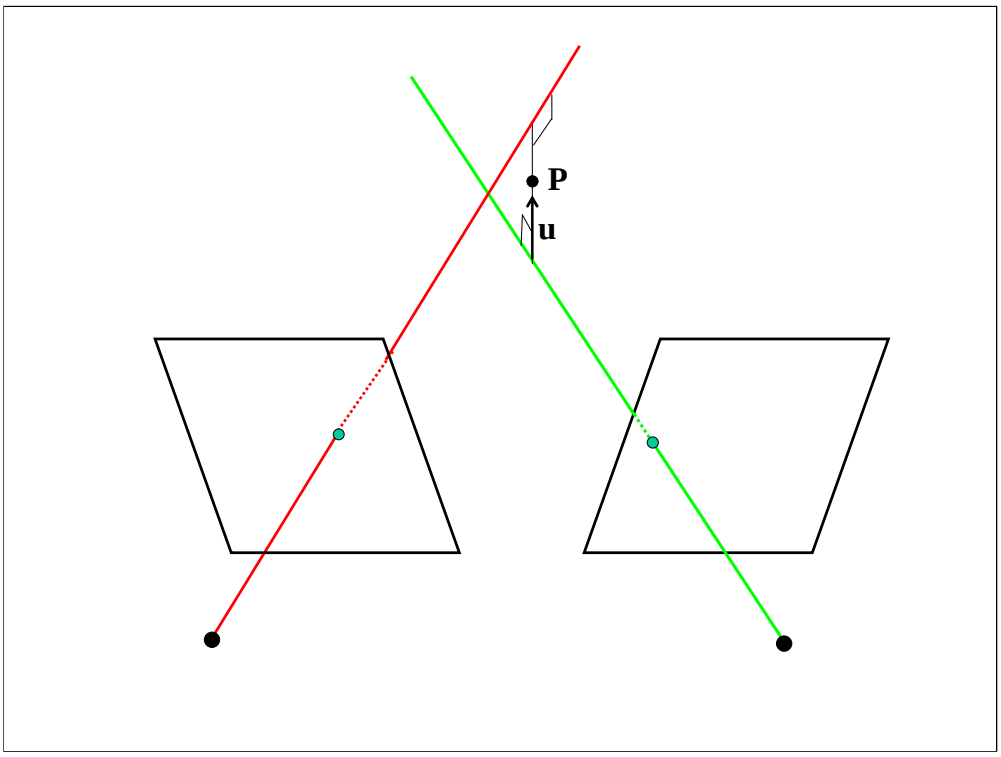
The final relation illustrates the fundamental relation between baseline, focal length and accuracy of stereo reconstruction. For an error of 1 pixel on disparity, we get an error in Z of:

$$\frac{Z^2}{Bf}$$

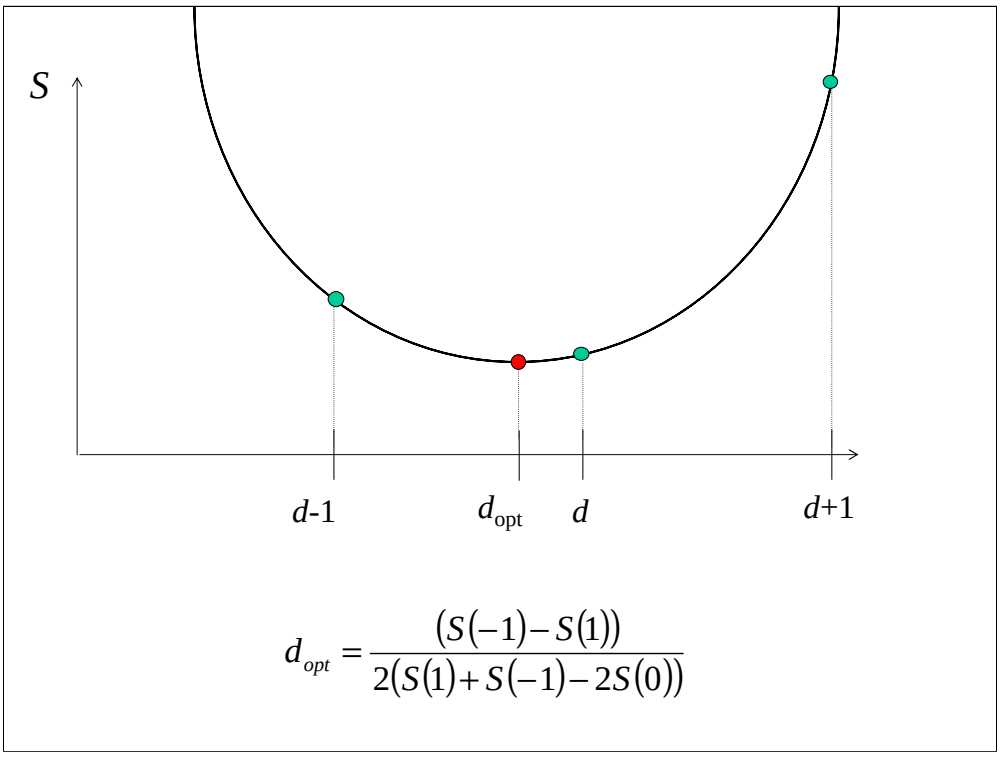
Depth: The resolution of the stereo reconstruction decreases quadratically with depth. This implies severe limitation on the applicability of stereo. Note that, if we assume sub-pixel disparity interpolation with a resolution Δd, the depth resolution becomes: $\frac{Z^2}{Bf}\Delta d$ but it still remains quadratic in depth.

Baseline: The resolution improves as the baseline increases. We would be tempted to always use a baseline as large as possible. However, the matching becomes increasingly difficult as the baseline increases (for a given depth) because of the increasing amount of distortion between the left and the right images. This is similar to the usual detection/localization trade-off.

Focal length: The resolution improves with focal length. Intuitively, this is due to the fact that, for a given image size, the density of pixels in the image plane increases as f increases. Therefore, the disparity resolution is higher. Although this was derived using a simplified model of the stereo system, the three principles hold for any stereo system.



The previous discussion assumes that the viewing rays from the left and right cameras intersect exactly. In fact, that is not usually the case because of small errors in calibration. In that case the two viewing rays pass close to each other but do not exactly intersect. The point **P** is reconstructed as the point that is the closest to both lines.



Sub-Pixel Disparity:

The disparity is computed by moving a window one pixel at a time. As a result, the disparity is known only up to one pixel. This limitation on the resolution of the disparity translates into a severe limitation on the accuracy of the recovered 3-D coordinates. One effective way to get this problem is to recover the disparity at a finer resolution by interpolating between the pixel disparities using quadratic interpolation.

Suppose that the best disparity at a pixel is obtained at d_o with a matching value (for example SSD) of $S(d_o)$. We can obtain a second order approximation of the (unknown) function $S(d)$ by approximating S by a parabola. At the position d_{opt} corresponding to the bottom of the parabola, we have $S(d_{opt}) <= S(d_o)$. Therefore, d_{opt} is a better estimate of the disparity than d_o.

The question remains as to how to find this approximating parabola. Let us first translate all the disparity values so that $d_o = 0$. The equation of a general parabola is: $S(d) = ad^2 + bd + c$. To recover the 3 parameters of the parabola we need 3 equations which we obtain by writing that the parabola passes through the point of disparity 0, -1, and +1:

$S(0) = c$ $S(1) = a + b + c$ $S(-1) = a - b + c$

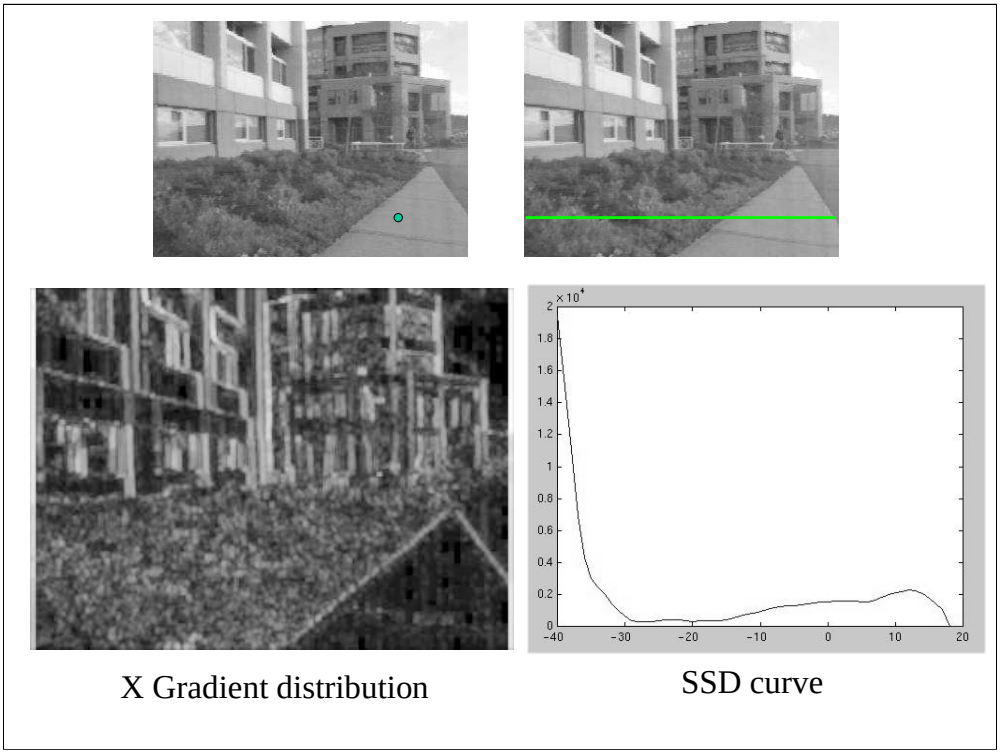
Solving this, we obtain: $c = S(0)$ $a = (S(1) + S(-1) - 2S(0))/2$ $b = (S(1) - S(-1))/2$

The bottom of the parabola is obtained at d_{opt} such that $S'(d) = 2ad + b = 0$. Therefore, the optimal disparity is obtained as:

$$d_{opt} = \frac{(S(-1) - S(1))}{2(S(1) + S(-1) - 2S(0))}$$

In practice, estimating disparity with a fraction of a pixel resolution is possible using this interpolation approach.

Note that the denominator is close to 0 if the function S is mostly flat, in which case there is no valid estimate of the disparity.



X Gradient distribution

SSD curve

Matching Confidence:

Stereo matching assumes that there is enough information in the images to discriminate between different positions of the matching window. That is not the case in regions of the image in which the intensity is nearly constant. In those regions, there is not a single minimum of the matching function and the disparity cannot be computed.

There are two ways of detecting that the disparity estimate is unreliable. The first approach is to look at the curvature of the matching function near its minimum. This curvature can be estimated in a variety of ways, for example, by using the standard deviation of the matching values for a few values of the disparity around the minimum. It can also be estimated by using the curvature of the fitted parabola (see the discussion below on sub-pixel interpolation.)

A second approach is to first apply a filter to the input images, such that the output of the filter at a pixel is high if there is enough information in its neighborhood. The main criterion is that there is enough variation of intensity in the direction of matching, ie., along the *x* direction. This can be done by compute the gradient of the image in the x direction, I_x and computing the local average of its squared magnitude: ΣI_x^2. This quantity can then be used as a measure of reliability of matching (confidence measure.)

Lighting adjusted Lighting corrupted

Lighting Issues:

A problem mentioned before is that the lighting conditions maybe substantially different between the left and right image. Because, for example, of different exposures or different settings of the camera. The problem is that, because ψ measures directly the difference in pixel values, its value will be corrupted. A simple experiment in which the intensities in the left image are corrupted by a smoothly varying function shows that the SSD curve is substantially corrupted.

A simple model would be that $I_l(x,y) = aI^o_l(x,y) + bx + c$, where I_l is the actual image, and I_l^o is the "ideal" image (ie, the one that matches the right image $I_l(x,y) = I_r(x-d,y)$). This amounts to applying a scaling followed by a linear ramp across the image. Using the SSD, we have:

$\psi(d) = (I_l(x,y) - I_r(x-d,y)^2 = ((a-1)I^o_l + bx+c)^2$

This shows that, even if the ideal images match perfectly, the matching function on the actual images can be arbitrarily far from 0.

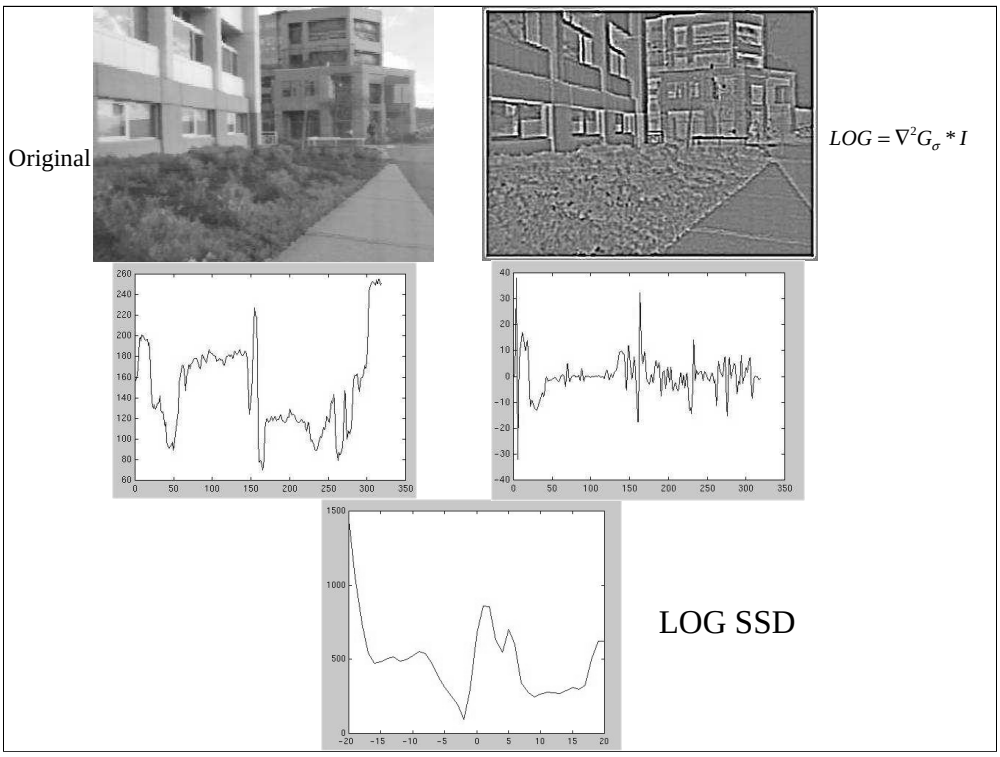
The normalized correlation reduces this problem by eliminating the terms *a* and *c*. However, it is still corrupted by the ramping factor *b*.

Lighting Conditions (Photometric Variations)

$W(\mathbf{P}_l)$ $W(\mathbf{P}_r)$

Comparing windows using SSD, etc. assumes that the pixels in $W(\mathbf{P}_l)$ and $W(\mathbf{P}_r)$ have (almost) the same values. In fact, the pixel values may be very different because the cameras and framegrabbers may have different aperture, gain settings, etc. Also, the intensity may change between images because they are looking from different viewpoints. The result is that, even if $\mathbf{P}_l$ and $\mathbf{P}_r$ are corresponding points, the intensity values of the pixels around them may be different.

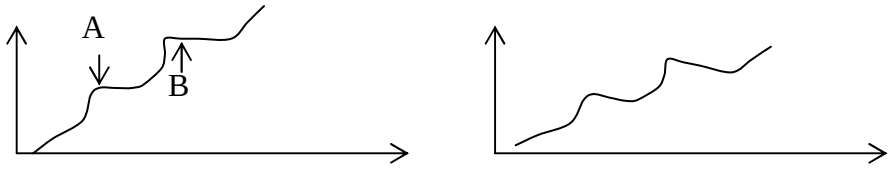
One way to address this problem is to use something like normalized correlation instead of straight comparison between pixels. Unfortunately, this is substantially more expensive. A better approach is to pre-process the images in order to reduce the photometric variations (using a Laplacian filter will be the preferred approach.)



Laplacian of Gaussian (LOG):

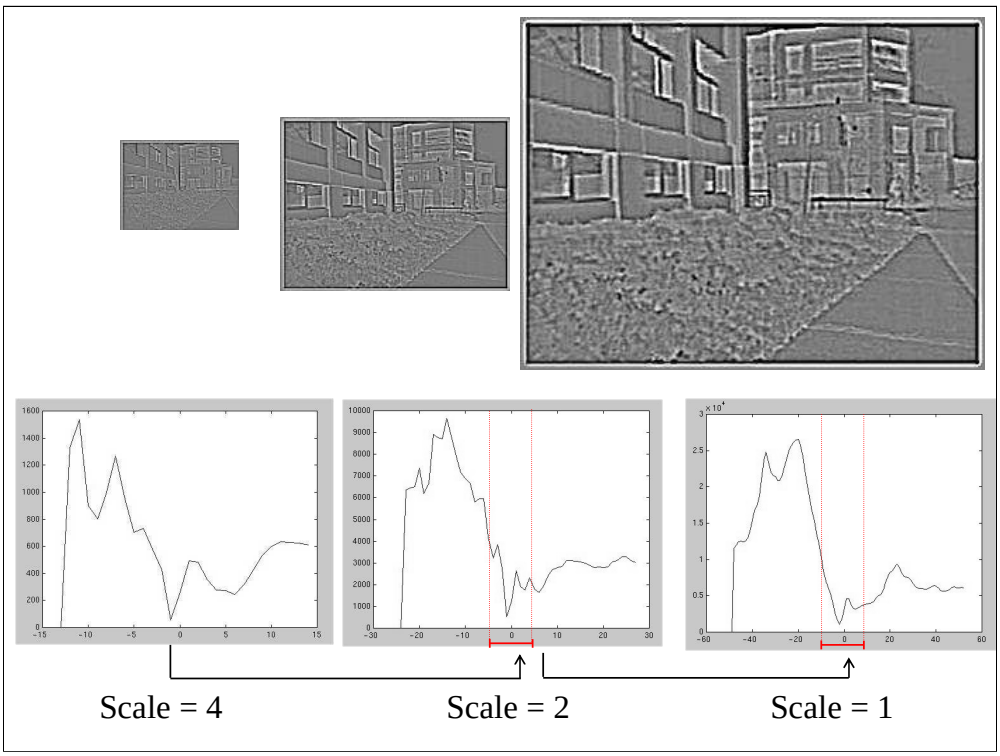
A better way of eliminating smooth variations across the images is to take the second derivative, because the second derivative of a linear function of the form $bx+c$ is 0, and therefore eliminates this component.

More generally, smoothly varying parts of the image do not carry much information for matching. The useful information is contained in higher-frequency variations of intensity. For example, in those graphs, only the pixels at the indicated locations carry useful information for matching. The smoothly varying part of the profile constitute the majority of the points and would confuse the matcher. The information that we really want to match is the fact that the intensity bumps at A and B have similar shape.



So we want to eliminate the slowly-varying parts of the image (low-frequency), and this can be done by using a second derivatives. Also, as usual, we want to eliminate the high-frequency components which correspond to noise in the image, which suggests blurring with a Gaussian filter. The combination of the two suggests the use of the Laplacian of Gaussian (LOG) previously defined in the context of edge detection. This filter is technically a band-pass filter (removes both high and low frequencies.) In practice, the images are first convoluted with the LOG filter:

Any of the definition of ∇ can be used directly on the Laplacian image. $\nabla^2 G_\sigma * I_l$ and $\nabla^2 G_\sigma * I_r$



Laplacian Pyramids:

The effect of σ is similar to the trade-off in edge detection:

Large σ: A lot of the noise is eliminated, the matching curves are smoother, and matching is less ambiguous. Therefore, we have better performance in terms of *detection* of matches.

Small σ: The image details are not blurred as much so the *localization* is better.

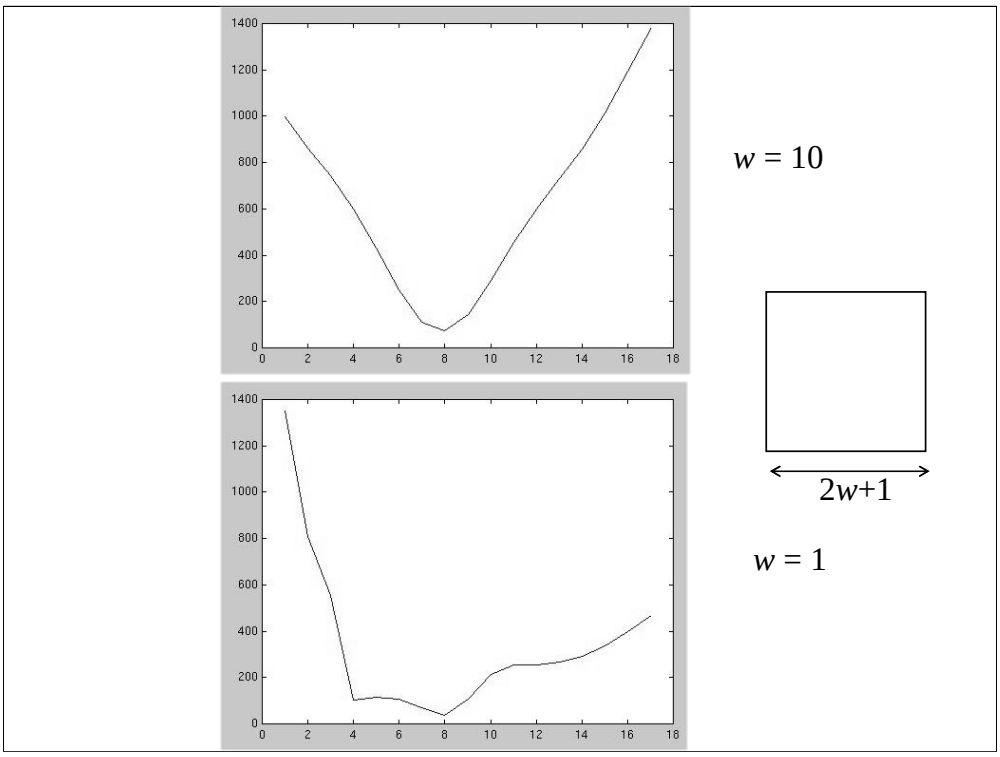
Just like for Gaussian smoothing, it is beneficial to perform stereo matching at different resolutions (ie., different values of σ.) The algorithm would look like:

- Find the best disparity $d_{\sigma k}(x,y)$ of every pixel at the current level of the image pyramid corresponding to σ_k by searching over the interval $[d_{min}(x,y)\ d_{max}(x,y)]$
- For every pixel, set $d_{min}(x,y) = d_{\sigma k}(x,y) - \delta$ and $d_{max}(x,y) = d_{\sigma k}(x,y) + \delta$
- Repeat at the next pyramid level for $\sigma_{k+1} < \sigma_{k.}$

This is interesting for two reasons:

Compromise localization/detection: Noise-free matching (good detection) is performed at coarse levels of the pyramids), those matches are then refined at the finer levels of the pyramid for better localization.

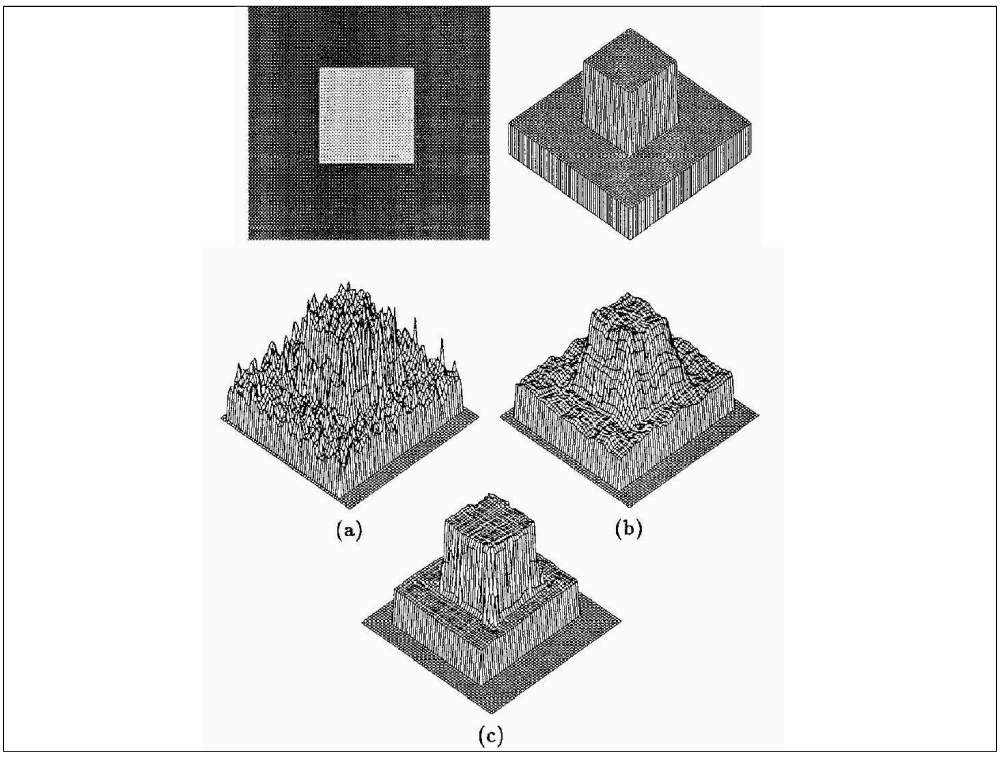
Computation: Matching is expensive. In this approach, the matching at coarse levels requires search over small intervals of disparity (because the image is small). At finer resolution, the search is still limited because we predict a small search interval from the previous level in the pyramid. Therefore, we never have to search the entire disparity interval in the full-resolution image.



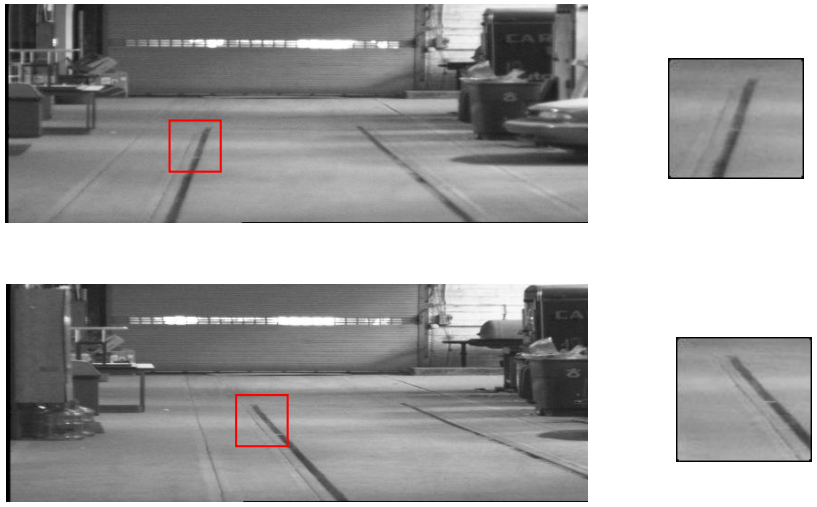
Effect of Window Size:

Window size has qualitatively the same effect as smoothing. Localization is better with smaller windows. In particular, the disparity image is corrupted less near occluding edges because of better localized match. Matching is better with larger windows because more pixels are taking into account, and there is therefore better discrimination between window positions.

Localization degrades as the window size increases, for the same reason. This can be seen geometrically by plotting the curve of the matching function S for different values of w. For large values of w, the minimum is flatter (lower curvature), leading to lower confidence in the matching as defined before.


(a)
(b)
(c)

Window Shape and Forshortening

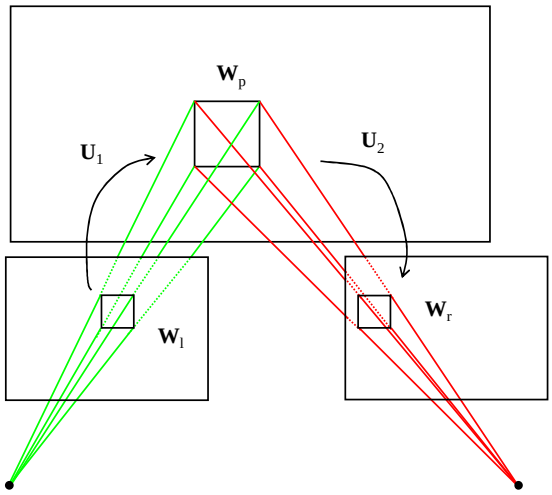


The discussion so far assumes that the window around $\mathbf{P}_l$ is compared to a window of the same size and shape around $\mathbf{P}_r$. In fact, given a square window in the left image, the corresponding pixels in the right image do not necessarily lies in a square window (let alone of the same size.) This is because the perspective projection distorts the geometry and does not preserve angles or parallelism. Without writing equations,it is easy to see on a pair of images taken from a camera low to the ground. A window taken on the ground is very distorted in the right image. Thus, matching results may be very poor by using the same window in the left and right image to do the matching.

The perspective effect involved in this example is called *foreshortening* (things farther away appear smaller.)

The only case in which the windows have the same shape is the case in which the point **P** lies (locally) on a plane parallel to the plane of the two images. This is called a *frontoparallel* configuration.

Window Shape: Fronto-parallel Configuration



The discussion so far assumes that the window around p_l is compared to a window of the same size and shape around p_r. In fact, given a square window in the left image, the corresponding pixels in the right image do not necessarily lie in a square window (let alone of the same size.) This is because the perspective projection distorts the geometry and does not preserve angles or parallelism. Without writing equations,it is easy to see on a pair of images taken from a camera low to the ground. A window taken on the ground is very distorted in the right image. Thus, matching results may be very poor by using the same window in the left and right image to do the matching.

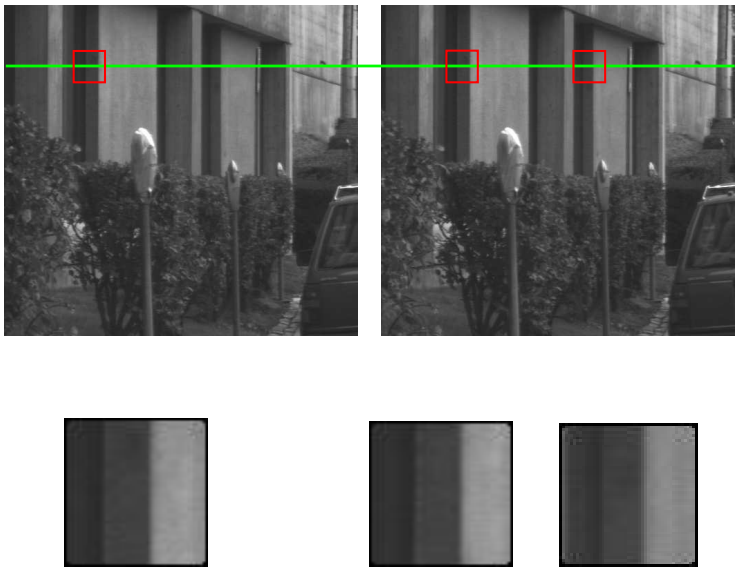
The perspective effect involved in this example is the *foreshortening* (things farther away appear smaller.)

The only case in which the windows have the same shape is the case in which the 3-D point **P** lies (locally) on a plane parallel to the plane of the two images. This is called a *frontoparallel* configuration.

In many cases, this effect is not significant enough to be worth correcting. In some cases, for example when the scene points are known to be near a plane for which the distortion would be significant, it may be worth correcting for it.

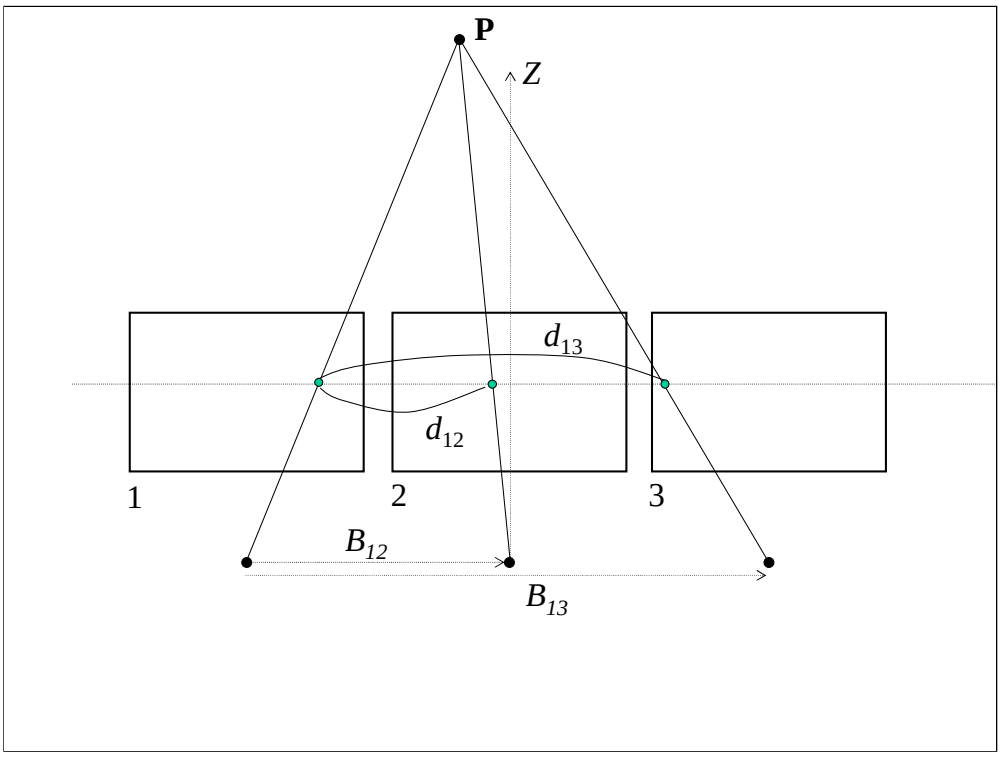
The square window in the left image corresponds to a distorted window on the reference plane. The distorted window can then be projected onto the right image plane to get the image window. The transformation from the left image plane to the reference plane is the same type of transformation as the rectification transformation seen before $\mathbf{W}_p = \mathbf{U}_1\mathbf{W}_l$, similarily the transformation from plane window to right image is $\mathbf{W}_r = \mathbf{U}_2\mathbf{W}_p$. So the transformation from window in the left image to window in the right image is $\mathbf{W}_r = \mathbf{U}_1\mathbf{U}_2\mathbf{W}_l$.

Ambiguity



In many cases, several positions along the epipolar line can match the window around the left pixel. In that case, there is an ambiguity as to which point leads to the correct 3-D reconstruction. If the ambiguous matches are far apart, they will correspond to very different points in space, thus leading to large errors.

In practice, it is nearly impossible to eliminate all such ambiguities, especially is environments with lots of regular structures. The solution is generally to use more than two cameras in stereo.



Another way to view the use of >2 cameras is the "multibaseline" approach. Suppose that the three cameras are now aligned. A point P corresponds to three points in the images, aligned along the horizontal epipolar line. The disparities of those points d12 and d13 between the first image and images 2 and 3 are in general different because the baselines are different:

$$d_{12} = \frac{B_{12}}{Z} \; d_{13} = \frac{B_{13}}{Z}$$

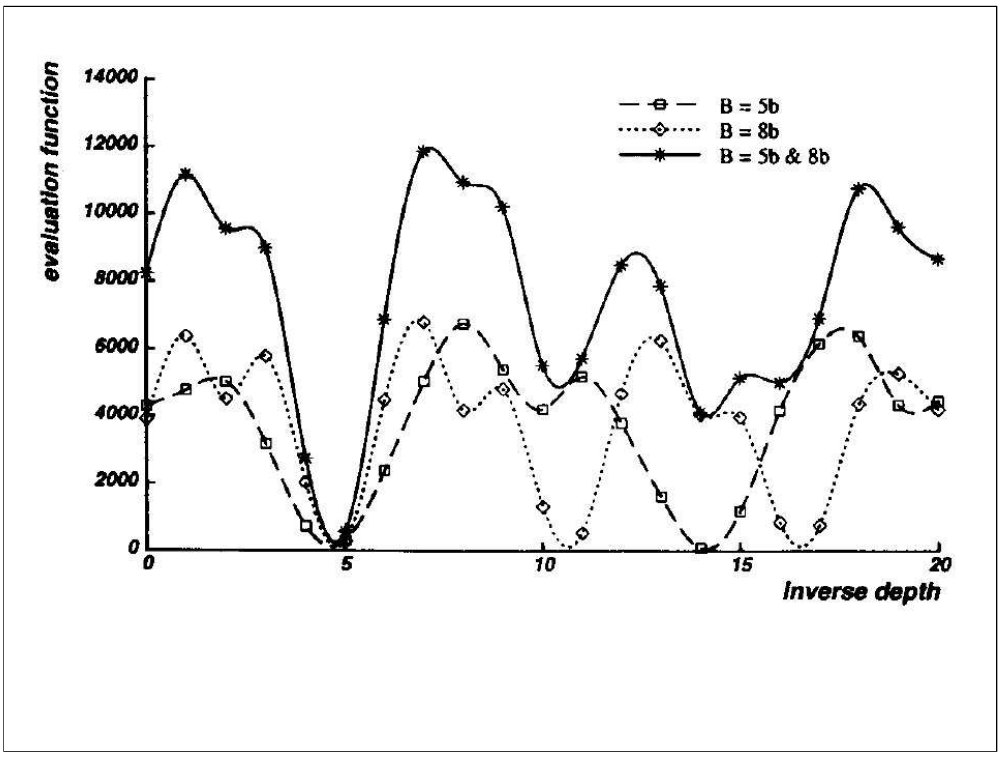
However, the depth (Z) of the physical point in space is the same. Therefore:

$$\frac{1}{Z} = \frac{d_{12}}{B_{12}} = \frac{d_{13}}{B_{13}}$$

The implication is that: if we plot the matching curves ($S_{12}(d')$) not as a function of the disparity but at a function of $d' = d/B$, assuming that the matches are correct, all the curves should have the same minimum at $d' = 1/Z$.

In fact, instead of using S_{12} and S_{13} separately, we can just combine them into a single matching function: $S(d') = S_{12} + S_{13}$

and find the minimum of $S(d')$. Such an approach is called multi-baseline stereo.

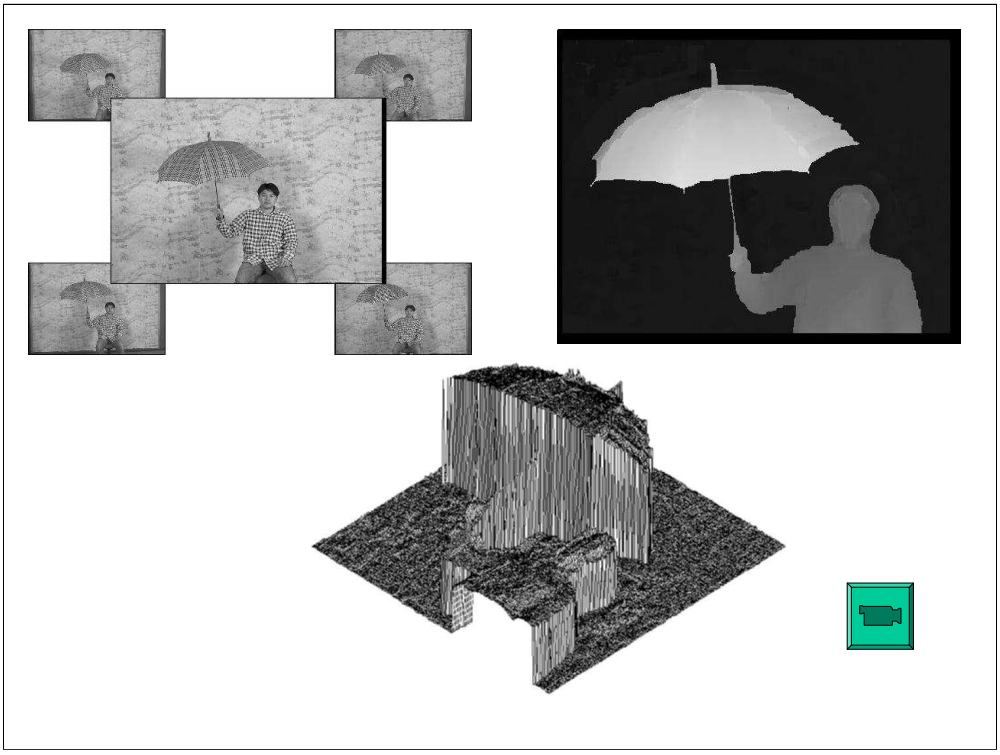


Using a multibaseline approach combines the advantages of both short and long baseline:

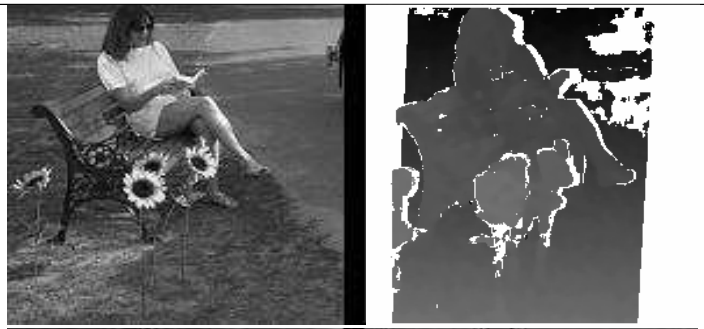
The short baseline will make the matching easier but leads to poorer localization is space. The longer baseline leads to higher precision localization but more difficult matching. Furthermore, ambiguous matches would not generally appear at multiple baselines, thus, by combining the matching function into a single function, we get a sharper, unique peak of the matching function.

Example (old) JPL

- Large baseline
- 5-10Hz
- LOG pyramid
- Rectification
- Sub-pixel parabolic interpolation



Conventional Stereo



High-Fidelity Stereo

- http://www .middlebury.edu/stereo/
- www. ptgrey.com
- www . ai.sri.com/~konolige/svs
- D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms.

  IJCV 47(1/2/3):7-42, April-June 2002.