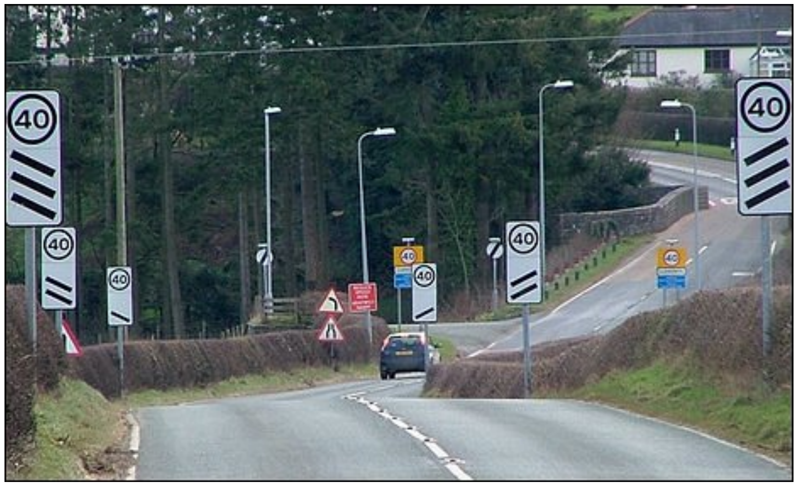
Scene Rep'n III
Building Ground Maps

D.A. Forsyth

Goal: Road Layout Map

- With minimal/no labelling
- In nasty geometries



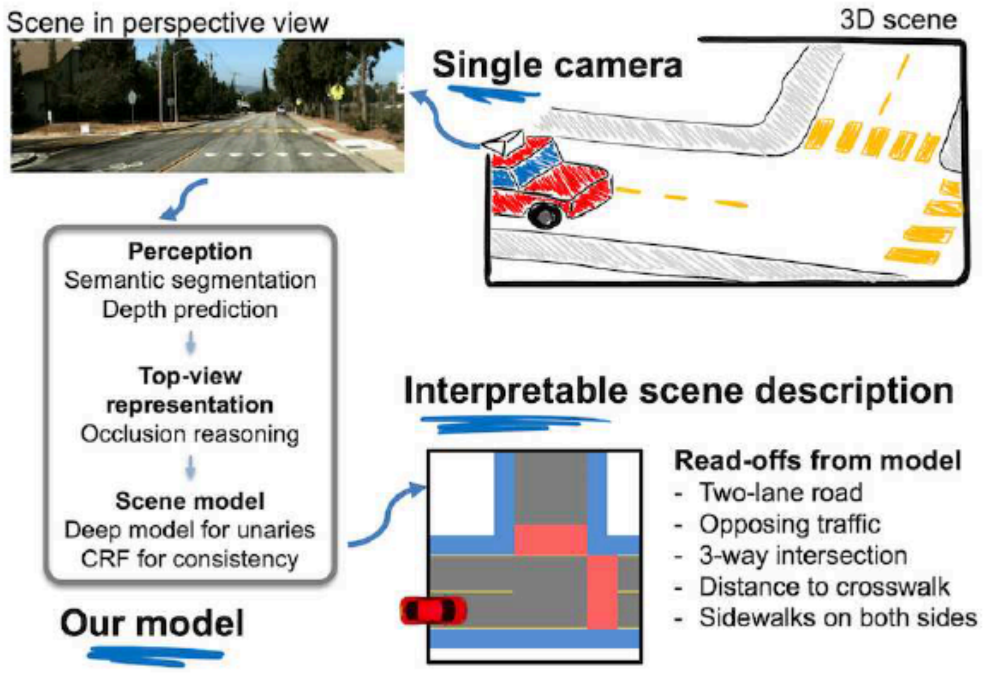


Figure 1: Our goal is to infer the layout of complex driving scenes from a single camera. Given a perspective image (top left) that captures a 3D scene, we predict a rich and interpretable scene description (bottom right), which represents the scene in an occlusion-reasoned semantic top-view.

Wang et al 19

Road layout maps

- A prediction of the layout of the main scene in front
  - distinguish between
    - transients (cars, pedestrians, etc)
    - and persistent (road, walkways, bicycle lanes, buildings)
  - including
    - intersections
    - lane boundaries
- Potential cues
  - streetview
  - openmaps
  - layout is stylized
  - persistent categories have coherent (but variable) appearance
  - scene flow/photometric consistency

Cues

- Incidental data
  - streetview+openmaps
- layout is stylized
- persistent categories have coherent appearance
- scene flow/photometric consistency

Layout is stylized

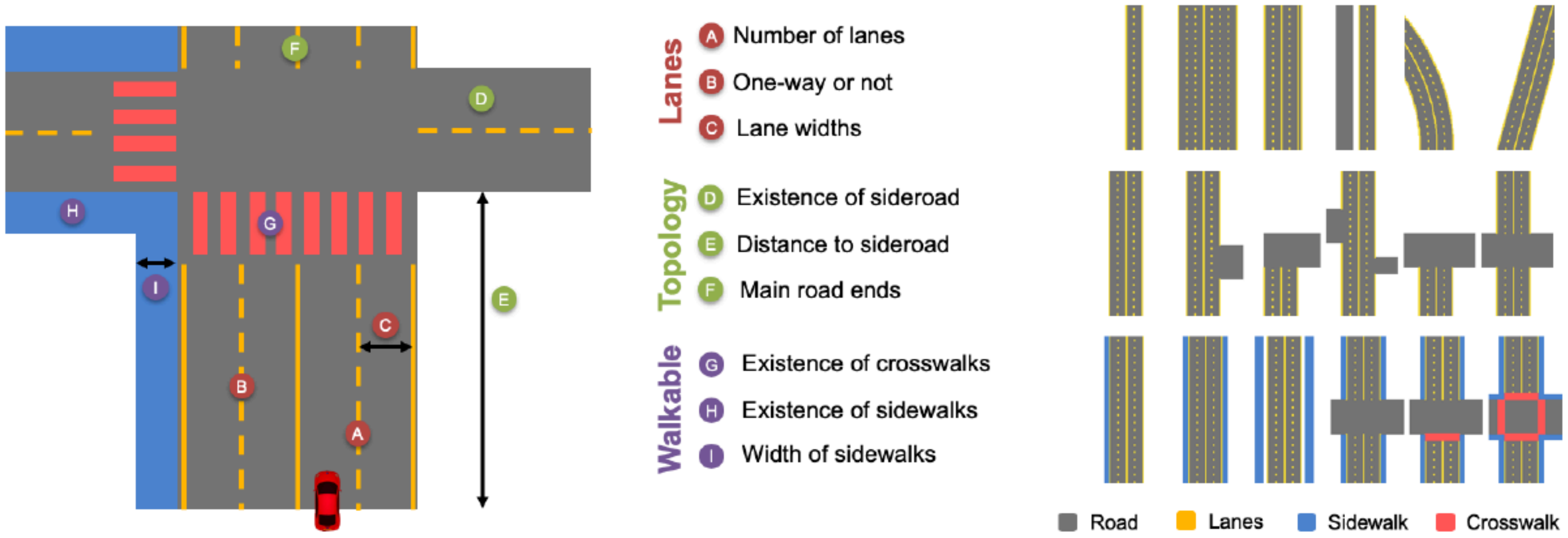


Figure 2: Our scene model consists of several parameters that capture a variety of complex driving scenes. (Left) We illustrate the model and highlight important parameters (A-I), which are grouped into three categories (middle): *Lanes*, to describe the layout of a single road; *Topology*, to model various road topologies; *Walkable*, describing scene elements for pedestrians. Our model is defined as a directed acyclic graph enabling efficient sampling and is represented in the top-view, making rendering easy. These properties turn our model into a simulator of semantic top-views. (Right) We show rendered examples for each of the above groups. A complete list of scene parameters and the corresponding graphical model is given in the supplementary.

Q: How do we impose structure?

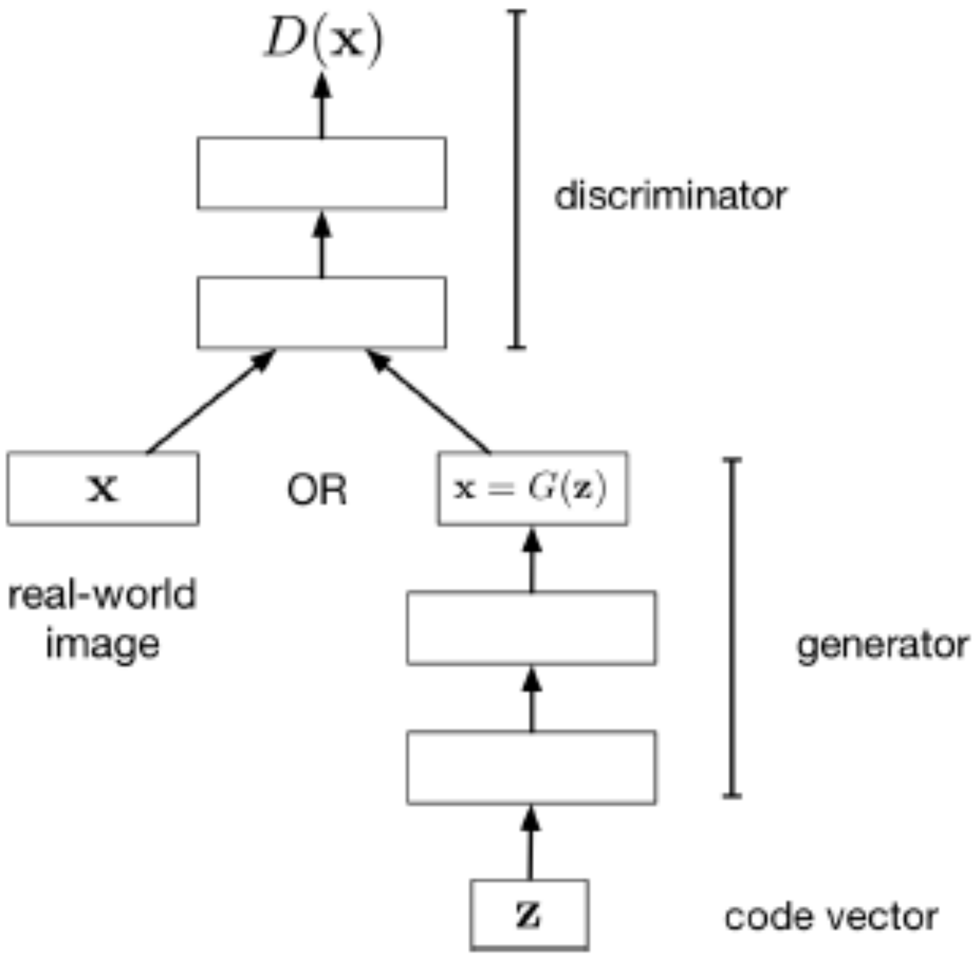
- We want to the network to produce layout maps that are “like real maps”
  - How?

Side topic - Adversarial losses

- Issue:
    - we are making pictures that should have a strong structure
        - albedo piecewise constant, etc.
        - but we don’t know how to write a loss that imposes that structure
- Strategy:
    - build a classifier that tries to tell the difference between
        - true examples
        - examples we made
    - use that classifier as a loss

A GAN

Generative
Adversarial
Network



Grosse slides

- Let D denote the discriminator's predicted probability of being data
- Discriminator's cost function: cross-entropy loss for task of classifying real vs. fake images

$$\mathcal{J}_D = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

Notice: we want the discriminator to make a 1 for real data, 0 for fake data

- One possible cost function for the generator: the opposite of the discriminator's

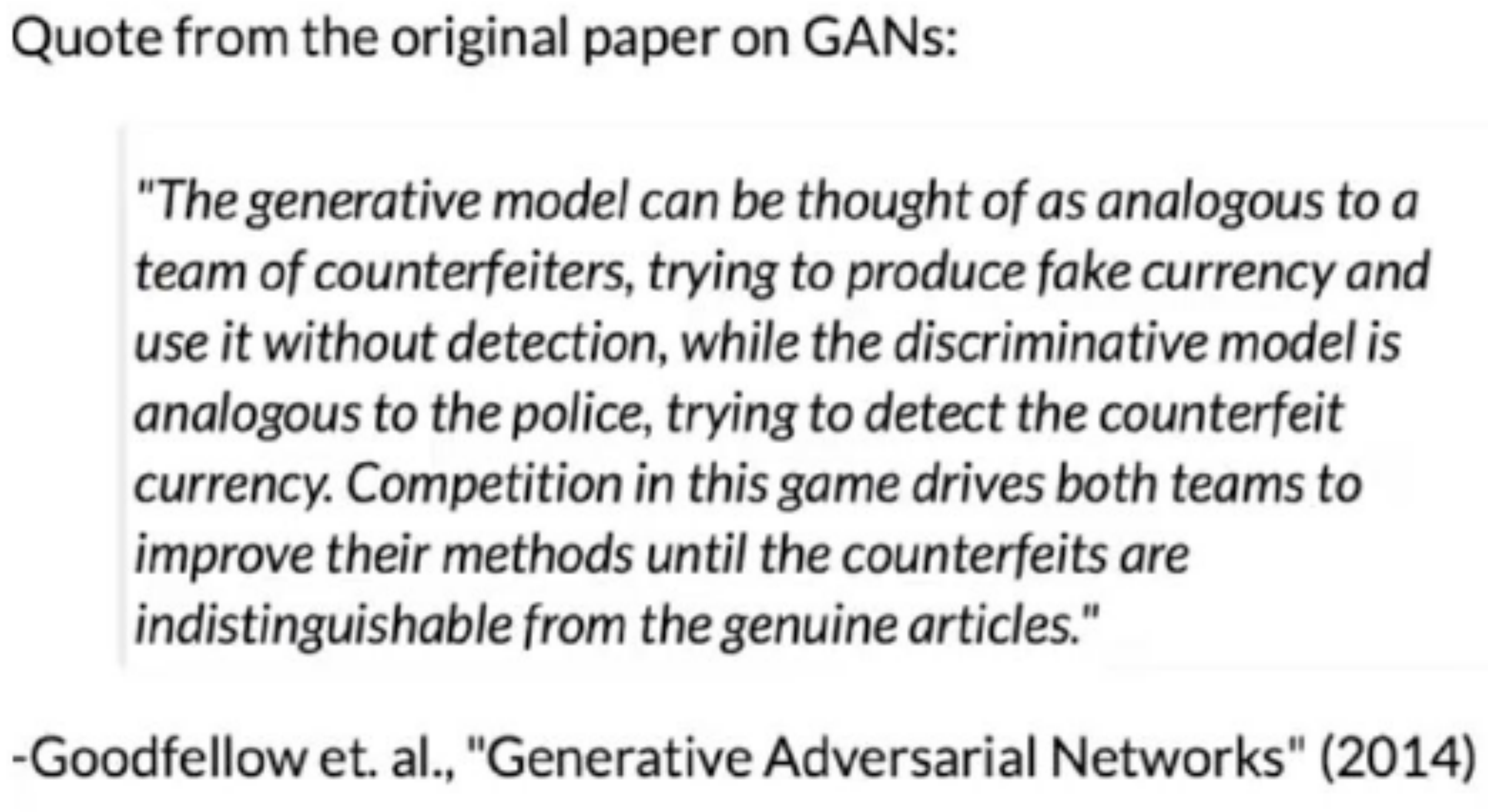
$$\begin{aligned}\mathcal{J}_G &= -\mathcal{J}_D \\ &= \text{const} + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]\end{aligned}$$

- This is called the minimax formulation, since the generator and discriminator are playing a zero-sum game against each other:

$$\max_G \min_D \mathcal{J}_D$$

Solution (if exists, which is uncertain; and if can be found, ditto) is known as a saddle point. It has strong properties, but not much worth talking about, as we don't know if it is there or whether we have found it.

Grosse slides



Quote from the original paper on GANs:

> *"The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles."*

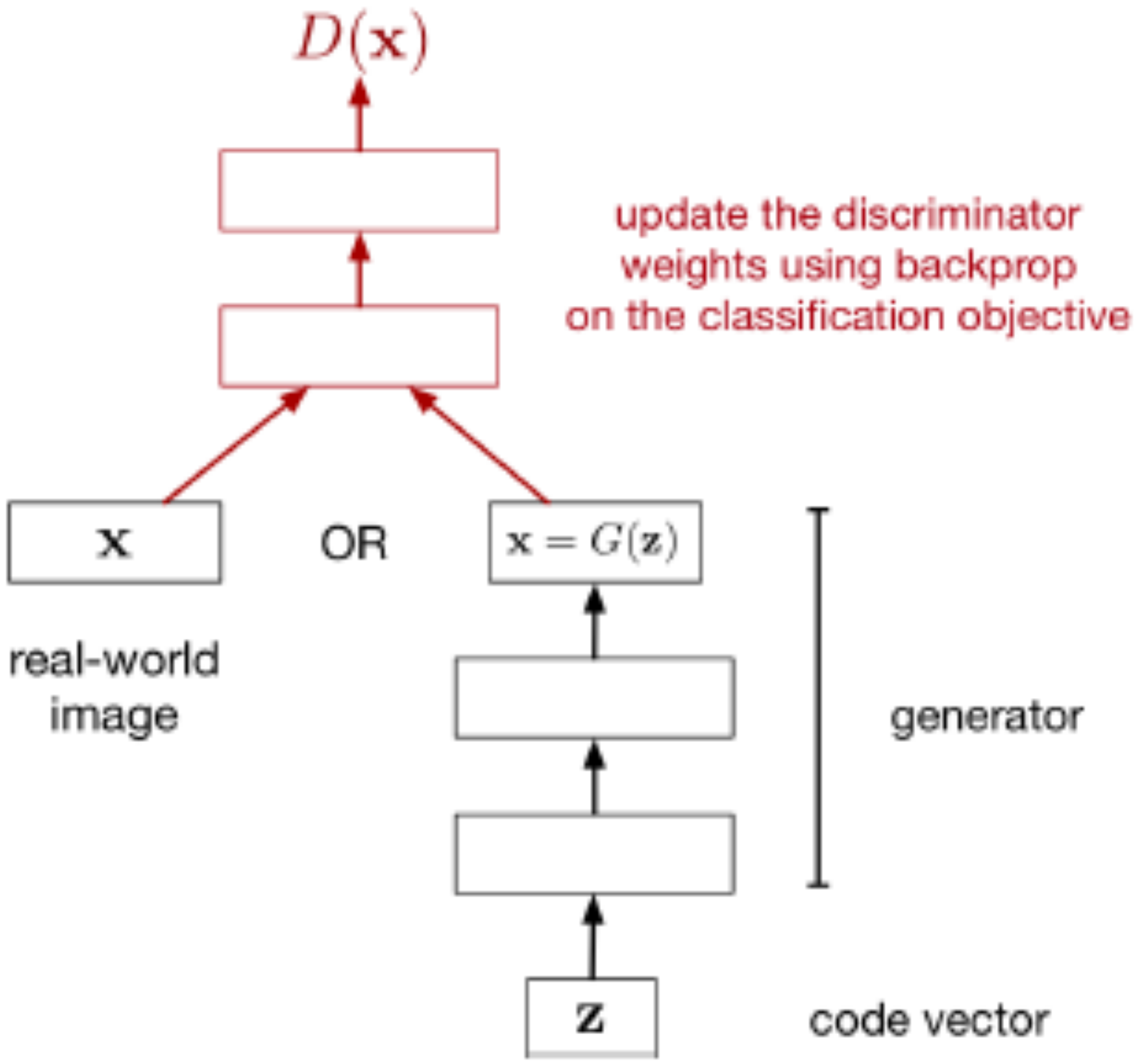
-Goodfellow et. al., "Generative Adversarial Networks" (2014)

Thakar slides

Important, general issue

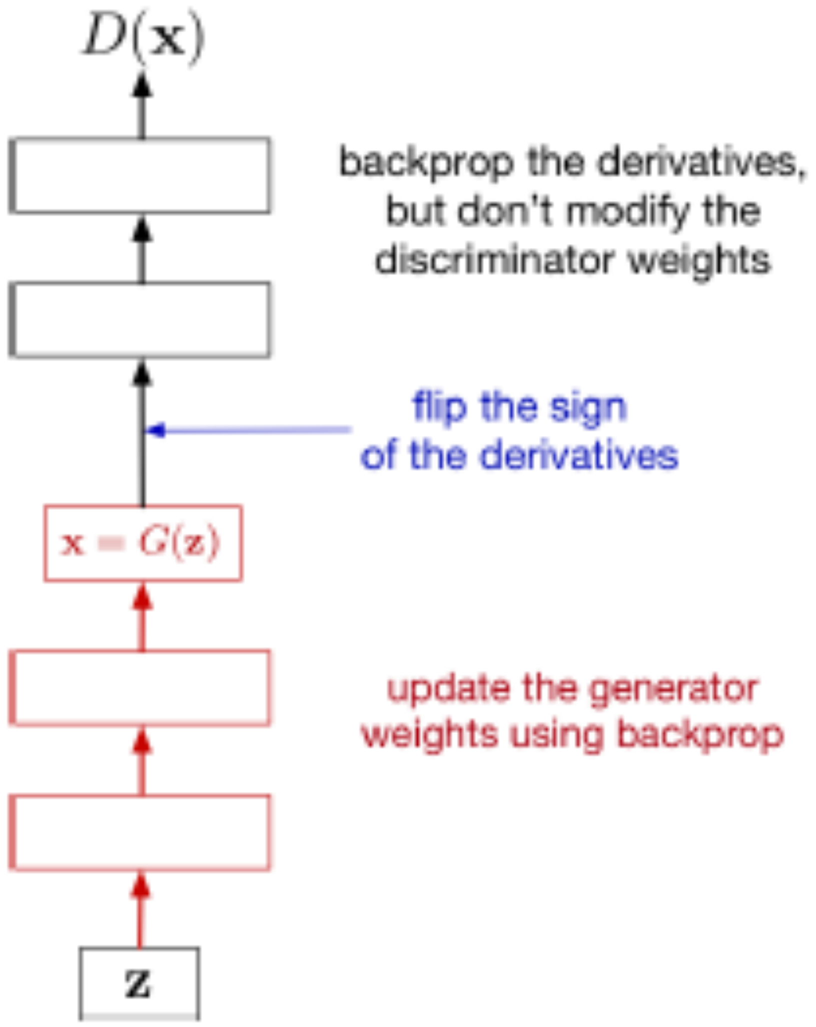
- If either generator or discriminator “wins” -> problem
- Discriminator “wins”
  - it may not be able to tell the generator how to fix examples
  - discriminators classify, rather than supply gradient
- Generator “wins”
  - likely the discriminator is too stupid to be useful
- Very little theory to guide on this point

Updating the discriminator:


$D(\mathbf{x})$
update the discriminator weights using backprop on the classification objective
$\mathbf{x}$
OR
$\mathbf{x} = G(\mathbf{z})$
real-world image
generator
$\mathbf{z}$
code vector

Grosse slides

Updating the generator:


$D(\mathbf{x})$
backprop the derivatives, but don't modify the discriminator weights
flip the sign of the derivatives
$\mathbf{x} = G(\mathbf{z})$
update the generator weights using backprop
$\mathbf{z}$

Grosse slides

One must be careful about losses…

- We introduced the minimax cost function for the generator:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- One problem with this is saturation.
- Recall from our lecture on classification: when the prediction is really wrong,
  - "Logistic + squared error" gets a weak gradient signal
  - "Logistic + cross-entropy" gets a strong gradient signal
- Here, if the generated sample is really bad, the discriminator's prediction is close to 0, and the generator's cost is flat.

Grosse slides

One must be careful about losses…

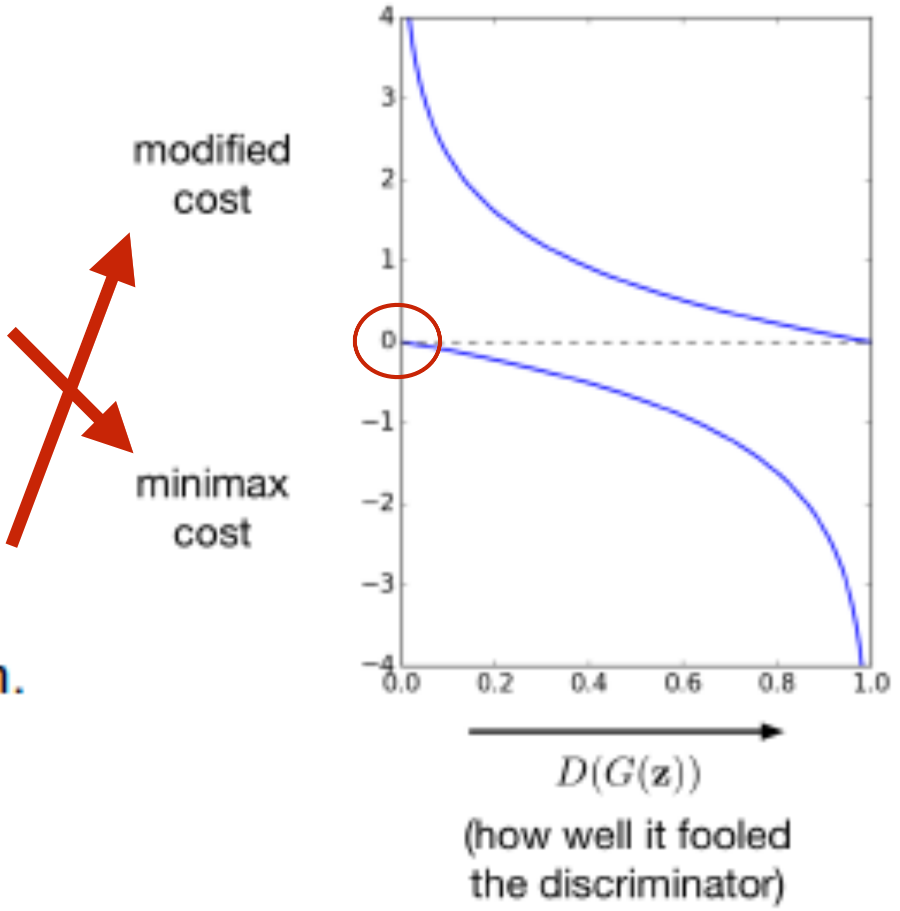
- Original minimax cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- Modified generator cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[-\log D(G(\mathbf{z}))]$$

- This fixes the saturation problem.



Grosse slides

Alternative losses

- Hinge:
  - Discriminator makes D(im)
    - want
      - real images -> -1
      - fake -> 1
  - Discriminator loss:

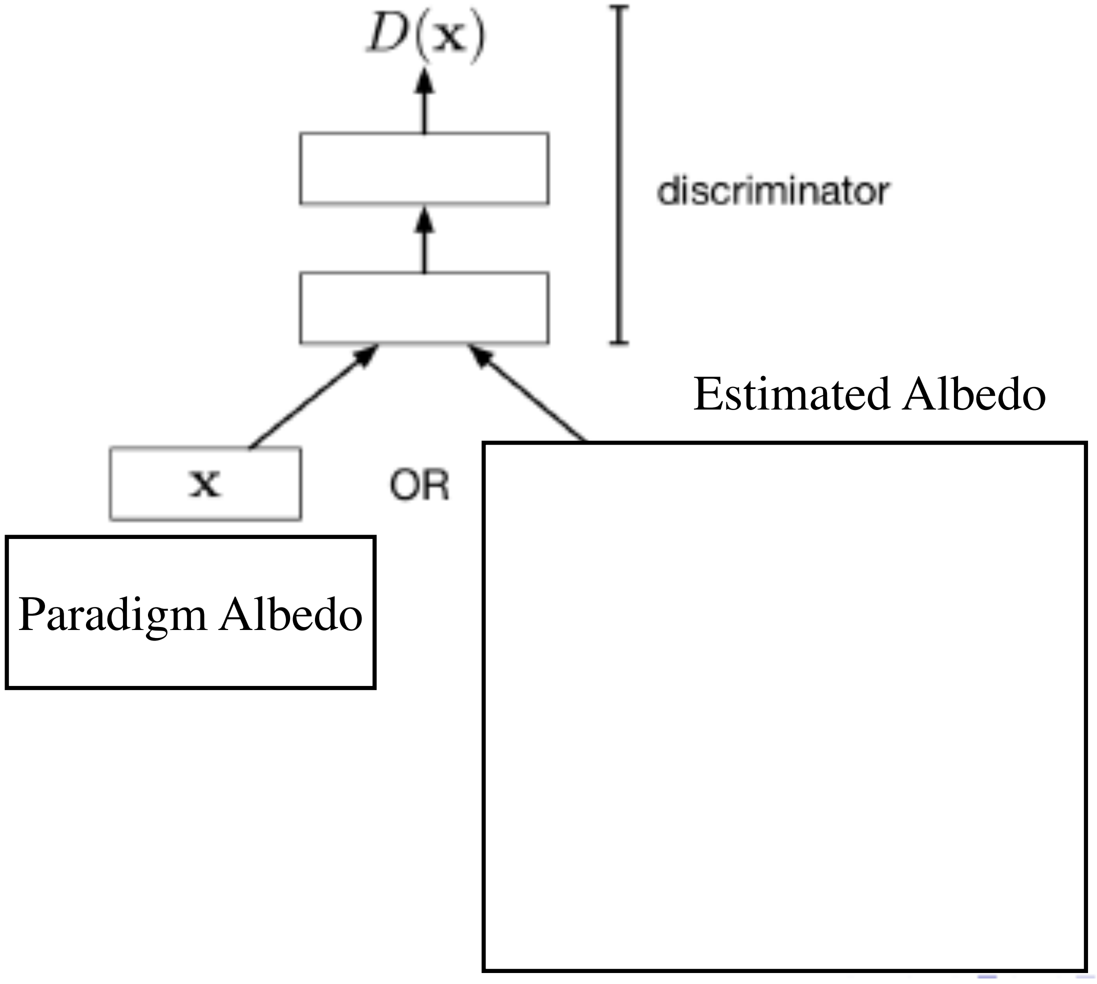
$$\sum_{\text{fakes and real}} \max(0, 1 - y_i D(I_i))$$

    - where y_i=-1 for real, y_i=1 for fake
  - Generator loss:
    -

$$\sum_{\text{fakes}} D(I_i)$$

Adversarial loss

Adversarial loss



Image

Grosse slides

Theory

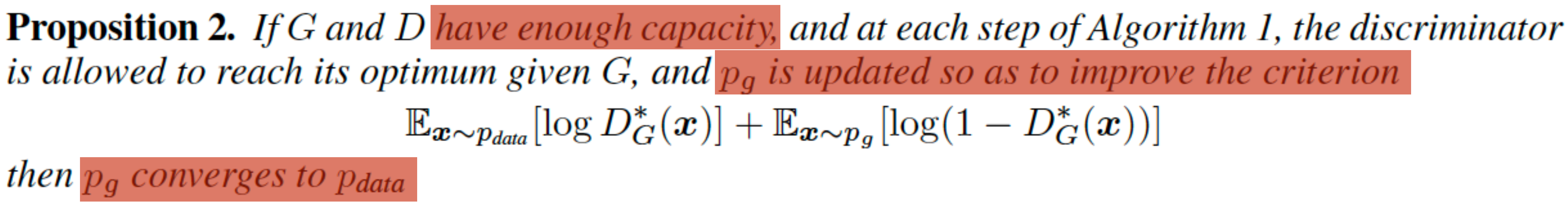
Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data}

Goodfellow et al 14

“Theory”


Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data}

- What if they don’t have enough capacity?
- What if p_g doesn’t make “enough progress”?
- In what sense converges?
  - p_data is a set of samples
  - we DON’T WANT usual convergences
  - we WANT convergence to some smoothed p_data
    - how smoothed? how controlled?

Goodfellow et al 14

Questions

- How do we hobble an adversary in a useful way?
  - dunno
- When is an adversarial smoother helpful?
  - dunno
-

Layout is stylized

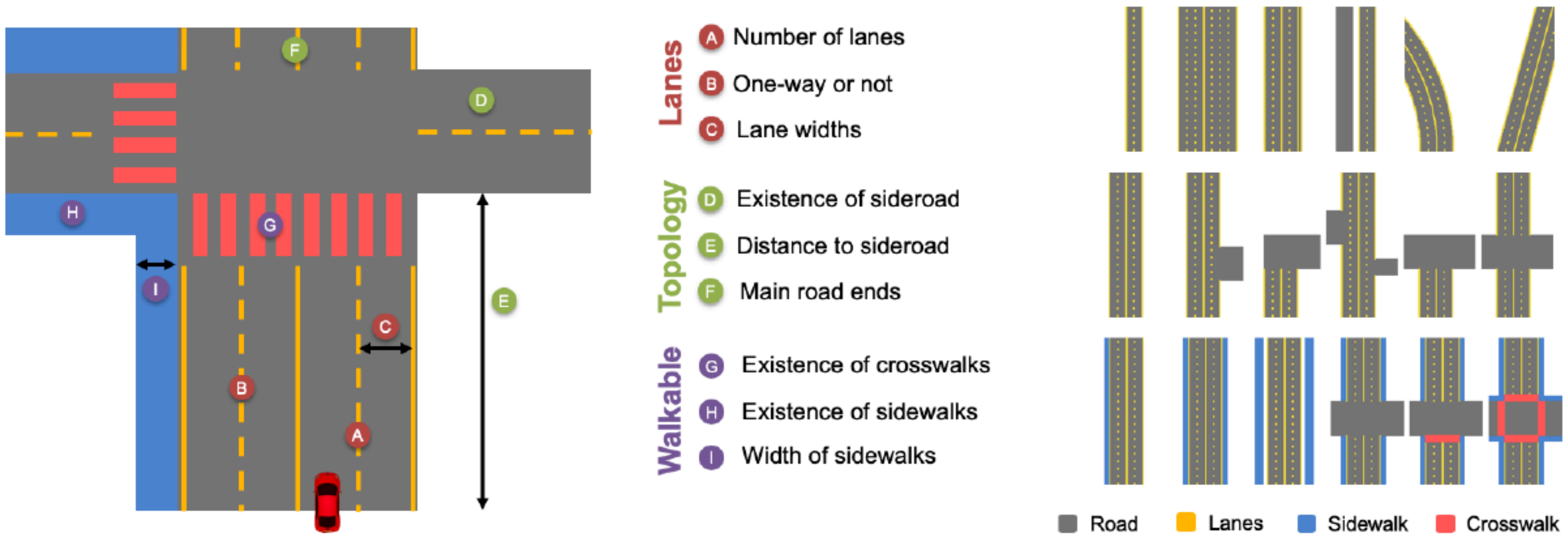


Figure 2: Our scene model consists of several parameters that capture a variety of complex driving scenes. (Left) We illustrate the model and highlight important parameters (A-I), which are grouped into three categories (middle): *Lanes*, to describe the layout of a single road; *Topology*, to model various road topologies; *Walkable*, describing scene elements for pedestrians. Our model is defined as a directed acyclic graph enabling efficient sampling and is represented in the top-view, making rendering easy. These properties turn our model into a simulator of semantic top-views. (Right) We show rendered examples for each of the above groups. A complete list of scene parameters and the corresponding graphical model is given in the supplementary.

Wang et al 18

In overhead view

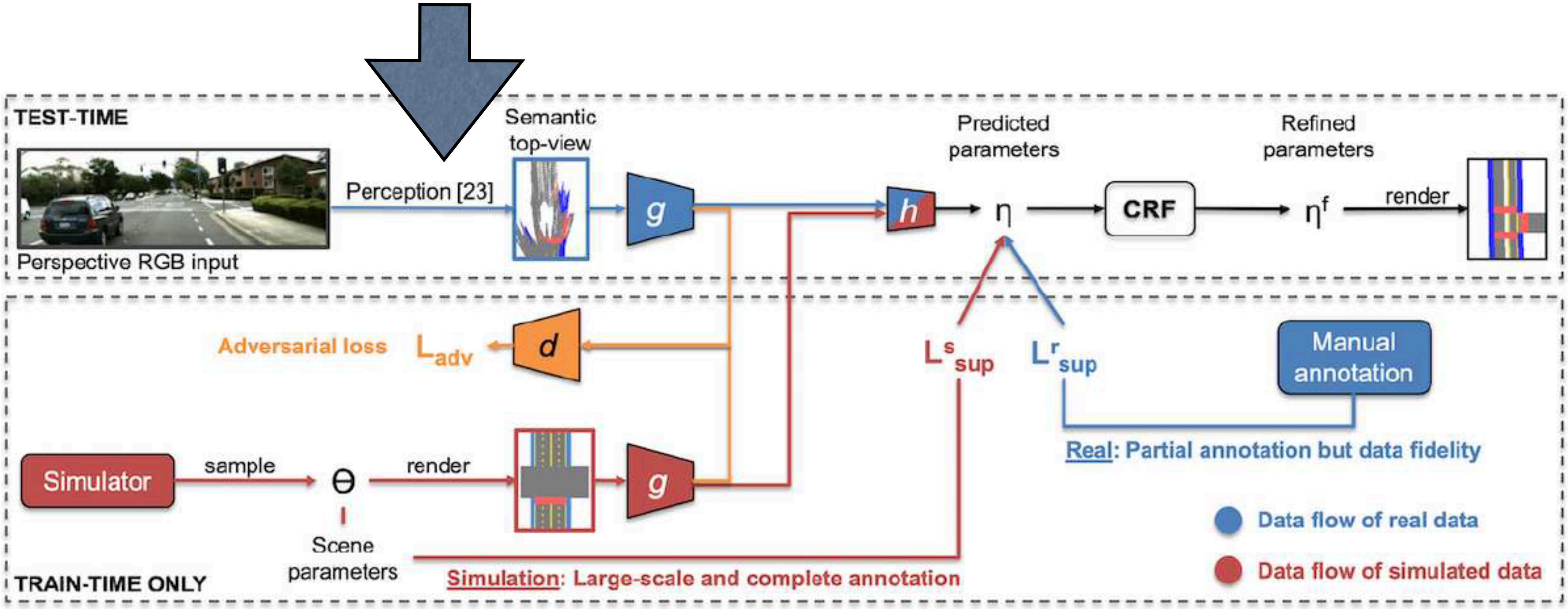
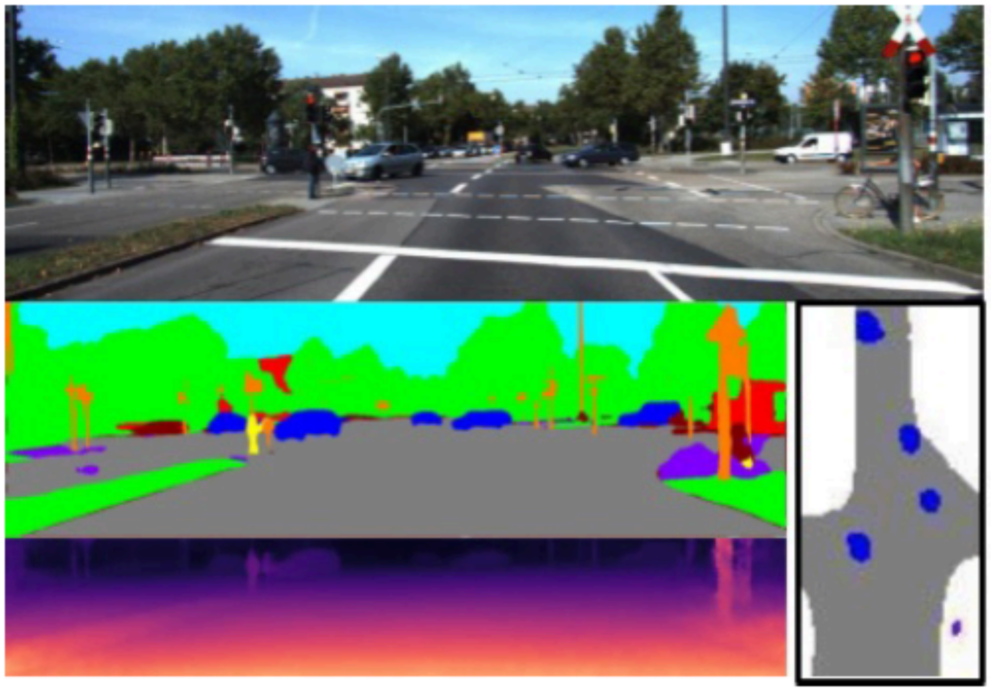
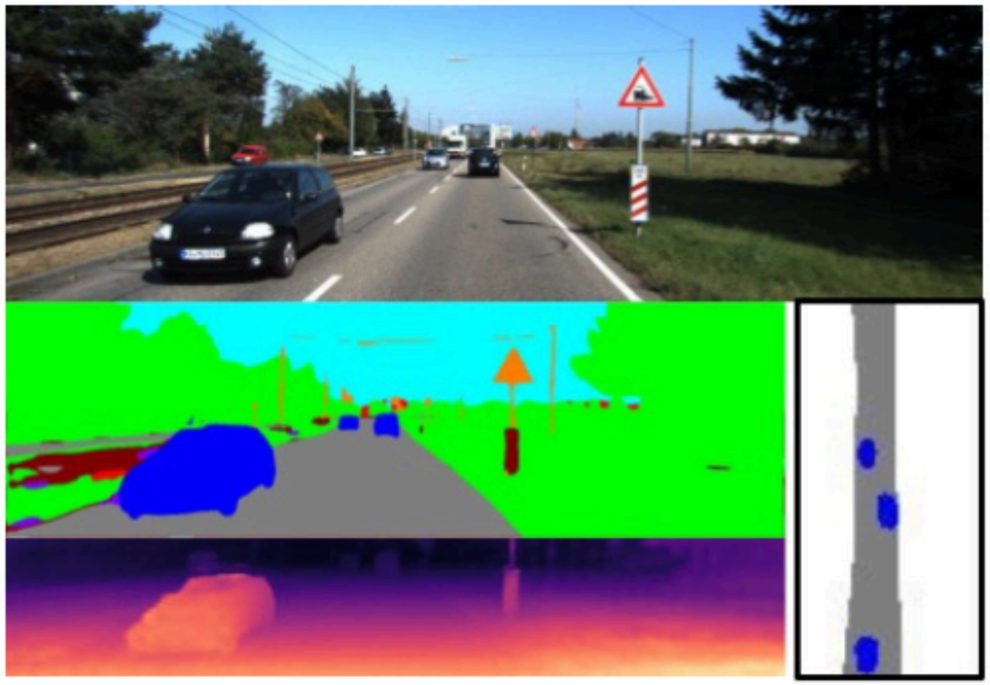


Figure 3: **Overview of our proposed framework:** At train-time, our framework makes use of both manual annotation for real data (blue) and automated annotation for simulated data (red), see Sec. 3.2. The feature extractors g convert semantic top views from either domain into a common representation which is input to h. An adversarial loss (orange) encourages a domain-agnostic output of g. At test-time, an RGB image in the perspective view is first transformed into a semantic top-view [23], which is then used by our proposed neural network (see Sec. 3.3), $h \circ g$, to infer our scene model (see Sec. 3.1). The graphical model defined in Sec. 3.4 ensures a coherent final output.

Wang et al 19

Issue - cars and pedestrians

- Moving objects obscure ground map
    - can be fixed

Schulter et al 18

Detector to mask

Inpaint semantics and depth

Fix the ground map

Map to ground

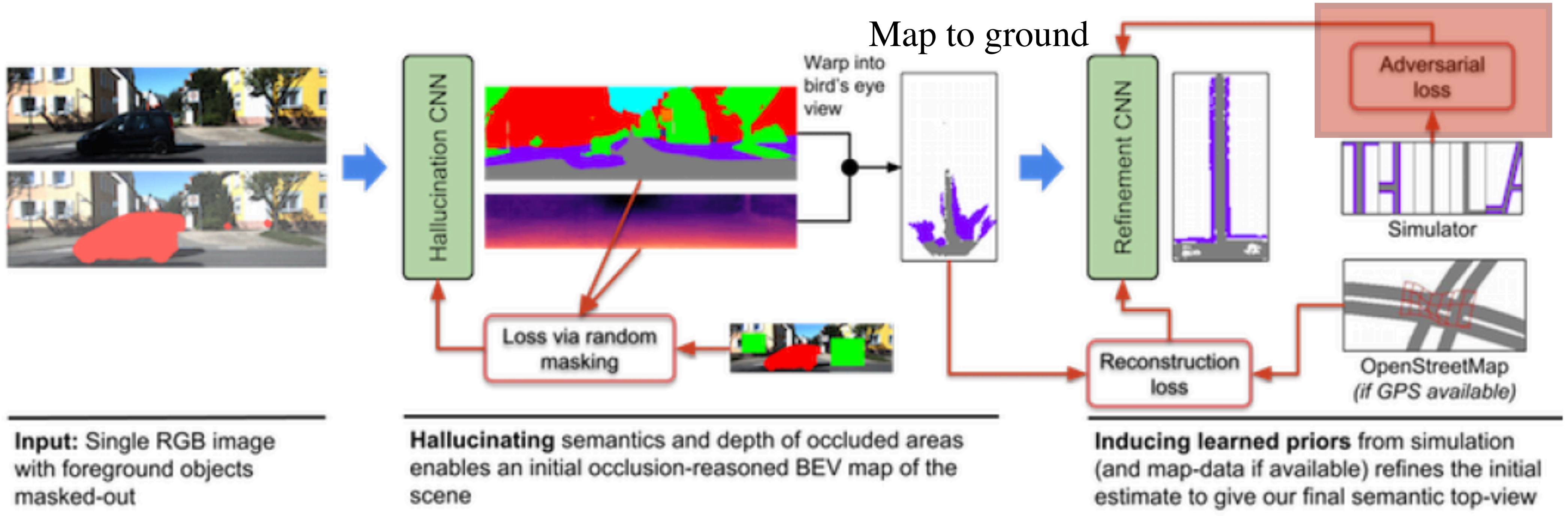
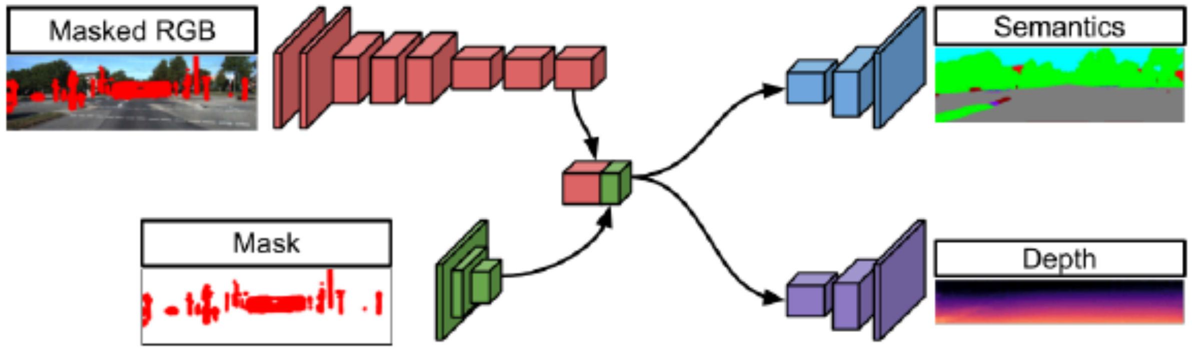

Hallucination CNN
Warp into bird's eye view
Loss via random masking
Refinement CNN
Adversarial loss
Simulator
Reconstruction loss
OpenStreetMap (if GPS available)
Input: Single RGB image with foreground objects masked-out
Hallucinating semantics and depth of occluded areas enables an initial occlusion-reasoned BEV map of the scene
Inducing learned priors from simulation (and map-data if available) refines the initial estimate to give our final semantic top-view

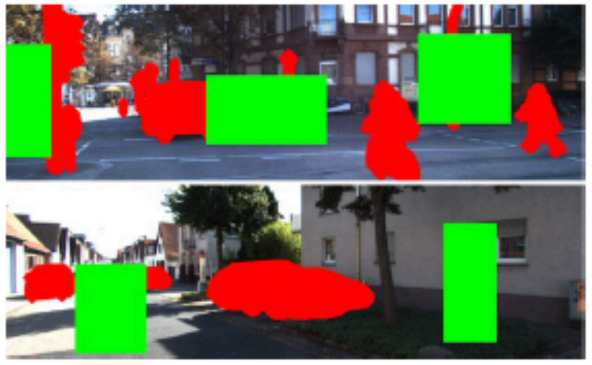
Fig. 1: Given a single RGB image of a typical street scene (left), our approach creates an **occlusion-reasoned semantic map of the scene layout in the bird's eye view**. We present a CNN that can hallucinate depth and semantics in areas occluded by foreground objects (marked in red and obtained via standard semantic segmentation), which gives an initial but noisy and incomplete estimate of the scene layout (middle). To fill in unobserved areas in the top-view, we further propose a refinement-CNN that induces learning strong priors from simulated and OpenStreetMap data (right), which comes at no additional annotation costs.

Schulter et al 18

Inpainting



(a)



(b)

Fig. 2: **(a)** The *inpainting CNN* first encodes a masked image and the mask itself. The extracted features are concatenated and two decoders predict semantics and depth for visible and occluded pixels. **(b)** To train the inpainting CNN we ignore foreground objects as no ground truth is available (red) but we *artificially add masks (green)* over background regions where full annotation is already available.

Notice: we inpaint labels and depth, NOT the image

Notice: depth is inferred from the image

Schulter et al 18

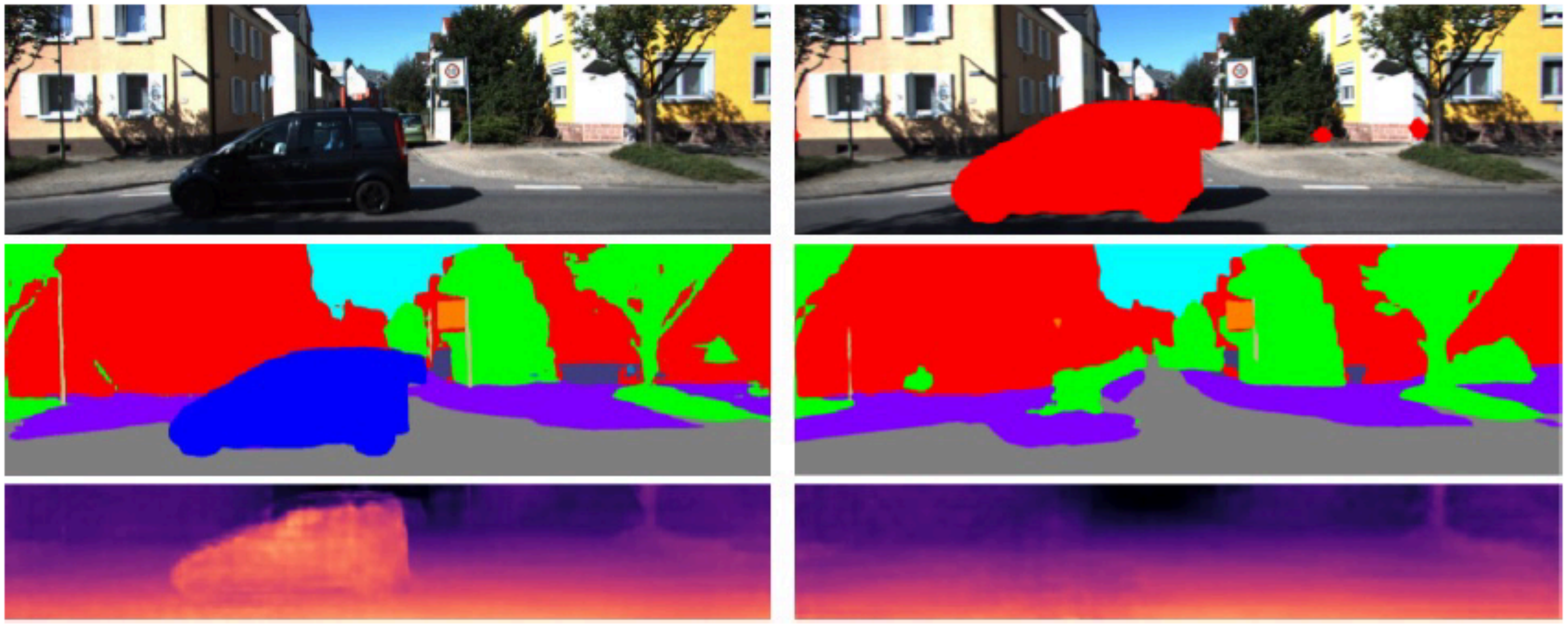


Fig. 6: Qualitative example of our hallucination CNN: Semantics and depth without (left) and with (right) hallucination.

Schulter et al 18

Birds eye view from depth + labels

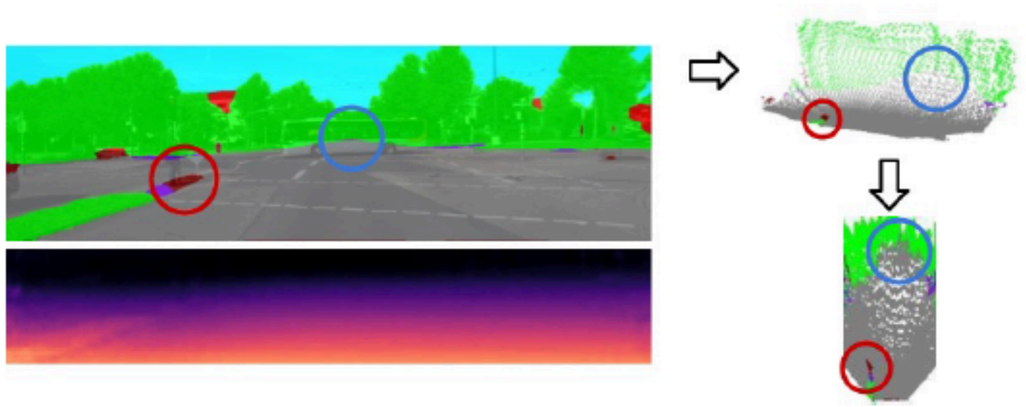


Fig. 3: The process of mapping the semantic segmentation with corresponding depth first into a 3D point cloud and then into the bird's eye view. The red and blue circles illustrate corresponding locations in all views.

Schulter et al 18

Refining birds eye predictions

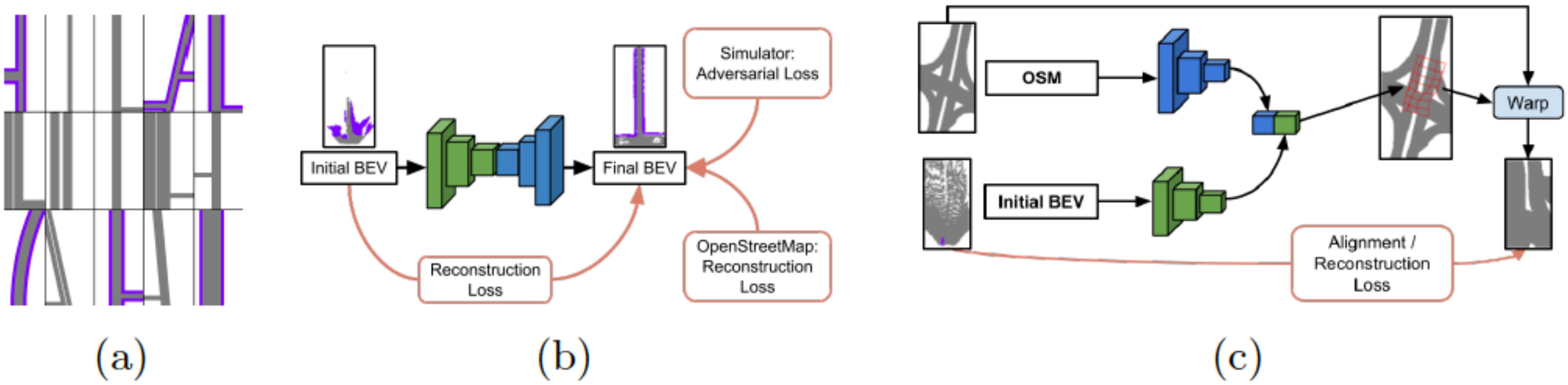
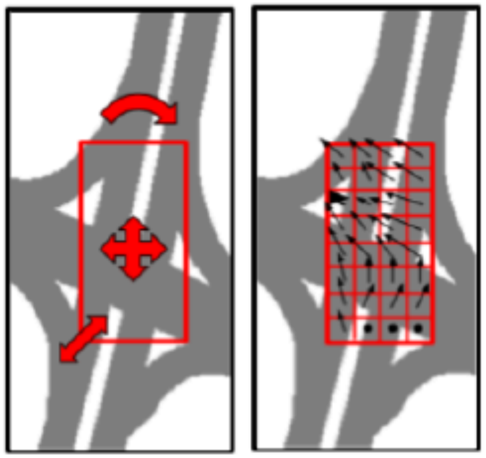

Initial BEV
Final BEV
Simulator: Adversarial Loss
Reconstruction Loss
OpenStreetMap: Reconstruction Loss
OSM
Initial BEV
Warp
Alignment / Reconstruction Loss
(a)
(b)
(c)

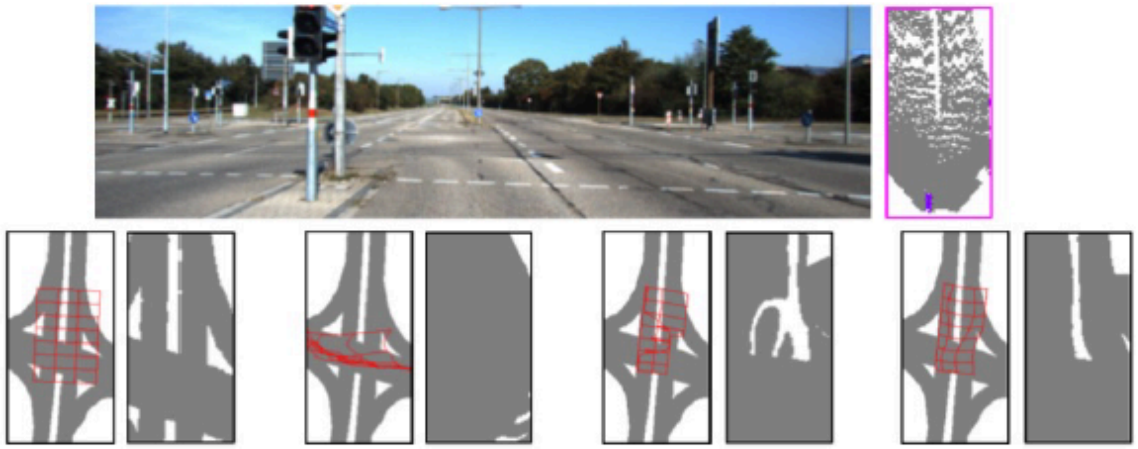
Fig. 4: **(a) Simulated road shapes** in the top-view. **(b) The refinement-CNN** is an encoder-decoder network receiving three supervisory signals: self-reconstruction with the input, adversarial loss from simulated data, and reconstruction loss with aligned OpenStreetMap (OSM) data. **(c) The alignment CNN** takes as input the initial BEV map and a crop of OSM data (via noisy GPS and yaw estimate given). The CNN predicts a warp for the OSM map and is trained to minimize the reconstruction loss with the initial BEV map.

Schulter et al 18

Warping OSM to map layout



(a)



(b)

Fig. 5: **(a)** We use a composition of similarity transform (left, "box") and a non-parametric warp (right, "flow") to align noisy OSM with image evidence. **(b, top)** Input image and the corresponding B^{init}. **(b, bottom)** Resulting warping grid overlaid on the OSM map and the warping result for 4 different warping functions, respectively: "box", "flow", "box+flow", "box+flow (with regularization)". Note the importance of composing the transformations and the induced regularization.

Schulter et al 18

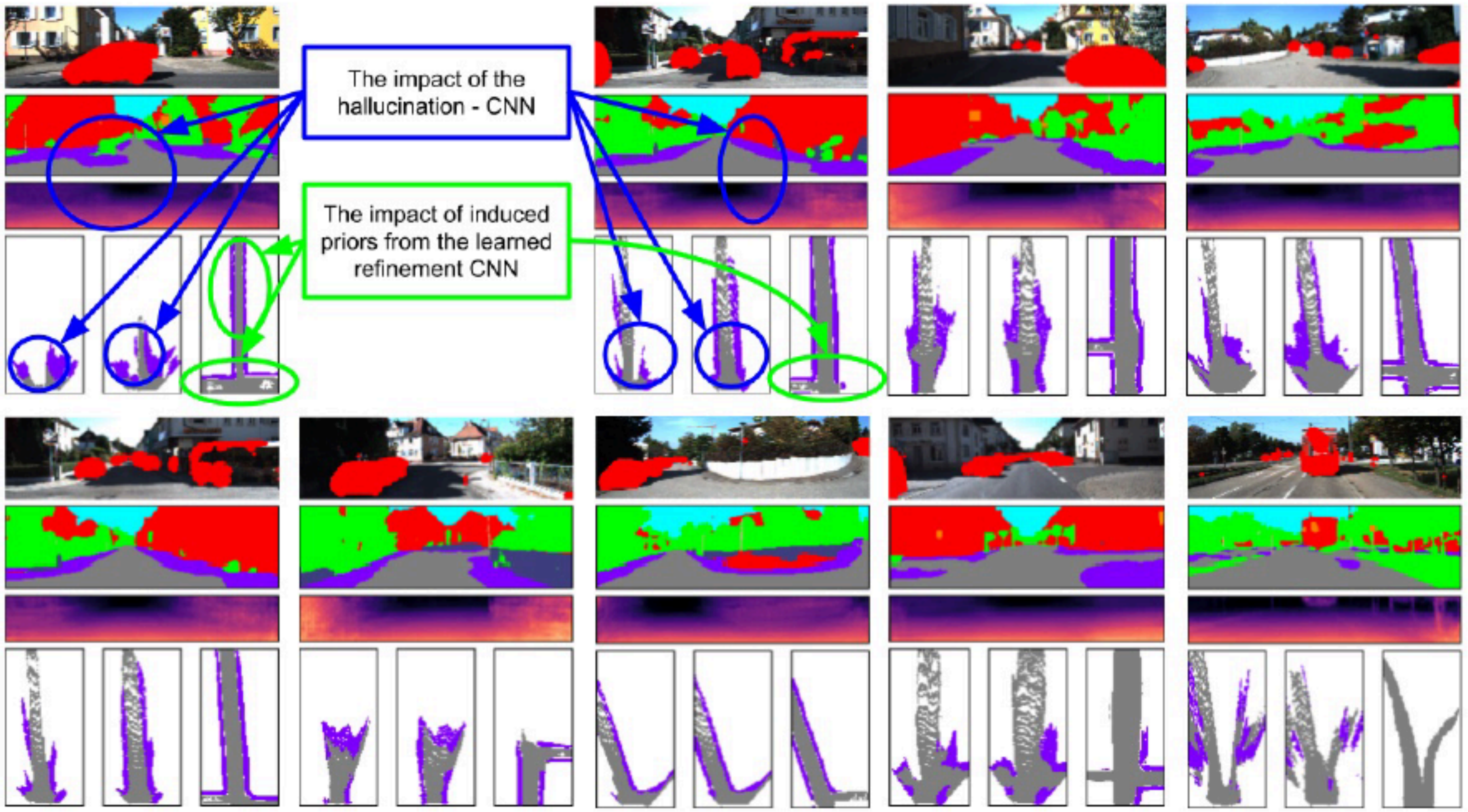
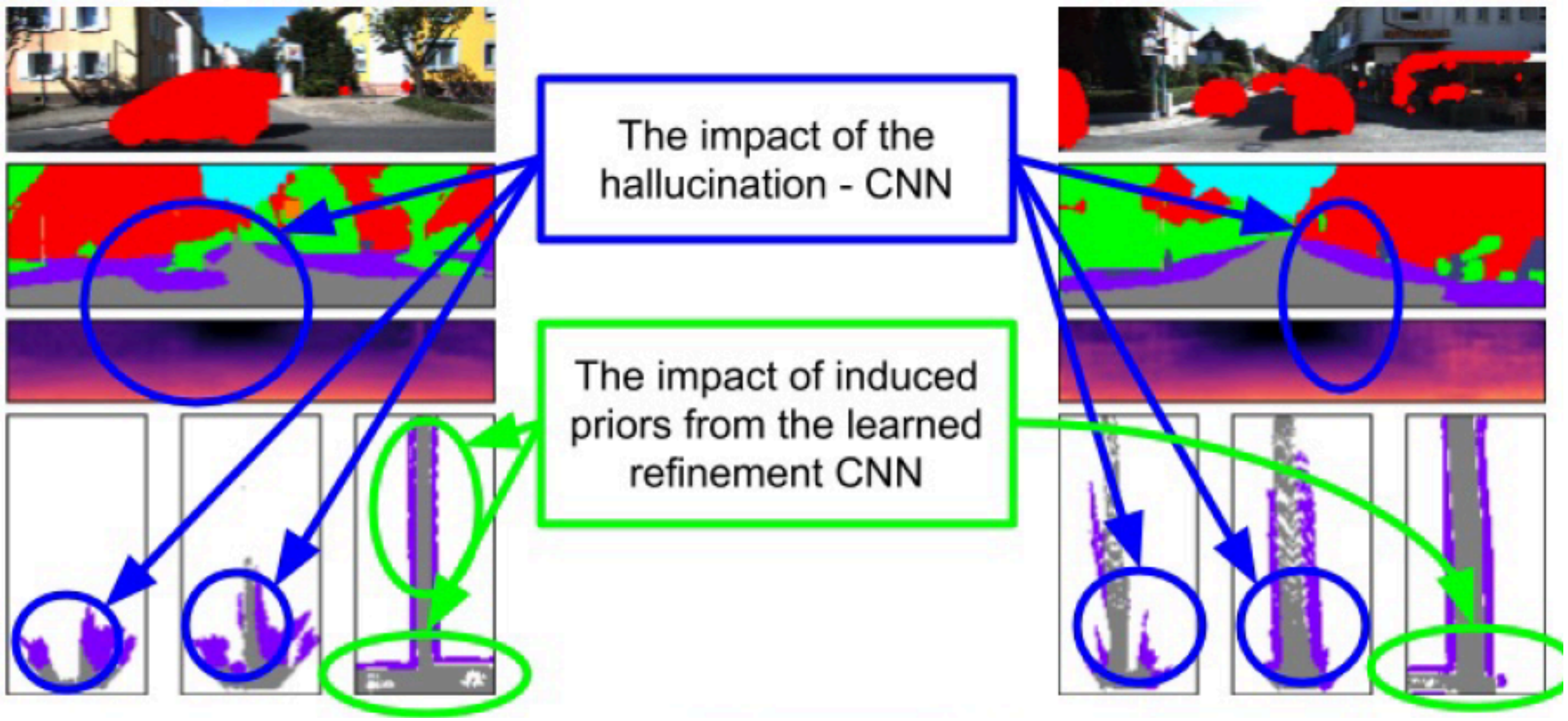
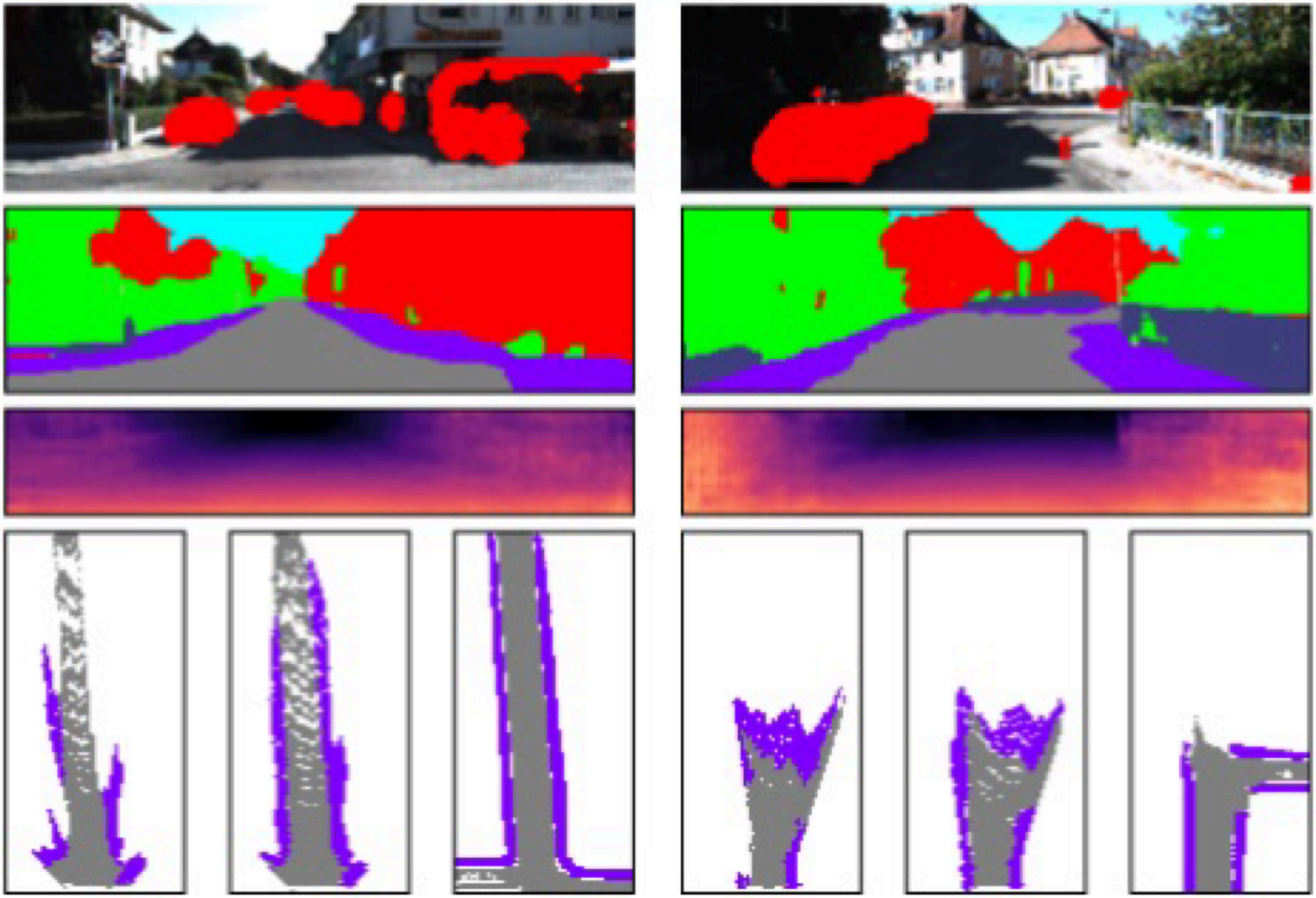

The impact of the hallucination - CNN
The impact of induced priors from the learned refinement CNN

Fig. 8: **Examples of our BEV representation**. Each one shows the masked RGB input, the hallucinated semantics and depth, as well as three BEV maps, which are (from left to right), The BEV map without hallucination, with hallucination, and after refinement. The last example depicts a failure case.

Schulter et al 18


The impact of the hallucination - CNN
The impact of induced priors from the learned refinement CNN

Schulter et al 18



Schulter et al 18

Good + bad

- Birds eye view is a good idea
    - right place to compare labels with models
- Label inpainting is good idea
    - but why in image?
    - the warping, registration seem to help A LOT with this
- It’s clear that warping, registration, adversary are helpful
    - adversary isn’t that helpful - why?
- If you’re going to warp OSM, why not use result of warp?
- Depth inference is a dubious idea
    - Why not use ground plane estimate?
    - and homography?

In overhead view

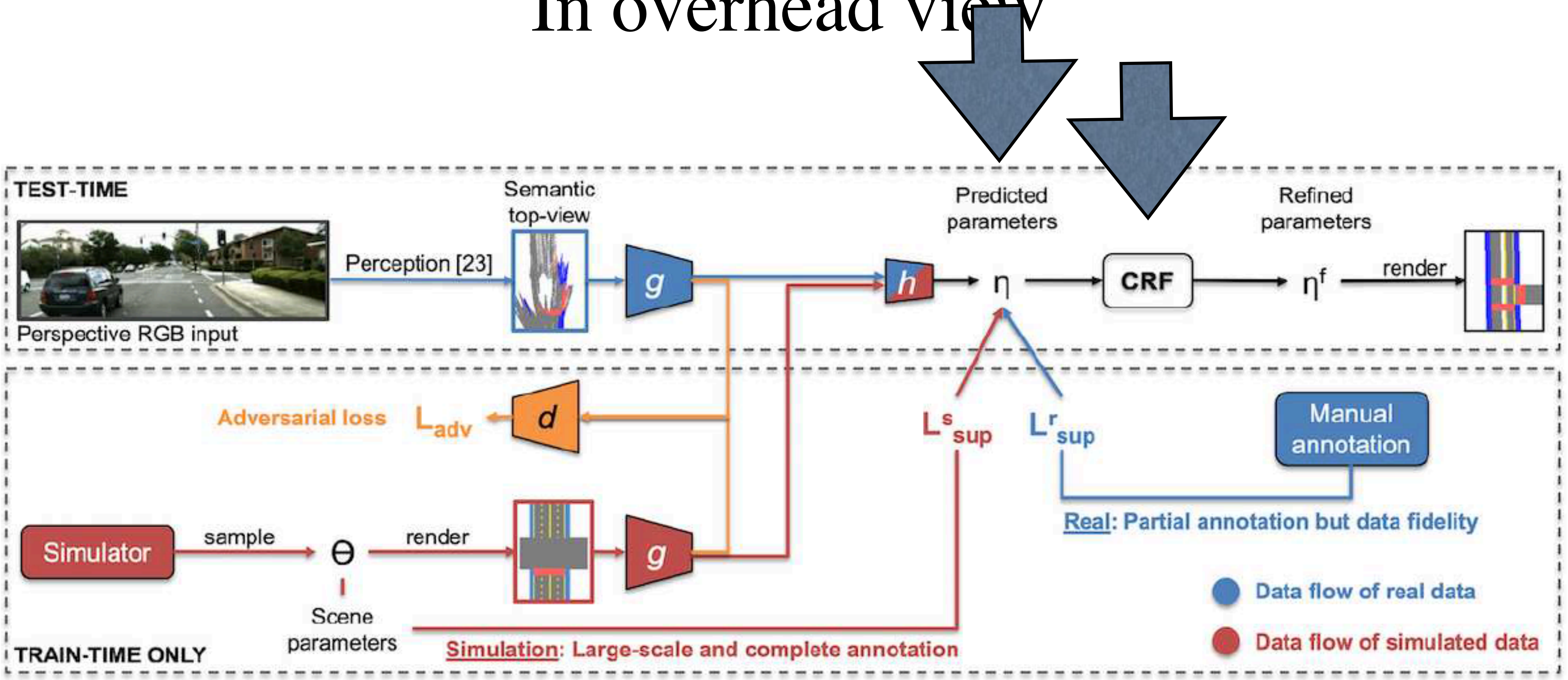


Figure 3: **Overview of our proposed framework:** At train-time, our framework makes use of both manual annotation for real data (blue) and automated annotation for simulated data (red), see Sec. 3.2. The feature extractors g convert semantic top views from either domain into a common representation which is input to h. An adversarial loss (orange) encourages a domain-agnostic output of g. At test-time, an RGB image in the perspective view is first transformed into a semantic top-view [23], which is then used by our proposed neural network (see Sec. 3.3), $h \circ g$, to infer our scene model (see Sec. 3.1). The graphical model defined in Sec. 3.4 ensures a coherent final output.

Wang 19

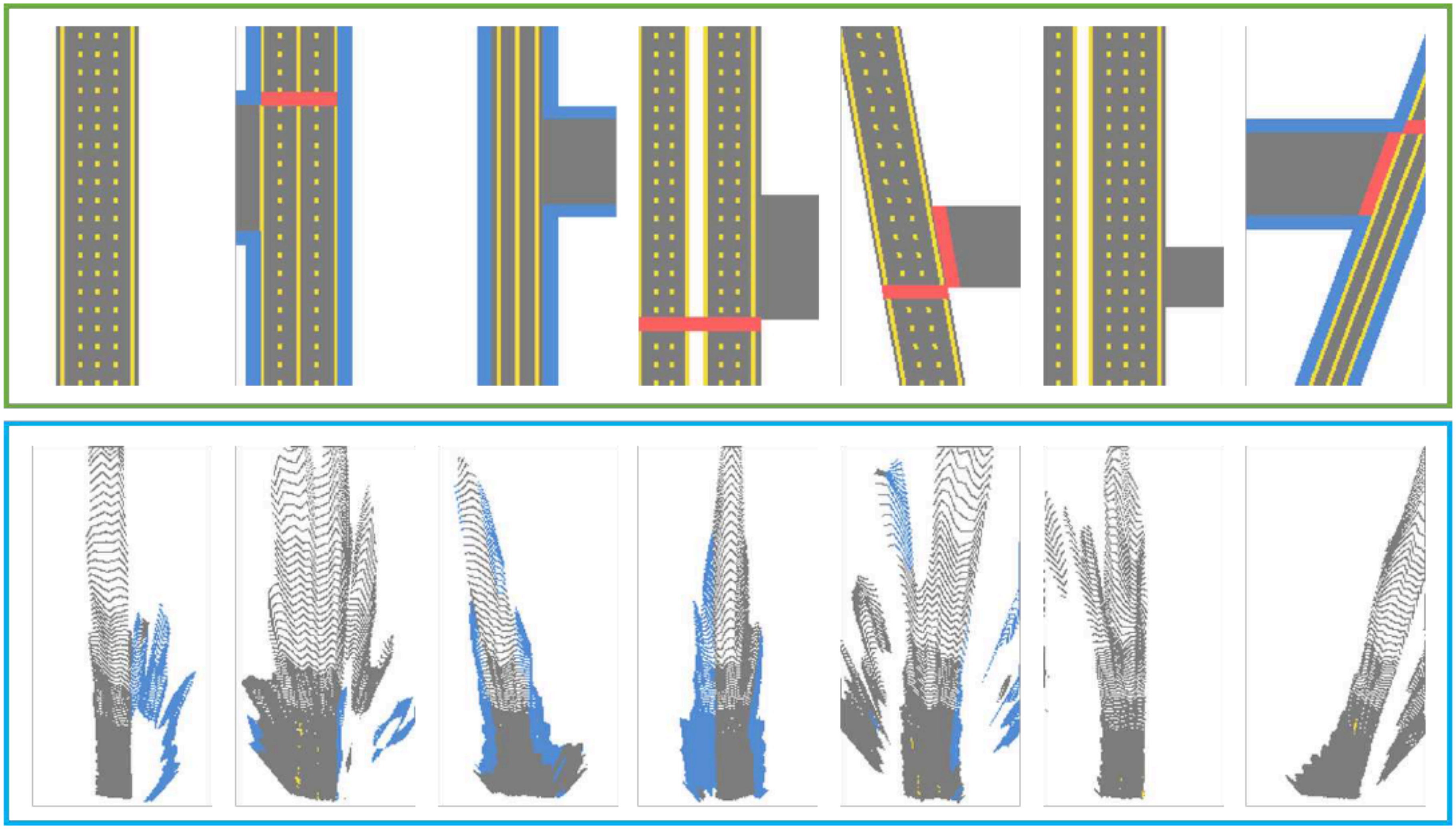


Figure 4: Unpaired examples of simulated semantic top-views (top) and real ones from [23] (bottom).

CRF

- Q: what does this apply to
  - I *think* predicted labels on “ground plane”
    - but what is discretization?

What is a CRF?

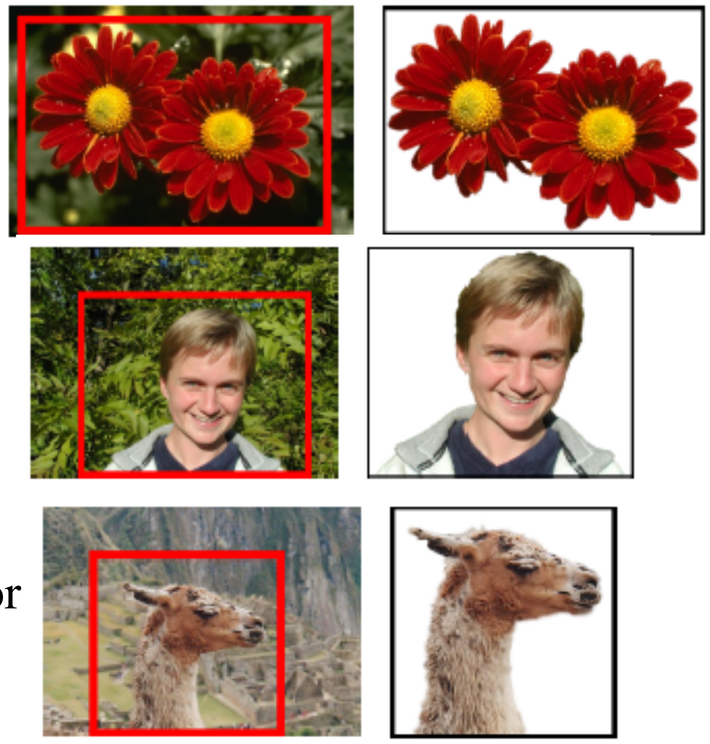
- At each pixel location i, j place a discrete label l_{ij}
- Construct a cost function
  - with unary (single label) terms $\phi_{ij}(l_{ij}, \mathrm{Data})$
  - and binary (label pairs) terms $\psi_{ij,uv}(l_{ij}, l_{uv}, \mathrm{Data})$
- Choose labels to minimize this cost
  - rich algorithmic tradition

Quick and dirty review of a topic that could occupy a course!

Example application of CRF: Grab Cut

- Originally for matting
  - extracting an object from an image
- Process
  - user places box
    - grabcut segments intended object
  - user perhaps iterates with strokes, etc.

- For us:
  - segments using graph cuts
    - clever iterative model of interior/exterior
  - extremely simple shape prior on object



GrabCut mapping

- Labels are either 0 (background) or 1 (foreground)
- Write
  - S for some function measuring similarity (smaller is better!)
  - p_ij for ij’th pixel value

$$\phi_{ij}(l_{ij}, \mathrm{Data}) = l_{ij} S(p_{ij}, \mathrm{fg}) + (1 - l_{ij}) S(p_{ij}, \mathrm{bg})$$

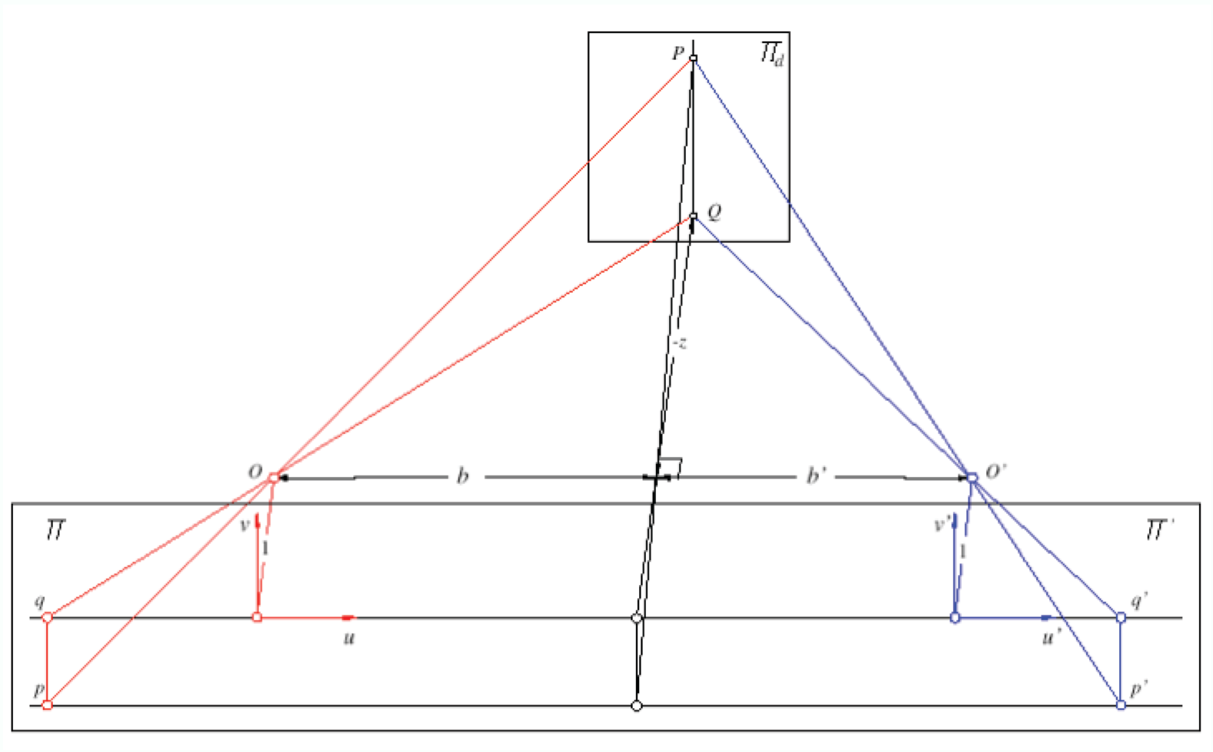
- We want the label of a pixel to agree with its neighbor

$$\psi_{ij,uv}(l_{ij}, l_{uv}, \mathrm{Data}) = \begin{cases} k \left[\begin{array}{c} l_{ij}(1 - l_{uv}) + \\ l_{uv}(1 - l_{ij}) \end{array} \right] & \text{if uv is neighbor o} \\ 0 & \text{otherwise} \end{cases}$$

GrabCut solving

- This is a discrete optimization problem
  - choose 0, 1 at labels
    - best case is when labels agree with foreground/background appearance
    - and with neighbors
- For this case, with $k>0$, solution is polynomial
  - well studied, graph cut
- Procedure
  - use pixels near center of box to build model of foreground
  - use pixels outside box to build model of background
  - choose k; solve for l_ij
  - you now have segmentation.

Stereo as a CRF



- Original:
  - find q, q’ that match, and infer depth
- Now:
  - choose value of depth at q; then quality of match at q’ is cost
  - optimize this

Stereo as a CRF

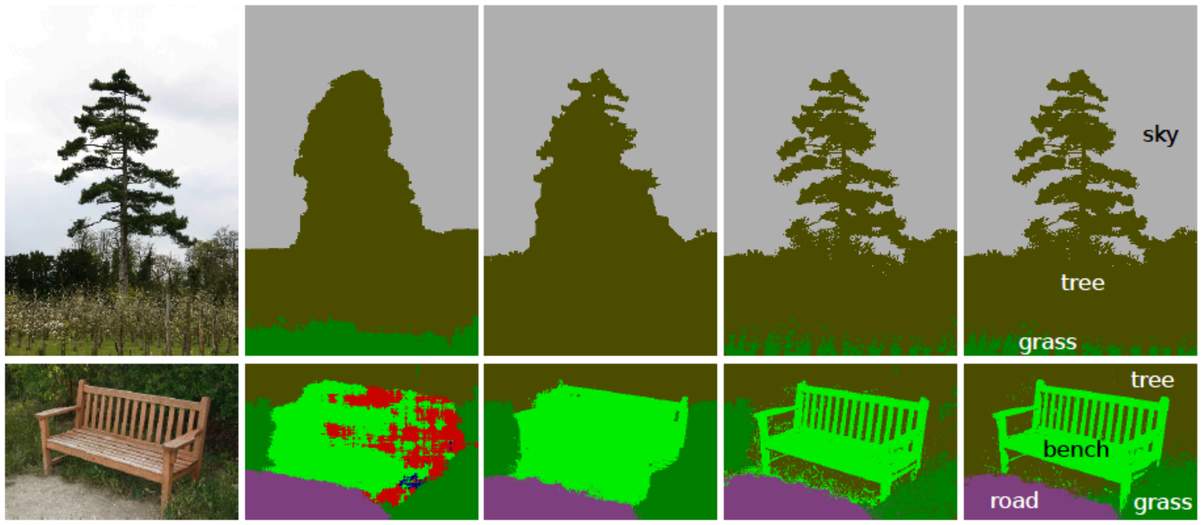
- Typically:
  - quantize depth to a fixed number of levels
    - label at pixel can be one of d depths
  - unary cost is color match
    - (photometric consistency constraint)
    - it can be helpful to match intensity gradients, too
  - pairwise cost from smoothness constraint on recovered depths
    - eg depth gradient not too big, etc.
  - massive discrete quadratic program

Semantic Segmentation as a CRF

- But now we have it for segmentation as well
  - one label per segment
  - costs:
    - per pixel:
      - how well does this label/pixel value go together
      - as in grabcut above
    - per pair:
      - how well does this label/pixel pair work together
      - usually, a form of smoothness
        - agree with your neighbors

Why go to all this trouble?

- Can impose some kinds of spatial prior
  - labels mostly agree with their labels
  - long scale agreement between labels



(a) Image (b) Unary classifiers (c) Robust P^n CRF (d) Fully connected CRF, MCMC inference, 36 hrs (e) Fully connected CRF, our approach, 0.2 seconds

Figure 1: Pixel-level classification with a fully connected CRF. (a) Input image from the MSRC-21 dataset. (b) The response of unary classifiers used by our models. (c) Classification produced by the Robust P^n CRF [9]. (d) Classification produced by MCMC inference [17] in a fully connected pixel-level CRF model; the algorithm was run for 36 hours and only partially converged for the bottom image. (e) Classification produced by our inference algorithm in the fully connected model in 0.2 seconds.

Why go to all this trouble?

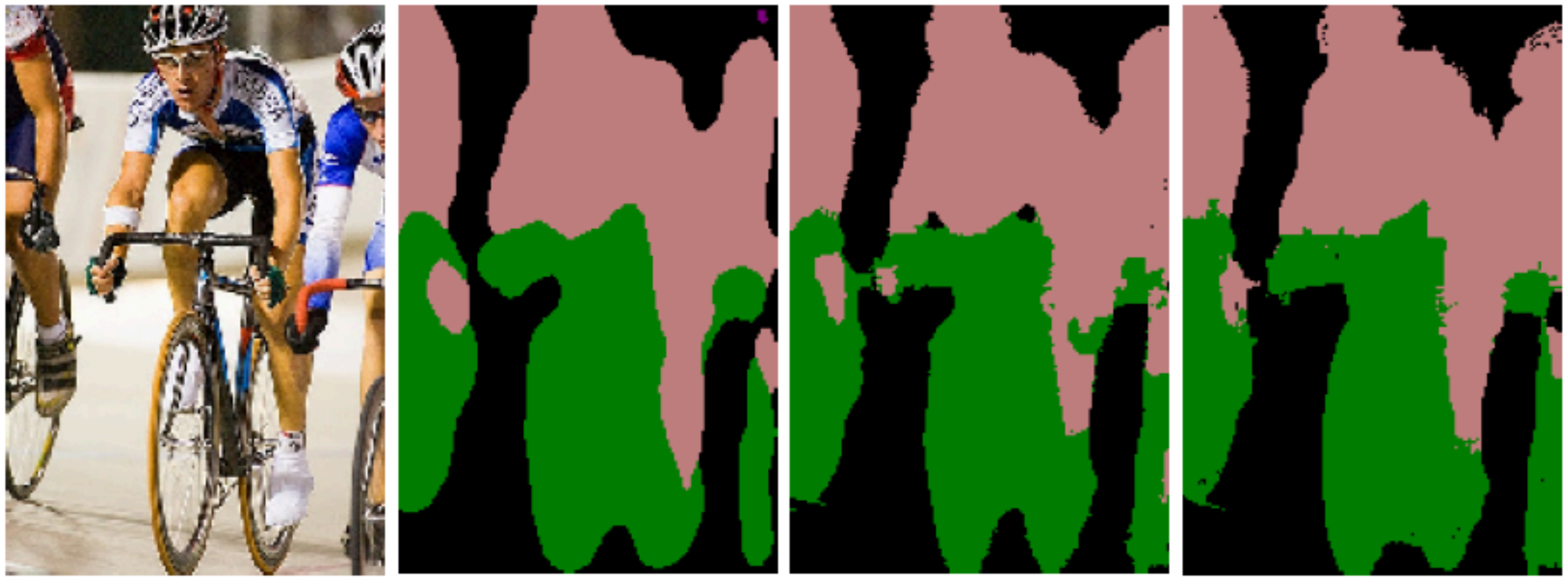


Fig. 5: Using the DeepLab CNN-based semantic segmentation algorithm [6] (5b) as input our bilateral solver can produce comparable edge-aware output (5d) to the DenseCRF [22] used in [6] (5c), while being 8× faster.

Solving CRF’s

- Solving is well understood for many cases
- Minimize:
  - x^T A x + b^t x
  - subject to: x is a vector of discrete values
- Immense, very active literature
  - settled down a bit over the last 10 years, but…
- Special case
  - Every pixel is connected to every other pixel
    - with weights
  - Yields a fast variational algorithm
    - based in non-local means

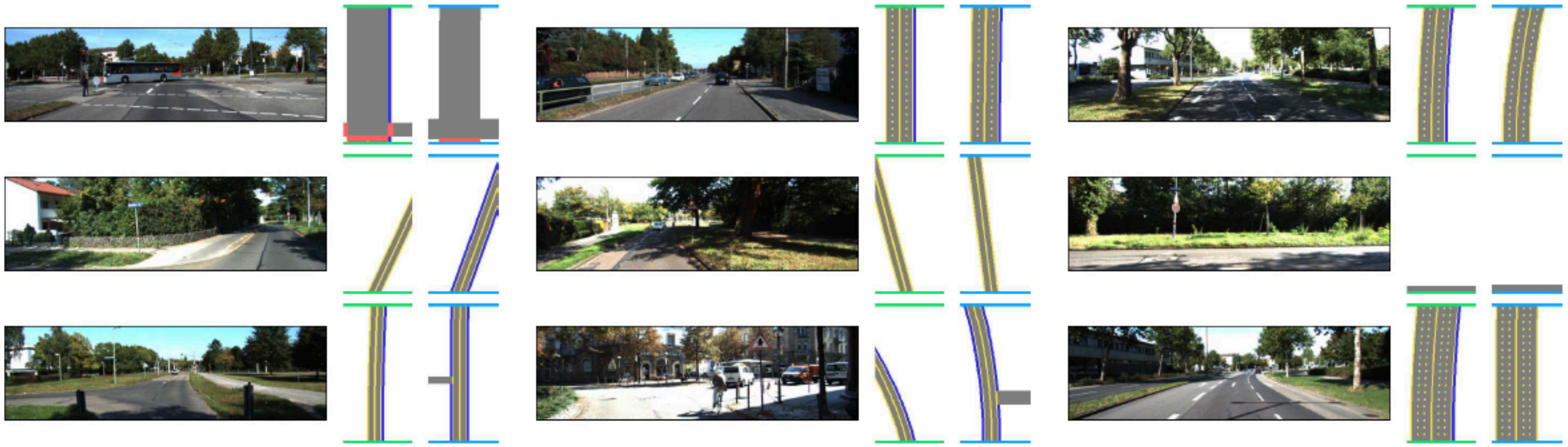


Figure 6: Qualitative results of H-BEV+DA+GM on individual frames from KITTI. Each example shows perspective RGB, ground truth and predicted semantic top-view, respectively. Our representation is rich enough to cover various road layouts and handles complex scenarios, *e.g.*, rotation, existence of crosswalks, sidewalks, side-roads and curved roads.

Wang 19

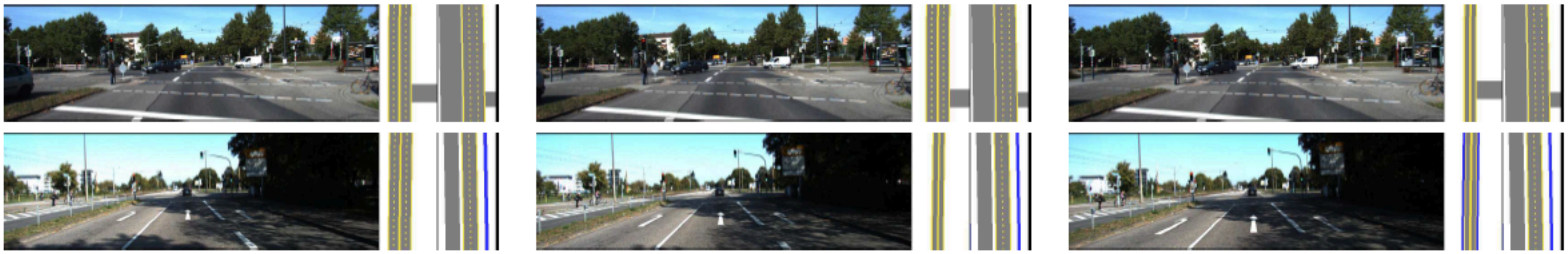


Figure 7: Qualitative results comparing H-BEV+DA and H-BEV+DA+GM in consecutive frames of two example sequences of the KITTI validation set. In each column, we have visualized the perspective RGB image, prediction from H-BEV+DA and that of H-BEV+DA+GM from left to right. Each row shows a sequence of three frames. We can observe more consistent predictions, *e.g.*, width of side-road and delimiter width, with the help of the temporal CRF.

Wang 19

Good + bad

- It’s clear that label fields are highly structured
  - but BEV construction is weird
- This structure is very important and valuable
  - Q: can we exploit without OSM, Streetview, etc.?