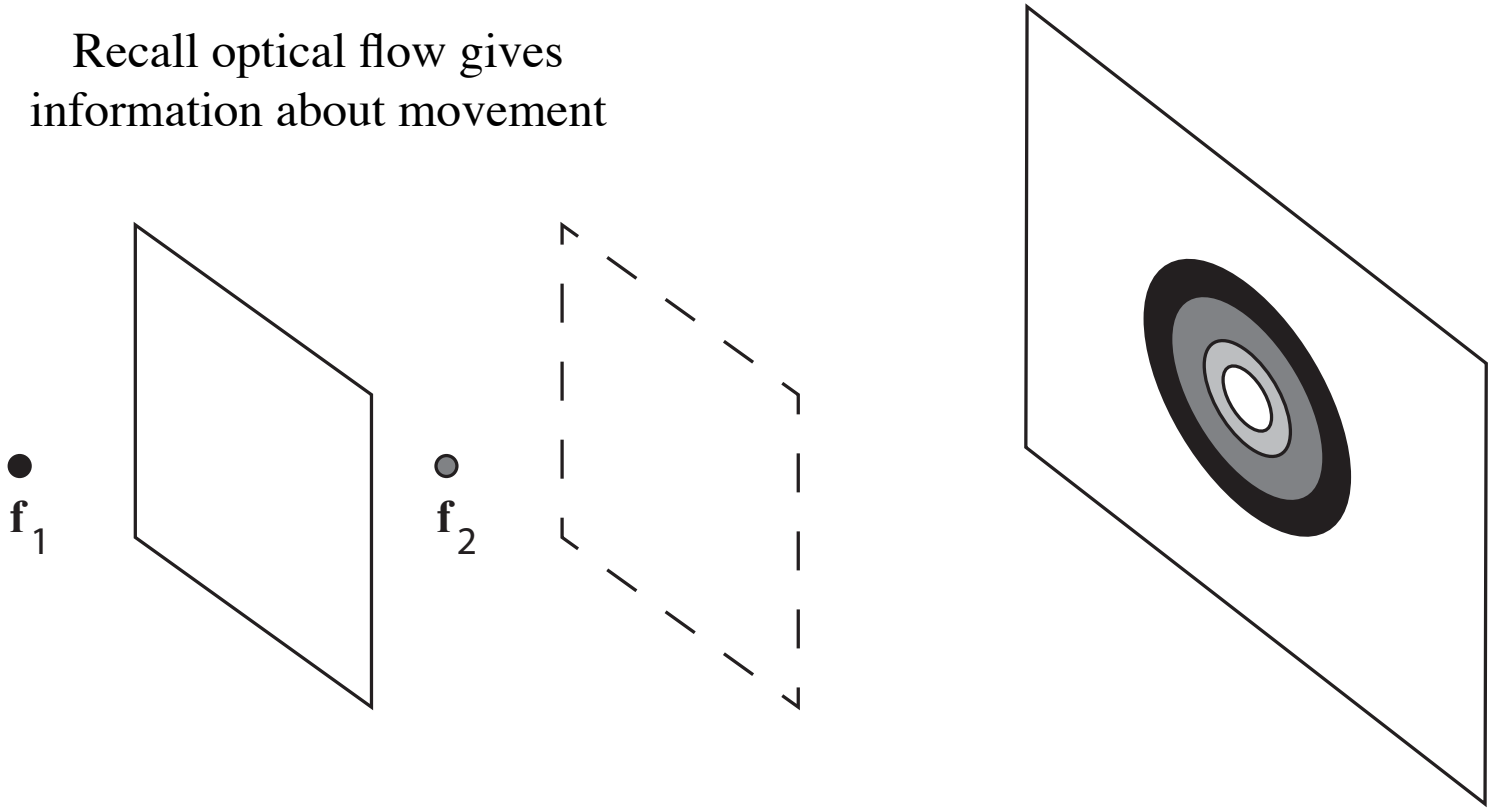
Scene Flow and Ways to Infer it

D.A. Forsyth, UIUC

Scene Flow and Ways to Infer It

- particularly photometric consistency
  - a version of this applies to scene inference

Recall optical flow gives information about movement



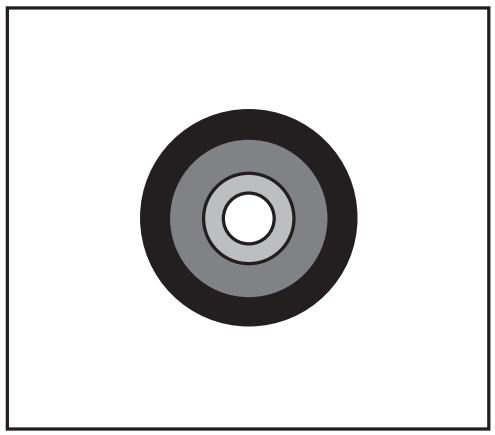


Image 1

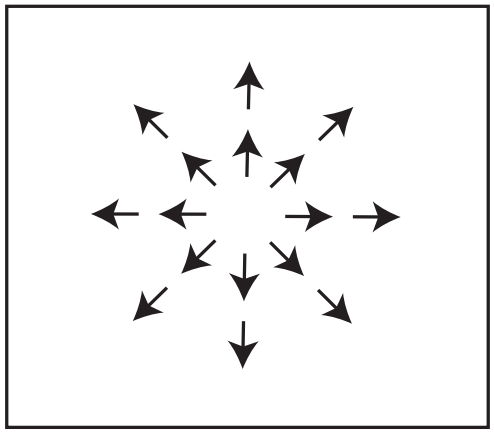


Image 1 optic flow

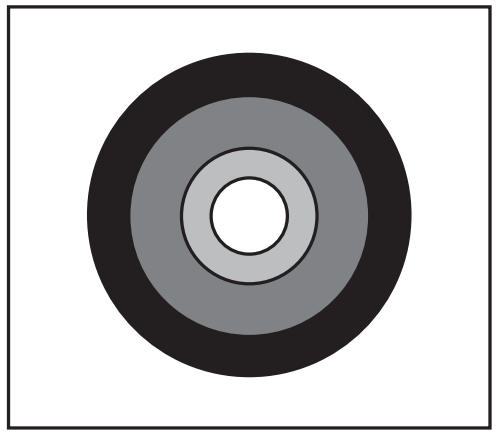
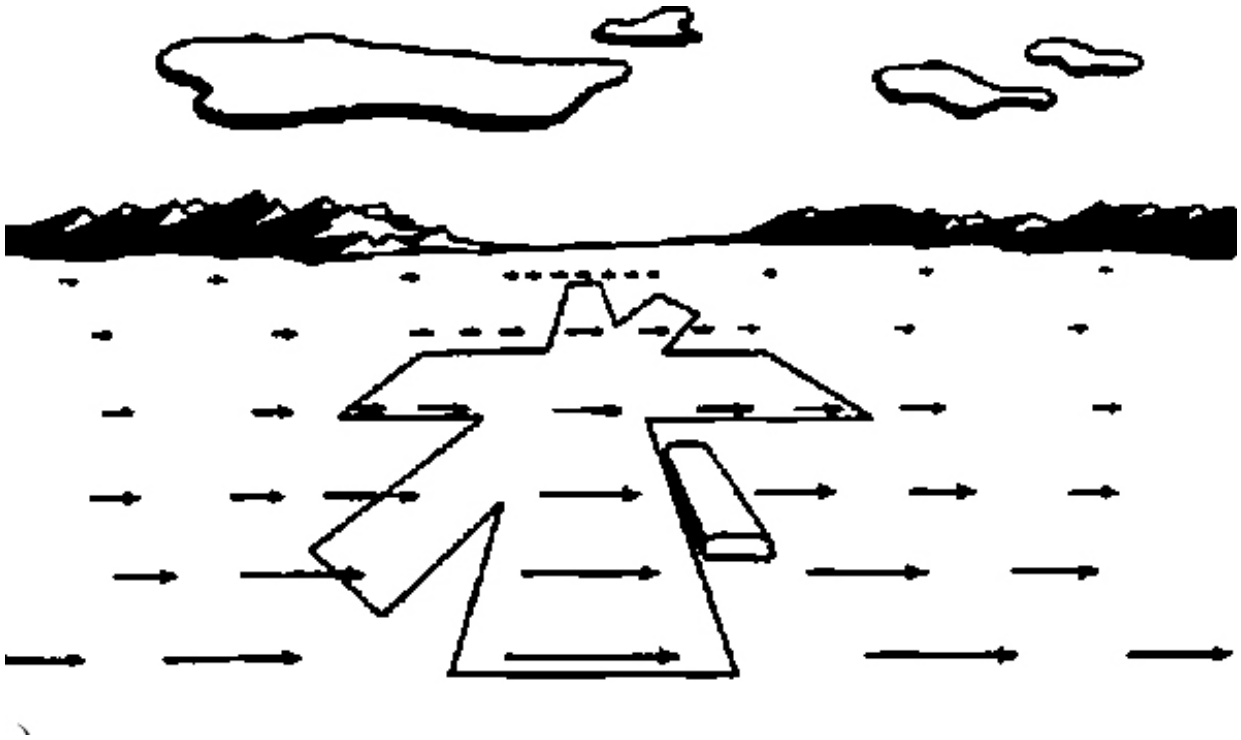


Image 2

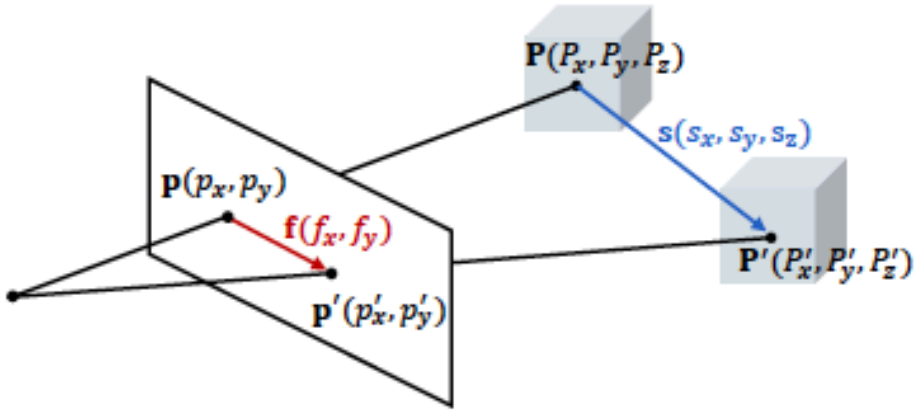
Recall optical flow gives
information about movement AND depth



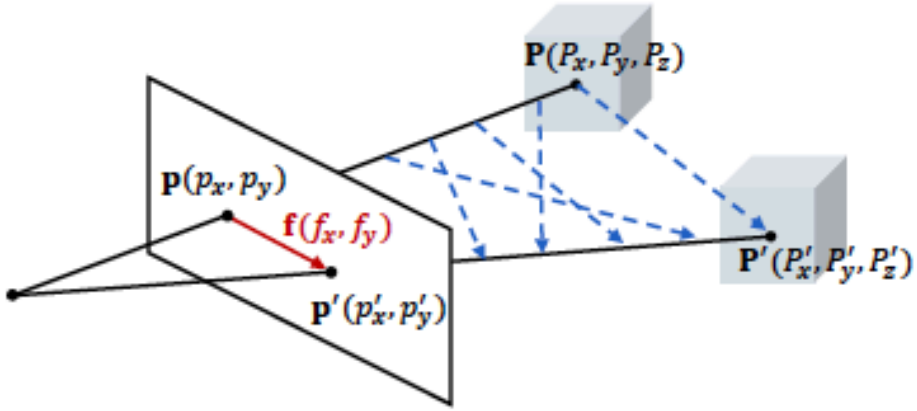
Scene Flow

- Mark (x, y, z) AND (v_x, v_y, v_z) at every image point
  - From pair of (image+depth) or (stereo pair) or (lidar) or even (image)
- Rigid scene
  - Easy for stereo pair/image+depth pair:
    - (v_x, v_y, v_z) follow from depth and camera ego-motion
  - Much harder for image pair
    - depth, scene flow ambiguity
- BUT assume there are moving objects

Depth/flow ambiguity



(a) Projecting scene flow into 2D space.



(b) Back-projecting optical flow into 3D space.

Figure 2. **Relating monocular scene flow estimation to optical flow**: *(a)* Projection of scene flow into the image plane yields optical flow [59]. *(b)* Back-projection of optical flow leaves an ambiguity in jointly determining depth and scene flow.

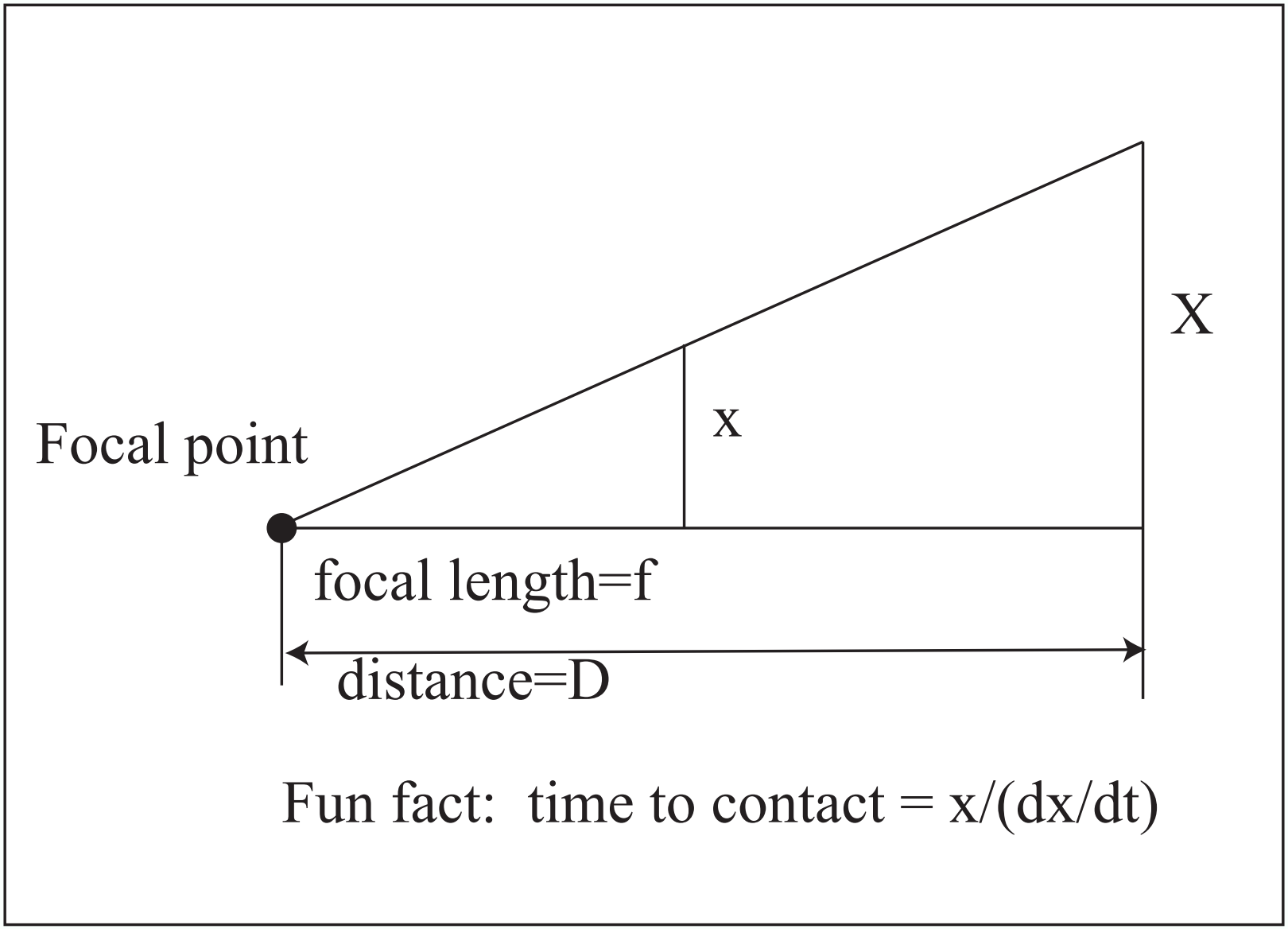
- Notice there are no problems if you know depth

Menze 2015

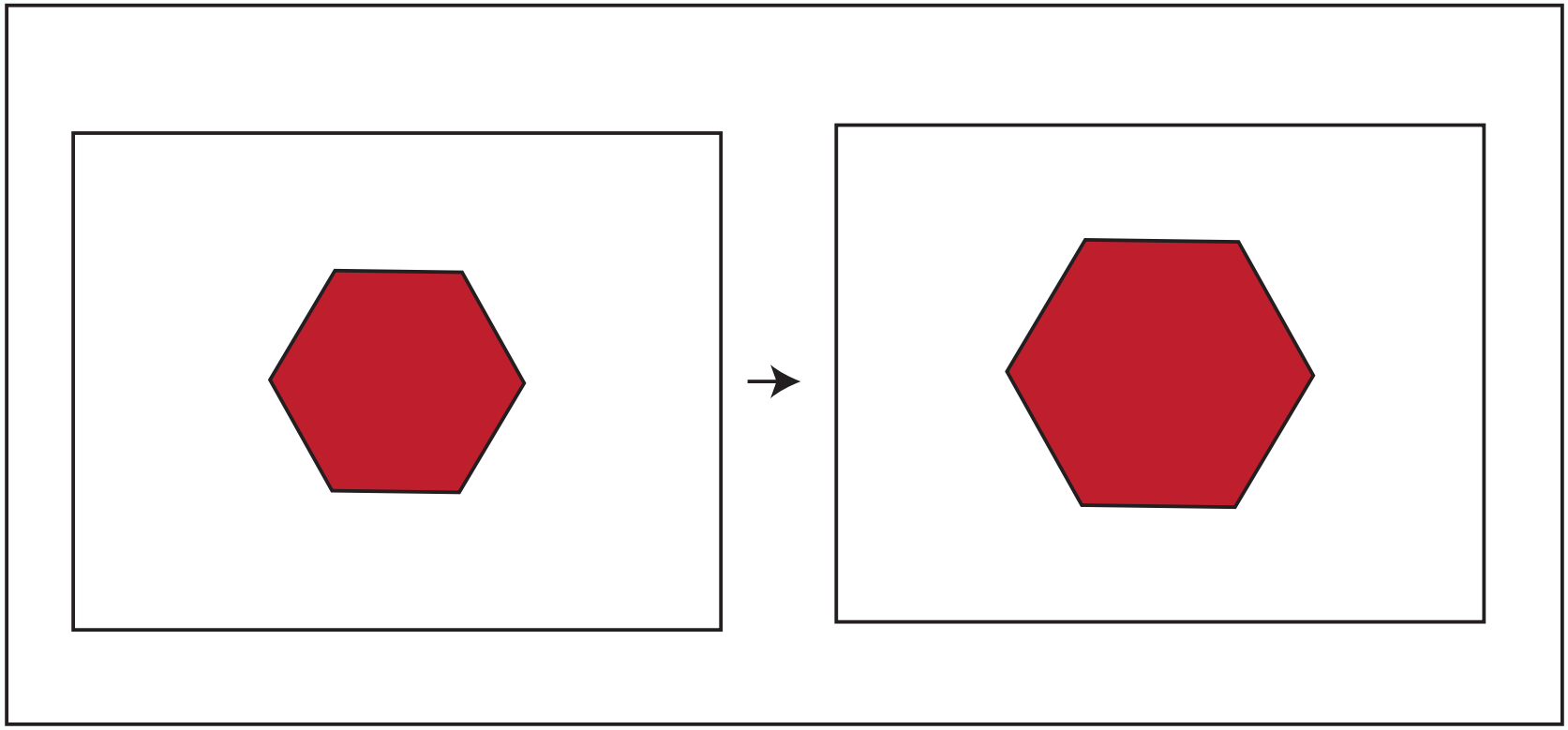
Harder when there are moving objects…

- Registering the depths (say) doesn't work
- Need to know which pixels are moving rigidly together
- Much more important case
  - Think cars
  - Time to contact

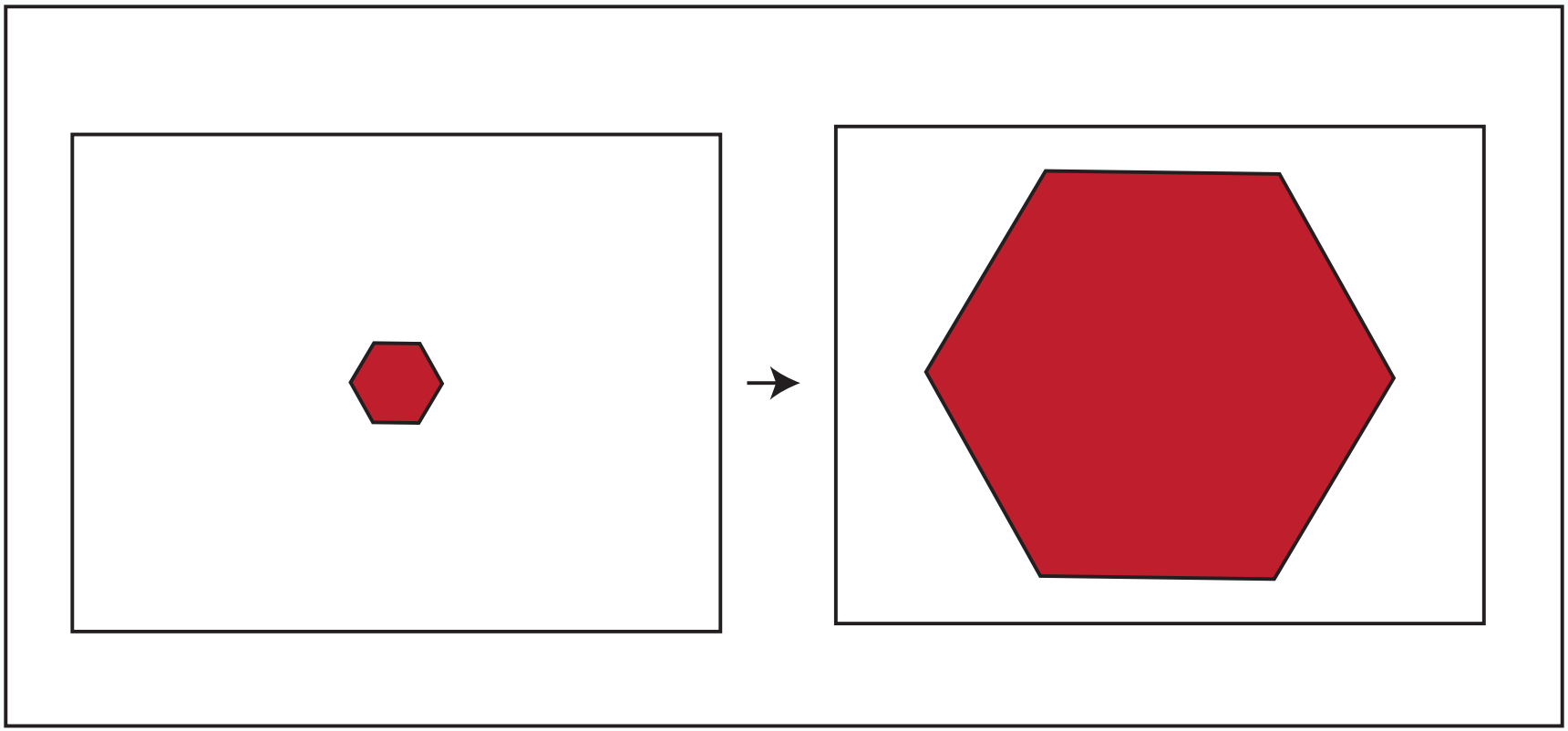
Fun fact about vision



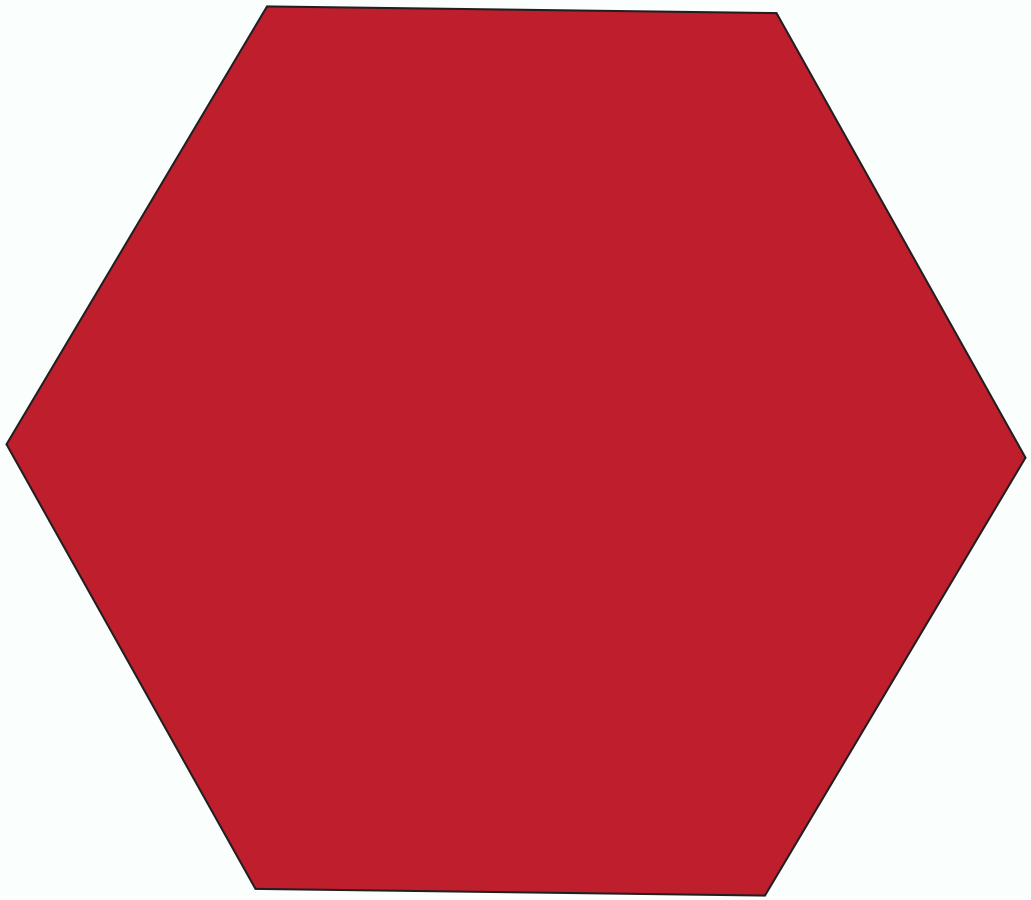
Fun fact: time to contact = x/(dx/dt)

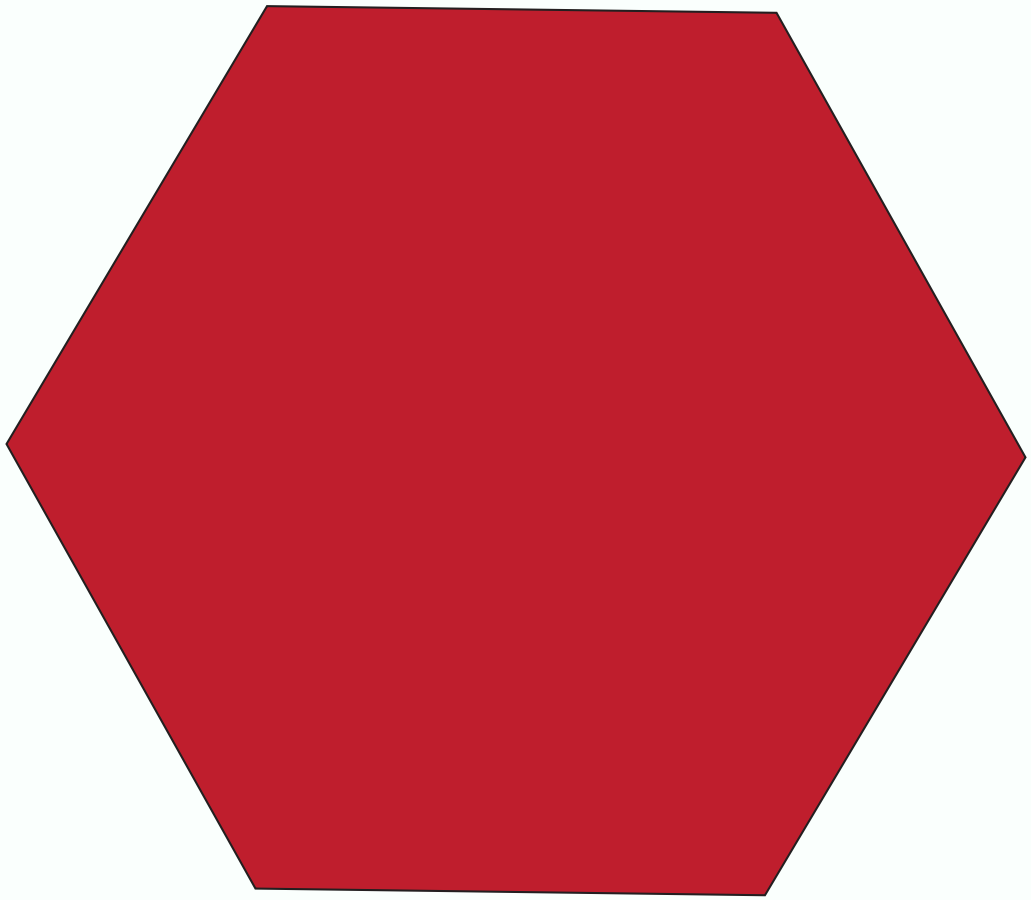


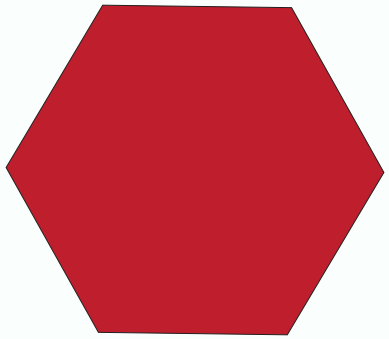
TTC - Long

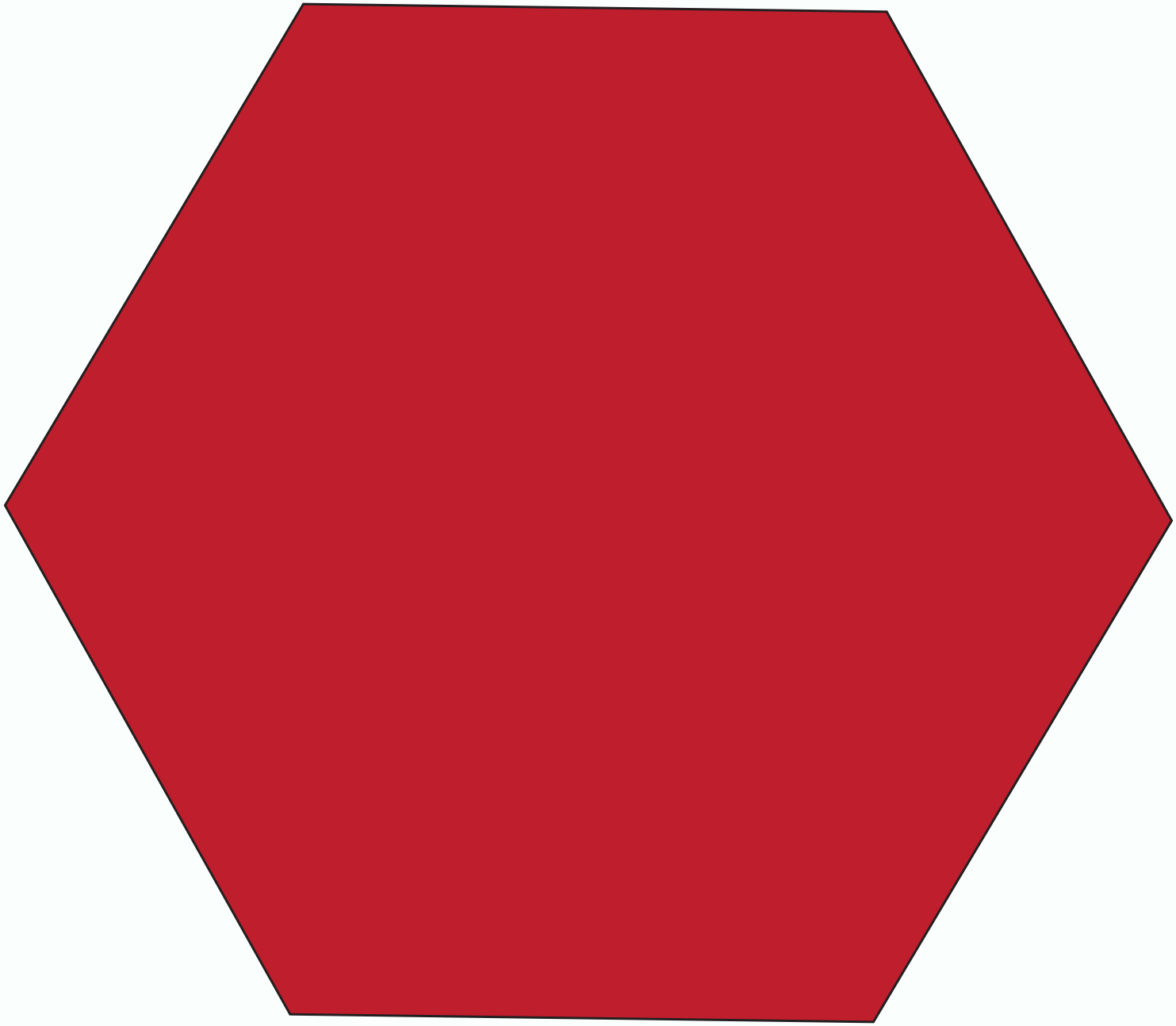


TTC - AAARGH!









Typical scene flows

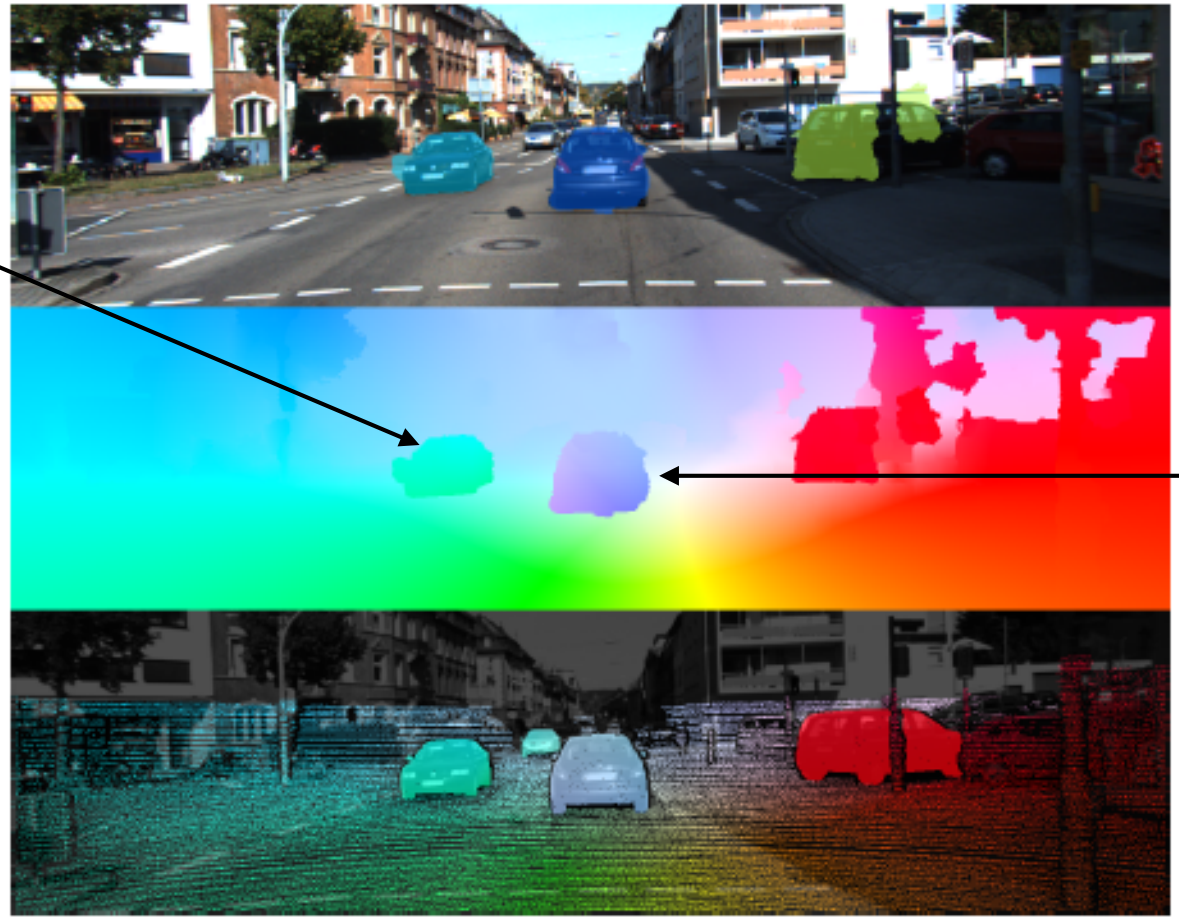


Figure 1: **Scene Flow Results on the proposed Dataset.** Top-to-bottom: Estimated moving objects with background object in transparent, flow results and flow ground truth.

Menze 2015

Estimation strategies -I

- RGB-D image pairs
  - segment
  - estimate correspondence using RGB
  - get v_x, v_y, v_z using D
- RGB stereo pairs
  - segment
  - estimate depth using stereo
  - as above
- LIDAR
  - segment; use registration from early lectures (with tricks, following slides)
  - get v_x, v_y, v_z

Estimation strategies -II

- Single image pairs
  - use single image depth predictor, proceed as above
  - use labelled scene flow images, predict w/net
- LIDAR - II
  - train a network to estimate from pairs with known scene flows

Estimation for stereo

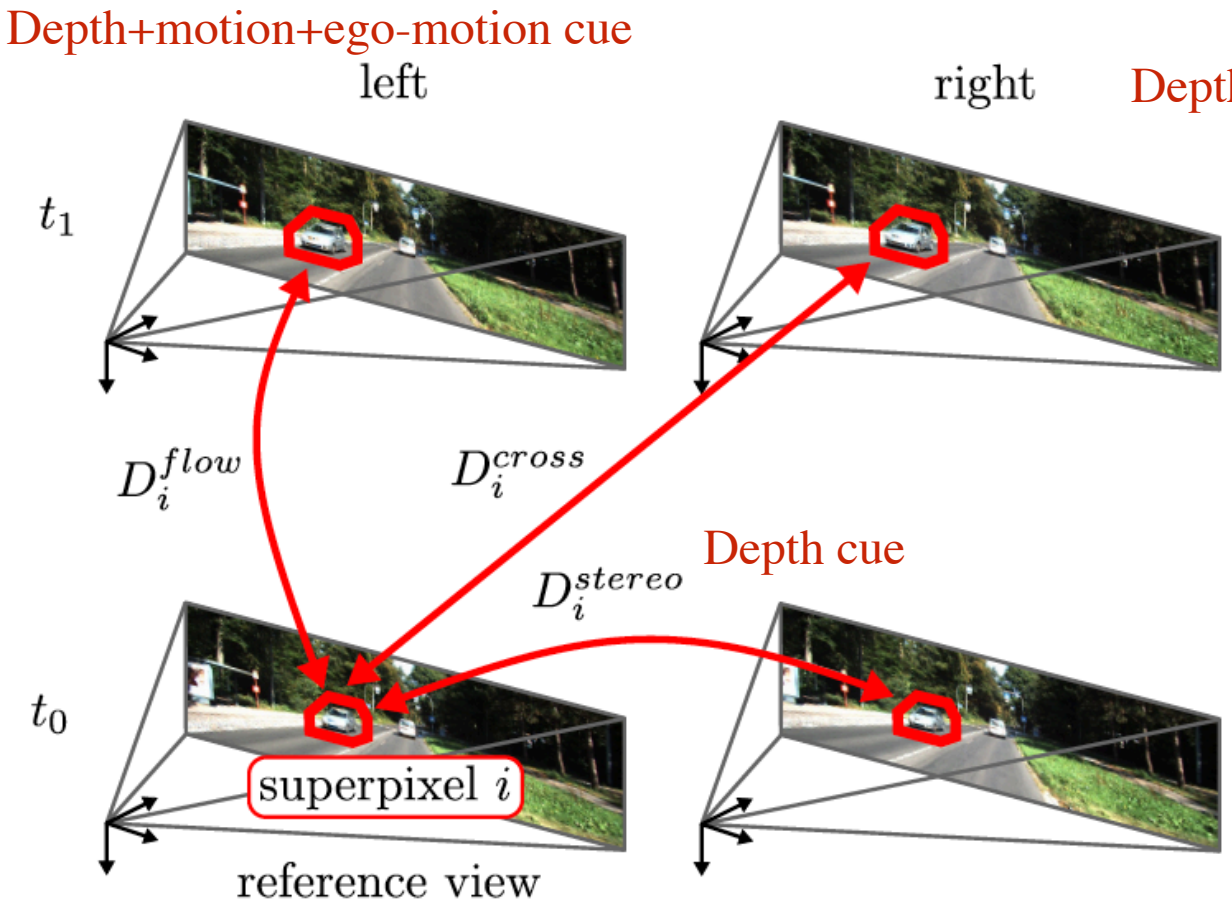


Figure 2: **Data Term.** Each superpixel i in the reference view is modeled as a rigidly moving 3D plane and warped into each other image to calculate matching costs. Each of the superpixels is associated with a 3D plane variable and a pointer to an object hypothesis comprising its rigid motion.

- Break into superpixels
- Each gets depth, flow
- Use this to predict appearance in other views
- This gives massive CRF
  - pile in and solve

Menze 2015

Lagniappe: Scene flow in LIDAR

- Learn without labelled data
- ICP isn't quite enough
  - objects might contract, for example
  - use a cycle consistency loss
    - f_ab = 3Da -> 3Db
    - we must have f_ba(f_ab(x))=x
    - trick:
      - as stated, this is unstable
      - instead, f_ba(0.5 f_ab(x)+ 0.5 NN(f_ab(x))) close to x
      - this also avoids problems with zero flow

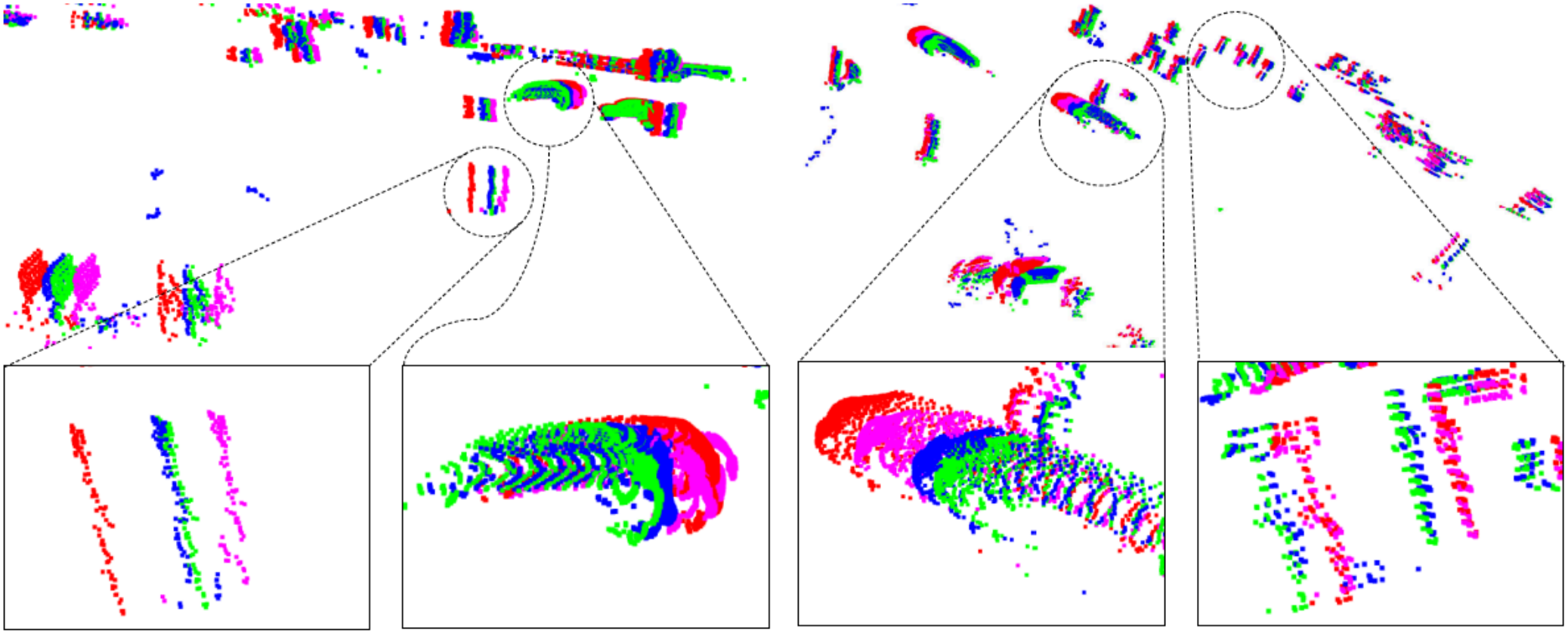
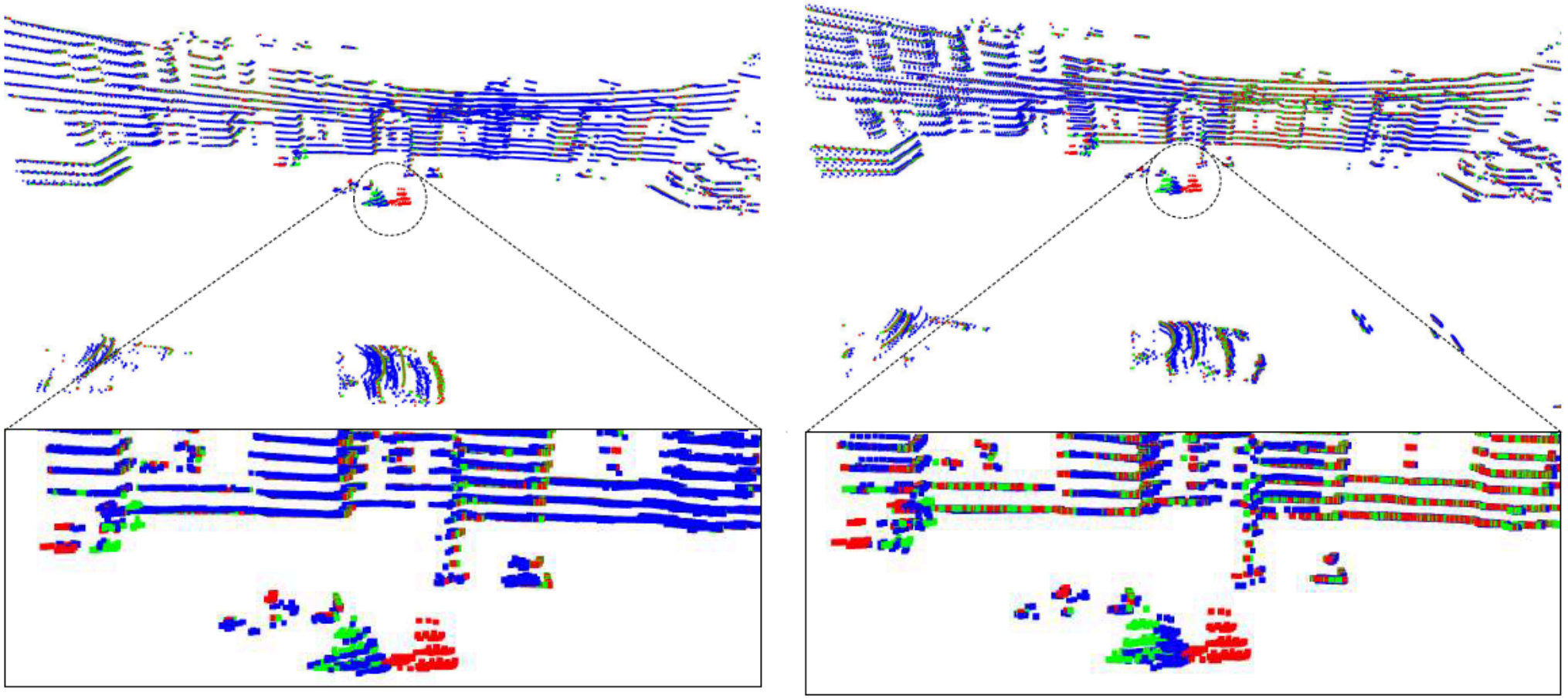


Figure 4: Scene flow estimation between point clouds at time t (red) and $t+1$ (green) from the KITTI dataset trained without any labeled LIDAR data. Predictions from our self-supervised method, trained on nuScenes and fine-tuned on KITTI using self-supervised learning is shown in blue; the baseline with only synthetic training is shown in purple. In the absence of real-world supervised training, our method clearly outperforms the baseline method, which overestimate the flow in many regions. (Best viewed in color)



(a) Ours (Self-Supervised Fine Tuning)

(b) Baseline (No Fine Tuning)

Figure 5: Comparison of our self-supervised method to a baseline trained only on synthetic data, shown on the nuScenes dataset. Scene flow is computed between point clouds at time t (red) and $t+1$ (green); the point cloud that is transformed using the estimated flow is in shown in blue. In our method, the predicted point cloud has a much better overlap with the point cloud of the next timestamp (green) compared to the baseline. Since nuScenes dataset does not provide any scene flow annotation, the supervised approaches cannot be fined tuned to this environment.

Mittal 20

Scene flow in single images

- Predict depth from single image
  - using network which makes mixture of normals in depth at location
  - trained using existing image-depth data
- Break image into superpixels
  - each is a plane section that moves rigidly
    - to infer: plane params, motion params (9 total per superpixel)
- 

Scene flow in single images

Photometric consistency

- CRF
  - unary losses:
    - plane section motion should predict next frame pixel values well
    - plane section should model predicted depth well
  - binary losses:
    - plane sections should have compatible depths on boundary
    - normals of neighbors should be similar

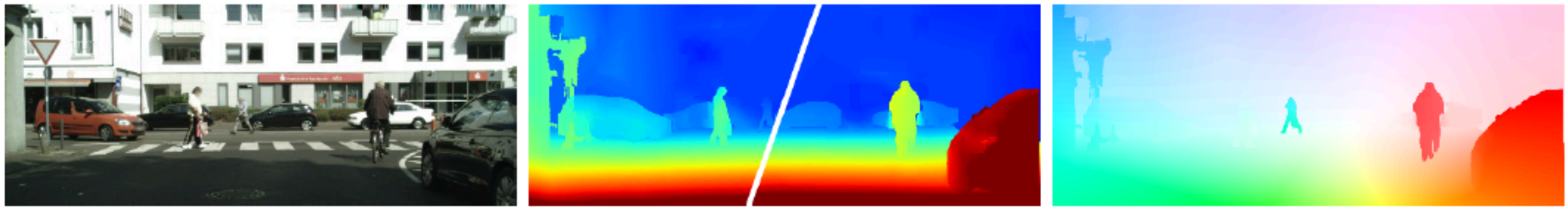
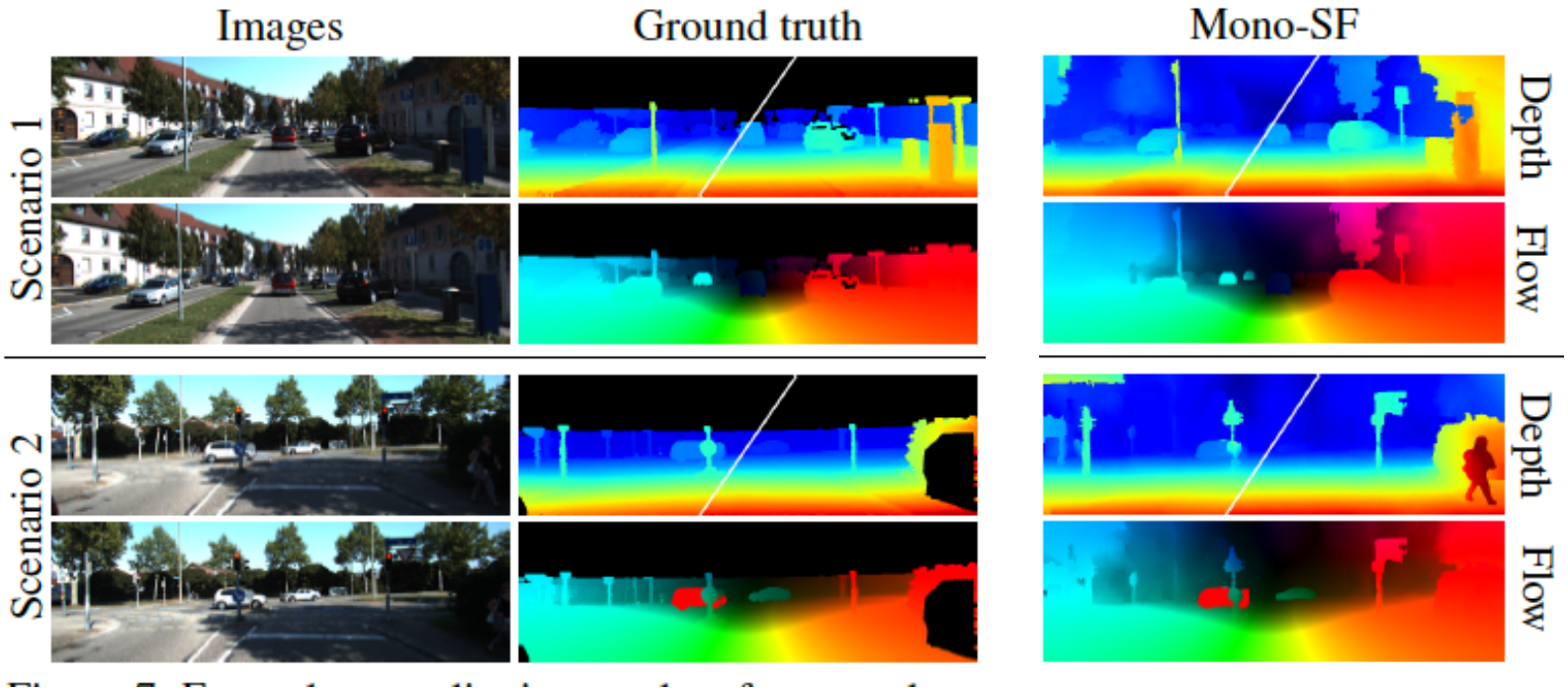


Figure 8. Exemplary qualitative result of Mono-SF on a crop of Cityscapes (removing car hood); left: first input image, middle: estimated depth values at time $t = 0$ (left half) and $t = 1$ (right half), right: estimated optical flow


Images
Ground truth
Mono-SF
Scenario 1
Scenario 2
Depth
Flow
Depth
Flow

Scene flow in single images

Figure 1. **Results of our monocular scene flow approach on the KITTI dataset [11].** Given two consecutive images *(left)*, our metho jointly predicts depth *(middle)* and scene flow *(right)*. (x,z)-coordinates of 3D scene flow are visualized using an optical flow color coding

- Straightforward network prediction of scene flow
  - depth ambiguity?
    - semantics, etc. resolve
    - *train* with stereo pairs
  - cues
    - single image depth cues (texture)
    - photometric consistency
      - optic flow

Scene flow in single images

Figure 1. **Results of our monocular scene flow approach on the KITTI dataset [11].** Given two consecutive images *(left)*, our method jointly predicts depth *(middle)* and scene flow *(right)*. (x,z)-coordinates of 3D scene flow are visualized using an optical flow color coding.

- Cute trick - this can be self-supervised
- Training time:
  - stereo images
- Test time
  - real images

Computing a loss for self supervision

Depth+motion+ego-motion cue

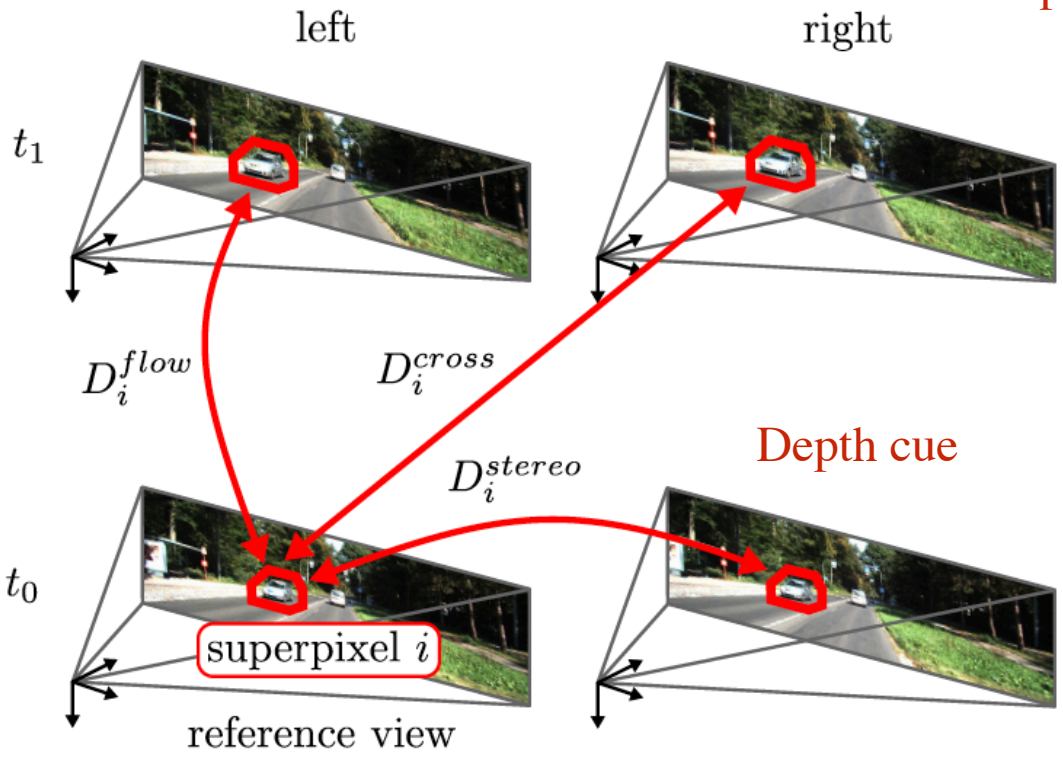


Figure 2: **Data Term.** Each superpixel i in the reference view is modeled as a rigidly moving 3D plane and warped into each other image to calculate matching costs. Each of the superpixels is associated with a 3D plane variable and a pointer to an object hypothesis comprising its rigid motion.

Depth+motion+ego-motion cue

- Each point in L gets
  - depth, flow
- Use depth to predict
  - appearance in R
- Use depth+flow to predict
  - appearance in t+1 L, R
  - 3D location in t+1 L
    - compare with depth
- This gives loss
- Train to minimize

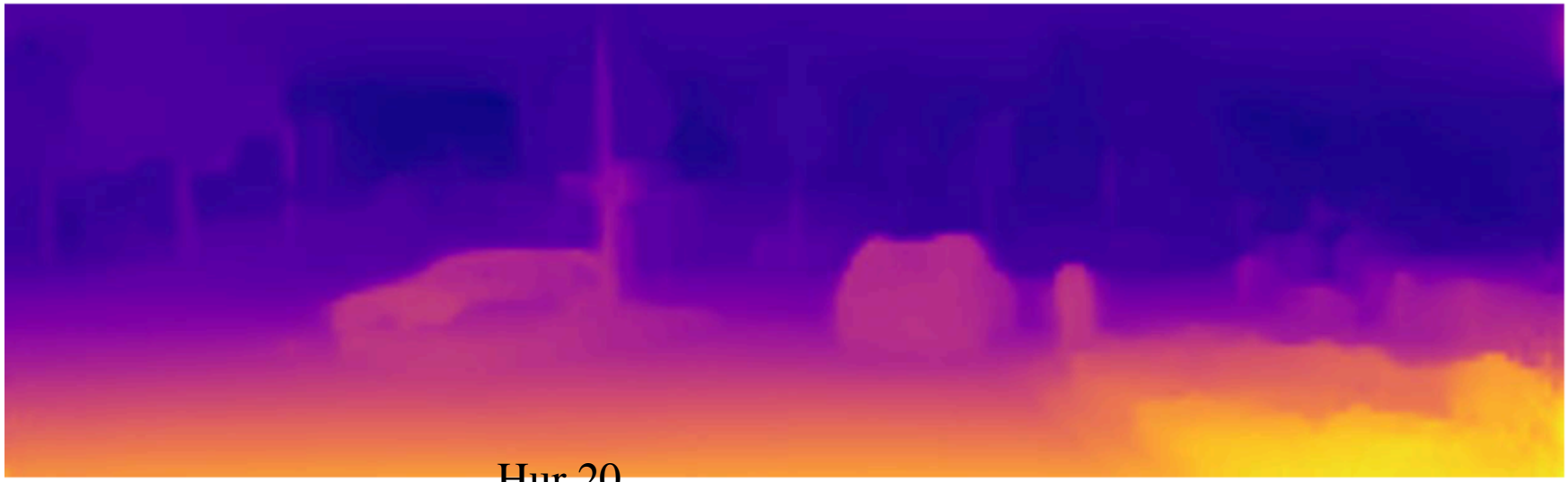
Menze 2015

Training losses

- Disparity predictions should be good
  - train with stereo pairs for this
  - disparity should predict color in other frame (in training)
  - disparity should be smooth
- Photometric consistency
  - scene flow should predict pixel values in next frame
- Point consistency
  - scene flow should predict depth in next frame
- Smoothness
  - scene flow at a point should be similar to neighbors

(a) Input images (b) Monocular depth (c) Optical flow (d) 3D visualization of scene flow

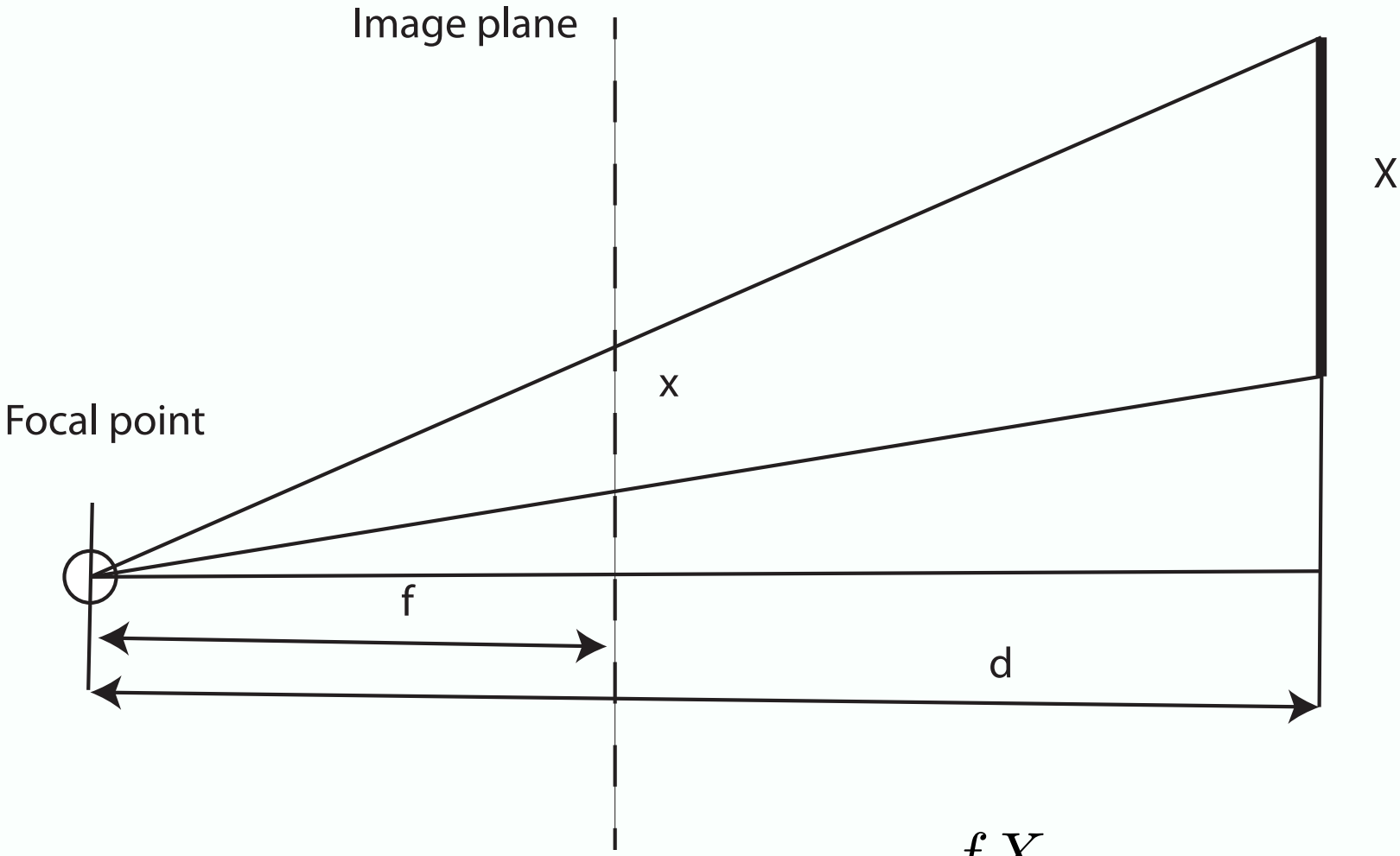
Figure 5. **Qualitative results of our monocular scene flow results (Self-Mono-SF-ft) on KITTI 2015 Scene Flow Test**: each scene shows *(a)* two input images, *(b)* monocular depth, *(c)* optical flow, and *(d)* a 3D visualization of estimated depth, overlayed with the reference image, and colored with the (x, z)-coordinates of the 3D scene flow using the standard optical flow color coding.



Hur 20

Hur 20

Motion in depth



$$x = \frac{fX}{d}$$

Motion in depth

- Now imagine object moves IN DEPTH
  - so d', x'
- We get

$$x' = \frac{fX}{d'}$$

$$s = \frac{x}{x'} = \frac{d'}{d}$$

$$x = \frac{fX}{d}$$

- this is important, because

$$d(s-1) = d' - d \propto v_z$$

- and we can estimate d

Scene flow from MiD

- Train optic expansion network
  - ptic expansion=1/s
  - using existing scene flow training data
- Then attach to optic flow, cleanup

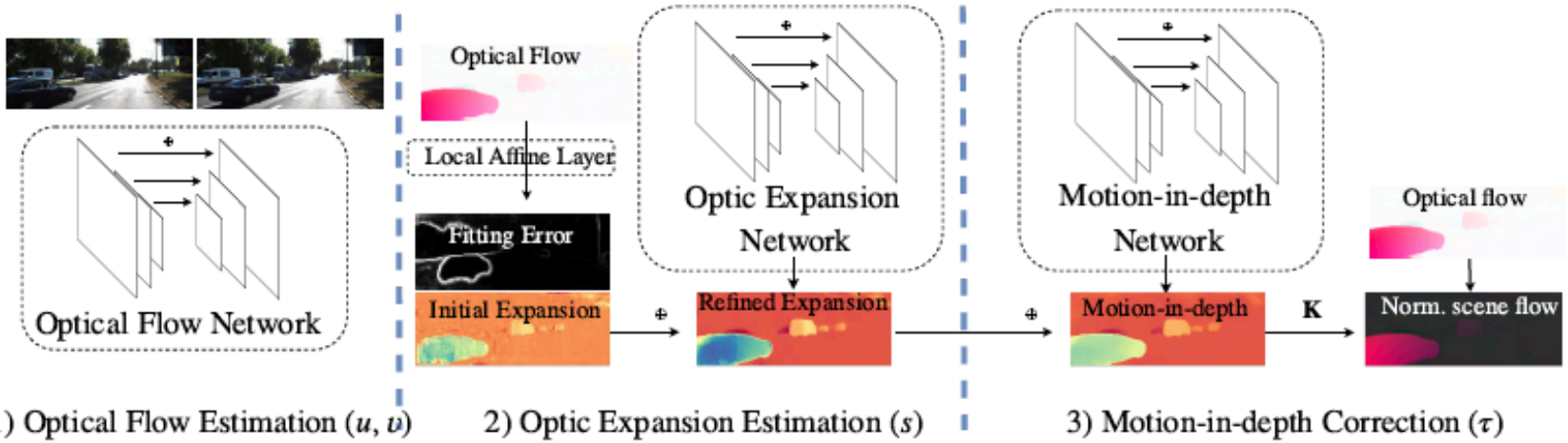


Figure 5. Network architecture for estimating normalized scene flow. 1) Given two consecutive images, we first predict dense optical flow fields using an existing flow network. 2) Then we estimate the initial optical expansion with a local affine transform layer, which is refined by a U-Net architecture taking affine fitting error and image appearance features as guidance [24]. 3) To correct for errors from the scaled-orthographic projection and rotation assumptions, we predict the difference between optical expansion and motion-in-depth with another U-Net. Finally, a dense normalized scene flow field is computed using Eq. 2 by combining (u, v, τ) with camera intrinsics $\mathbf{K}$.

Learning to predict SF from point clouds

- Point clouds
  - Eg LiDAR
  - problem:
    - given point cloud at t, t+1
      - place a 3D motion vector on each point in t
  - hard, because:
    - there may be no corresponding point in t+1
    - representing a point cloud is hard
- Strategy:
  - don’t need corresponding points - use segments
  - use pointnet features

Pointnet - a neat trick

- Required: learned feature representation of a point cloud
- Difficulty: point cloud has no order
  - you can get the same point cloud in a different order
  - could impose order, but…
- Permutation invariants:
  - the basis for permutation invariants are the symmetric functions
    - mostly, a nuisance to work with
- Idea:
  - for any point cloud of n points in d dimensions,

$$\begin{bmatrix} \max(x_{1,1}, x_{2,1}, \ldots x_{n,1}) \\ \ldots \\ \max(x_{1,d}, x_{2,d}, \ldots x_{n,d}) \end{bmatrix}$$

is permutation invariant

Pointnet - a neat trick - II

- So:
  - embed points in high dimension (K)
  - compute this pooling
  - now compute embedding of this feature vector
  - the resulting object is permutation invariant
    - and “general”
      - assume
        - f(S) continuous in hausdorff distance on point sets
          - hausdorff distance on point sets = max dist to nearest neighbor
      - choose eps, and K big enough
      - then there is some g(S) of this form st $|f(S)-g(S)|<eps$

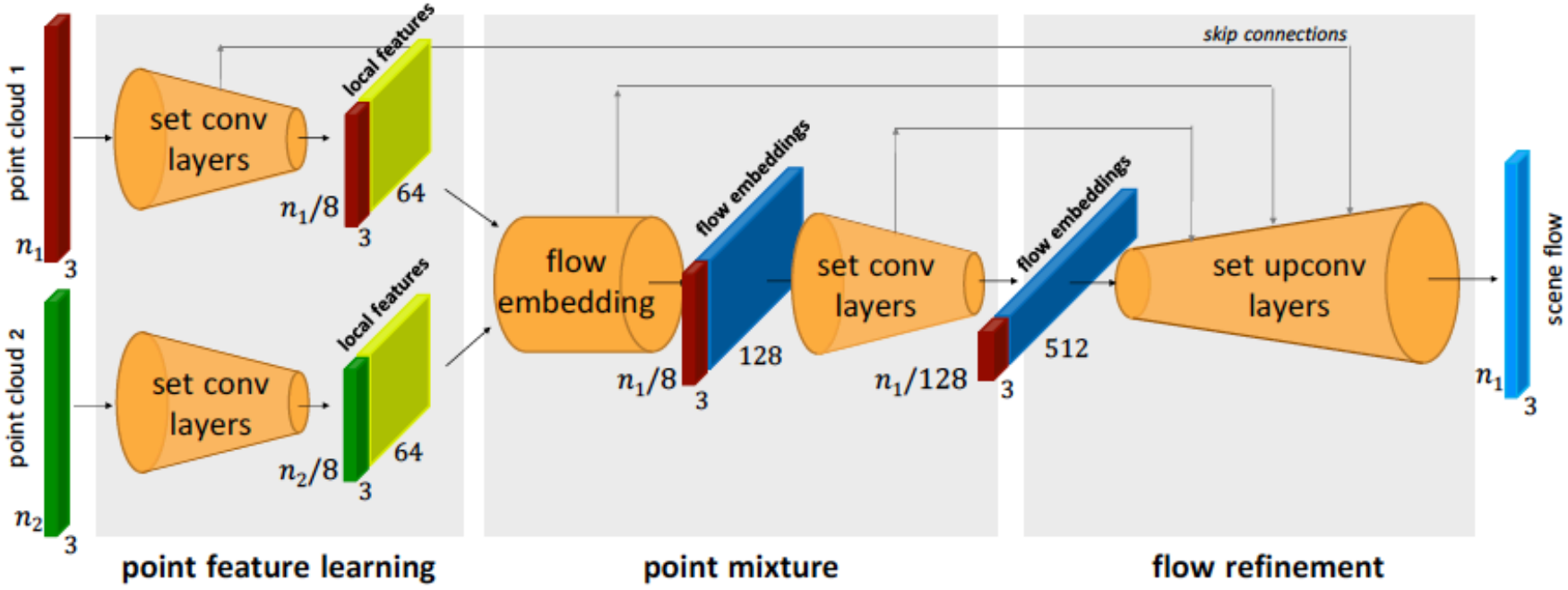


Figure 3: **FlowNet3D architecture.** Given two frames of point clouds, the network learns to predict the scene flow as translational motion vectors for each point of the first frame. See Fig. 2 for illustrations of the layers and Sec. 4.4 for more details on the network architecture.

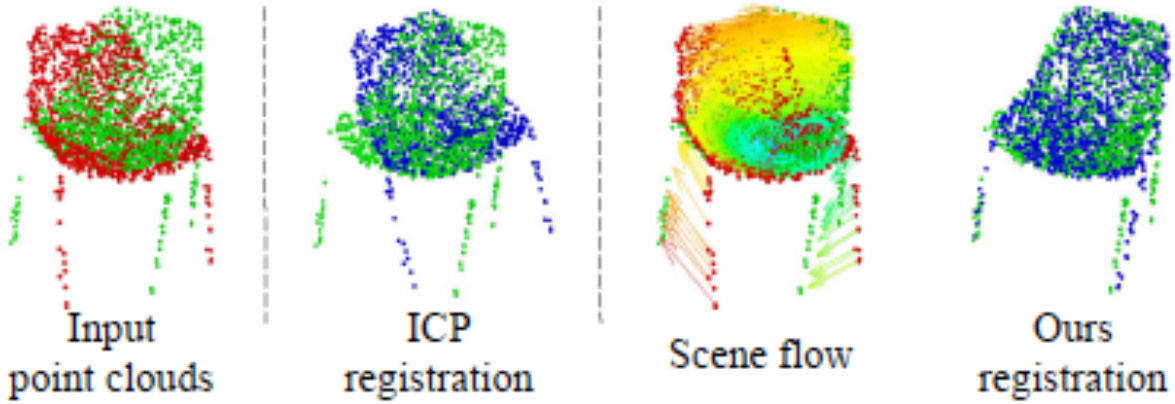


Figure 6: **Partial scan registration of two chair scans.** The goal is to register point cloud 1 (red) to point cloud 2 (green). The transformed point cloud 1 is in blue. We show a case where ICP fails to align the chair while our method grounded by dense scene flow succeeds.

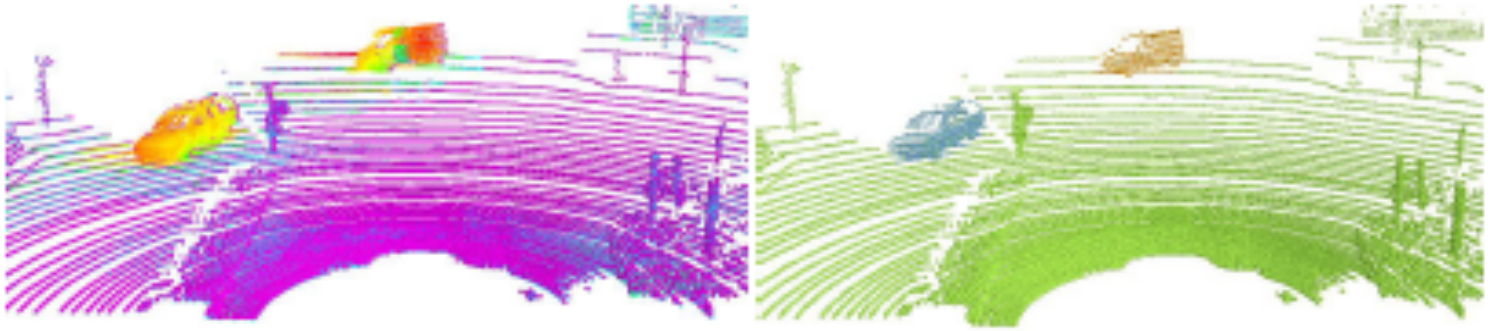


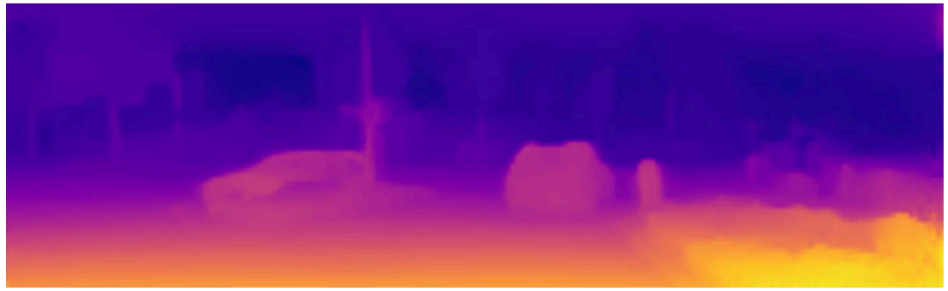
Figure 7: **Motion segmentation of a Lidar point cloud.** *Left:* Lidar points and estimated scene flow in colored quiver vectors. *Right:* motion segmented objects and regions.

How do we deal with relief?

- Surely some form of height field
  - estimated by consistency
  - changing slowly
- Horizon estimation gets complicated in tilted planes
  - you might get distracted by distant horizon
  - Local horizon estimator has problems

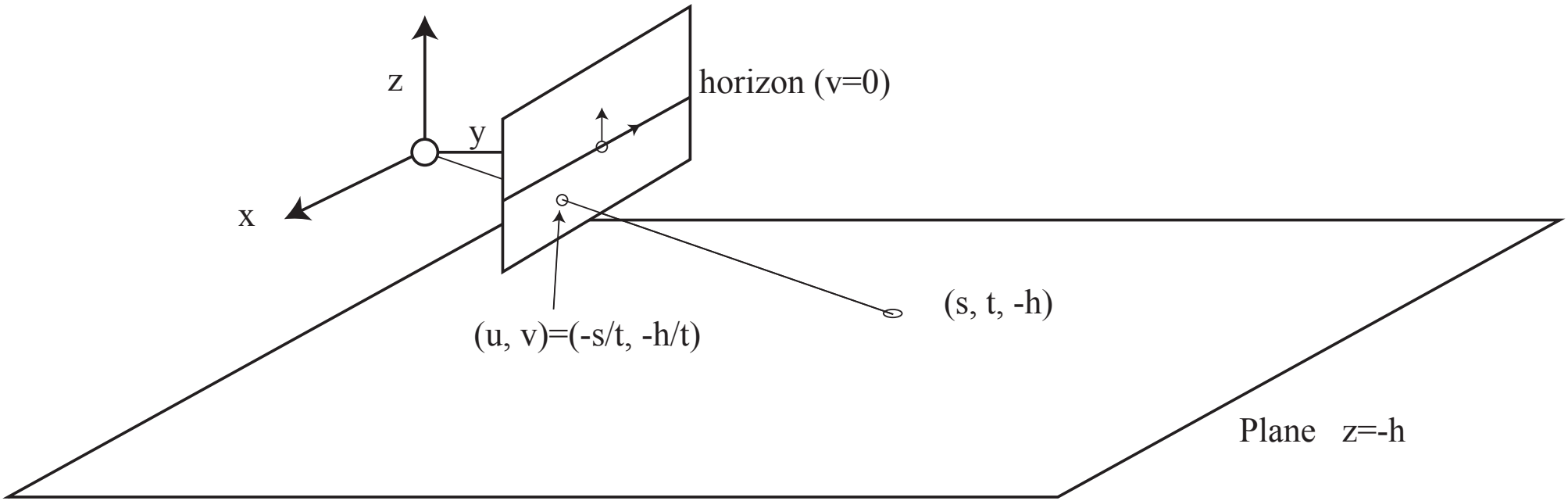
Nasty geometries

- Single image depth prediction likely doesn't work here
  - weird relief and dip in road
- Ground plane estimates likely don't work here either



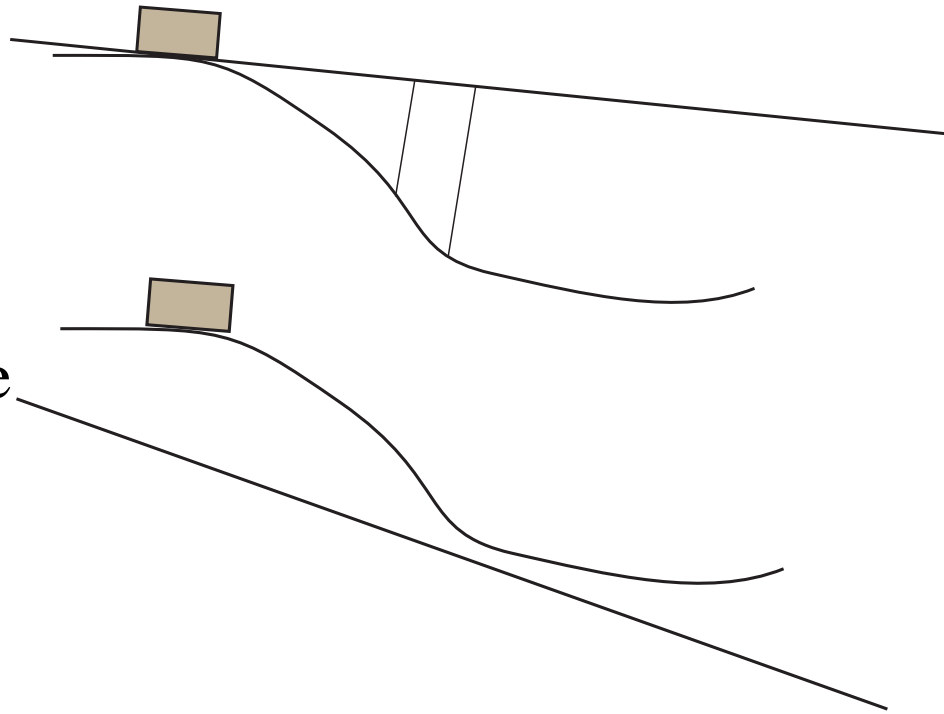
Estimating the camera

- Height
  - from car (calibrated and known)
- Roll and pitch
  - from horizon
    - roll is why horizon isn't parallel to image plane
    - pitch is why it isn't centerline



Sources of variation in the label map

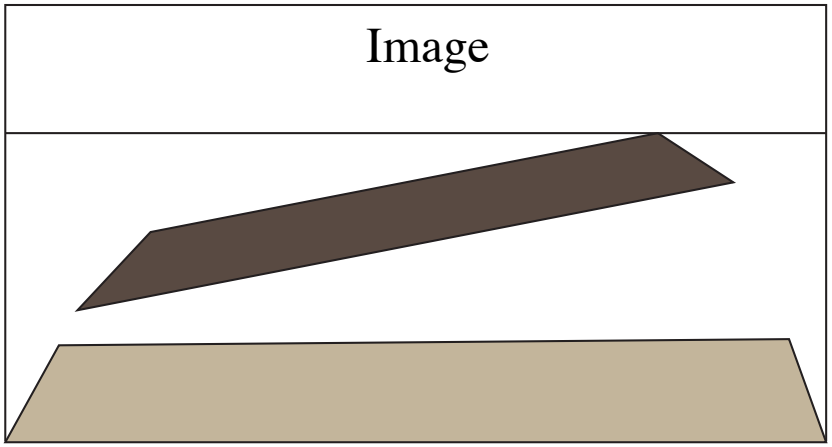
- Foreshortening

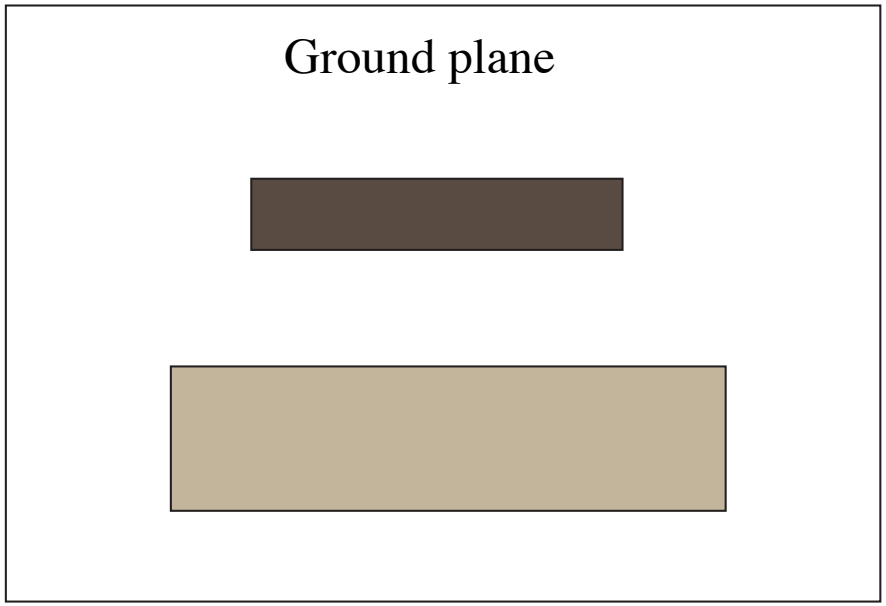


- Wrong ground plane estimate

Sources of variation in the label map

- Torsion





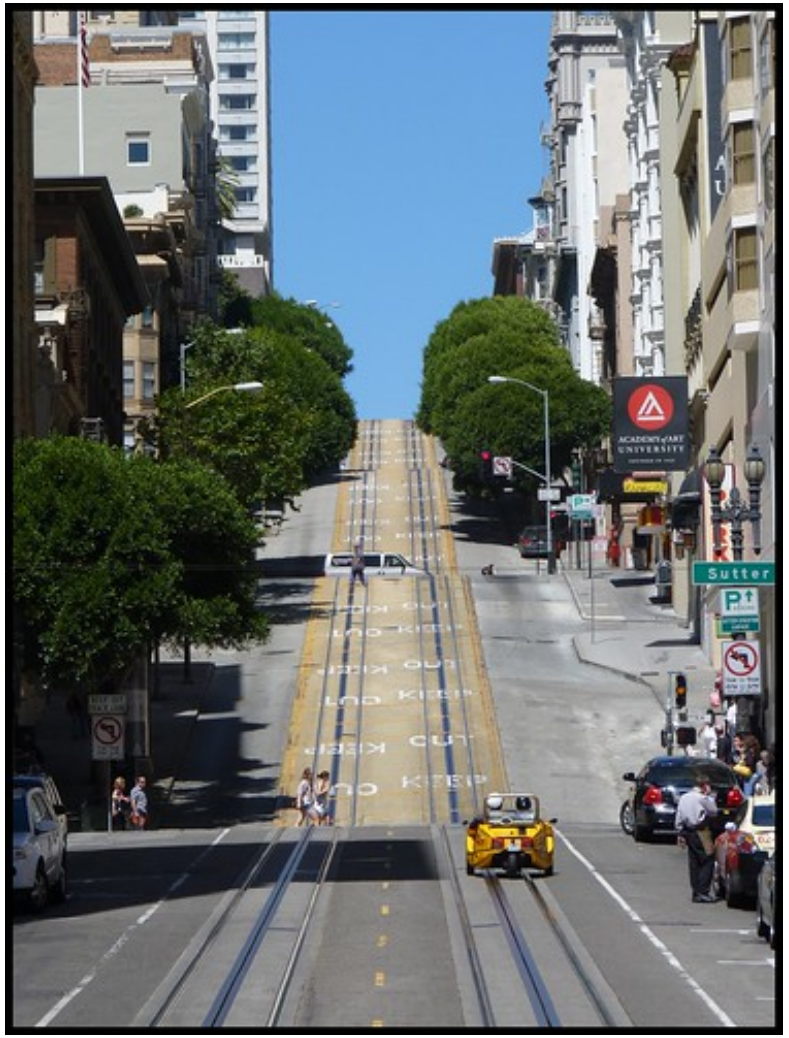
Horizon estimation

- Khan et al - vanishing points from road lines + fudge
- Workman et al - mark up dataset, classify

Figure 5: Example results showing the estimated distribution over horizon lines. For each image, the ground truth horizon line (dash green) and the predicted horizon line (magenta) are shown. A false-color overlay (red = more likely, transparent = less likely) shows the estimated distribution over the point on the horizon line closest to the image center.

Horizons

- Horizon estimation gets complicated in tilted planes
  - you might get distracted by distant horizon (picture)



Horizons

- Horizon estimation gets complicated in tilted planes
  - local cues are a problem

What to do?

- (Likely)
  - build sources of variance into simulated label fields
  - work on best available ground plane
    - (possibly) estimate several planes to rectify label fields
  - train without labelled images, as above
    - note this is a clusterer

Notice

- Straightforward consistency losses are very powerful
- Minimal use of labelled data
  - (augmentation by stereo pairs, but no labelling)
- Some form of photometric consistency loss for labels
  - eg
    - predict layout map 1
    - move forward
    - predict layout map 2
    - they should register
    - things that have the same label (tar, paint, junction, etc.)
      - should look similar

Appearance Consistency and Clustering

- Map image into some feature space so that
  - patches that “look similar” are “close”
  - without markup
- Why?
  - because doing so would help produce a layout map eg
    - attach labels to clusters using current maps
    - improve maps using labels

Deep Embedding Clustering

- Compute embedding that
  - autoencodes
  - clusters well

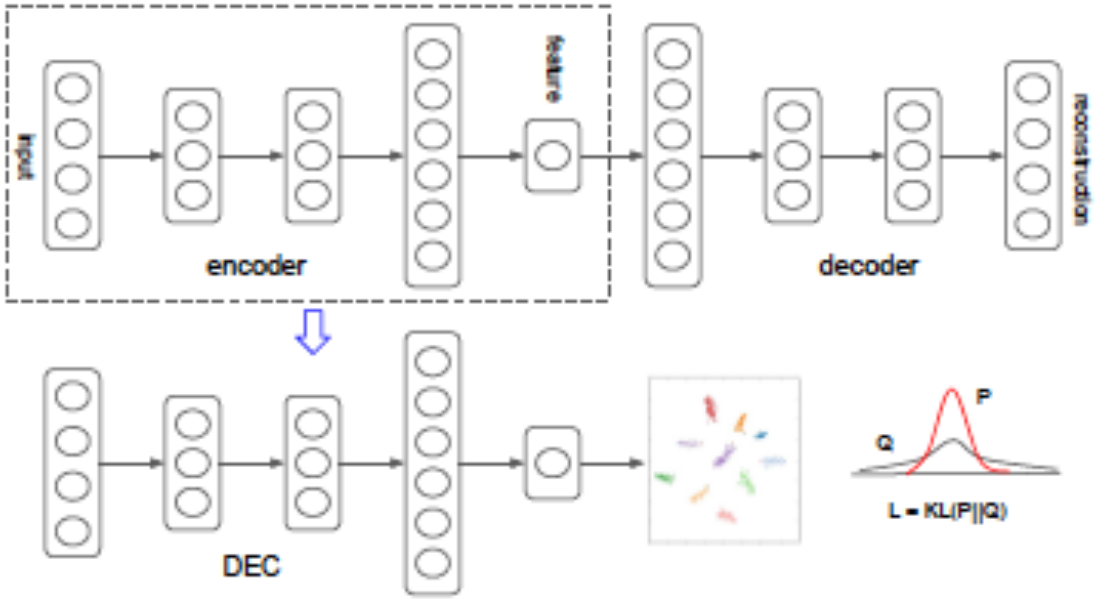


Figure 1. Network structure

Xie et al 15

Clustering

- Cluster centers mu_j must be estimated
  - form membership weights as in TSNE (alpha=1) ->

Following van der Maaten & Hinton (2008) we use the Student's t-distribution as a kernel to measure the similarity between embedded point z_i and centroid μ_j:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + \|z_i - \mu_{j'}\|^2/\alpha)^{-\frac{\alpha+1}{2}}}, \quad (1)$$

- We want these weights to match a target distribution
  - p_ij=target for j'th cluster on i'th point
  - KL divergence (as in TSNE)

$$L = \mathrm{KL}(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \quad (2)$$

Clustering-II

- But what are p?
  - notice we have some form of reestimation going on here

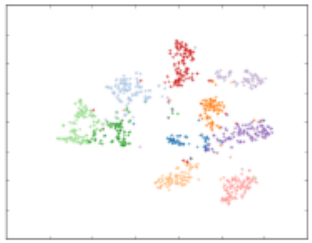
In our experiments, we compute p_i by first raising q_i to the second power and then normalizing by frequency per cluster:

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f_{j'}}, \tag{3}$$

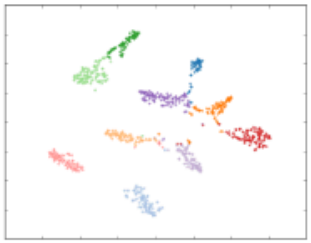
where $f_j = \sum_i q_{ij}$ are soft cluster frequencies. Please refer to section 5.1 for discussions on empirical properties of L and P.

- After that, just descend
  - note autoencoder initialization would probably be done differently now

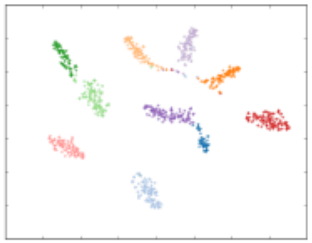
Clustering



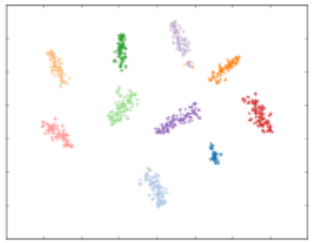
(a) Epoch 0



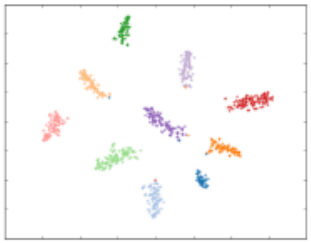
(b) Epoch 3



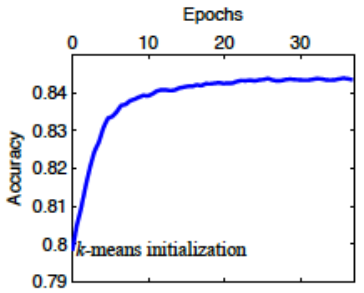
(c) Epoch 6



(d) Epoch 9



(e) Epoch 12



(f) Accuracy vs. epochs

Xie et al 15

Clustering

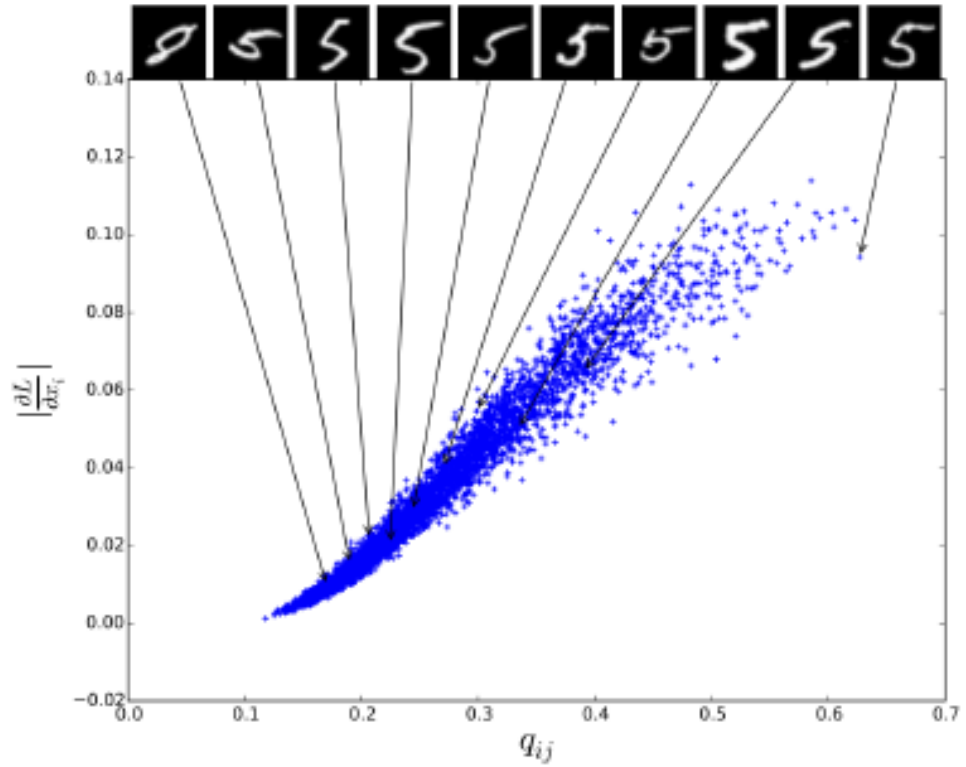
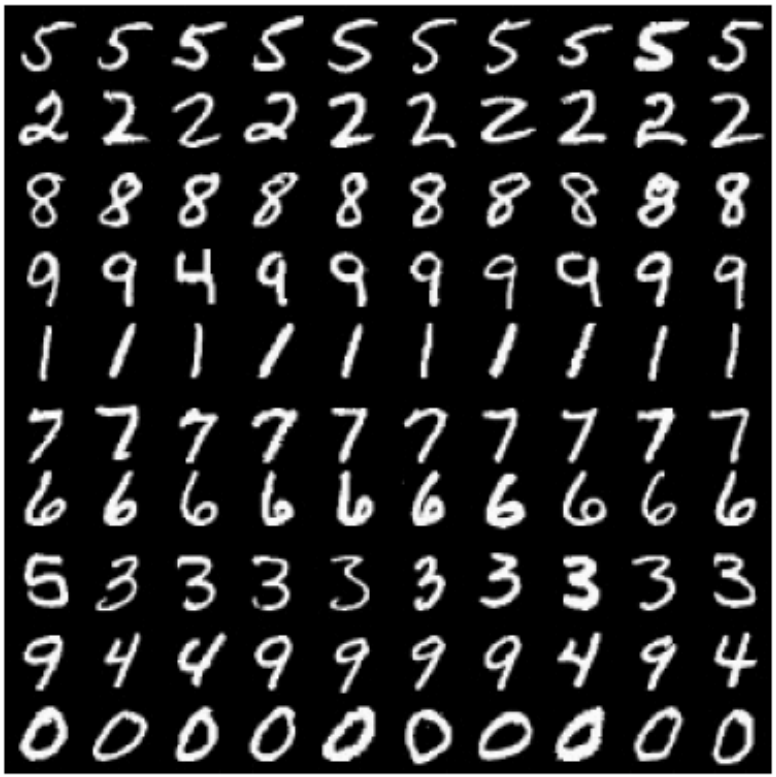


Figure 4. Gradient visualization at the start of KL divergence minimization. This plot shows the magnitude of the gradient of the loss L vs. the cluster soft assignment probability q_{ij}. See text for discussion.



(a) MNIST

(b) STL-10

Figure 3. Each row contains the top 10 scoring elements from one cluster.

Xie et al 15

Attribute discovery

- We have:
  - a set of images labelled with class, but not attribute
  - a feature construction (now very old fashioned)
- We want:
  - to associate each image with a bit vector
    - attribute present/absent
  - where
    - bits are “easily predicted”
    - bits are “informative”
    - bit vectors within a category cluster

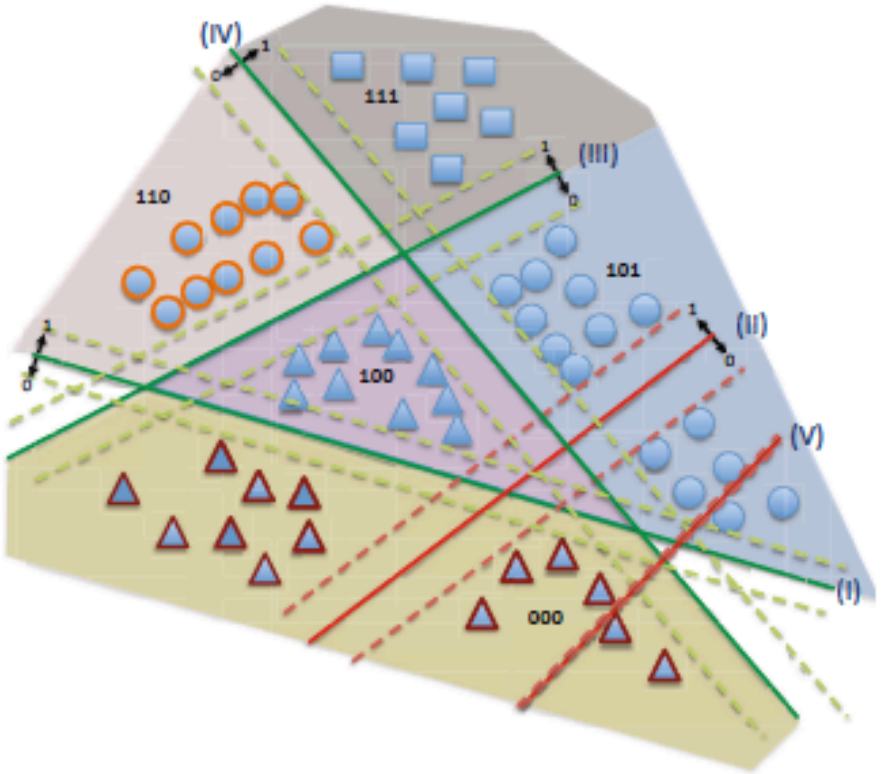


Fig. 1. Each bit in the code can be thought of as a hyperplane in the feature space. We learn arrangements of hyperplanes in a way that the resulting bit codes are discriminative and also all hyperplanes can be reliably predicted (enough margin). For example, the red hyperplanes (II,V) are not desirable because II is not informative(discriminative) and IV is not predictable (no margin). Our method allows the green hypeplanes (good ones) to sacrifice discrimination for predictability and vice versa. For example, our method allows the green hyperplane (I) to go through the triangle class because it has strong evidence that some of the triangles are very similar to circles.

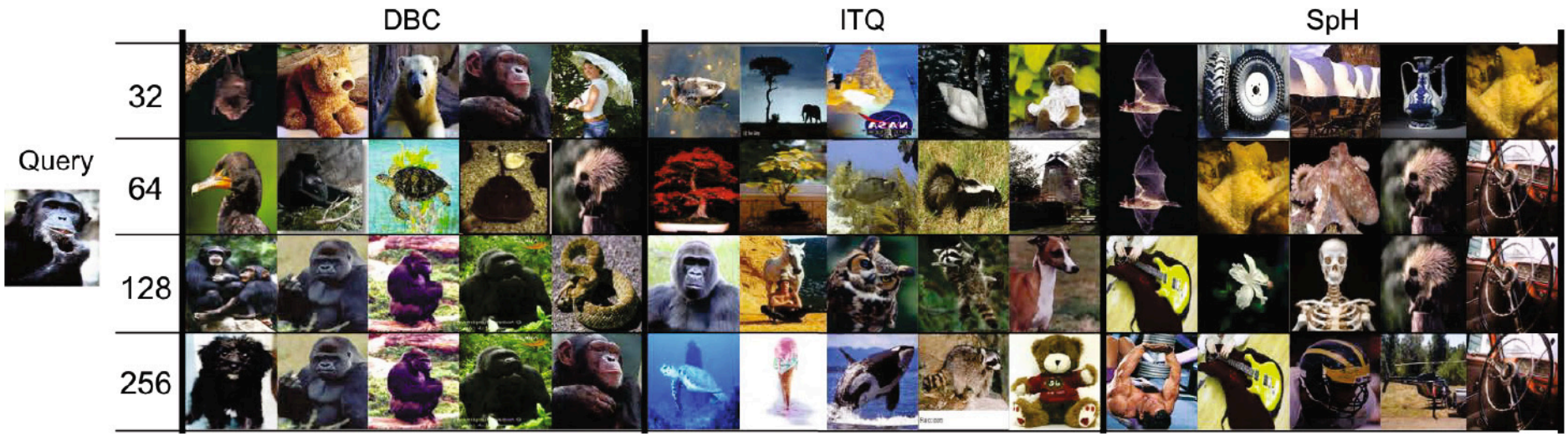


Fig. 6. This figure qualitatively compares the quality of retrieved images by our method comparing to that of ITQ and SpH. Each row corresponds to the top five images returned by three different methods: ours, ITQ and spectral hashing. This retrieval is done by first projecting the query image to the space of binary codes and then running KNN in that space. Notice how, even with relatively short codes(32 bits), our method recovers relevant objects. This menas that the discriminative training of the code has forced our code learning to focus on distinctive shared properties of categories. Our method consistently becomre more accurate as we increase the code size.

Fig. 8. Discovering attributes: Each bit corresponds to a hyperplane that group the data according to unknown notions of similarity. It is interesting to show what our bits have discovered. On two sides of the black bar we show 8 most confident images for 5 different hyperplanes/bits (Each row). Note that one can easily provide names for these attributes. For example, the bottom row corresponds to all round objects versus objects with straight vertical lines. The top row has silver, metalic and boxy objects on one side and natural images on the other side, the second row has water animals versus objects with checkerboard patterns. Discovered attributes are in the form of contrast: both sides have its own meaning. These attributes are compact representations of standard attributes that only explain one property. For more examples of discovered attributes please see supplementary material.

Why do we care?

- Each imputes labels by
  - compelling the label space to have strong properties
    - variant clustering
- DEC suggests that this is enough to learn features
  - DBC has fixed feature stack (but this is discriminative)
- Idea:
  - a feature stack that is discriminative
    - and perhaps has autoencoding properties
  - likely clusters appearance in a useful way
    - so you can impose labels by just compelling them to have spatial structure