Stereo Correspondence
CSE 6367: Computer Vision

Instructor: William J. Beksi

Introduction

- **Stereo matching** is the process of taking two more more images and estimating a 3D model of the scene by finding matching pixels in the images and converting their 2D positions into 3D depths
- A complete 3D model of an object can be built from a sparse or dense **depth map** that assigns relative depths to pixels in the input images

Introduction

- Why are people interested in stereo matching?
- From the earliest inquires into visual perception, it was known that we perceive depth based on the differences in appearance between the left and right eye

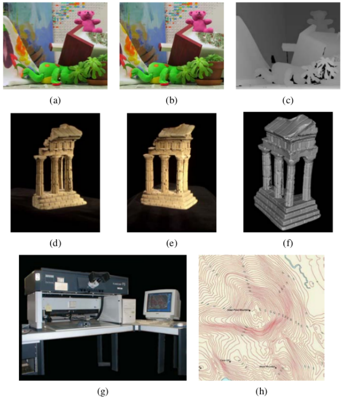
Introduction

- Under simple imaging configurations (both eyes or cameras looking straight ahead), the amount of horizontal motion or **disparity** is inversely proportional to the distance from the observer
- While the basic physics and geometry relating visual disparity to scene structure are well understood, automatically measuring this disparity by establishing dense and accurate inter-image *correspondences* is a challenging task

Introduction

- In computer vision, the topic of stereo matching has been one of the most widely studied and fundamental problems, and continues to be an area of high research activity
- While early stereo matching algorithms for photogrammetric matching concentrated mainly on aerial imagery, computer vision applications include modeling the human visual system, robotic navigation and manipulation, view interpolation, image-based rendering, 3D model building, etc.

Introduction



- Stereo reconstruction techniques can convert (a-b) a pair of images into (c) a depth map; or (d-e) a sequence of images into (f) a 3D model; (g) an analytical stereo plotter can generate (h) contour plots

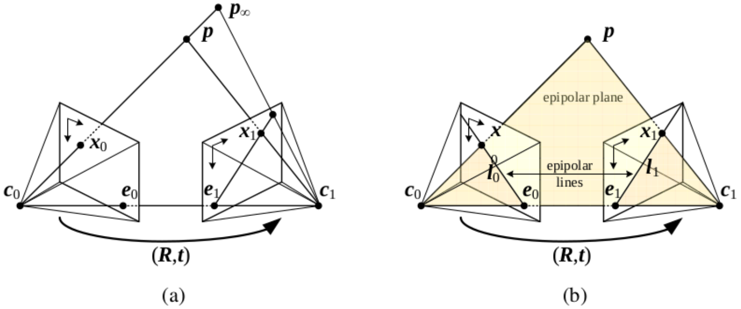
Finding Point Correspondences

- Given a pixel in one image, how can we compute its correspondence in the other image?
- We've seen a variety of techniques to match pixels based on their local appearance
- In the case of stereo matching, we have some additional information available, namely the positions and calibration data for the cameras that took the pictures of the same static scene

Finding Point Correspondences

- How can we exploit this information to reduce the number of potential correspondences and therefore both speed up the matching and increase its reliability?
- Using epipolar geometry, we find a pair of corresponding epipolar lines, which are the intersection of the two image planes with the epipolar plane that passes through both camera centers as well as the point of interest

Finding Point Correspondences



- Epipolar geometry: (a) epipolar line segment corresponding to one ray; (b) corresponding set of epipolar lines and their epipolar plane

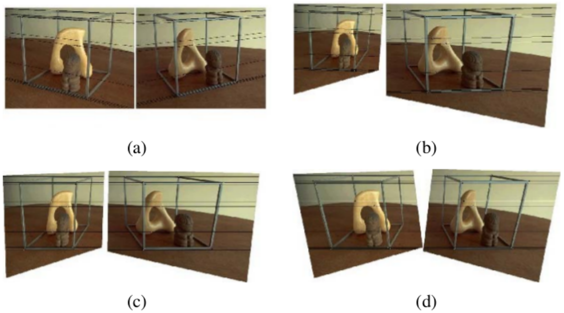
Rectification

- The epipolar geometry is implicit in the relative pose and calibrations of the camera, and can easily be computed from seven or more point matches using the fundamental matrix
- Once this geometry has been computed, we can use the epipolar line corresponding to a pixel in one image to constrain the search for corresponding pixels in the other image

Rectification

- An efficient way to perform this search is by first **rectifying** (i.e. warping) the input images so that corresponding horizontal scanlines are epipolar lines
- Afterwards, it is possible to match horizontal scanlines independently or to shift images horizontally while computing matching scores

Rectification



- Multi-stage stereo rectification: (a) original image overlaid with several epipolar lines; (b) images transformed so that epipolar lines are parallel; (c) images rectified so that epipolar lines are horizontal and vertical in correspondence; (d) final rectification that minimizes horizontal distortions

Rectification

- A simple way to rectify the two images is to first rotate both cameras so that they are looking perpendicular to the line joining the camera centers $\mathbf{c}_0$ and $\mathbf{c}_1$
- Since there is a degree of freedom in the *tilt*, the smallest rotations that achieve this should be used

Rectification

- Next, to determine the desired twist around the optical axes, make the *up vector* (the camera y axis) perpendicular to the camera center line
- This ensures the corresponding epipolar lines are horizontal and that the disparity for points at infinity is 0
- Finally, rescale the images if necessary to account for different focal lengths, magnifying the smaller image to avoid aliasing

Rectification

- The resulting **standard rectified geometry** is employed in stereo camera setups and algorithms, and leads to a simple inverse relationship between 3D depths Z and disparities, d,

  $$d = f\frac{B}{Z}$$

  where f is the focal length (measured in pixels), B is the baseline, and

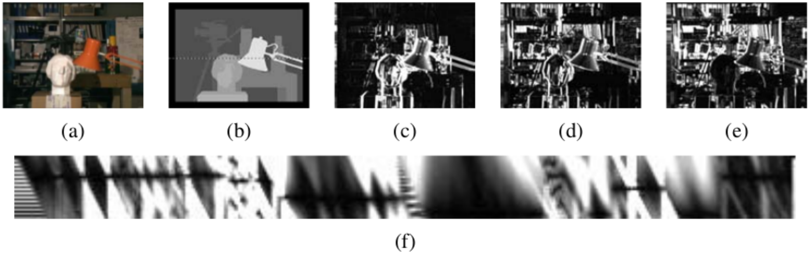
  $$x' = x + d(x, y), \quad y' = y$$

  describes the relationship between corresponding pixel coordinates in the left and right images

Rectification

- The task of extracting depth from a set of images then becomes one of estimating the **disparity map** $d(x, y)$
- After rectification, we can compute the similarity of pixels at corresponding locations (x, y) and $(x', y') = (x + d, y)$ and store them in a **disparity space image** (DSI), $C(x, y, d)$, for further processing

Rectification



(a) (b) (c) (d) (e)

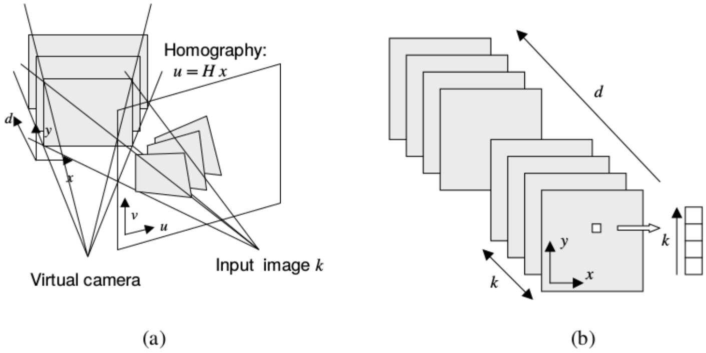
(f)

- Slices through a typical disparity space image: (a) original image; (b) ground truth disparities; (c-e) three (x, y) slices for $d = 10, 16, 21$; (f) an (x, d) slice for $y = 151$ (the dashed line in (b))

Plane Sweep

- An alternative to pre-rectifying the images before matching is to sweep a set of planes through the scene and to measure the **photoconsistency** of different images as they are re-projected onto these planes
- This process is commonly known as the **plane sweep** algorithm

Plane Sweep



- Sweeping a set of planes through a scene: (a) the set of planes seen from a virtual camera induces a set of homographies in any other source camera image; (b) the warped images from all the other cameras can be stacked into a generalized disparity space volume

Plane Sweep

- The last row of a full rank 4×4 projection matrix $\tilde{P}$ can be set to an arbitrary plane equation $\mathbf{p}_3 = s_3[\hat{\mathbf{n}}_0 \,|\, c_0]$
- The resulting four-dimensional projective transform (*collineation*) maps 3D world points $\mathbf{p} = [X, Y, Z, 1]$ into screen coordinates $\mathbf{x}_s = [x_s, y_s, 1, d]$, where the **projective depth** (or **parallax**) is 0 on the reference plane

Plane Sweep

- Sweeping d through a series of disparity hypotheses corresponds to mapping each input image into the **virtual camera** $\tilde{P}$ defining the disparity space through a series of homographies

$$\tilde{\mathbf{x}}_k \sim \tilde{P}_k \tilde{P}^{-1} \mathbf{x}_s = \tilde{H}_k \tilde{\mathbf{x}} + \mathbf{t}_k d = (\tilde{H}_k + \mathbf{t}_k \begin{bmatrix} 0 & 0 & d \end{bmatrix}) \tilde{\mathbf{x}}$$

  where $\tilde{\mathbf{x}}_k$ and $\tilde{\mathbf{x}}$ are the homogeneous pixel coordinates in the source and virtual (reference) images

- The members of the family of homographies $\tilde{H}_k(d) = \tilde{H}_k + \mathbf{t}_k \begin{bmatrix} 0 & 0 & d \end{bmatrix}$ are related to each other through a **planar homography**

Plane Sweep

- The choice of virtual camera and parameterization is application dependent and gives this framework its flexibility
- In many applications one of the input cameras (the *reference* camera) is used
- In other applications a camera centrally located between the two input cameras is preferable since it provides the needed per-pixel disparities to hallucinate the virtual middle image

Plane Sweep

- The choice of disparity sampling (i.e. the setting of the zero parallax plane and the scaling of integer disparities) is usually set to bracket the range of interest (*working volume*) while scaling disparities to sample the image in pixel shifts
- For example, when using stereo vision for obstacle avoidance in robot navigation it is most convenient to set up disparity to measure per-pixel elevation above the ground

Plane Sweep

- As each input image is warped onto the current planes parameterized by disparity d, it can be stacked into a **generalized disparity space image** (DSI) $\tilde{I}(x, y, d, k)$ for further processing
- In most stereo algorithms, the photoconsistency (e.g. sum of squared or robust differences) w.r.t the reference image I_r is calculated and stored in the DSI

$$C(x, y, d) = \sum_k \rho(\tilde{I}(x, y, d, k) - I_r(x, y))$$

Plane Sweep

- It is also possible to compute alternative statistics such as robust variance, focus, or entropy
- In addition, planes are not the only surfaces that can be used to define a 3D sweep through the space of interest (e.g. cylindrical surfaces are often used)

Plane Sweep

- Once the DSI has been computed, the next step in most stereo correspondence algorithms is to produce a univalued function in disparity space $d(x, y)$ that best describes the shape of the surfaces in the scene
- This can viewed as finding a surface embedded in the disparity space image that has some optimality property, such as lowest cost and best (piecewise) smoothness

Stereo Matching Algorithms

- Early stereo matching algorithms were *feature-based*, they extracted a set of potential matchable image locations, using interest operators or edge detectors, and then searched for corresponding locations in other images using a patch-based metric
- This limitation to **sparse correspondences** was partially due to computational resource limitations and also driven by a desire to limit the answers produced by stereo algorithms to matches with high certainty

Stereo Matching Algorithms

- More recent work in this area has focused on first extracting highly reliable features and then using these as *seeds* to grow additional matches
- Similar approaches have also been extended to wide baseline multi-view stereo problems and combined with 3D surface reconstruction

3D Curves and Profiles

- Another example of sparse correspondence is the matching of **profile curves** (or **occluding contours**)
- Profile curves occur at the boundaries of objects and interior self occlusions where the surface curves away from the camera viewpoint

3D Curves and Profiles

- The difficulty in matching profile curves is that in general, the locations of profile curves vary as a function of camera viewpoint
- Therefore, matching curves directly in two images and then triangulating these matches can lead to erroneous shape measurements

3D Curves and Profiles

- Fortunately, if three or more closely spaced frames are available, it is possible to fit a local circular arc to the locations of corresponding edgels and thus obtain semi-dense curved surface meshes directly from the matches
- Another advantage of matching such curves is that they can be used to reconstruct surface shape for untextured surfaces so long as there is a visible difference between foreground and background colors

3D Curves and Profiles

- Over the years, a number of different techniques have been developed for reconstructing surface shape from profile curves
- In one such technique, assume that the camera is moving smoothly enough that the local epipolar geometry varies slowly, i.e. the epipolar planes induced by the successive camera center and an edgel under consideration are nearly co-planar

3D Curves and Profiles

- The first step in the processing pipeline is to extract and link edges in each of the input images
- Next, edgels in successive images are matched using pairwise epipolar geometry, proximity and (optionally) appearance
- This provides a linked set of edges in the spatio-temporal volume, which is sometimes called the **weaving wall**

3D Curves and Profiles

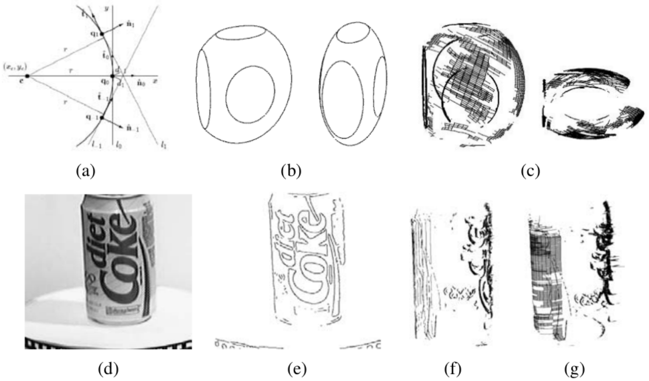
- To reconstruct the 3D location of an individual edgel, we project the viewing rays corresponding to its neighbors onto the instantaneous epipolar plane defined by the camera center, the viewing ray, and the camera velocity

- We then fit an **osculating circle** to the projected lines, parameterizing the circle by its centerpoint $\mathbf{c} = (x_c, y_c)$ and radius r

$$c_i x_c + s_i y_c + r = d_i$$

where $c_i = \hat{\mathbf{t}}_i \cdot \hat{\mathbf{t}}_0$ and $s_i = -\hat{\mathbf{t}} \cdot \hat{\mathbf{n}}_0$ are the cosine and sine of the angle between viewing ray i and central viewing ray 0, and $d_i = (\mathbf{q}_i, -\mathbf{q}_0) \cdot \hat{\mathbf{n}}_0$ is the perpendicular distance between i and the local origin q_0

3D Curves and Profiles


(a) (b) (c) (d) (e) (f) (g)

- Surface reconstruction from occluding contours

3D Curves and Profiles

- The resulting set of 3D points, along with their spatial (in-image) and temporal (between image) neighbors, form a 3D surface mesh with local normal and curvature estimates
- Note that whenever a curve is due to a surface marking or a sharp crease edge rather than a smooth surface profile curve, this shows up as a 0 or small radius of curvature
- Such curves resulted in isolated 3D space curves, rather than elements of smooth surface meshes, but can still be incorporated into the surface model at a later stage of interpolation

Dense Correspondence Algorithms

- While sparse matching algorithms are still occasionally used, most stereo matching algorithms today focus on **dense correspondence** since this required for applications such as image-based rendering or modeling
- This problem is more challenging than sparse correspondence since inferring depth values in textureless regions requires a certain amount of guesswork

Dense Correspondence Algorithms

- The algorithmic building blocks from which a set of dense correspondence stereo algorithms can be constructed using the following steps:
  1. Matching cost computation
  2. Cost (support) aggregation
  3. Disparity computation and optimization
  4. Disparity refinement

Dense Correspondence Algorithms

- *Local* (window-based) algorithms, where the disparity computation at a given point depends only on intensity values within a finite window, usually make implicit smoothness assumptions by aggregating support
- Some of these algorithms can be cleanly broken down into steps 1, 2, 3

Dense Correspondence Algorithms

- For example, the traditional sum-of-squared differences (SSD) algorithm can be described as:
    1. The matching cost is the squared difference of intensity values at a given disparity
    2. Aggregation is done by summing the matching cost over square windows with constant disparity
    3. Disparities are computed by selecting the minimal (winning) aggregated value at each pixel
- However, some local algorithms combine steps 1 and 2 and use a matching cost that is based on a support region, e.g. normalized cross-correlation and the rank transform

Dense Correspondence Algorithms

- *Global* algorithms make explicit smoothness assumptions and then solve a global optimization problem
- Such algorithms do not perform an aggregation step, but rather seek a disparity assignment (step 3) that minimizes a global cost function that consists of data (step 1) terms and smoothness terms

Dense Correspondence Algorithms

- The main distinctions among these algorithms is the minimization procedure used, e.g. simulated annealing, probabilistic (mean-field) diffusion, expectation maximization (EM), graph cuts, loopy belief propagation, etc.
- In between these two broad classes are certain iterative algorithms that do not explicitly specify a global function to be minimized, but whose behavior mimics closely that of iterative optimization algorithms

Similarity Measures

- The first component of any dense stereo matching algorithm is a similarity measure that compares pixel values in order to determine how likely they are to be in correspondence
- The most common pixel-based matching costs include sums of squared intensity differences (SSD) and absolute intensity differences (SAD)

Similarity Measures

- Other traditional matching costs include normalized cross-correlation, which behaves similarly to sum-of-squared-differences (SSD), and binary matching costs (i.e. match or no match)
- Due to their poor discriminability, simple binary matching costs are no longer used in dense stereo matching

Local Aggregation Methods

- **Local** and **window-based methods** aggregate the matching cost by summing or averaging over a **support region** in the DSI $C(x, y, d)$
- A support region can be either 2D at a fixed disparity, or 3D in the x-y-d space

Local Aggregation Methods

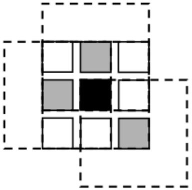
- Aggregation with a fixed support region can be performed using 2D or 3D convolution

$$C(x, y, d) = w(x, y, d) * C_0(x, y, d)$$

  or in the case of rectangular windows, using efficient moving average box-filters

- Shiftable windows can also be implemented efficiently using a separable sliding min-filter

Local Aggregation Methods

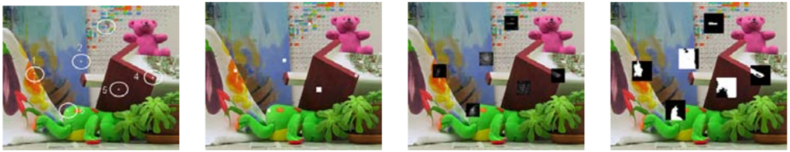


- Shiftable window: The effect of trying all 3×3 shifted windows around the black pixel is the same as taking the minimum matching scores across all *centered* (non-shifted) windows in the same neighborhood

Local Aggregation Methods

- Selecting among windows of different shapes and sizes can be performed more efficiently by first computing a summed area table
- Selecting the right window is important, windows must be large enough to contain sufficient texture yet small enough so that they do not straddle depth discontinuities

Local Aggregation Methods


(a) (b) (c) (d)

- Aggregation window sizes and weights adapted to image content: (a) original image with selected evaluation points; (b) variable windows; (c) adaptive weights; (d) segmentation-based
- Notice how the adaptive weights and segmentation-based technique adapt their support to similarly colored pixels

Local Aggregation Methods

- In local methods, the emphasis is on the matching cost computation and cost aggregation steps
- Computing the final disparities is trivial: simply choose at each pixel the disparity associated with the minimum cost value

Local Aggregation Methods

- Thus, these methods perform a local “winner take all” (WTA) optimization at each pixel
- A limitation of this approach (and many other correspondence algorithms) is that uniqueness of matches is only enforced for one image (the *reference image*), while points in the other image might match multiple points, unless cross-checking and subsequent hole filling is used

Sub-pixel Estimation and Uncertainty

- Most stereo correspondence algorithms compute a set of disparity estimates in some discretized space, and for applications such as robot navigation or people tracking these may be adequate
- However, for image-based rendering, such quantized maps lead to very unappealing view synthesis results, i.e. the scene appears to be made up of many thin shearing layers
- To remedy this situation, many algorithms apply a sub-pixel refinement stage after the initial discrete correspondence stage

Sub-pixel Estimation and Uncertainty

- Sub-pixel disparity estimates can be computed in a variety of ways, including iterative gradient descent and fitting a curve to the matching costs at discrete disparity levels
- This provides an easy way to increase the resolution of a stereo algorithm with little computation
- However, to work well, the intensities being matched must vary smoothly and the regions over which these estimates are computed must be on the same (correct) surface

Sub-pixel Estimation and Uncertainty

- Besides sub-pixel computations, there are other ways of post-processing the computed disparities
- Occluded areas can be detected using cross-checking, i.e. comparing left-to-right and right-to-left disparity maps
- A median filter can be applied to clean up spurious mismatches, and holes due to occlusion can be filled by surface fitting or by distributing neighboring disparity estimates

Sub-pixel Estimation and Uncertainty

- Another kind of post-processing, which can be useful in later processing stages, is to associate **confidences** with per-pixel depth estimates
- This can be done by looking at the curvature of the correlation surface, i.e. how strong the minimum in the DSI image is at the winning disparity

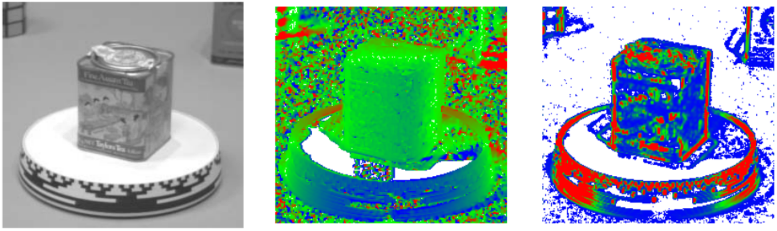
Sub-pixel Estimation and Uncertainty

- Under the assumption of small noise, photometrically calibrated images, and densely sampled disparities, the variance of a local depth estimate can be estimated as

$$Var(d) = \frac{\sigma_I^2}{a}$$

where a is the curvature of the DSI as a function of d, which can be measured using a local parabolic fit or by squaring all the horizontal gradients in the window, and σ_I^2 is the variance of the image noise, which can be estimated from the minimum SSD score

Local Aggregation Methods


(a) (b) (c)

- Uncertainty in stereo depth estimation: (a) input image; (b) estimated depth map (blue is closer); (c) estimated confidence (red is higher)
- More textured areas have higher confidence

Application: Stereo-based Head Tracking

- A common application of real-time stereo algorithms is for tracking the position of a user interacting with a computer or game system
- The use of stereo can dramatically improve the reliability of such a system compared to trying to use monocular color and intensity information
- Once recovered, this information can be used in a variety of applications including controlling a virtual environment or game

Application: Stereo-based Head Tracking

- The use of head tracking to control a users' virtual viewpoint while viewing a 3D object or environment on a computer monitor is sometimes called *fish tank virtual reality*
- Early versions of these systems used mechanical head tracking devices and stereo glasses
- Today, such systems can be controlled using stereo-based head tracking and stereo glasses can be replaced with autostereoscopic displays

Global Stereo Matching

- Stereo matching methods perform some **global optimization** or iteration steps after the disparity computation phase
- The aggregation step is often skipped since the global smoothness constraints perform a similar function

Global Stereo Matching

- Many global methods are formulated in an energy-minimization framework where the objective is to find a solution d that minimizes a global energy

$$E(d) = E_d(d) + \lambda E_s(d)$$

- The data term, $E_d(d)$, measures how well the disparity function d agrees with the input image pair
- Using our previously defined disparity space image, we define this energy as

$$E_d(d) = \sum_{(x,y)} C(x, y, d(x, y))$$

where C is the (initial or aggregated) matching cost DSI

Global Stereo Matching

- The smoothness term $E_s(d)$ encodes the smoothness assumptions made by the algorithm
- To make the optimization computationally tractable, the smoothness term is often restricted to measuring only the differences between neighboring pixels' disparities

$$E_s(d) = \sum_{(x,y)} \rho(d(x,y) - d(x+1,y)) + \rho(d(x,y) - d(x,y+1))$$

where ρ is some monotonically increasing function of disparity difference

Global Stereo Matching

- The terms in E_s can also be made to depend on the intensity differences, e.g.

$$\rho_d(d(x,y) - d(x+1,y)) \cdot \rho_I(||I(x,y) - I(x+1,y)||)$$

  where ρ_I is some monotonically decreasing function of intensity differences that lowers smoothness costs at high-intensity gradients

- This idea encourages disparity discontinuities to coincide with intensity or color edges and improves the performance of global optimization approaches

Global Stereo Matching

- Once the global energy has been defined, a variety of algorithms can be used to find a (local) minimum
- Efficient methods to solve the global optimization problem include *max flow* and *graph cuts*
- While global optimization techniques currently produce the best stereo matching results, alternative approaches worth studying include *cooperative algorithms* and *coarse-to-fine and incremental warping*

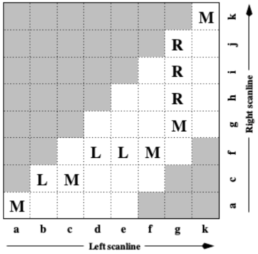
Dynamic Programming

- A different class of global optimization is based on **dynamic programming**
- Dynamic programming can find the global minimum for independent scanlines in polynomial time and was first used for stereo vision in sparse, edge-based methods

Dynamic Programming

- More recent approaches have focused on the dense (intensity-based) scanline matching problem
- These approaches work by computing the minimum-cost path through the matrix of all pairwise matching costs between two corresponding scanlines, i.e. through a slice of the DSI

Dynamic Programming



- Stereo matching using dynamic programming: For each pair of corresponding scanlines, a minimizing path through the matrix of all pairwise matching costs (DSI) is selected
- Lowercase letters (a-k) symbolize the intensities along each scanline, uppercase letters represent the selected path through the matrix
- Matches are indicated by M, while partially occluded points (which have a fixed cost) are indicated by L (only visible in left image) or R (only visible in right image)

Dynamic Programming

- To implement dynamic programming for a scanline y, each entry (state) in a 2D cost matrix $D(m, n)$ is computed by combining its DSI value

$$C'(m, n) = C(m + n, m - n, y)$$

with one of its predecessor cost values

Dynamic Programming

- The aggregated match costs can be recursively computed as

$$\begin{aligned} D(m,n,M) &= \min(D(m-1,n-1,M), D(m-1,n,L), \\ &\qquad D(m-1,n-1,R)) + C'(m,n) \\ D(m,n,L) &= \min(D(m-1,n-1,M), D(m-1,n,L)) + O \\ D(m,n,R) &= \min(D(m,n-1,M), D(m,n-1,R)) + O \end{aligned}$$

where O is a per-pixel occlusion cost

Dynamic Programming

- Problems with dynamic programming stereo include the selection of the right cost for occluded pixels and the difficulty of enforcing inter-scanline consistency
- Another problem is that the dynamic programming approach requires enforcing *monotonicity* or *ordering constraint*
- This constraint requires that the relative ordering of pixels on a scanline remain the same between the two views, which may not be the case in scenes containing narrow foreground objects

Dynamic Programming

- An alternative to traditional dynamic programming is to neglect the vertical smoothness constraints and simply optimize independent scanlines in the global energy function using a recursive algorithm

$$D(x, y, d) = C(x, y, d) + \min_{d'}\{D(x-1, y, d') + \rho_d(d-d')\}$$

- The advantage of this **scanline optimization** algorithm is that it computes the same representation and minimizes a reduced version of the same energy function as the full 2D energy function

- Unfortunately, it still suffers from the same streaking artifacts as dynamic programming

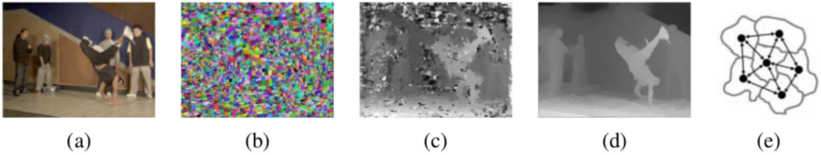
Dynamic Programming

- A better approach is to evaluate the cumulative cost function from multiple directions, e.g. from the eight cardinal directions: N, E, W, S, NE, SE, SW, NW
- The resulting *semi-global* optimization performs quite well and is extremely efficient to implement
- Even though dynamic programming and scanline optimization algorithms do not generally produce *the* most accurate stereo reconstructions, when combined with sophisticated aggregation strategies, they can produce very fast and high-quality results

Segmentation-based Techniques

- While most stereo matching algorithms perform their computations on a per-pixel basis, some techniques first segment the images into regions and then try to label each region with a disparity
- For example, such algorithms segment the reference image, estimate per-pixel disparities using a local technique, and then do local plane fits inside each segment before applying smoothness constraints between neighboring segments

Segmentation-based Techniques



- Segmentation-based stereo matching: (a) input color image; (b) color-based segmentation; (c) initial disparity estimates; (d) final piecewise-smoothed disparities; (e) MRF neighborhood defined over the segments in the disparity space distribution

Matching Multiple Images

- While matching pairs of images is a useful way of obtaining depth information, matching more images can lead to even better results
- **Multi-view stereo** (matching multiple images) can not only improve the quality of depth maps, but also allow for the creation of complete 3D object models

Matching Multiple Images

- As seen using the plane sweep technique, it is possible to resample all neighboring k images at each disparity hypothesis d into a generalized disparity space volume $\tilde{I}(x, y, d, k)$
- The simplest way to take advantage of these additional images is to sum up their differences from the reference image I_r

$$C(x, y, d) = \sum_k \rho(\tilde{I}(x, y, d, k) - I_r(x, y))$$

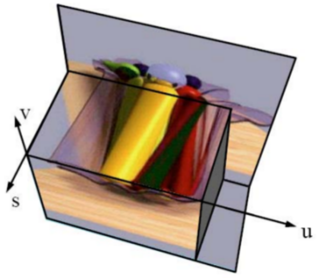
Matching Multiple Images

- This is the basis of the well-known sum of squared-distances (SSD) and SSAD approaches, which can be extended to reason about likely patterns of occlusion
- In addition, the baselines can be adapted to the expected depth in order to get the best tradeoff between geometric accuracy (wide baseline) and robustness to occlusion (narrow baseline)

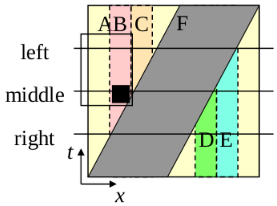
Matching Multiple Images

- A useful way to visualize the multi-frame stereo estimation problem is to examine the **epipolar plane image** (EPI) formed by stacking corresponding scanlines from all the images
- As the camera translates horizontally (in a standard horizontally rectified geometry), objects at different depths move sideways at a rate inversely proportional to their depth

Matching Multiple Images



(a)



(b)

- Epipolar plane image (EPI): (a) the Lumigraph (light field) is the 4D space of all light rays passing through a volume of space, objects at different depths move sideways with velocities (slopes) proportional to their inverse depth; (b) the EPI showing the three images (left, middle, right) as slices through the EPI volume

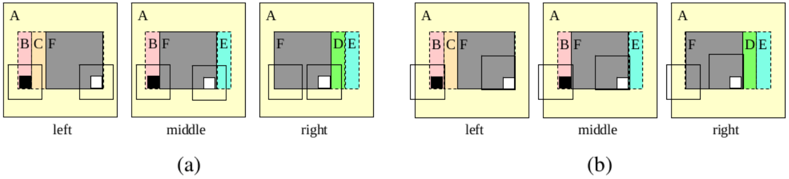
Matching Multiple Images

- Foreground objects occlude background objects, which can be seen as *EPI-strips* occluding other strips in the EPI
- If we are given a dense enough set of images, we can find such strips and reason about their relationships in order to both reconstruct the 3D scene and make inferences about translucent objects
- Alternatively, we can treat the series of images as a set of sequential observations and merge them using Kalman filtering or maximum likelihood inference

Matching Multiple Images

- When fewer images are available, it becomes necessary to fall back on aggregation techniques such as sliding windows or global optimization
- With additional images, however, the likelihood of occlusions increases
- To handle occlusions, a subset of neighboring frames can be selected in order to discount those images where the region of interest is occluded

Matching Multiple Images



- Spatio-temporally shiftable windows: A three-image sequence which has a moving frontal gray square and a stationary background, regions B, C, D, and E are partially occluded; (a) the SSD will make mistakes when matching pixels in these regions; (b) shiftable windows help mitigate the problems in partially occluded regions and near depth discontinuities

Matching Multiple Images

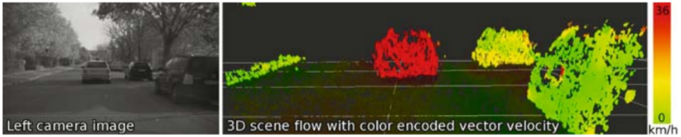
- While computing a depth map from multiple inputs outperforms pairwise stereo matching, even more dramatic improvements can be obtained by estimating multiple depth maps simultaneously
- The existence of multiple depth maps enables more accurate reasoning about occlusions, as regions which are occluded in one image may be visible (and matchable) in others
- The multi-view reconstruction problem can be formulated as the simultaneous estimation of depth maps at key frames while maximizing consistency between disparity estimates at different frames

Matching Multiple Images

- A closely related topic to multi-frame stereo estimation is **scene flow** where multiple cameras are used to capture a dynamic scene
- The task is then to simultaneously recover the 3D shape of the object at every instant in time and to estimate the full 3D motion of every surface point between frames
- Scene flow can be used to support both spatial and temporal view interpolation

Matching Multiple Images

(a)



(b)

- 3D scene flow: (a) computed from a multi-camera dome surrounding a human subject; (b) computed from stereo cameras mounted on a moving vehicle

Summary

- The field of stereo correspondence and depth estimation is one of the oldest and most widely studied topics in computer vision
- Early algorithms focused on finding sparse correspondences due to computational limitations and the desire to find appearance-invariant correspondences
- Upon setting up a DSI, the actual depth map can be computed from the costs of dense correspondences using some form of optimization or selection criterion
- Algorithms for multi-view stereo can be used to compute depth maps using several images