An Introduction to the SGM Algorithm for Dense Matching in Binocular Stereo

Avinash Kak

Purdue University

March 29, 2023

10:26pm

An RVL Tutorial Presentation on Semi-Global Matching

Originally presented in Fall 2022

CONTENTS

	Section Title	*Page*
1	**Introduction**	3
2	**Some Basic Vocabulary**	10
3	**Constraints on Disparity Values for Rectified Cameras**	16
4	**Markov Random Field Modeling of Disparity Maps**	22
5	**Cost Minimization over MRF for Solving the Disparity Labeling Problem**	26
6	**The SGM Algorithm**	34
7	**Clean-up of the SGM Computed Disparities**	47

Back to TOC

1: Introduction

- In every branch of knowledge, roughly every ten to fifteen years, you see a publication in a journal or a conference that creates what's referred to as a paradigm shift in the community. [The phrase "paradigm shift" was coined by the philosopher Thomas Kuhn in his highly influential book "The Structure of Scientific Revolutions" that was published in 1962.] In Computer Vision, the following publication falls in that category:

  > "Stereo Processing by Semiglobal Matching and Mutual Information" by Heiko Hirschmüller, IEEE-PAMI, 2008.
  >
  > `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4359315`

- Hirschmüller provided a beautiful solution — a solution that actually works in real-world applications — to a very difficult problem: The problem of assigning disparity labels to the pixels in stereo imaging.

- From a purely algorithmic standpoint, what Hirschmüller gifted to the humanity was a computationally efficient polynomial-time approximation to an NP-Hard problem in combinatorial optimization.

- Given a pair of rectified images — a chosen *reference* image and the *other* image — you want to estimate the disparity to be associated with each pixel in the reference image based on which pixel it best matches with in the other image. For rectified cameras, the disparity is the distance between the column index values for the pixel in the reference image and its corresponding pixel in the other image.

- When measured as described above, the disparities are simply integer values. In many applications, one can usually set bounds, $[d_{min}, d_{max}]$, in which to search for the corresponding pixel in the other image.

- The numeric nature of the disparities might cause one to think of a convex-optimization based approach for figuring out the best values for the disparities at each pixel in the reference image. As it turns out, the problem does not lend itself to that sort of optimization. What it does lend itself to is *combinatorial optimization* in which we think of each disparity value as a possible label for a pixel from a set of known labels as determined by the the set of integers in the range $[d_{min}, d_{max}]$. The combinatorial optimization is carried out subject to the maximization of the smoothness of the disparities while also maximizing the quality of the match for the corresponding pixels in the two images.

- As you will see in this tutorial, the problem described above is best cast within the framework of Markov Random Fields, which allows us to achieve global-scale effects through calculations that are local in nature.

- Even with the simplification made possible by MRF modeling, the problem of making the best choices of the disparity labels to the pixels is an NP-Hard problem in combinatorial optimization.

- As I mentioned earlier in this introduction, Hirschmüller demonstrated it was possible to solve the optimization problem approximately in a highly computationally efficient manner and with results of higher quality than had ever been achieved with the earlier solutions.

- To illustrate the power of Hirschmüller's SGM algorithm, shown below is a reconstruction of a portion of Jacksonville, FL, from 40 pairs of multi-date and multi-view satellite images from Maxar (these used to be referred to as WorldView images previously). These were created by ARA's LEGO pipeline that uses Purdue RVL's implementation of a variant of the SGM algorithm. The variant is known as the tSGM algorithm and it was first proposed by Rothermel in 2017 in his "*Development of a SGM-based Multi-View Reconstruction Framework For Aerial Imagery*". Our implementation of Rothermel's variant and another variant from our own lab are presented in the publication "*A*

Comparative Evaluation of SGM Variants (including a New Variant, tMGM) for Dense Stereo Matching" by Patil et al. that you can access at

`https://arxiv.org/pdf/1911.09800.pdf`

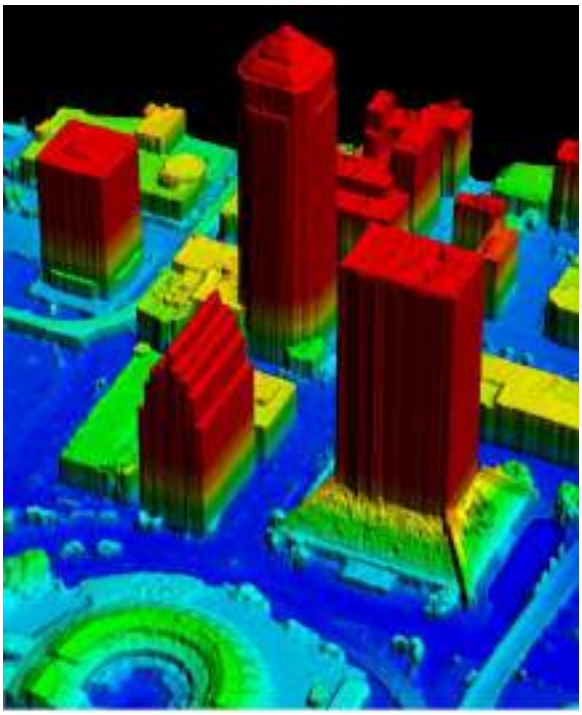

Figure 1: *A reconstruction of a portion of the downtown area of Jacksonville, FL, using 40 pairs of multi-view and multi-date Maxar satellite images. The dense stereo matching algorithm used in this result was developed by Purdue-RVL. As you would expect, there are many other software components that go into producing the kind of result you see in the figure. The result shown was produced by ARA's LEGO pipeline under an IARPA contract.*

- Lest you think that our stereo framework can reconstruct only

small areas on the ground, shown in Figure 2 is a section from from a 120 km^2 area in Ohio from our following publication "*Semantic Labeling of Large-Area Geographic Regions Using Multi-View and Multi-Date Satellite Images and Noisy OSM Training Labels*" by Comandur and Kak in the IEEE-JSTARS journal that you can access here

https://arxiv.org/pdf/2008.10271.pdf

- If you would like to see the flyby videos over the areas reconstructed, there are available here:

https://engineering.purdue.edu/RVL/CORE3D/DSM/

- It is important to bear in mind that the precision of detail in the large-area reconstructions in Figure 2 on the next page is exactly the same as what you saw for a more localized reconstruction in Figure 1. If were were to zoom into the reconstructions shown in Figure 2, the quality of the detail would be no different from what is shown in Figure 1.

- The stereo reconstructions I have shown so far were all at the native resolution of the Maxar images, roughly around 0.3m per pixel. For large-area monitoring of the earth, there is interest in downshifting the resolution to see how that affects the quality of the DSMs. Toward that end, shown in Figure 3 is a

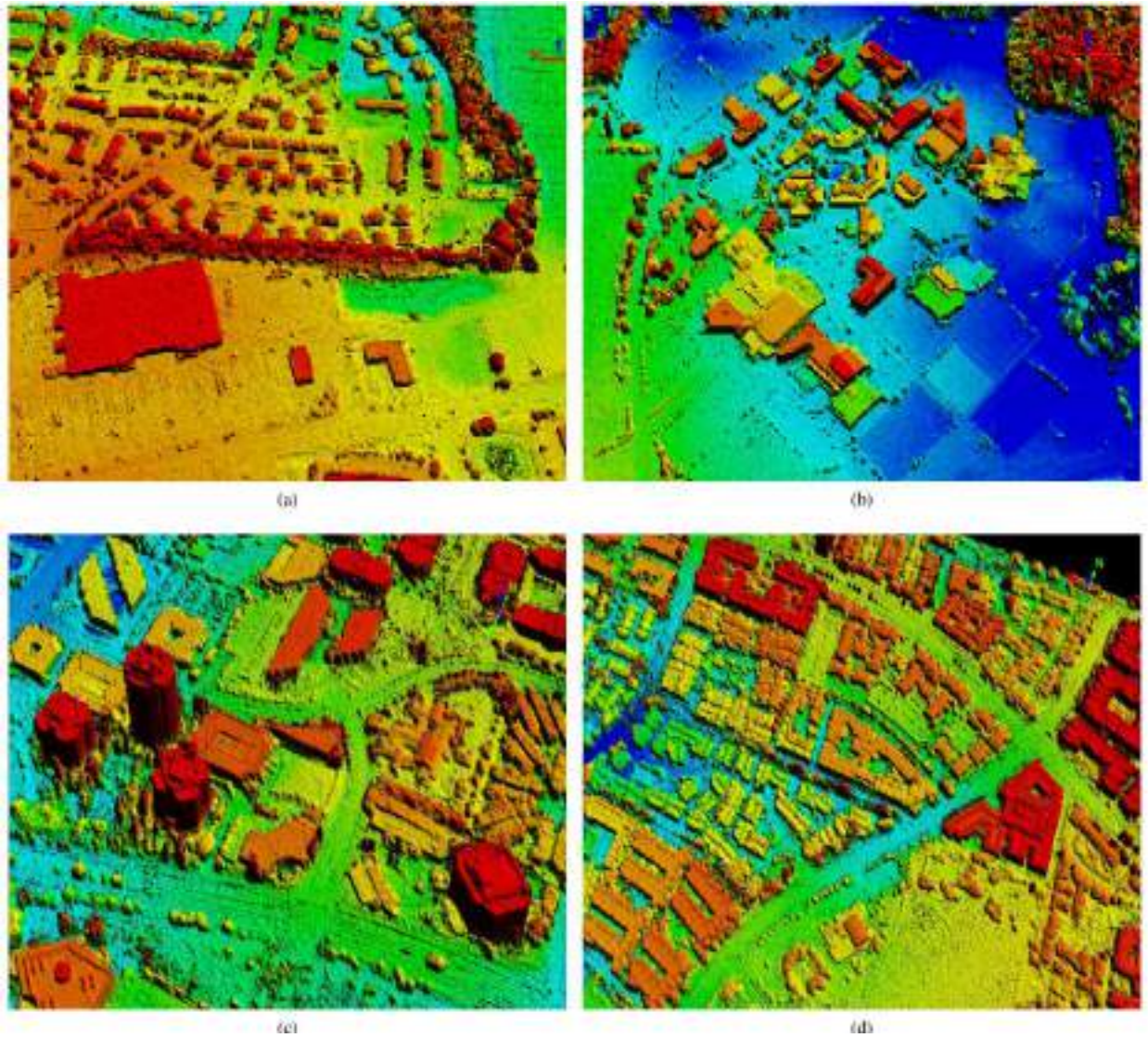

Figure 2: *Sections of the DSMs for two large areas reconstructed by Purdue's RVL-LEGO pipeline: a 120 km^2 area in Ohio and a 62 km^2 area in California. The top two images depict two small sections from the Ohio DSM, and the bottom two images depict two small sections from the California DSM. If you were to zoom into the reconstructions, you will see the same precision of detail as in the more localized reconstruction in Figure 1. The large-area reconstructions shown here are from the IEEE-JSTARS paper by Comandur and Kak.*

reconstruction of a 105 km^2 area around Jacksonville, FL, at 1.5m GSD. This DSM was produced by Purdue's RVL-LEGO pipeline.

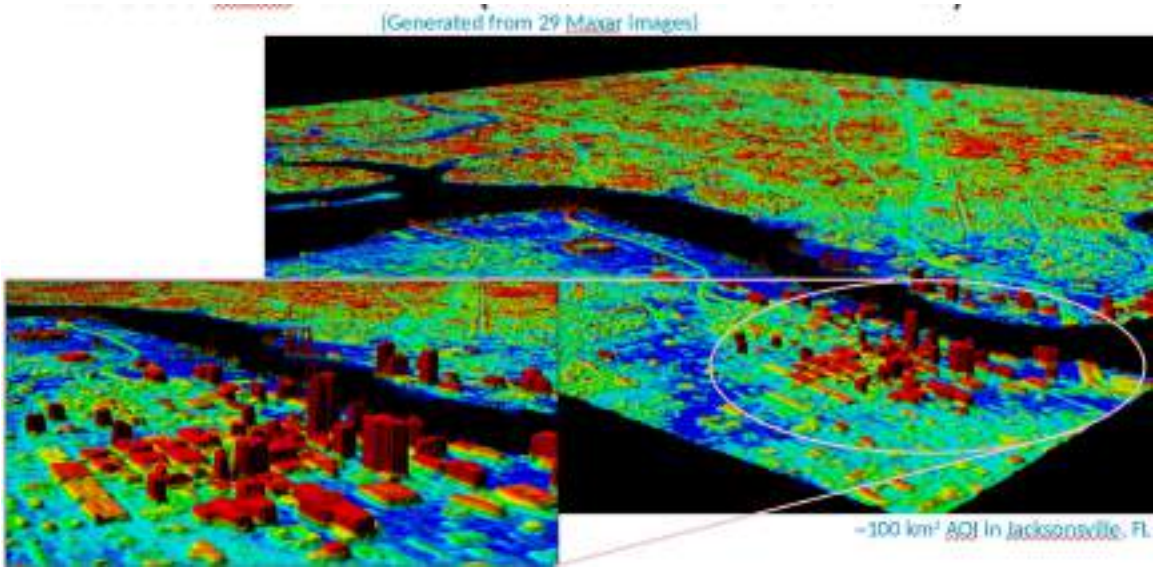

Figure 3: *An example of a DSM produced at 1.5m GSD for the Jacksonville area of size 105 km^2. This DSM was produced by Purdue's RVL-LEGO pipeline.*

- Finally, the main reason for this tutorial is for it to serve as a handout for Lecture 25 of my class on Computer Vision at Purdue University. Here is a link to the course website so that you can see for yourself where this lecture belongs in an overall organization of the course:

`https://engineering.purdue.edu/kak/ComputerVision`

Back to TOC

2: Some Basic Vocabulary of Stereo Imaging

- For obvious reasons, any discussion of stereo in computer vision must begin with a pair of images (I, I') of the same scene from two different viewpoints. Shown below is an example.

I I'

Figure 4: *The two images of a stereo pair*

- We choose one of the two images, say I, as our *reference image*. The goal is to scan the pixels in I and, for each pixel $(i, j) \in I$, search in the other image, I', for the corresponding pixel (meaning the pixel for the same object point in the scene) which, if found, would be located at, say, the coordinates $(m, n) \in I'$.

- Subsequently, we refer to the distance d between the reference image pixel $(i, j) \in I$ and the corresponding pixel $(m, n) \in I'$ in the other image as the *disparity* associated with the reference image pixel.

- The goal of a stereo matcher is to find for each pixel $(i, j) \in I$ in the reference image its corresponding pixel in the other image. In other words, the goal of a stereo matcher is to calculate the *disparity map* over the pixels in the reference image.

- The search for the corresponding pixels in the two images becomes a little bit easier if the two images are rectified — meaning that if a pair of corresponding pixels exist in the same row in the two images. As you know from my Lecture 24, for any given pixel $(i, j) \in I$, the search for its corresponding pixel in the other image must be conducted on the former pixel's *epipolar line* in the other image. Also, as I explained in Lecture 24, a good *rectification algorithm*, like the Loop and Zhang algorithm presented in my "Loop and Zhang Reader", will do a good job of rectifying the two stereo images (I, I') so the epipolar lines become the corresponding rows in the images. Here is a direct link to the "Loop and Zhang Reader":

  `https://engineering.purdue.edu/kak/Tutorials/StereoRectification.pdf`

- From this point on, I'll assume that the image pair (I, I') has been appropriately rectified. As a consequence, for a given pixel

$(i, j) \in I$, its corresponding pixel can be expected to be $(i, n) \in I'$ for some value of integer n. Should we be able to find the matching pixel at (i, n), the *disparity* at the reference-image pixel $(i, j) \in I$ will be $j - n$. [**As you will see, for the case of rectified cameras, given a pair of corresponding pixels that should be in the same row in the two images, the left-image column-index will always be greater than the right-image column-index. More accurately speaking, the left-image column-index will not be less that the right-image column-index.**]

- It may appear that the assumption of images being rectified, as stated above, would make trivial the problem of disparity estimation. **Unfortunately, that's not the case.**

- Shown below is a stereo pair that appears as if it might have been rectified. Let's go with that assumption just so that I can explain that dense stereo is challenging even when the two images are rectified.

- To show how difficult it would be to find the pixel correspondences, I have drawn a line through the two images as shown below. Let's say we want to find the right-image pixel that corresponds to the left-image pixel marked 'X'. As you see, all the pixels in the vicinity of where we may hope to find the X's corresponding pixel look more or less the same. Therefore any logic that searches for the corresponding pixels just on the basis of the similarity of their neighborhoods is bound to fail.

I I'

Figure 5: *Let's pretend for a moment that these two stereo images are rectified.*

- The difficulty of finding pixel correspondences as illustrated above is greatly resolved by the Hirschmüller's Semiglobal Matching (SGM) algorithm cited earlier in the Introduction section.

- The key ideas that allows the SGM algorithm to create a dense disparity map (meaning that it tries to find a disparity at *every* pixel in the reference image) is that the search for the best disparity value at a pixel is conducted simultaneously along multiple directions in an image as illustrated in Figure 7. The algorithm estimates the costs associated with the assignment of all possible disparities to the pixels along a certain number of designated directions. From amongst the choices made available in this manner at each pixel, the disparity that has associated with it the least cost is accepted for the pixel.

Figure 6: *Shows the great difficulty of finding matching points. For the pixel marked 'X' in the left image, where exactly is the corresponding pixel in the right image?*

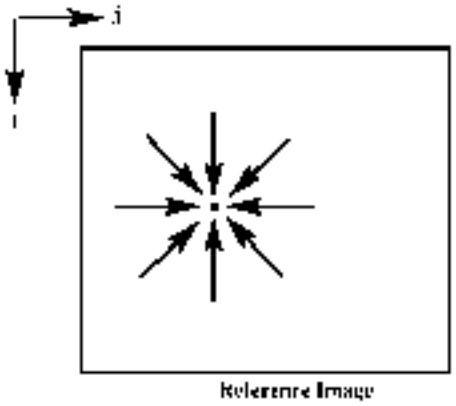

Figure 7: *For the pixel shown with a small circle, the arrows desginate the directions along which the disparity assignment costs are estimated in the SGM-8 algorithm by Hirschmüller. The disparity that has associated with it the least cost is accepted for the pixel.*

- At this point, the reader might ask: If we know in advance the disparity range $[d_{min}, d_{max}]$ in which to search for, why is the problem any more complex that reaching into the other image and simply looking for the pixel has the best match for the pixel in question in the reference image. And even if we had to use multiple directions as advocated by Hirschmüller, given again a known search bound, why is that any more difficult than looking for the best pixel to match with along just one line?

- As you will see in the rest of this tutorial, the matching of the pixels in the two images of a stereo pair cannot be carried out in isolation from the matching decisions elsewhere in the image. A decision for a disparity value at any given pixel must take into account the previously calculated disparities at the neighboring pixels and those, recursive speaking, must depend on the disparity values computed before them. The main reason for that is that we want the calculated disparities to be maximally smooth — subject, of course, to the presence of genuine depth discontinuities in the scene.

- It is the inter-dependencies between the disparity assignments at the different pixels in the reference image that turn the problem into an exercise in combinatorial optimization.

Back to TOC

3: Constraints on Disparity Values for Rectified Cameras

- As you know, to the extent possible, our goal is to estimate the disparity at every pixel in the reference image vis-a-vis the corresponding pixels in the other images. **In this section, I will assume that that the image produced by the left camera ('left' for an observer standing on the stereo baseline and looking in the same direction as the cameras) is the *reference image*.** We also assume that the two images have been rectified and are therefore row aligned for disparity calculations. Recall that, for rectified cameras, each row is the epipolar line for the pixels in the same row in the other image and vice versa.

- As it turns out, under the conditions described above, the disparity values estimated at the pixels in the reference image must obey an ordering constraint that will become obvious from the explanations that follow. Since the constraint plays an important role in how we clean up the noise in the computed disparities, it is important to understand it. I will use the three figures that follow to explain some key "properties" of disparity values and, subsequently, the ordering constraint.

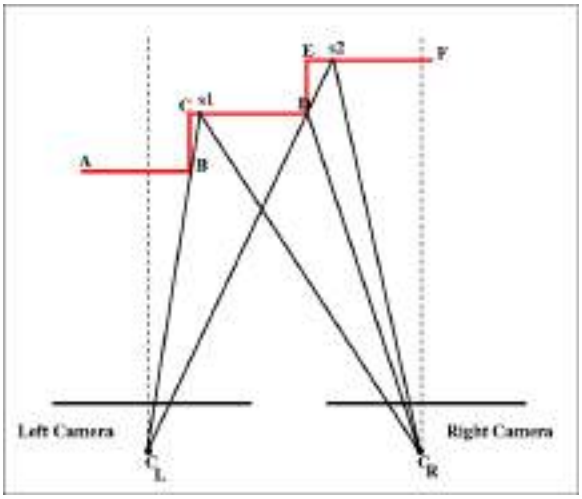

Figure 8: *Understanding the disparities calculated for rectified cameras - I*

- Consider the case when the depth variations in the scene in front of the cameras are as shown by the red line in Figure 8. This is a good first example to convey the idea that even with the best algorithms available, you may **not** be able to estimate the disparity at all pixels in the reference image. As shown in Figure 8, an important reason for this is that one portion of the scene may obscure other portions in either or both of the two cameras.

- For the case depicted in Figure 8, the sharp depth discontinuity at point B in the scene creates a self-obscuration in the reference camera. As a consequence, the scene points in the segment B-C-s_1 are not visible to the reference camera. Another example of the same is the obscuration in the reference camera created by the depth discontinuity at point D in the scene. As a result, the estimated disparities will also miss out on the scene points in the segment D-E-s_2 of what's in front of the cameras.

- The above observation is important for understanding the cause of disparity jumps. We will have a sudden jump in the disparity map when we go from the reference-image pixel for point B in the scene to the next pixel that would correspond to point s_1 in the scene. You will see a similar jump in the disparities when you go from the pixel for point D in the scene to the pixel for point s_2.

- Let's now look at the scene example shown in Figure 9. The overall conclusion here is the same as before: Even when we are able to estimate the disparities at every pixel in the reference camera, some portions of the scene will remain uncovered on account of the obscurations in — in this case — the right camera.

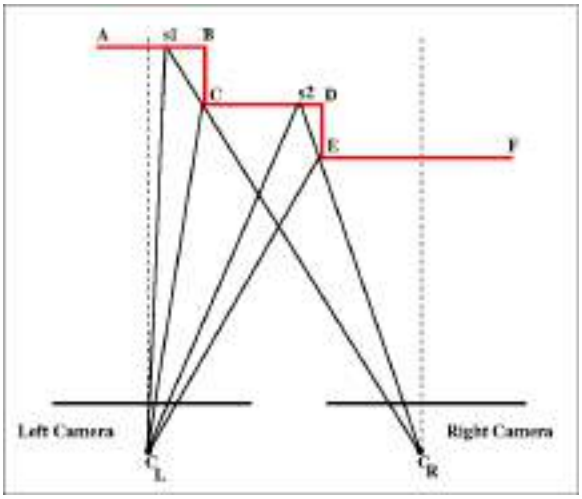

Figure 9: *Understanding the disparities calculated for rectified cameras - II*

- While the overall conclusion is the same for the previous example and the current one, there is one very important difference: **Now**

there will be *gaps* in the pixels in the reference image where there will be no disparities. For example, there will be no disparities for the reference-image pixels corresponding to the portion s_1-B-C of the scene because that portion is obscured in the right camera. The same would be the case for the potion s_2-D-E of the scene.

- To summarize, whereas the example in Figure 8 leads to sudden one-pixel-to-the-next jumps in the disparity map over the reference image, the example in Figure 9 results in gaps in the disparity map.

- The more complex example shown in Figure 10 will exhibit both gaps and jumps in the disparity map over the reference image. Under the best of conditions when you are able to find *all possible* pixel correspondences for the pixels in the reference image, you will see a gap for the portion s_1-s_2 of the scene followed immediately by a jump at the next pixel corresponding to point C in the scene.

- To continue with the example in Figure 10, subsequently you will see a continuous disparity map for the portion C-s_2 of the scene. That will be followed by a gap over the pixels corresponding to the portion s_2DE in the left camera. followed by another continuous run of disparity values for the portion EF in the scene, following a pixel-to-pixel sudden jump in disparities created by

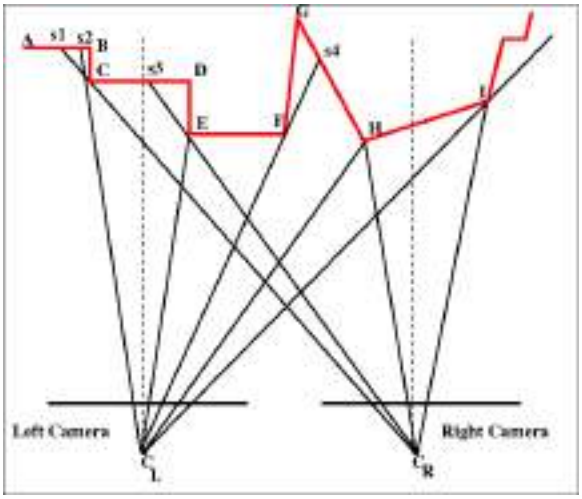

Figure 10: *Understanding the disparities calculated for rectified cameras - III*

the right-camera obscuration for the portion F-G-s_4 of the scene, and followed, finally, by a continuous run of disparity values for the portion s_4-H-I of the scene. Note that we cannot refer to the last few pixels in the reference camera as a gap since there is nothing there to see of the object at those pixels.

- To formalize these observations, let's use the notation (i, j) to denote the pixels in the reference image and (m, n) to denote the pixels in the other image. Assuming the images to be rectified and that we are talking about the corresponding pixels in the same row, indexed i, in the two cameras, let the column index values for the k^{th} correspondence in the i^{th} row in the two images be denoted j_k in the reference image and n_k in the other image. In this case, the disparity d_k would be given by

$$d_k \;=\; j_k \;-\; n_k \tag{1}$$

- From the geometry of pinhole imaging, for the case of rectified cameras, the value of d_k is always positive. It will approach zero for very distant objects in the scene. **The closer a scene point to the camera baseline, the larger the disparity.** [In general, though, for two cameras with converging optic axes, the disparity will be positive for scene points closer than the point of convergence and negative for the scene points beyond that.]

- Let $M = [(j_1, n_1), (j_1, n_1), (j_2, n_2), ..., (j_K, n_K)]$ denote a sequence of *consecutive* matches along the i^{th} row in the two images. The matches obey the following ordering property: For any two consecutive matches $[(j_{k_1}, n_{k_1}), (j_{k_2}, n_{k_2})]$, $j_{k_1} > j_{k_2}$, and $n_{k_1} > n_{k_2}$

- For a more elaborate discussion of the constraints that apply to the disparities for a pair of rectified cameras, see the paper "*A Hierarchical Symmetric Stereo Algorithm using Dynamic Programming*" by Van Meerbergen et el.:

`https://https://people.inf.ethz.ch/pomarc/pubs/VanMeerbergenIJCV.pdf`

Back to TOC

4: Markov Random Field Modeling of Disparity Maps

- As mentioned in Section 2, it makes no sense to calculate the disparity at each pixel in isolation from the other pixels in the reference image. Speaking strictly, the disparity at each pixel must depend on the disparities at all the other pixels. The question then becomes: What is a good computationally feasible model for the interdependence of the disparity values?

- A good computationally feasible model of interdependence that has served well a vast majority of problems in computer vision is known as the Markov Random Field (MRF). The most general definition of MRF is for graphs — it addresses the properites of probability distributions over the nodes in a graph.

- In our context, you can fit a first-order MRF model to an image if the probability of the disparity at each pixel acquiring a particular value depends directly on only the disparities at its immediate neighbors.

$$prob\Big(d(i,j) \mid I\Big) \;\; = \;\; prob\Big(d(i,j) \mid (m,n) \in \mathcal{N}_{i,j}\Big) \qquad (2)$$

where I represents all the pixel in the reference image and $\mathcal{N}_{i,j}$ is

the set of pixels in the immediate neighborhood of the pixel at (i, j). The notation $prob\Big(d(i,j)\Big)$ stands for the probability of the disparity at pixel (i, j) being what is specified by the disparity map $d(i, j)$ over the reference image.

- Connecting the above image-based definition of an MRF model with its more general graph-theoretic definition, you can think of each pixel as a node in a graph formed by all the pixels and each direct pixel-to-pixel dependency by an edge in the graph. **The MRF assumption stated above boils down to saying that the graph representing all the nodes is made up of cliques, with each clique "centered" at a pixel (i, j) and with the other nodes in the clique consisting of the immediate neighbors of (i, j).**

- The model in Eq. (2) and its connection with the more general graph-theoretic MRF modeling allows us to talk about the joint probability of an entire disparity map. That is, we can now talk about the probability to be associated with the disparities over all the pixels in the image I. In principle, we would want to accept that disparity map over all the pixels in I that has the highest overall probability associated with it. We therefore want to be able to talk about different possible disparity maps and their associated probabilities.

- **That takes us to a most famous theorem, known as Hammersley-Clifford Theorem (HC), in probabilistic inference**

over graphs.

- The HC theorem tells us that if you are thinking about a joint probability distribution over all the nodes in a graph, all it takes is assigning arbitrary positive real numbers to the individual cliques in the graph, normalizing the values so that you are not violating the basic tenets of probability (the unit summation rule, for example), and, subsequently, taking a product of the numbers.

- In the graph-theoretic jargon, the numbers you assign to the cliques are referred to as the *potentials*. As mentioned, the product of the potentials constitutes a valid joint distribution over all the nodes in the graph in pretty much the same sense as assigning *arbitrary* nonnegative numbers to the outcomes in a random experiment constitutes a valid probability distribution as long as the numbers add up to 1. For a general graph $G = (V, E)$ consisting of the nodes V and the edges E, the theorem is expressed compactly as

$$prob(V) \;=\; \prod_{k} \phi(clique_k) \tag{3}$$

where $\phi(clique_k)$ is the potential assigned to the k^{th} clique in the graph G. Note that this is a necessary and sufficient condition for the LHS above to be a *legitimate* probability joint distribution over the set V of nodes in G.

- **The HC theorem gives us the license to think *locally* for drawing inferences that work *globally*.**

- While this may give us the permission to make each local decision in isolation from all such decisions elsewhere in an image without violating the HC theorem for creating a valid joint probability distribution, in practice we must make the decisions interdependently for generating the best solutions. **Nevertheless, the HC theorem is indispensable in making those interdependent decisions more computationally efficient.**

- In other words, while the HC Theorem prescribes a necessary and sufficient condition for a joint probability distribution over the nodes in a graph to be legitimate, it is only a necessary condition for the probability distribution to be of practical utility.

Back to TOC

5: Cost Minimization over MRF for Solving the Disparity Labeling Problem

- The HC theorem obviously allows us to make local decisions in order to achieve global effects. While the theorem take care of the necessary conditions to be satisfied for the localized processing of an image, in and of itself it does not constrain the final solution sufficiently for it to be of practical usefulness.

- **The additional constraints on the final solution are supplied by what's come to be known as *cost minimization*. As you will see, the process of cost minimization takes full advantage of the localized nature of computations made possible by the HC theorem.**

- To explain how the cost minimization works algorithmically, I'll follow the basic definitions provided by Szeliski, et al. in their 2008 paper "*A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors*" that you can access here:

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4420084

- Before explaining the rudiments of cost minimization for disparity estimation, I must point out that (as pointed out in the above publication), this approach to problem solving is much more general than would be reflected by the concerns of this tutorial. Basically, in the context of computer vision, it should be possible to solve *any pixel labeling problem* over an MRF with this approach. Dense disparity estimation is obviously a pixel labeling problem in the sense that you would like to associate a disparity value with every pixel in the reference image. Other important pixel labeling problems include semantic segmentation, object detection, object tracking in videos, etc.

- At this point, you are probably surprised that the calculation of a disparity map over the pixels in the reference image can be considered to be an example of a pixel labeling problem. As it turns out, yes, it can. We can place depth bounds on the scene for which we are estimating the disparities. For a given separation between two optical centers of two *rectified* cameras, the farther a scene point, the smaller will be value for its disparity. That is, a far-off scene point will be at approximately the same column index in the two images. By the same token, a nearby scene point will produce a large disparity in the reference image. So if we say that the closest scene points will produce a disparity of, say, 10 pixels, we can then claim that the disparity value assigned to each pixel in the reference image will be from the set of integers $\{0, 1, 2, ..., 10\}$. **As you can see, that definitely looks like a pixel labeling problem.**

- When the probabilities associated with the assignment of labels to pixels can be modeled as a MRF, **a cost minimization framework for the solution requires that you define two costs**: the cost of assigning a disparity label to the pixel (considering, say, the noise at the pixel and the other pixels in its neighborhood) and the cost associated with assigning two *different* disparity labels to the pixels that are in each other's immediate neighborhoods. The former is referred to as the *data cost* and the latter as the *discontinuity cost*. The two may also be referred to as *data penalty* and *discontinuity penalty*. In the literature, you will also see the following usage for the two costs: *data energy* and *discontinuity energy*.

- Note again that in the usages *discontinuity cost*, *discontinuity energy*, etc., the term *discontinuity* refers to the change in the values of the disparity labels in a small neighborhood around each pixel in the reference image.

- Denoting the data cost by $\mathcal{C}_{data}$ and the discontinuity cost by $\mathcal{C}_{discon}$, we use the symbol $\mathcal{C}$ to denote a weighted combination of the two:

$$\mathcal{C} \;=\; \mathcal{C}_{data} \;+\; \lambda \mathcal{C}_{discon} \tag{4}$$

- The "best" labeling solution is then defined as the one that minimizes the total cost $\mathcal{C}$.

- The purpose of the data cost at each pixel that gets a disparity label is to capture the degree to which the assigned label agrees with the *observed data* at and around the pixel where the label was assigned. To calculate $\mathcal{C}_{data}$, we must sum the data costs at all the pixels.

- For example, if $d(i,j)$ is the disparity label assigned to the pixel at (i,j) and if c_{data} is the cost associated with this assignment, we can write the following expression for the overall data cost:

$$\mathcal{C}_{data} \;=\; \sum_{(i,j)\in I} c_{data}\Big(d(i,j)\Big) \tag{5}$$

- For a deeper insight into what is meant by the data cost, let's say you are in the process of constructing a disparity map over the pixels in the reference image. At the moment, you are at the pixel (i,j) in the reference image and you are looking at a neighborhood in the same row in the other image in a window whose second coordinate goes from $n-d$ to $n+d$ for some value of the integer n. Let's say that the pixels in that window are very noisy and make it difficult to decide as to which of those pixels would make a good match for the $(i,j)^{th}$ pixel in the reference image. **So any choice you make for the disparity value could prove to be a wrong choice.** In other words, you are not likely to be very confident about whatever choice you make for the disparity at (i,j) in the reference image. If $d(i,j)$ is the disparity you decide to assign to the (i,j) pixel, you would make the value

of $c_{data}\Big(d(i,j)\Big)$ large as a cost (or penalty, if that makes it easier for your intuitions) for having chosen $d(i,j)$ under rather *difficult* and *uncertain* circumstances.

- In summary, in the context of disparity mapping, at any pixel (i,j) in the reference image, $c_{data}\Big(d(i,j)\Big)$ measures the how well the choice of the disparity $d(i,j)$ harmonizes with the other pixels in the neighborhoods of both the (i,j) pixel in the reference image and the $(i,j+d)$ pixel in the other image. If you want to factor in both neighborhoods, we could make it proportional to a measure of the difference between the two neighborhoods with, say, what's known as the census transform.

- The formula for the total data cost is written more compactly as:

$$\mathcal{C}_{data} \;=\; \sum_{\mathbf{p}\in I} c_{data}(d_{\mathbf{p}}) \tag{6}$$

- Let's now turn our attention to the disparity discontinuity term in Eq. (4). The goal of this term is to minimize sudden discontinuities in the disparity values within small neighborhoods in the reference image. In most practical applications, it is highly unlikely that you would want the disparity labels assigned to the pixels to be chaotic. To the extent possible, we want the disparity maps to be piecewise continuous. In light of the discussion in Section 3, piecewise continuity means that the disparity map

along, say, each row may only have a finite (hopefully just small) number of sudden jumps in disparity values, and also a finite (hopefully just a small) number of intervals where there do not exist any disparities on account of the *gaps* we talked about in Section 3. Outside of such gaps and sudden jumps, we want the disparity labels to be continuous over runs of pixels in the reference image (and that includes runs in different directions in the reference image). **Minimization of the second term in Eq. (4) is supposed to give us that sort of continuity.**

- In general, we may write the following expression for the discontinuity cost:

$$\mathcal{C}_{discon} = \sum_{(i,j)\in I} \sum_{(i',j')\in\mathcal{N}_{(i,j)}} c_{discon}(d(i,j) \neq d(i',j')) \tag{7}$$

where $\mathcal{N}_{(i,j)}$ is a neighborhood of the pixel (i,j) in the reference image. **This formula uses the fact that, as discussed in Section 4, the HC theorem allows us to limit the testing for disparity continuity at each pixel in the reference image to the immediate neighborhood of that pixel.**

- We can write the above formula more compactly as:

$$\mathcal{C}_{discon} = \sum_{\mathbf{p}\in I} \sum_{\mathbf{q}\in\mathcal{N}_{\mathbf{p}}} c_{discon}(d_{\mathbf{p}} \neq d_{\mathbf{q}}) \tag{8}$$

- The formula shown above extracts the same penalty for each case when the disparity label assigned at a neighborhood pixel $\mathbf{q}$ is different from the disparity label assigned at the pixel $\mathbf{p}$ in question. In general, you are more likely to make this cost a function of the difference in the values of the disparities:

$$\mathcal{C}_{discon} = \sum_{\mathbf{p}\in I} \sum_{\mathbf{q}\in\mathcal{N}_\mathbf{p}} c_{discon}(d_\mathbf{p} - d_\mathbf{q}) \tag{9}$$

- Since the disparity labels are ordinary integers, it is for us to set up the cost function c_{discon} so that the larger the difference $d_\mathbf{p} - d_\mathbf{q}$ in the disparity values at two neighboring pixels, the greater the cost of assigning the disparity $d_\mathbf{p}$ to the current pixel $\mathbf{p}$.

- Since the discontinuity cost deals with disparity *discontinuities*, in order to use the formula shown above we would need to define c_{discon} for all disparity differences $d_\mathbf{p} - d_\mathbf{q} \geq 1$.

- **Even with the localization of the cost calculations made possible by MRF modeling, a truly global minimization of the total cost defined earlier is computationally impossible for realistic images. Just imagine the combinatorics of trying out different possible ddisparities at all the pixels, computing the data cost and the discontinuity cost for each choice in order to figure out the best values to use for the disparity labels at each pixel.**

- In the next section, I'll review the SGM algorithm by Hirschmüller that is a computationally efficient approach to the minimization, albiet an approximation one, of the overall data and discontinuity costs. Since it is a solution to an NP-Hard combinatorial optimization problem, it can never archieve a truly global minimum. That raises the question how good is the approximation achieved by the algorithm. As for that, the quality of the results that can be achieved with this algorithm speak for themselves.

Back to TOC

6: The SGM Algorithm

- As previously mentioned, SGM is an approximate but computationally feasible solution to the cost minimization problem presented in the previous section.

- A key concept in the explanation of the algorithm that follows is that of a Disparity Volume over the reference image as shown in Fig. 11. If the expected disparity range is $[d_{min}, d_{max}]$, the Disparity Volume would be of size $W \times H \times (d_{max} - d_{min})$ where W and H are the width and the height of the reference image.

- At each pixel $\mathbf{p}$ in the reference images, the goal of the SGM algorithm is to estimate the total cost of the disparity $d_{\mathbf{p}}$ at that pixel being equal to each possible disparity label above the point $\mathbf{p}$ in the Disparity Volume.

- You could say that for each of the $W \times H \times (d_{max} - d_{min})$ points in the Disparity Volume the job of the SGM algorithm is to estimate the cost of the disparity at the pixel $\mathbf{p}$ shown in the figure to be equal to each of the disparity values above that point.

- After the Disparity Volume is filled with cost values, for our final

solution, to each pixel $\mathbf{p}$ in the reference image, we assign that disparity that has associated with it the last total cost.

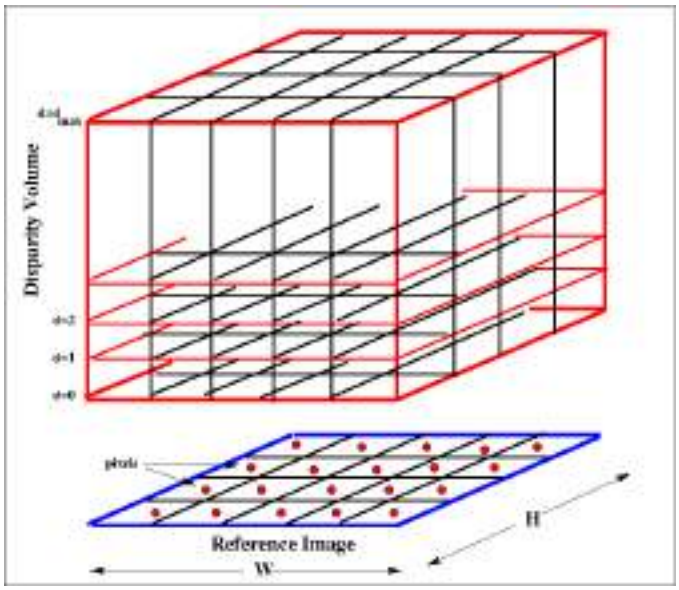

Figure 11: *The Disparity Volume is shown in red and you can think of it as sitting on top of the reference image.*

- Fundamental to the SGM algorithm is the following expression for the total cost $\mathcal{C}$ that one would want to minimize for obtaining the best solution for the disparities at all the pixels in the reference image:

$$\mathcal{C}(d) \;=\; \sum_{p} \left(\mathcal{C}_{data}(\mathbf{p}, d_{\mathbf{p}}) \;+\; \sum_{\mathbf{q} \in \mathcal{N}_{\mathbf{p}}} P_1 \cdot T[|d_{\mathbf{p}} - d_{\mathbf{q}}| = 1] \;+\; \sum_{\mathbf{q} \in \mathcal{N}_{\mathbf{p}}} P_2 \cdot T[|d_{\mathbf{p}} - d_{\mathbf{q}}| > 1] \right) \quad (10)$$

- For the notation used in the above equation:

d **:** a disparity map over the pixels in the reference image; you can think of the map as $d : \{(i,j) => \{d_{min}, d_{max}\} \mid (i,j) \in I\}$ where I is reference image

$\mathbf{p}$ **:** a pixel in the reference image

$\mathbf{q}$ **:** a pixel in the neighborhood of the pixel $\mathbf{p}$ in the reference image

$\mathcal{N}_{\mathbf{p}}$ **:** local neighborhood around pixel $\mathbf{p}$ in the reference image I

$d_{\mathbf{p}}$ **:** disparity label assigned to the pixel $\mathbf{p}$ in the reference image

$\mathcal{C}_{data}(\mathbf{p}, d_{\mathbf{p}})$ **:** The data cost at the pixel $\mathbf{p}$ in the reference image for the disparity label $d_{\mathbf{p}}$ at that pixel

$\mathbf{T}$ **:** defined as $T[arg] = 1$ if arg is True and $T[arg] = 0$ when arg is false

P_1 **and** P_2 **:** User defined weights

- One of the key concepts of the SGM algorithm is that the total cost defined above must be computed for each disparity level in a prespecified range of disparities and, subsequently, **in a winner take all strategy**, we retain at each pixel the disparity that has associated with it the least cost at that pixel.

- Therefore, you must specify in advance the disparity range you are interested in. I'll use the notation (d_{min}, d_{max}) to express this range. Obviously, the minimum expected disparity d_{min} puts a constraint on the the farthest object point from the cameras and the maximum expected disparity d_{max} a constraint on how close

the object points are allowed to be. [In connection with this specification, note that, as described previously in Section 3, the disparities cannot be negative for rectified cameras. In general, though, disparities may be positive or negative for convergent cameras depending on whether the 3D point is closer than the point where the optic axes intersect or farther. When the 3D point is on the camera side of the point of intersection of the axes, the disparity is negative. And when the 3D point is farther than the point of intersection of the axis, the the disparity is positive.

- Comparing the formulas for the discontinuity costs in Eq. (10) and Eq. (9), SGM uses only two user-defined "weights" P_1 and P_2 to make a distinction between just two case of disparity discontinuities: when $|d_{\mathbf{p}} - d_{\mathbf{q}}| = 1$ and when $|d_{\mathbf{p}} - d_{\mathbf{q}}| > 1$. As you would expect, it must be case that $P_2 > P_1$. What that means is that we want the disparities to change only slowly to the largest extent possible but as dictated by the cost minimization. You could say that SGM considers the assigned disparities to be continuous if they change by at most a unit value when you go from one pixel to the next in the reference image. **Any larger disparity changes on a pixel-to-pixel basis are to be discouraged by the higher cost P_2.**

- Note that the essence of the cost minimization here — especially in comparison with the other cost minimization solutions you have seen in my class. **The cost calculation involves just summing numbers, some based on the differences in the pixel values and others based on the differences in the assigned disparity values.** We are NOT visualizing the cost as defining a surface over a parameter space and, therefore, we are NOT computing any

Jacobians and Hessians here. On the other hand, now we are solving a combinatorial problem: To the extent possible, we want to try out all possible disparity labels at all the pixels and, finally, retain for each pixel that disparity which minimizes the overall cost.

- As you try to understand what follows in this section, it's good to keep at the back of your head the following thought: Our goal is NOT to find the disparity value at each pixel of the reference image that minimizes the cost at just that pixel. On the other hand, our goal is to find the best disparity at each pixel that minimizes the *overall* cost $\mathcal{C}$.

- The most important aspect of SGM is how it resolves the computational impracticality of using the cost formula in Eq. (10) for finding the best disparity map d over the reference image. As mentioned toward the end of the previous section, to find the disparity map that truly minimizes the overall cost over all possible disparity assignments over all the pixels in the reference image is an NP-Hard problem. What that implies is that while you might be able to get it to work on small images, it would be prohibitively computationally expensive for images of the size you are likely to encounter in non-trivial applications.

- In order to address this computational dilemma, SGM takes inspiration from the following paper by Van Meerbergen et el.

that I cited previously: "*A Hierarchical Symmetric Stereo Algorithm using Dynamic Programming*." An important message in this paper is that while the overall cost minimization problem is NP-Hard for a calculation that takes into account all the pixels in the reference image, it is of polynomial-time complexity if the disparity assignments and the cost minimization is carried out for a single line of pixels at a time. Van Meerbergen et el. demonstrated that such was the case for at least a dynamic-programming based solution to disparity assignments along what they called was a "scan-line" in the images.

- **Hirschmüller generalized the computational feasibility of cost minimization along one scanline into a multi-path approach to solving the cost minimization problem for all the pixels in the reference images.**

- Let the symbol $\mathbf{r}$ denote the direction along which we wish to estimate the cost as defined in Eq. (10). Let the estimated cost along the direction $\mathbf{r}$ be designated $\mathcal{C}^{\mathbf{r}}$.

- Based on the formula in Eq. (10), we now write the following formula for estimating the total cost along the direction $\mathbf{r}$ in the reference image:

$$\mathcal{C}^{\mathbf{r}}(\mathbf{p},d) \;=\; \mathcal{C}_{data}(\mathbf{p},d) \quad + \\ \min\left(\mathcal{C}^{\mathbf{r}}(\mathbf{p}-\mathbf{r},d-1)+P_1, \quad \mathcal{C}^{\mathbf{r}}(\mathbf{p}-\mathbf{r},d+1)+P_1, \quad \min_i\Big(\mathcal{C}^{\mathbf{r}}(\mathbf{p}-\mathbf{r},i)\Big)+P_2 \right) \tag{11}$$

- To understand the above formula, note that for any direction vector $\mathbf{r}$ our goal is fill the "Disparity Volume" over the reference image with cost estimates by $\mathcal{C}^{\mathbf{r}}(\mathbf{p},d)$ as defined above. To the extent it is possible, we need this value for all possible disparity labels at every pixel $\mathbf{p}$ in the reference image. Each disparity label in the $[d_{min}, d_{max}]$ range defines a disparity level in the Disparity Volume over the reference image. Figure 12 depicts the case when we are interested in estimating $\mathcal{C}^{\mathbf{r}}(\mathbf{p},d)$ for the disparity label d_k at the pixel $\mathbf{p}$. The direction of $\mathbf{r}$ would be the arrow from the pixel at $\mathbf{p}-\mathbf{r}$ to the pixel at $\mathbf{p}$ in the figure.

- To revisit Eq. (11) and elaborate on it through the depiction in the above figure, the formula in Eq. (11) says that for the Discontinuity Cost we need to choose the minimum of the following three values: (1) We reach down to the level d_{k-1} in the Disparity Volume and, for the previously addressed pixel at $\mathbf{p}-\mathbf{r}$, we fetch the value stored there for $\mathcal{C}^{\mathbf{r}}$. To that value we add the weight P_1; (2) We reach up into the level d_{k+1} in the disparity value and, for the previously addressed pixel $\mathbf{p}-\mathbf{r}$, we fetch the

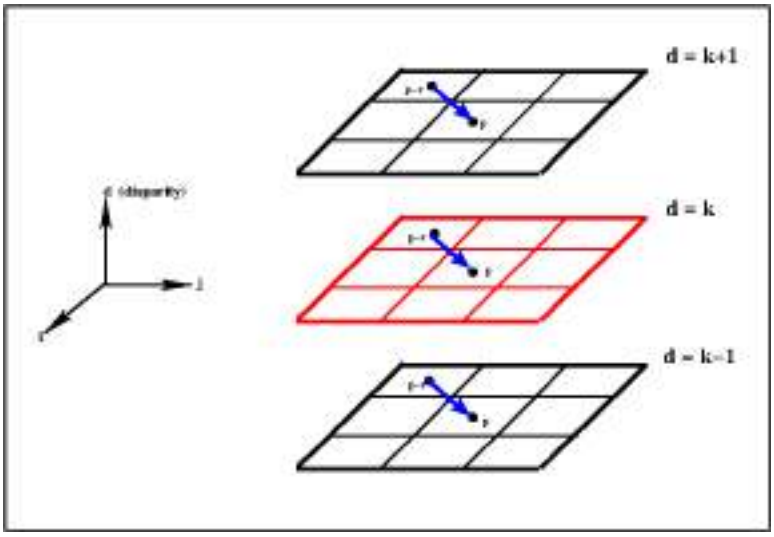

Figure 12: *Recursive calculation of the disparity discontinuity cost at pixel* $\mathbf{p}$ *for the disparity label* $d_p = k$ *in the direction* $\mathbf{r}$. *The recursive calculation is based on the previously computed discontinuity costs at the pixel* $\mathbf{p} - \mathbf{r}$ *through the formula shown in Eq. (11).*

value stored there for $\mathcal{C}^{\mathbf{r}}$. To that value we again add the weight P_1; and (3) We vertically scan the Disparity Volume at the location $\mathbf{p} - \mathbf{r}$ for all other disparity labels and record the minimum of the $\mathcal{C}^{\mathbf{r}}$ values stored there. To the value we fetch we add the weight P_2. The Discontinuity Cost at pixel $\mathbf{p}$ at the disparity level d_k is then set to the minimum of these three numbers.

- In addition to the above described recursive computation of the discontinuity costs at pixel $\mathbf{p}$ for every possible disparity label, the formula shown in Eq. (11) also requires us to compute the data costs. Data cost computational are straightforward — all they require is a comparison the pixel neighborhoods at the pixel

$\mathbf{p} = (i, j)$ in the reference image with the same neighborhood in the other image at the pixel $(i, j - d)$.

- In a practical implementation of the SGM algorithm, the computation of the data costs at all points in the Disparity Volume is a part of the initialization that is carried out before invoking the recursions for calculating the Discontinuity Costs.

- After the total costs $\mathcal{C}^{\mathbf{r}_i}(\mathbf{p}, d)$ have been computed for a set of directions $\mathbf{r}_i, i = 1, 2, ...$, they are simply aggregated as follows:

$$S(\mathbf{p}, d) \quad = \quad \sum_i \mathcal{C}^{\mathbf{r}_i}(\mathbf{p}, d) \tag{12}$$

  This is referred to as the *aggregation step* in the algorithm.

- From the Disparity Volume, of size $H \times W \times (d_{max} - d_{min})$, filled with value of $S(\mathbf{p}, d)$ as explained above, we now retain at each pixel $\mathbf{p}$ of the reference image that disparity d for which the $S(\mathbf{p}, d)$ value is the least:

$$d_{\mathbf{p}} \quad = \quad \underset{d}{\text{argmin}} \; S(\mathbf{p}, d) \tag{13}$$

- That takes us to the question of how many directions $\mathbf{r}_i$ to use in the aggregation step described above. In the most commonly used

implementations of SGM, one uses either 8 or 16 directions. We will refer to these as SGM-8 and SGM-16.

- Figure 13 on the next page shows the 8 directions for computing the direction dependent costs $\mathcal{C}^{\mathbf{r}_i}$ for $i = 1 \ldots 8$.

- In all cases, the current pixel is in the 4^{th} row and 4^{th} column, counting the rows and columns at the upper-left hand corner.

- The red arrows for each direction indicate how the recursion works in each case. For example, in (a), in order to estimate $\mathcal{C}^{\mathbf{r}}$ at the current pixel, you would need to have previously computed the values for this cost at the pixel immediately to the left. On the other hand, for the case shown in (e), the current value for $\mathcal{C}^{\mathbf{r}}$ depends on the previously computed values for the same at the pixel that is the upper-left diagonal neighbor of the current pixel.

- Figure 13 also presents the scanning strategy to be used for each of the 8 directions. For example, for the direction of recursion shown in (a), it makes to sense to scan the pixels left to right and top to bottom. The two perpendicular arrows you see at top left of the image array in (a) indicate that you start scanning the image at the upper left corner and go from pixel-to-pixel in each row and row-to-row. When you are the current pixel, the shaded area of the image array indicates the pixels already visited.

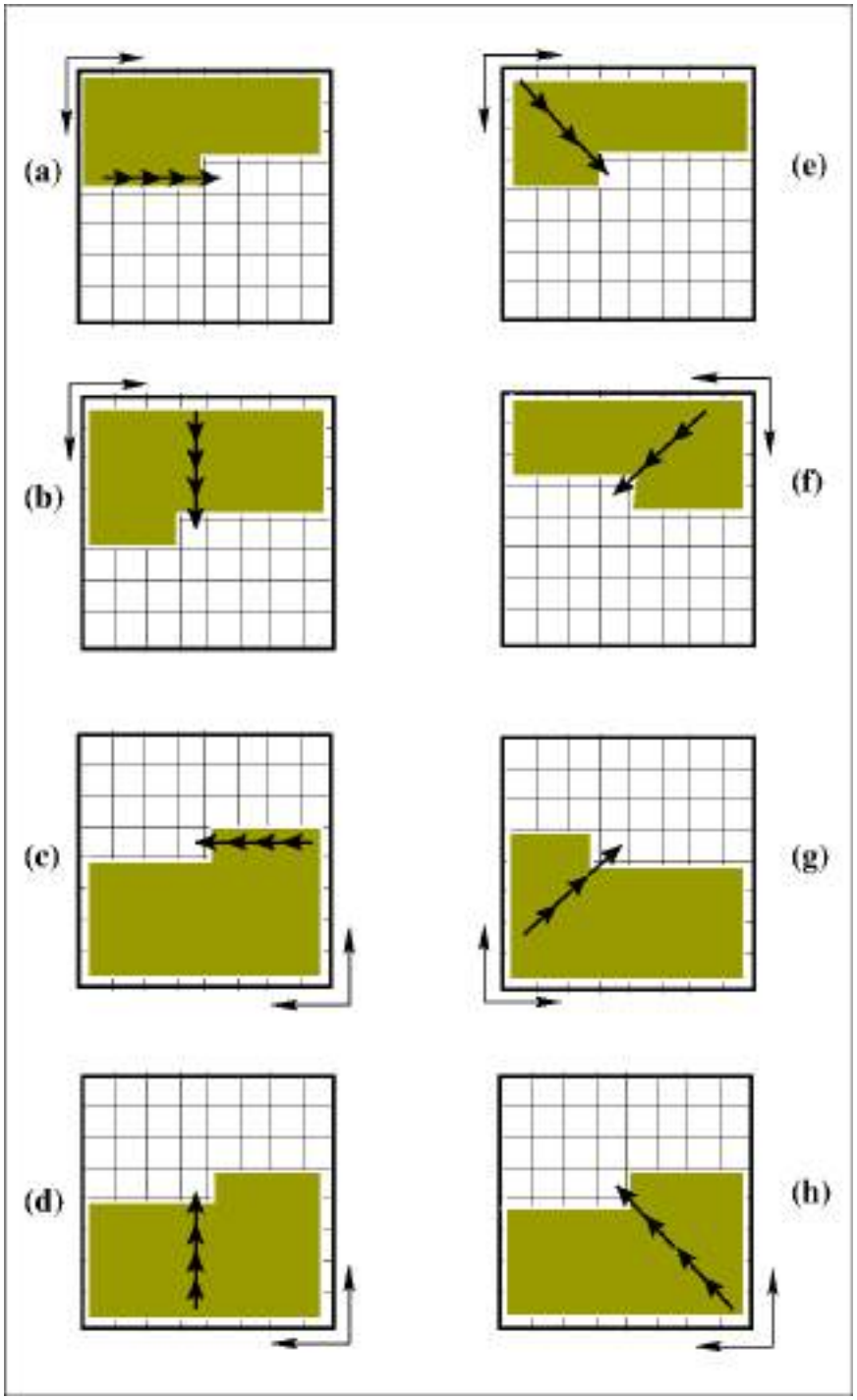

Figure 13: *Computation of the $\mathcal{C}^{\mathbf{r}_i}$ for 8 different values of the direction vectors $\mathbf{r}_i$.*

- It's interesting to see that the left-to-right and top-to-bottom scanning strategy works best for the recursion directions shown in (a), (b), and (e). On the other hand, the image scanning would need to be from right-to-left and top-to-bottom for the recursion direction depicted in (f).

- For the rest of the recursion directions, for cases shown in (c), (d), (h), you would need to scan the image right-to-left and bottom-to-top. Finally, for the recursion direction in (h), you would to scan the image right-to-left and bottom-to-top.

- The next figure, Figure 14, shows the SGM recursion being implemented in a sweep mode. For each of the eight directions of recursive estimation, you simultaneously estimates the costs along a sweep line. Since the sweep directions align with the cardinal axes or the diagonals as shown, updating the coordinates of the pixels that would need to be updated at the next position of the sweep-line is straightforward. For the sweep lines parallel to the image axes, it is entirely trivial. And, if you think about it, it is just as easy for the diagonal sweep lines.

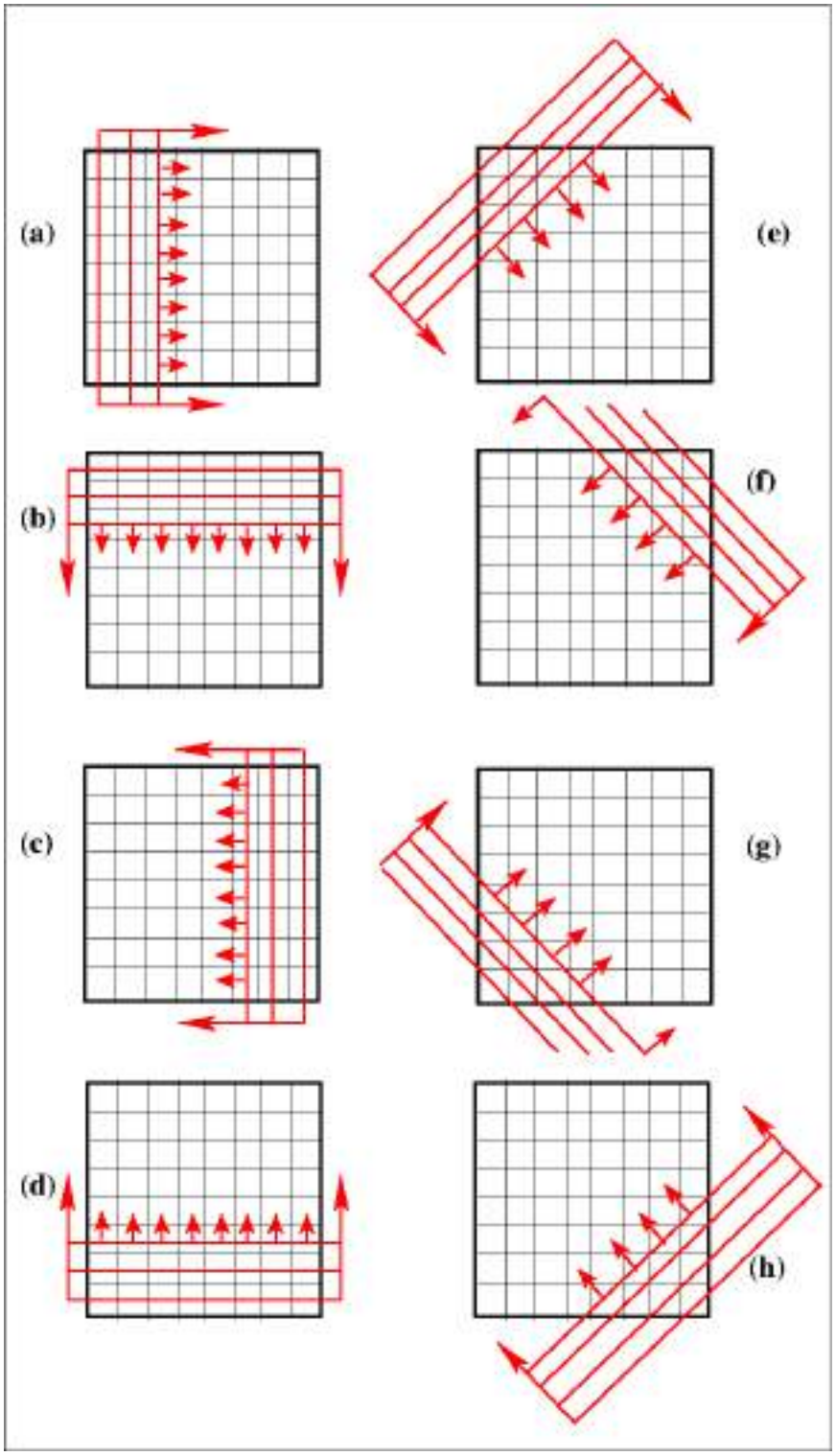

Figure 14: *The sweep mode for calculating the costs* $\mathcal{C}^{\mathbf{r}_i}$ *for the 8 different directions in SGM-8.*

Back to TOC

7: Clean-up of the SGM Computed Disparities

- There are various kinds errors that can creep into the process of disparity mapping the pixels in the reference image. The sources of these errors are: (1) image regions with no or low or repetitive textures; and (2) gaps in the disparity map on account of occlusions in the scene as we talked about in Section 3.

- In regions with no or low textures, the data costs at neighboring pixels in the reference image lose their ability to discriminate between different possible matches in the other image for a given pixel in the reference image. This creates disparity noise during the execution of the algorithm.

- When a disparity map has gaps on account of occlusions, the logic of SGM will willy-nilly assign disparity values there since the final disparity assigned to a pixel is the argmax along the disparity axis at that location in the reference image. Depending on the extent of the gap, and also depending on the coherence of the disparity assignments created by the argmax operation at adjoining pixels, you may either see disparity outliers or you could end up with random floating patches of disparity values over the values elsewhere.

- The outliers can easily be rejected by applying a small 3×3 median filter to the estimated disparity map.

- Getting rid of the random floaters in a disparity map is a bit trickier since you do not want to get rid of the disparities corresponding to small protrusions in the 3D surface being imaged. Usually, for those portions of a disparity map that actually correspond to meaningful structures in the scene, the disparity labels are likely to vary by just one unit as you go from one pixel to the next. That makes it easy to identify the regions delimited by rather large jumps in disparity labels. All you have to do is to scan a computed disparity map and mark the pixels with large jumps in disparity labels. If the bounded regions thus identified are small enough in size, you just discard them. However, getting rid of "disparity floaters" in this manner will create gaps in the disparity map. You would declare the pixels thus identified as possessing invalid disparities Recovering the disparities may be possible under certain conditions. See the paper by Hirschmüller for his recommendations on this issue.

- An important method for detecting invalid disparities over the reference image is constructing a second disparity map, but this time after you have switched the roles of two images. That is, what was previously the *other* image now becomes the new *reference* image vis-a-vis the other image.

- For obvious reasons, the SGM algorithm would work in exactly the same fashion as before.

- We will denote the first disparity map by $d_{L\rightarrow R}$ and the second by $d_{L\leftarrow R}$.

- Let $(\mathbf{p}, \mathbf{q})$ be a pair of corresponding pixels discovered when we constructed the $d_{L\rightarrow R}$ disparity map. For the pixel $\mathbf{p}$ in the "left" image, the corresponding pixel $\mathbf{q}$ in the "right" image.

- When there are no errors in the assignment of the disparity labels, we woulld $d_{L\rightarrow R}(\mathbf{p})$ to be equal to $d_{L\leftarrow R}(\mathbf{q})$. However, to account for any residual errors even under the best of conditions, we may allow one of these two disparitiy assignments to differ by at most one unit. That leads to the following rule for discovering additional invalid pixels in your original reference images:

$$\begin{aligned} d_(\mathbf{p}) &= d_{L\rightarrow R}(\mathbf{p}) \quad if \quad |d_{L\rightarrow R}(\mathbf{p}) - d_{L\leftarrow R}(\mathbf{q})| \leq 1 \\ &= d_{invalid} \qquad otherwise \end{aligned} \tag{14}$$