CSE 659A: Advances in Computer Vision

Spring 2020: T-R: 1:00-2:20pm @ Whitaker 216

Instructor: Ayan Chakrabarti (ayan@wustl.edu).

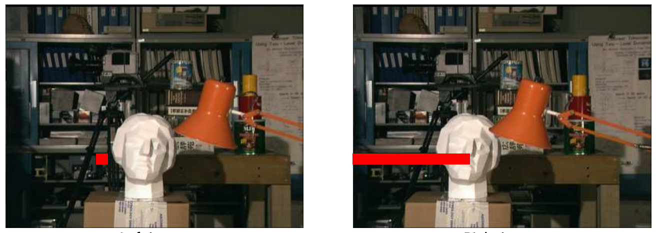
http://www.cse.wustl.edu/~ayan/courses/cse659a/

Feb 11, 2020

GENERAL

- Homework review 2 posted. Due Feb 20 (next Thursday).
- Project 1 Proposals due through Canvas this Friday, Feb 14, 11:59pm.
- Text-box submission: Mention paper(s) you will analyze, and write 3-4 sentences on what you plan to do.
- If you select a paper we have discussed in class, there will be a higher bar: you will be evaluated on the analysis that goes beyond what was done in lecture.
- Other ways to find good papers: look at the references in the papers we have discussed. Also look at work that came after those papers and cite them. You can use Google Scholar (https://scholar.google.com/) to find these.

PLANE SWEEP STEREO



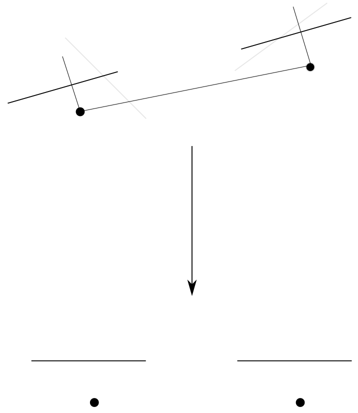
Standard Rectified Stereo

(x,y) matches to (x-d,y) for d > 0

We build cost-volume C[x,y,d]

Cameras differ only by translation in the x direction

PLANE SWEEP STEREO



Arbitrary camera pair

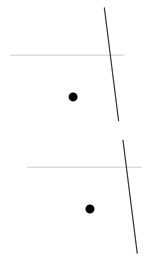
Rotate views about camera center so that viewing direction is orthogonal to translation between two cameras.

Can be done with a Homography

Standard Rectified Stereo

Cameras differ only by translation in the x direction

PLANE SWEEP STEREO



What if the two cameras are translated in pretty much the viewing direction ?

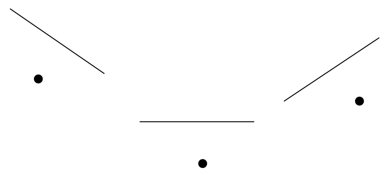
For example, you don't have a camera pair but two images from a camera moving forward ?

You can use a homography, but there's very little overlap between the new and old views.

Standard Rectified Stereo

Cameras differ only by translation in the x direction

PLANE SWEEP STEREO



What if you have more than 2 cameras ?

Rectification is pair-wise: you can't rectify all three together

Standard Rectified Stereo

Cameras differ only by translation in the x direction

PLANE SWEEP STEREO

Left Image

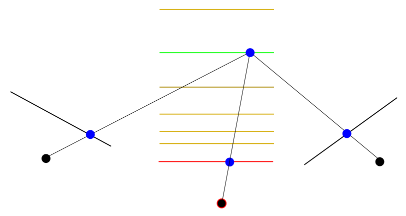
Right Image

We build cost-volume C[x,y,d]

- Since we work in pixels, we consider discrete values of d: 0, 1, 2, 3, ...
- We know z = f t / d.
- So C[x,y,d] = C[x,y,z = ft/d]
- We're building hypothesis for z = ∞, ft/1, ft/2, ft/3, ...

Cameras differ only by translation in the x direction by amount t

PLANE SWEEP STEREO



Plane-sweep Stereo: Assume cameras are calibrated

- Choose a reference frame.
- Build a cost volume C[x,y,z] where z is viewing direction in the reference frame.
- Discretize z in some way. Typically, take uniform intervals of 1/z (just like for binocular rectified stereo)
- For each possible value of z for a pixel (x,y) in the reference frame, we can compute the co-ordinate of (x',y') in camera 2, (x",y") in camera 3, etc.
- Define C[x,y,z] as the sum of matching costs between all these pairs.
- Now apply an MRF / planarity model on this cost-volume.

MONOCULAR FLOW

- General case: compute optical flow between two image frames $I_t[x, y]$ and $I_{t+1}[x, y]$.
  - Find flow field $u[x, y], v[x, y]$ so that $I_t[x, y]$ matches $I_{t+1}[x + u[x, y], y + v[x, y]]$.
- Consider the case where the flow is due to movement of the camera, but the scene is static.
  - Camera moves with respect to the scene, but objects in scene don't move with respect to each other.

This is an epipolar system. You can add additional constraints the following way:

- First find a sparse set of matches (more on this later).
- Then, estimate fundamental matrix F such that $p_t^T F p_{t+1} = 0$ is a good fit for the pairs homogeneous co-ordinates (p_t, p_{t+1}) of these matches.
- Now your 2D flow-search problem turns into a 1-D search (don't rectify, use plane-sweep, search along a different line for each (x, y)).
  - You are adding the constraint that: $[x + u(x, y), y + v(x, y), 1]^T \quad F \quad [x, y, 1]^T = 0$
- Once you get a flow-field (and F) this way, does this give you depth ?

NO ! Because your cameras aren't calibrated.

MONOCULAR FLOW

- Once you get a flow-field (and F) this way, does this give you depth ?

NO ! Because your cameras aren't calibrated.

- If camera translates by $[t_x, t_y, t_z]$ or by $[2t_x, 2t_y, 2t_x]$, you will get the same F.
- Your flow corresponds to a disparity along each epipolar line. $d = t/z$. So there is a scale ambiguity because you don't know t.
- Your camera could be moving twice as fast but your scene could also be twice as far away. There is a global scaling in the world that you can't resolve.
- Nevertheless, you can use this to get a more accurate flow (subject of homework paper review).

MONOCULAR FLOW

Basic idea: estimate F, assume $|[t_x, t_y, t_z]| = 1$ (you know direction of translation, but not magnitude).

- Build a cost-volume $C[x, y, z]$
- z here corresponds to true z' as $z' = z|[t_x, t_y, t_z]|$
- But irrespective, the planarity model $1/z = \alpha x + \beta y + \gamma$ holds. Use our stereo tricks !

But what if objects in the scene *are* moving ?

- If there is only one object in the scene, and it is moving "rigidly"
  - We're back to an epipolar system.
  - Doesn't matter if camera moves relative to object or object moves relative to camera.

What if we have multiple objects moving in different ways ?

- Option 1: Assume that movement is small: most of the movement is from camera.
  - Require that the true flow is close to the epipolar lines instead of exactly on them.
  - Penalize $\left\|[x + u(x, y), y + v(x, y), 1]^T \quad F \quad [x, y, 1]^T\right\|^2$ instead of saying it is exactly 0.

MONOCULAR FLOW

But what if there are different objects, each moving differently but rigidly.

- Rigidly = within an object, no change in the relative distance between points between two frames.
- We can assume each object k has it's own epipolar system.
- Segment image into K different objects, each with it's own F_k.
- Flow for each pixel inside an object is given by a combination of it's F_k, and a single per-pixel scalar $z(x, y)$.
- You can further say that each object is piecewise planar. Constrain $z(x, y) = \alpha x + \beta y + \gamma$.
- Image = Collection of Objects. Each Object = Collection of planes.
- One F_k for each object. One (α, β, γ) for each plane within each object.

Can put this all inside one giant MRF ...

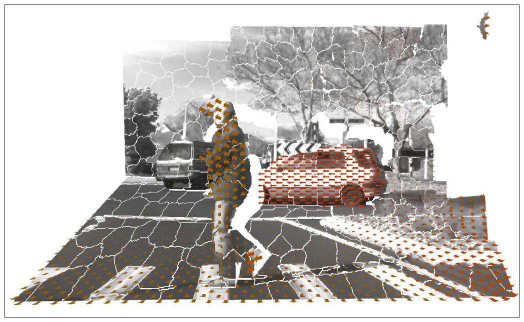
SCENE FLOW

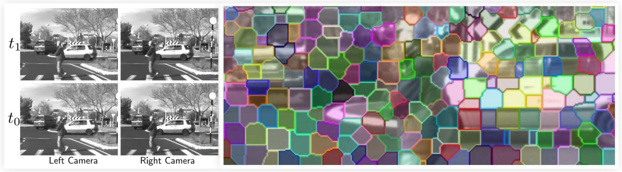
- What if you had a stereo pair of cameras that was moving ?
- So you have I_t^{left}, I_t^{right} and $I_{t+1}^{left}, I_{t+1}^{right}$.
- Since you have left-right calibration, you can get absolute depth.
- You can then also turn the flow-field on the image plane, to actual 3D motion flow.
  - $(x, y) \rightarrow (x', y')$ corresponds to (X, Y, Z) moved to (X', Y', Z').
- Aggregate information from matching between both left-right images and consecutive frames t and $t + 1$.

Vogel et al., "3D scene flow estimation with a piecewise rigid scene model," IJCV 2015.

SCENE FLOW

Vogel et al., "3D scene flow estimation with a piecewise rigid scene model," IJCV 2015.





OPTICAL FLOW

- But all of this processing still requires computing correspondences.
  - Even for the simplest case of monocular video with no moving objects, you need some matches to estimate the common F.
- Optical flow methods we talked about in 559A assumed "infinitesimal" motion
  - Lucas Kanade, Horn Schunck
  - Based on Taylor series expansion around (x, y)
- In these cases, the actual flow values u, v can have large magnitudes.
- So at the initial stage (prior to epipolar reasoning), each (x, y) in frame t can match to a large number of possible locations in frame $t + 1$.
- Pretty expensive to build a cost volume of $C[x, y, x', y']$

MATCH SEARCH FOR OPTICAL FLOW

Hu et al., "Efficient Coarse-to-Fine PatchMatch for Large Displacement Optical Flow," CVPR 2016.

- But before that, let's talk about the PatchMatch algorithm.
- The goal is to find a "flow" field $u(x, y), v(x, y)$ between images I and J which minimizes

$$\sum_{x,y} D(u(x,y), v(x,y), x, y), \qquad D(u, v, x, y) = \sum_{x'=x-P/2}^{x+P/2} \sum_{y'=y-P/2}^{y+P/2} \| I[x', y'] - J[x' + u, y' + v] \|^2$$

- Basically, find nearest neighbors for all overlapping $P \times P$ matches, in terms of euclidean distance in intensity space.
  - Can also use other distance metrics.
- Note that no need for one to one correspondence. No expectation that u, v is small. Doesn't even need to correspond to actual motion.
- In fact, the original PatchMatch paper was a graphics paper---used for image retargetting, inpainting, texture synthesis, ...

MATCH SEARCH FOR OPTICAL FLOW

Hu et al., "Efficient Coarse-to-Fine PatchMatch for Large Displacement Optical Flow," CVPR 2016.

- PatchMatch Algorithm

$$\sum_{x,y} D(u(x,y), v(x,y), x, y), \qquad D(u,v,x,y) = \sum_{x'=x-P/2}^{x+P/2} \sum_{y'=y-P/2}^{y+P/2} \|I[x',y'] - I[x'+u, y'+v]\|^2$$

- Iteration 0: Initialize $f(x,y) = [u(x,y), v(x,y)]$ randomly.
- At odd iterations: Check if replacing $f(x,y)$ with $f(x-1,y)$ or $f(x,y-1)$ reduces the cost.
- At even iterations: Check if replacing $f(x,y)$ with $f(x+1,y)$ or $f(x,y+1)$ reduces the cost.
- At all iterations, for each (x,y) try randomly sampling a (x',y') that is within some distance r of (x,y). Check if $f(x',y')$ lowers distance.
- Reduce the value of r across iterations.

MATCH SEARCH FOR OPTICAL FLOW

Hu et al., "Efficient Coarse-to-Fine PatchMatch for Large Displacement Optical Flow," CVPR 2016.

- PatchMatch Algorithm

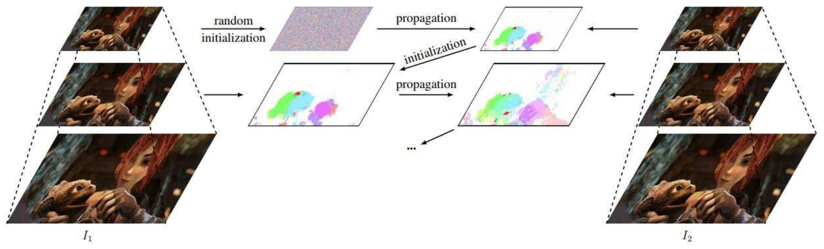
Works well in practice (at minimizing euclidean distance)!

- Note that looking at neighbors is not a smoothness 'constraint' like in MRF models. We are still making decisions based on the euclidean distance metric.
- Instead, it is way of getting to better results without doing an exhaustive search.
  - Example of a randomized algorithm. . . .
- If you apply PatchMatch to a pair of images, this gives you a flow-field where matching pixels will have similar appearance.
- This is enough if you want to closeness in appearance is all you care about. But these aren't accurate as motion estimates.
- Because, this doesn't enforce coherence in $u(x,y)$, $v(x,y)$---only minimizes intensity difference.
  - And we know that in many smooth regions, multiple matches will have small intensity difference.

MATCH SEARCH FOR OPTICAL FLOW

Hu et al., "Efficient Coarse-to-Fine PatchMatch for Large Displacement Optical Flow," CVPR 2016.

Key Insight: Apply patch-match in a coarse to fine way.

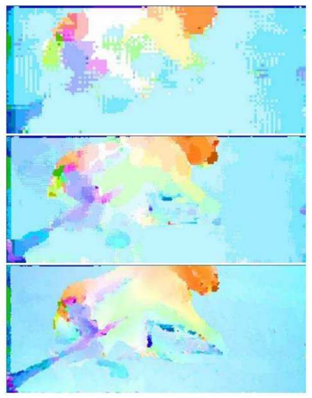


- Build image pyramid.
- Start at coarsest level apply patch-match.
- Initialize next level with those flow values.
- Restrict search to a fixed radius around matches from the coarse level.

Restriction = we are minimizing a different cost

MATCH SEARCH FOR OPTICAL FLOW

Hu et al., "Efficient Coarse-to-Fine PatchMatch for Large Displacement Optical Flow," CVPR 2016.



Coarse to fine: Colors visualize $u(x,y)$, $v(x,y)$ values.

MATCH SEARCH FOR OPTICAL FLOW

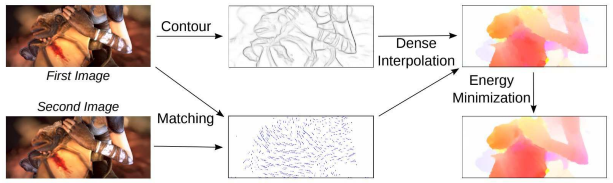
- These constrained nearest neighbors will give you pretty good motion estimates.
- But you'll still get a fraction of bad matches: where estimated flow is very different from true flow.

Detect and Remove Outliers

- Compute flow this way between $I_t \rightarrow I_{t+1}$ and separately $I_{t+1} \rightarrow I_t$
- Throw away flow values that don't complete the cycle:
  $$f_{t+1}([x, y] + f_t(x, y)) \neq -f_t(x, y)$$
- Now you are left with fewer matches, but you are sure most of them are accurate.

SPARSE TO DENSE FLOW FIELDS

Revaud et al., "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow," CVPR 2015.

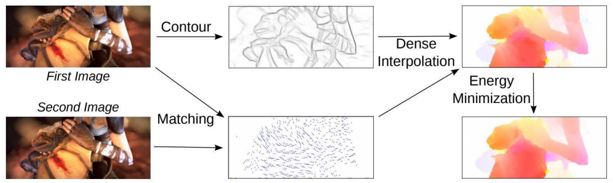


Dense Interpolation

- Solve energy minimization problem which says:
  - Output flow should be close to input sparse matches at locations they are available.
  - Neighboring pixels should have similar flow (smooth or explained by same 'affine' model)
  - Discontinuities should align with reference frame intensity edges (by lowering the smoothness weights at those locations).

SPARSE TO DENSE FLOW FIELDS

Revaud et al., "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow," CVPR 2015.



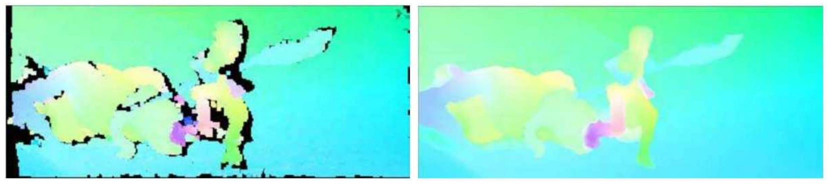
Energy Minimization

- Further refine interpolated flow-field by going back and looking at original image pair.
- Corresponds to minimizing Lucas Kandae / Horn Schunck type energy

but now with at Taylor expansion around the input flow field, instead of $(u = 0, v = 0)$.

SPARSE TO DENSE FLOW FIELDS

Revaud et al., "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow," CVPR 2015.



Going from Coarse-to-Fine PatchMatch to interpolated and refined version

NEURAL NETWORK-BASED STEREO

Let's go back to the basic anatomy of a stereo algorithm.

Two steps:

- Build cost volume based on some matching metric.
- Smooth cost-volume using smoothness (SGM) / planarity (MRF) models.

Both involve hand-crafted choices:

- Matching Metric = expert assumption that census features are discriminative and robust
- Smoothing = hand chosen smoothness costs and models

NEURAL NETWORK-BASED STEREO

Zbontar and LeCun, Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches, JMLR 2016.

- Conference version came out in CVPR 2015.
- First paper to apply deep learning to stereo.
- Significant improvements from only modifying matching cost !

NEURAL NETWORK-BASED STEREO

Zbontar and LeCun, Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches, JMLR 2016.

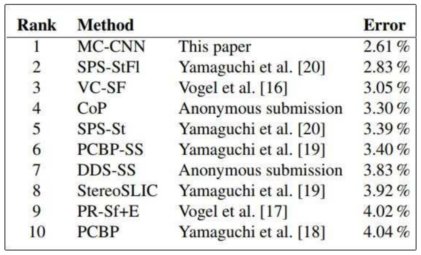
- Build cost volume C using a neural network f

$$C[x, y, d] = f(p_{left}(x, y), p_{right}(x - d, y)$$

Where $p_{left}(x, y)$ and $p_{right}(x, y)$ are 9×9 patches centered at (x, y).

- Apply standard cost-volume/filtering and SGM on this cost-volume.
- No MRFs or planar models.

No. 2 and 3 used multiple frames + flow reasoning.



Rank	Method		Error
1	MC-CNN	This paper	2.61 %
2	SPS-StFl	Yamaguchi et al. [20]	2.83 %
3	VC-SF	Vogel et al. [16]	3.05 %
4	CoP	Anonymous submission	3.30 %
5	SPS-St	Yamaguchi et al. [20]	3.39 %
6	PCBP-SS	Yamaguchi et al. [19]	3.40 %
7	DDS-SS	Anonymous submission	3.83 %
8	StereoSLIC	Yamaguchi et al. [19]	3.92 %
9	PR-Sf+E	Vogel et al. [17]	4.02 %
10	PCBP	Yamaguchi et al. [18]	4.04 %

NEURAL NETWORK-BASED STEREO

Zbontar and LeCun, Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches, JMLR 2016.

$$S[x, y, d] = f(p_{left}(x, y), p_{right}(x - d, y)$$

Using S for similarity instead of C for cost.

- Want $S[x, y, d]$ to be higher for the correct value of d than for incorrect values.
- Standard approach in learning similarity "metrics". Training set is a bunch of "triplets".

Triplet = (reference, correct match, in-correct match) = (p_r, p_c, p_i)

- For $p_r = p_{left}(x, y)$, select $p_c = p_{right}(x, y, \hat{d})$ and $p_i = p_{right}(x, y, \bar{d})$
- $\hat{d}$ = True disparity.
- $\bar{d}$ = Randomly selected in-correct disparity s.t. $|\hat{d} - \bar{d}| > \epsilon$

NEURAL NETWORK-BASED STEREO

Zbontar and LeCun, Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches, JMLR 2016.

$$S[x,y,d] = f(p_{left}(x,y), p_{right}(x-d,y)$$

- Given a training example (p_r, p_c, p_i), train using a "hinge" loss:

$$L = \max\left(0, m + (f(p_r, p_i) \quad - \quad f(p_r, p_c))\right)$$

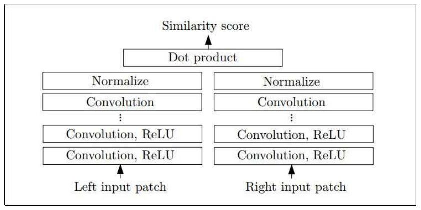
where m is some selected margin (hyper-parameter).

- Loss is high when similarity score for incorrect match is higher than for correct match.
- Clipped from below when correct match has score at least m higher than in-correct match.

NEURAL NETWORK-BASED STEREO

Zbontar and LeCun, Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches, JMLR 2016.

Architectures: Fast (Siamese Network)

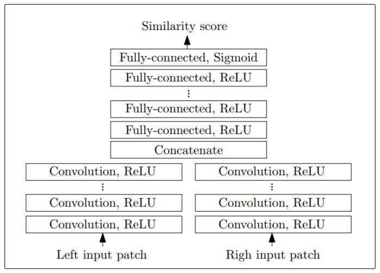


- The left and right half of the network share weights.
- Similarity score is basically normalized dot-product between "feature" vectors,< br/> extracted by applying a common network to both patches.
- This is fast because you compute feature vectors once for all patches in both the left and right image. (Can apply the network in a fully convolutional way).
- Compute all $S[x,y,d]$ which involves computing all epipolar pairs, by taking only dot-products

NEURAL NETWORK-BASED STEREO

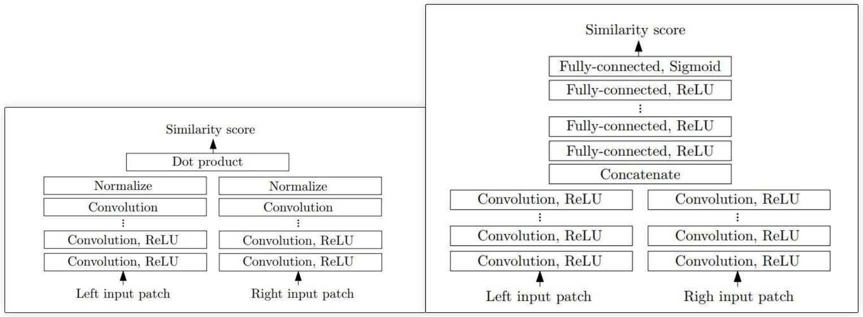
Zbontar and LeCun, Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches, JMLR 2016.

Architectures: Accurate (Siamese + learned distance metric)



- This allows the similarity score to express more complex interactions between per-patch features.
- But also means that you need to re-compute the last fully connected layers for every epipolar pair: i.e., for every entry of $S[x,y,d]$.
- Convolution layers need to be computed only once per patch like before.

NEURAL NETWORK-BASED STEREO



- Fast: Error = 2.82 %, Time = 0.8 secs
- Accurate: Error = 2.43 %, Time = 67 secs

NEURAL NETWORK-BASED STEREO

- But Zbontar and LeCun only looked at improving the matching cost.
- Rest of the pipeline is the same: cost-volume filtering + SGM
- What if we replaced the entire pipeline with a neural network ?
- Need to choose an architecture that mimics the processing steps required.