Improved monocular depth prediction using distance transform over pre-semantic contours with self-supervised neural networks

Supplementary Material

The supplementary material includes multiple details and insights that complement the main paper.

A. Theoretical Analysis

In this section, we elaborate on the mathematical formalism of the distance transform. We also give more details on the toy experiments that corroborate the theoretical properties.

A.1. Maximal variance under constraints

Let us give a proof of Theorem 1. As a reminder:

Theorem 1 *The distance transform is the unique solution, up to an isomorphism, of the following optimization problem:*

$$\max_{f \in \mathcal{I}_\mathcal{T}} \int_\Omega \left(f(x) - \overline{f}\right)^2 dx \quad \textit{s.t.} \quad \|\nabla f\| = 1 \tag{A.1}$$

Proof:
We only provide a sketch of the proof to keep things simple. Level sets of a function f are the $\{f^{-1}(v)\}$ for each specific value v. For example, a function defined as the distance to a fixed point has circular level sets (Figure A.1, right). In contrast, in the case of the distance transform, the level sets conform to the shape. Let us decompose the shape Ω into a countably infinite number of level set slices. Then the level set lengths form a decreasing series, $\{f_n\}_{n\in\mathbb{N}}$, as shown in Figure A.1 left, with:

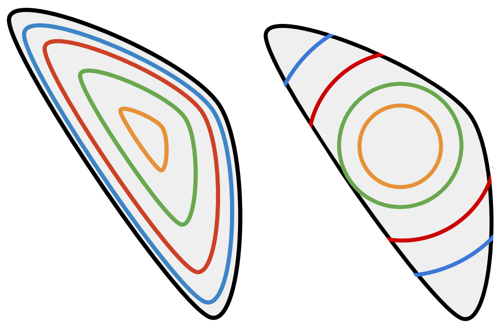
$$f_n \xrightarrow{n\to\infty} 0 \tag{A.2}$$

This unique property of the level sets in the distance transform case enables the existence of an isomorphism Φ that can be defined as an increasing function of the inverse of f_n

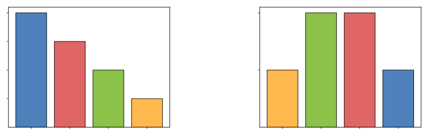
$$\begin{gathered}\Phi(n) \sim h\left(\frac{1}{f_n}\right) \quad \text{such that} \\ \sum_{n=1}^{\infty} \phi(n) f_n < \infty, \\ \sum_{n=1}^{\infty} \phi(n)^2 f_n < \infty \end{gathered} \tag{A.3}$$

where h is an increasing function.

In this way the mean value as well as the variance can be increased drastically. This is not possible for circular level sets, where the histogram is not monotonic as illustrated in Figure A.1.



Level sets of distance transform (left) and distance to the centre (right)



Histogram of distance transform values (left) and distance to centre (right)

Figure A.1. Illustration of the histogram of the number of pixels for each value for different level set.

A.2. Convergence properties

Let us give a proof of theorem 2. As a reminder:

Theorem 2 *The loss function l (See Equation 6) is, with respect to the first argument $\hat{y}$:*
- *α-Lipschitz*
- *β-smooth*
- *strictly convex if a regularization term $\eta\|y\|^2$ is added*

Proof:
Let y be the ground truth pixel as explained in Section 3.2 and let g be the function as defined in Equation 5, with g' and g'' bounded respectively by K_1 and K_2.
Lipschitz: We want to prove that:

$$\exists \alpha \text{ s.t } \forall u, v \in \Omega : |l(u) - l(v)| \leq \alpha \|u - v\| \tag{A.4}$$

$$\begin{aligned} &|l(u) - l(v)| \\ &= \left\| \left(g \circ \delta_\Omega(u) - g \circ \delta_\Omega(y)\right)^2 - \left(g \circ \delta_\Omega(v) - g \circ \delta_\Omega(y)\right)^2 \right\| \\ &= \Big\| \left(g \circ \delta_\Omega(u) - g \circ \delta_\Omega(v)\right) \\ &\quad \underbrace{\left(g \circ \delta_\Omega(u) + g \circ \delta_\Omega(v) - 2g \circ \delta_\Omega(y)\right)}_{\leq 4 \text{ as } \|g\| \leq 1} \Big\| \end{aligned} \tag{A.5}$$

Then,

$$\begin{aligned}|l(u) - l(v)| &\leq 4\|g \circ \delta_\Omega(u) - g \circ \delta_\Omega(v)\| \\ |l(u) - l(v)| &\leq 4\|g'\|_\infty \|\delta\|_\infty \|u - v\|\end{aligned} \tag{A.6}$$

We have $\|\delta\|_\infty \leq 1$ by the Eikonal equation in Definition 2. Then:

$$|l(u) - l(v)| \leq \underbrace{4K_1}_{=\alpha} \|u - v\| \tag{A.7}$$

Smooth: We want to prove that:

$$\exists \beta \text{ s.t } \forall u, v \in \Omega, \|\nabla l(u) - \nabla l(v)\| \leq \beta \|u - v\| \tag{A.8}$$

Computing the gradient of l gives:

$$\nabla l(u) = 2\big(g \circ \delta_\Omega(u) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(u) g'\big(\delta_\Omega(u)\big) \tag{A.9}$$

Then we introduce intermediate values that cancel each other in the gradient differences to be able to compute a bound:

$$\begin{aligned}\|\nabla l(u) - \nabla l(v)\| = \\ \Big\| 2\big(g \circ \delta_\Omega(u) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(u) g'\big(\delta_\Omega(u)\big) \\ -2\big(g \circ \delta_\Omega(v) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(u) g'\big(\delta_\Omega(u)\big) \\ +2\big(g \circ \delta_\Omega(v) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(u) g'\big(\delta_\Omega(u)\big) \\ -2\big(g \circ \delta_\Omega(v) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(v) g'\big(\delta_\Omega(u)\big) \\ +2\big(g \circ \delta_\Omega(v) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(v) g'\big(\delta_\Omega(u)\big) \\ -2\big(g \circ \delta_\Omega(v) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(v) g'\big(\delta_\Omega(v)\big) \Big\|\end{aligned} \tag{A.10}$$

We have, for each part of the sum:

$$\begin{aligned}\Big\| 2\big(g \circ \delta_\Omega(u) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(u) g'\big(\delta_\Omega(u)\big) \\ -2\big(g \circ \delta_\Omega(v) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(u) g'\big(\delta_\Omega(u)\big) \Big\| \\ \leq 2\alpha K_1 \|u - v\| \\ \Big\| 2\big(g \circ \delta_\Omega(v) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(u) g'\big(\delta_\Omega(u)\big) \\ -2\big(g \circ \delta_\Omega(v) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(v) g'\big(\delta_\Omega(u)\big) \Big\| \\ \leq 4K_1 \|\nabla \delta_\Omega(u) - \nabla \delta_\Omega(v)\| \\ \Big\| 2\big(g \circ \delta_\Omega(v) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(v) g'\big(\delta_\Omega(u)\big) \\ -2\big(g \circ \delta_\Omega(v) - g \circ \delta_\Omega(y)\big)\nabla \delta_\Omega(v) g'\big(\delta_\Omega(u)\big) \Big\| \\ \leq 4K_2 \|u - v\|\end{aligned} \tag{A.11}$$

In order to bound the second term, let $x = \Pi(u)$ be the orthogonal projection of u on Ω. And let $\kappa(x)$ be the mean curvature at x. Note that Ω being convex, we have $0 \leq \kappa(x)$. Then the Hessian of δ_Ω at u is diagonalizable in the surface tangent and normal orthonormal basis :

$$H_{\delta_\Omega}(u) = \begin{pmatrix} 0 & 0 \\ 0 & \frac{\kappa(x)}{1+\delta_\Omega(u)\kappa(x)} \end{pmatrix} \tag{A.12}$$

Let us define the maximal curvature:

$$K_3 = \max_{x \in \partial\Omega} \kappa(x) \tag{A.13}$$

Then,

$$\frac{\kappa(x)}{1 + \delta_\Omega(u)\kappa(x)} \leq K_3 \tag{A.14}$$

We can therefore bound the second term:

$$\|\nabla \delta_\Omega(u) - \nabla \delta_\Omega(v)\| \leq K_3 \|u - v|\| \tag{A.15}$$

and then bound the gradient different:

$$\|\nabla l(u) - \nabla l(v)\| \leq \underbrace{(2\alpha K_1 + 4K_2 + 4K_1K_3)}_{=\beta} \|u - v\| \tag{A.16}$$

Strictly convex: We want to prove that the Hessian of l is definite positive

$$\nabla l(u) = 2\underbrace{\big(g \circ \delta_\Omega(u) - g \circ \delta_\Omega(y)\big)}_{=\epsilon} \nabla \delta_\Omega(u) g'\big(\delta_\Omega(u)\big) \tag{A.17}$$

and

$$\begin{aligned}H_l(u) = 2g'(\delta_\Omega(u))^2 \nabla \delta_\Omega^T \nabla \delta_\Omega \\ +2\epsilon g''(\delta_\Omega(u)) \nabla \delta_\Omega^T \nabla \delta_\Omega \\ +2\epsilon g'(\delta_\Omega(u)) H_{\delta_\Omega}(u)\end{aligned} \tag{A.18}$$

In the surface tangent, normal orthonormal basis, it can be written as follows:

$$H_l(u) = \begin{pmatrix} \underbrace{2g'(\delta_\Omega(u))^2 + 2\epsilon g''(\delta_\Omega(u))}_{\lambda_1} & 0 \\ 0 & \underbrace{2\epsilon g'(\delta_\Omega(u))}_{\lambda_2} \end{pmatrix} \tag{A.19}$$

If we add a regularization term, then it becomes:

$$H_l(u) = \begin{pmatrix} 2\eta + \lambda_1 & 0 \\ 0 & 2\eta + \lambda_2 \end{pmatrix} \tag{A.20}$$

If the prediction u and the ground truth y are close enough, then ϵ is low enough and:

$$\begin{aligned}0 < 2\eta + 2g'(\delta_\Omega(u))^2 + \underbrace{2\epsilon g''(\delta_\Omega(u))}_{\sim 0} \\ 0 < 2\eta + \underbrace{2\epsilon g'(\delta_\Omega(u))}_{\sim 0}\end{aligned} \tag{A.21}$$

A.3. A toy experiment for maximal variance

Let us now show that the maximal variance property mentioned in 3.1 can be retrieved with a simple experiment.
We built a large dataset comprising 10 000 polygons I of shape 64×64 (see examples on Figure A.2). Then we trained a U-Net architecture Θ detailed in Figure A.3 and Table A.1 [62] with self-attention [71] to maximize the variance in polygon areas while respecting translation, flip and rotation invariance. We also promoted low Laplacian values to satisfy the Eikonal equation and avoid clustering patterns as mentioned previously. Training was conducted for 50 epochs, with a learning rate of $lr = 0.01$ and a batch size of 16, optimizing the following loss function:

$$\begin{aligned} \mathcal{L} = \lambda_{\text{repro}}\mathcal{L}_{\text{repro}} + \lambda_{\text{var}}\mathcal{L}_{\text{var}} \\ +\lambda_{\text{laplacian}}\mathcal{L}_{\text{laplacian}} + \lambda_{\text{control}}\mathcal{L}_{\text{control}} \end{aligned} \tag{A.22}$$

with

$$\mathcal{L}_{\text{repro}} = \|\Theta(I) - T^{-1}\big(\Theta\,(T(I))\,\big)\| \tag{A.23}$$

Here T is chosen to be either a translation, rotation, or flip. The objective of this loss is to best satisfy the constancy assumption.

$$\begin{aligned} \mathcal{L}_{\text{var}} &= \frac{\text{mean}(I)}{\text{var}(I)} \\ \mathcal{L}_{\text{laplacian}} &= \|\Delta I\|_1 \\ \mathcal{L}_{\text{control}} &= \|I\|_1 \end{aligned} \tag{A.24}$$

Here $\mathcal{L}_{\text{var}}$ is used to promote variance within the polygon. A normalization by the mean is applied to prevent large values. $\mathcal{L}_{\text{laplacian}}$ helps to approximate the Eikonal equation. Finally, we found that introducing $\mathcal{L}_{\text{control}}$ improves training stability..
We set the following values: $\lambda_{\text{repro}} = 1$, $\lambda_{\text{var}} = 1$, $\lambda_{\text{laplacian}} = 0.1$, and $\lambda_{\text{control}} = 0.01$. Our experiment shows that Θ indeed converges towards the distance transform solution as seen on Figure A.2.

A.4. A toy experiment for convergence

Let us now show that the improved convergence property mentioned in 3.2 can be retrieved with a simple experiment.
We built $n = 10\,000$ pairs of rectangles $\{(R_0^i, R_1^i)\}_{i\in\{1,\dots,n\}}$ each of shape 500×500 with R_1^i being R_0^i after a random rigid translation $\Delta_i = (u_i, v_i)$. We trained a multi-layer perceptron detailed in Table A.2 to take as input the pair of rectangle images and to predict the shift Δ. Training was done for 20 epochs, using a learning rate of 0.01 and a batch size of 32. We discarded any random translation that caused parts of the rectangle to fall outside the boundaries. We considered two scenarios: first, rectangles are only filled with a white colour. Second, rectangles are filled with the distance transform values. We re-iterated this procedure with stars instead of rectangles. Some images of the dataset are displayed on Figure A.5. Figure A.4 show that the convergence on the training is much faster with the distance transform.

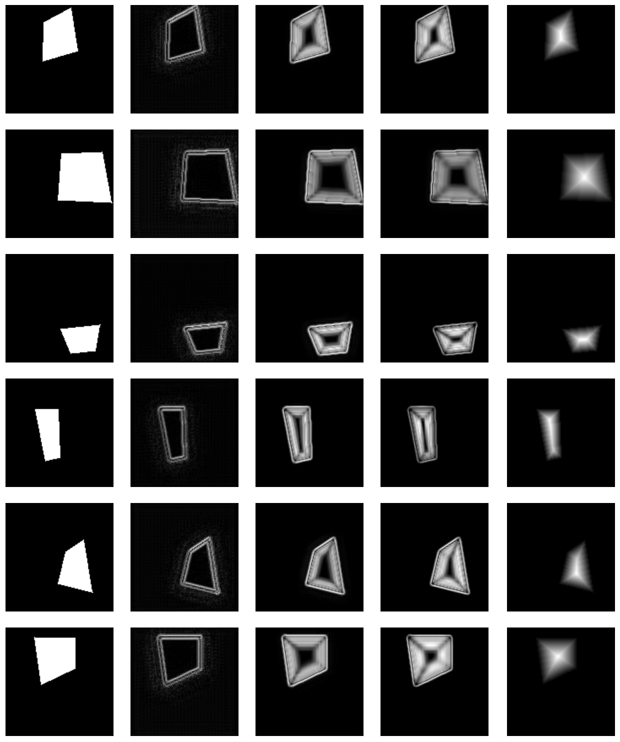


Figure A.2. Results of the U-Net training. From left to right: (i) Input shape, (ii) Gradient norm of the U-Net output after 1 epoch, (iii) after 5 epochs, (iv) after 10 epochs, (v) Distance transform. We show the gradient norm of the output for better visualization of the convergence process. The gradient of the resulting image is uniform except on the medial axis (a.k.a. skeleton), thus corresponding to the distance transform.

B. Extra Results

B.1. Depth

We give extra quantitative results of our method in Table B.1. As expected, when we increase the input resolution from 256×832 to 320×1024 and apply the online refinement procedure of [4], we improve even more the metrics.

We also provide more depth images in Figure F.1. In general, our method renders very sharp depth images, which is the sign that our pipeline indeed reduces the ill-posed nature of the optimization problem.

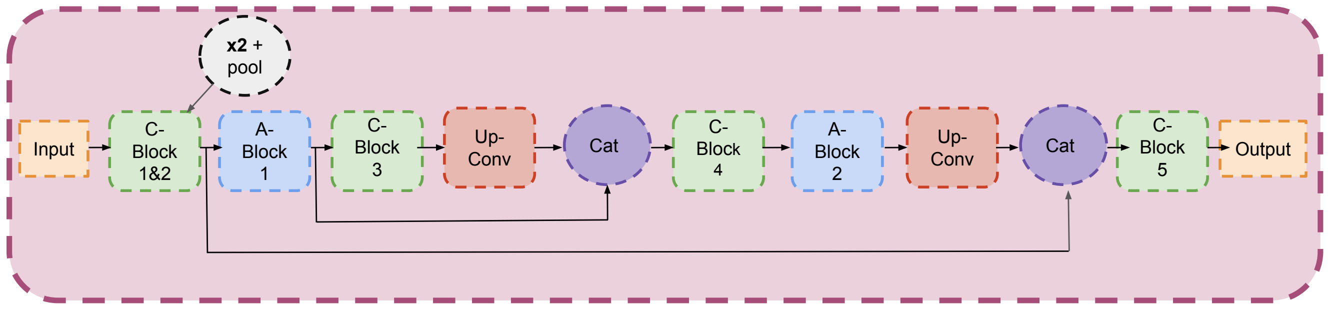


Figure A.3. U-Net architecture used for our toy experiment on the maximal variance. Details of C-Blocks and A-Blocks given in Table A.1.

C-Block(in, out, dilation)
Conv2D(in_ch=in, out_ch=out, kernel_size=3, dilation=dilation, padding=dilation), BatchNorm2d(out_ch=out), ReLU, Conv2D(in_ch=out, out_ch=out, kernel_size=3, dilation=dilation, padding=dilation), BatchNorm2d(out_ch=out), ReLU
A-Block(in)
f=Conv2D(in_ch=in, out_ch=in / 8, kernel_size=1) g=Conv2D(in_ch=in, out_ch=in / 8, kernel_size=1) h=Conv2D(in_ch=in, out_ch=in, kernel_size=1) s=Softmax(dim=-1)
Blocks
C-Block 1 = C-Block(1, 32, 1) C-Block 2 = C-Block(32, 64, 2) C-Block 3 = C-Block(64, 128, 1) C-Block 4 = C-Block(128, 64, 2) C-Block 5 = C-Block(64, 32, 1) + Conv2D(in_ch=32, out_ch=1, kernel_size=1)
A-Block 1 = A-Block(64) A-Block 2 = A-Block(64)

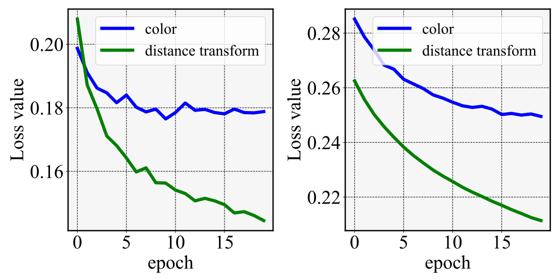
Table A.1. Details of blocks used for our UNet architecture, see diagram of our architecture on Figure A.3.

Multi-layer perceptron
Linear(in_features=2 $\times$ 500 $\times$ 500, out_features=128)
Linear(in_features=128, out_features=64)
Linear(in_features=64, out_features=2)

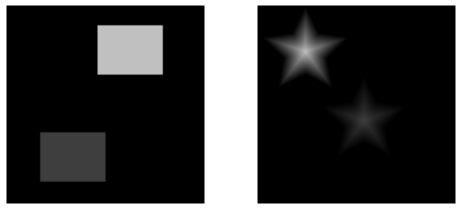
Table A.2. Neural Network Layer Definitions for SelfAttention Class

B.2. Flow

Our method uses the strategy of [28] to remove moving pixels from the computation of the photometric loss. It takes advantage of a self-supervised optical flow network. Quantitative results are given in Table B.2. We can see that our improved framework also improves flow metrics. The flow is also trained using the variance augmented image. To assess optical flow we use the KITTI 2015 flow dataset containing 200 annotated training images as test images.



Training loss for translation prediction for rectangles (left) and stars (right)



Translation of two uniform rectangles (left) and stars with distance transform (right)

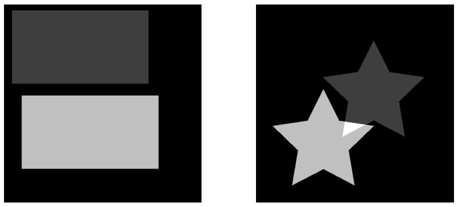
Figure A.4. Convergence study for a translation prediction model with and without the distance transform.

Method	*Lower is better* ↓				*Higher is better* ↑		
	Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
Ours HR(320×1024)	0.101	0.703	4.422	0.176	0.895	0.963	0.984
Ours HR*	0.083	0.655	4.11	0.166	0.916	0.965	0.984

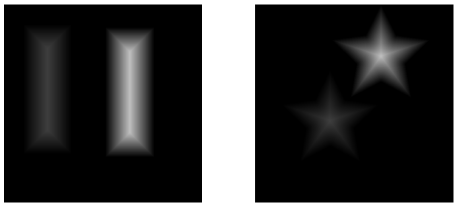
Table B.1. **Results of depth estimations on KITTI 2015**. *: Using the online refinement technique of [4].

B.3. Odometry

We give results for odometry in Table B.3. To assess odometry we use Sequence 9 and 10 of the KITTI Odometry



Colour dataset with rectangles (left) and stars (right). Lighter shapes are before the random translation.



Distance transform dataset with rectangles (left) and stars (right).

Figure A.5. Example of images used in the experiment to analyse convergence properties.

Method	Noc	All
FlowNetS[17]	8.12	14.19
FlowNet2[33]	4.93	10.06
GeoNet[86]	8.05	10.81
GLNet[7]	4.86	8.35
CoopNet[28]	5.10	9.43
Ours	**4.59**	**7.82**

Table B.2. **Optical Flow:** Average end point error (in pixels) for non occluded (Noc) and for all (All) pixels on the KITTI 2015 flow dataset.

dataset.

C. Discussion on the distance transform

C.1. Algorithm

The distance transform in the case of 8-neighbours(d_8) is shown on Algorithm 1.

C.2. Functions of the type $g(d)$

Different functions of the distance transform are illustrated in Figure F.2, while the corresponding quantitative results for depth are shown on Table C.1.

C.3. Random Walk

Here is the algorithm of the random walk, provided here in the 2D case for simplicity.
Once the random walk is performed, the mapping is done as follows:

$$\mathrm{RW}\,(i) = X_i \tag{C.1}$$

Algorithm 1 Distance Transform d_8

```
Require: $I$ image of size $H \times W$, $\mathcal{C}$ binary contour of $I$
 1: Initialize $F(i,j) = \infty$ for all $(i,j) \in I$
 2: Forward pass:
 3: for $i = 1$ to $H$ do
 4:     for $j = 1$ to $W$ do
 5:         if $\mathcal{C}\,(i,j) = 1$ then
 6:             $F(i,j) \leftarrow 0$   ▷ Set to 0 if contour is present
 7:         else
 8:             $F(i,j) \leftarrow \min\big(F(i-1,j-1), F(i-1,j), F(i-1,j+1), F(i,j-1)\big) + 1$   ▷ Update distance using neighbours
 9:         end if
10:     end for
11: end for
12: Backward pass:
13: for $i = 1$ to $H-1$ do
14:     for $j = 1$ to $W-1$ do
15:         if $\mathcal{C}\,(i,j) = 1$ then
16:             $F(i,j) \leftarrow \min\big(F(i,j), F(i+1,j+1)+1, F(i+1,j)+1, F(i+1,j-1)+1, F(i,j+1)+1\big)$   ▷ Update distance using neighbours
17:         end if
18:     end for
19: end for
```

Algorithm 2 Random Walk

```
Require: $N$ number of steps, $\varepsilon$ size of one step, $\{\theta_i\}_{i=1}^{k}$ is
    a discrete partition of the unit disk in $k$ angles.
 1: Initialize $X_0 = (0,0)$
 2: for $i = 1$ to $N$ do
 3:     $\theta = \text{random}\,(\theta_1, ..., \theta_k)$   ▷ choose a random angle
 4:     $X_i = X_{i-1} + \varepsilon e^{i\theta}$
 5: end for
```

One example of a random walk function is shown on Figure F.2. It was performed using $\varepsilon = 0.01$ and a partition number $k = 1000$ for polar and azimuthal angles . Quantitative results for different dimensions of random walks are shown on Table C.1.

D. Discussion on the contour

D.1. Complementarity Depth - Normal

Depth contours and normal to surface contours target different parts of the contour. The depth is better at localizing edge resulting from occlusions, i.e. between foreground instances and background, where a large distance gradient is expected. In contrast, the normal is better at locating edges related to sharp angle changes, as shown on Figure D.1.

Methods	Seq. 09		Seq. 10	
	t_{err} (%)	r_{err} ($°/100m$)	t_{err} (%)	r_{err} ($°/100m$)
ORB-SLAM[51]	15.30	0.26	3.68	0.48
Zhou et al.[91]	17.84	6.78	37.91	17.78
Bian et al.[3]	11.2	3.35	10.1	4.96
CoopNet[28]	8.42	2.66	7.29	**2.14**
Ours	**8.39**	**2.31**	**7.17**	2.81

Table B.3. **Odometry**: Average Translation and Rotation errors for sequence 09 and 10 of the KITTI Odometry Dataset. Bold indicate the best learning based method.

Functions	Abs Rel (↓)	RMSE log (↓)	δ_1 (↑)
$x \mapsto x$	0.106	0.183	0.884
$x \mapsto \sin(\pi x)$	0.105	0.182	0.884
$x \mapsto 4x(1-x)$	0.107	0.185	0.882
RW$_2$	0.105	0.181	0.885
RW$_3$	**0.104**	**0.180**	**0.885**
RW$_4$	0.107	0.185	0.880

Table C.1. Results on KITTI 2015 for various functions as defined in Section 3.1, and different RW encoding of the distance transform.

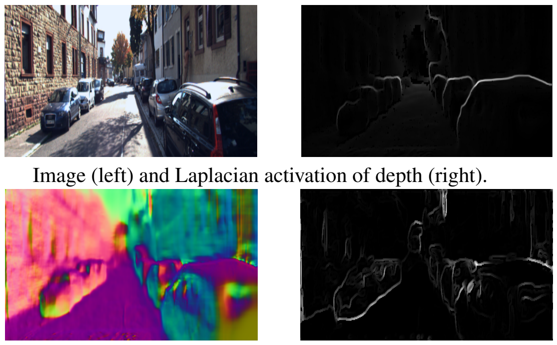


Image (left) and Laplacian activation of depth (right).
Normal (left) and gradient activation of normal (right).

Figure D.1. Illustration of the complementary of depth and normals.

D.2. Comparison to Lego

In the Lego method [84], instead of considering zero-crossing to learn the edges, the authors consider the whole positive part of the second-derivative. This creates coarse edges that are not aligned on the true semantic borders but either towards the exterior or the interior of the instances. Besides, the loss derived from the depth includes a normalization term that poorly addresses the bias of large distances. Our proposed procedure solves all these issues. We give a comparison of qualitative results in Figure F.3. We observe that our contour estimations perform better at large distances and with orientation changes. Additionally, the predictions are more tightly aligned with the objects, which is crucial for the effectiveness of our framework.

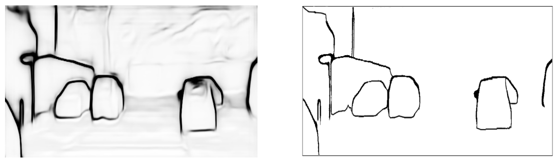


Figure D.2. Output of the edge neural network (left) and contour estimation after post-processing (right)

D.3. Post-processing

For post-processing, we first apply hysteresis thresholding to the output of the edge network, using a low threshold of 80 and a high threshold of 100 to obtain edge$_h$.
Next, we perform non-maximum suppression along the gradient direction on the output of the edge network to obtain edge$_n$. Finally we compute:

$$\text{edge}_{\text{binary}} = \begin{cases} 1 & \text{if } 0 < \text{edge}_h \times \text{edge}_n, \\ 0 & \text{otherwise.} \end{cases} \tag{D.1}$$

The final estimated pre-semantic contour, $\widehat{\mathcal{C}}(I)$, is derived from edge$_{\text{binary}}$ by applying morphological transformations using OpenCV's "morphologyEx" to fill holes, performing a contour closing procedure, and filtering out small, isolated contours. A post-processing result is shown on Figure D.2.

E. Discussion on the constancy assumption

In this experiment, we focus on the constancy assumption, i.e., the fact that any change introduced in image t should be reproduced identically in image $t+1$ We aim to evaluate the validity of the constancy assumption for our distance transform map and compare it to the deep features of a ResNet-18 pre-trained on ImageNet. To achieve this, we use the KITTI MOTS dataset [72], which provides several sequences of images with ground-truth instances, as shown in Figure E.1. We considered a mask with a radius of 3 pixels around the center of the object tracked across the sequence, as illustrated in Figure F.4.

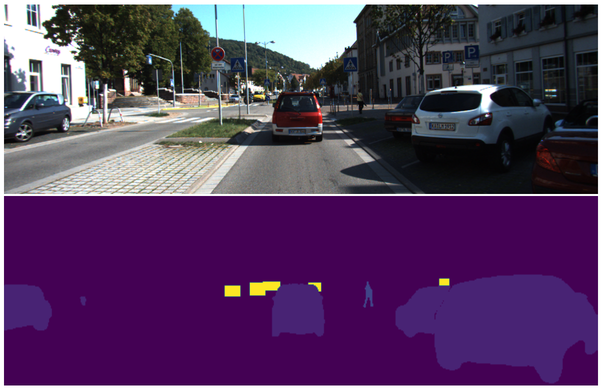


Figure E.1. One Image of sequence 11 of dataset KITI MOTS [72] (above) and the associated ground-truth instance segmentation map (below).

Sequences	Object Indexes
4	1002
5	1031
8	1008
10	1000
11	1000
18	1003
20	1012

Table E.1. Sequences of KITTI MOT, with the corresponding object indexes, used for our experiment.

After tracking the position of the mask along the sequence, we computed the variance over time, normalized by the mean over time, for both our distance transform map and the deep feature maps. For the deep features, we considered the output of the four-layer blocks in the encoding part (referred to as layer1, layer2, layer3, and layer4 in PyTorch). We conducted these experiments on 7 sequences, summarized in Table E.1, along with their corresponding object IDs. Figure 4 illustrates the improved constancy achieved when using the distance transform.

F. Training hyper-parameters

F.1. ResNet50 encoder

As a reminder, the loss used to train the edge network is:

$$\mathcal{L}^{\text{edge}} = \sum_p \left(\lambda_d w_d(p) + \lambda_n w_n(p)\right)(1 - E(p)) + \lambda_c \mathcal{L}_c + \lambda_e \mathcal{L}_e$$

with $\mathcal{L}_e = \sum_p E(p)^2$ to avoid trivial solutions.

The loss weights are set as follows: $\lambda_d = 0.5$, $\lambda_n = 0.25$, $\lambda_e = 1.0$ and $\lambda_c = 0.001$. Contrastive loss $\mathcal{L}_c$ is only introduced after 15 epochs.

The loss used to train the depth network is:

$$\mathcal{L}^{\text{depth}} = \lambda_{\text{dist}}\mathcal{L}_{\text{dist}} + \lambda_{\text{photo}}\mathcal{L}_{\text{photo}} + \lambda_s \mathcal{L}_s$$

The weights are set as follows: $\lambda_{\text{photo}} = 1$, $\lambda_{\text{dist}} = 1$ and $\lambda_s = 0.001$. Normal smoothing is also added to the total loss to better predict edges with $\lambda_{ns} = 0.01$:

$$\mathcal{L}_{ns} = \sum_p \left(|\nabla N_x(p)| e^{-|\nabla I_x(p)|} + |\nabla N_y(p)| e^{-|\nabla I_y(p)|}\right)$$

F.2. Dino encoder

We also considered a DinoV2 [56] encoder following the implementation provided by Facebook Research. We followed the implementation of [82] for the depth head. We considered the large ViT ”dinov2_vitl14” with 304M parameters. For this experiment a OneCycle learning rate was chosen for the depth network with $lr = 2.10^{-6}$ for the encoder part and $lr = 10^{-5}$ for the decoder part and a weight decay of 0.01. Other network training parameters and hyper-parameters remained unchanged from the original implementation.

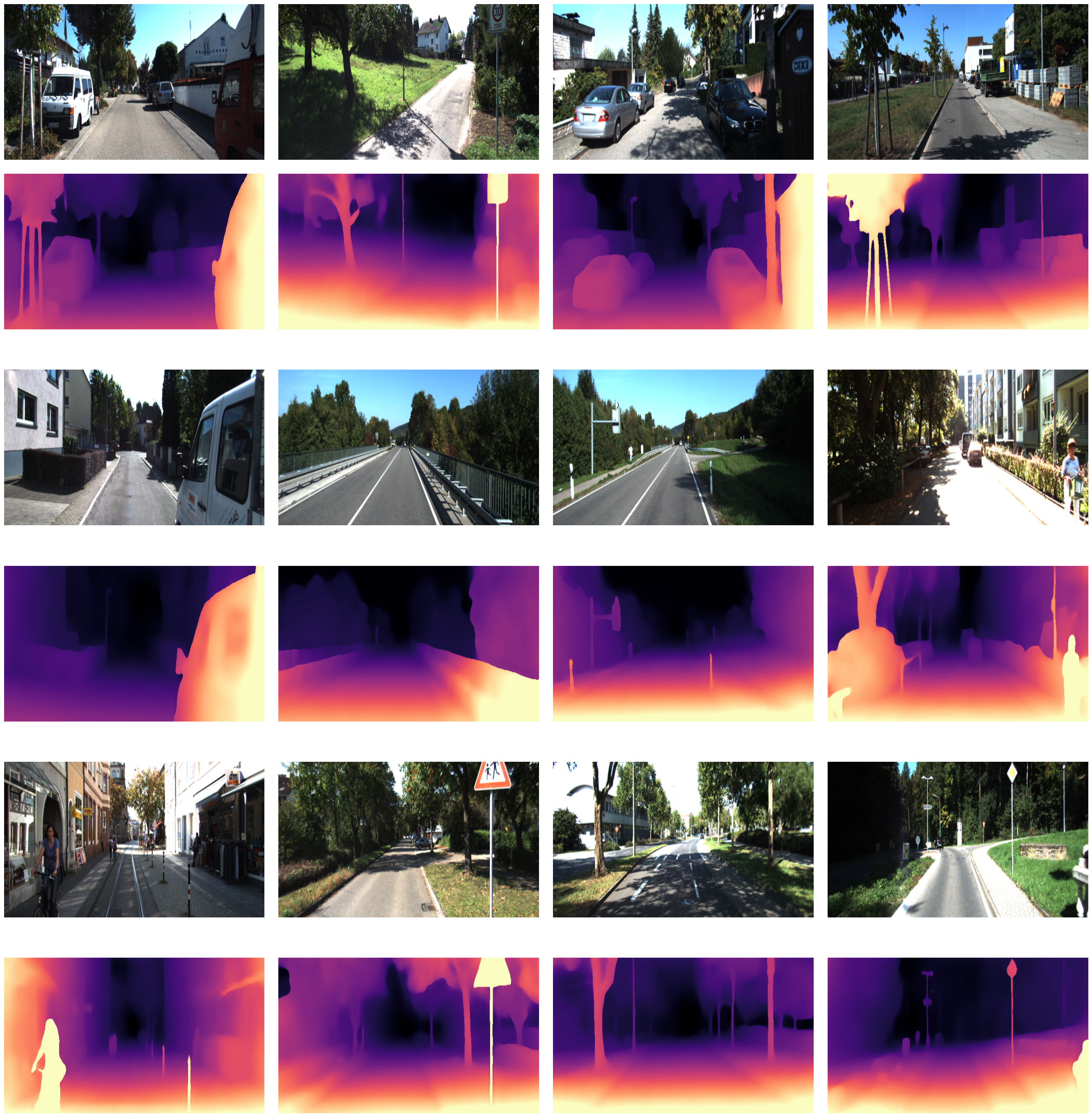


Figure F.1. Qualitative results of our method.

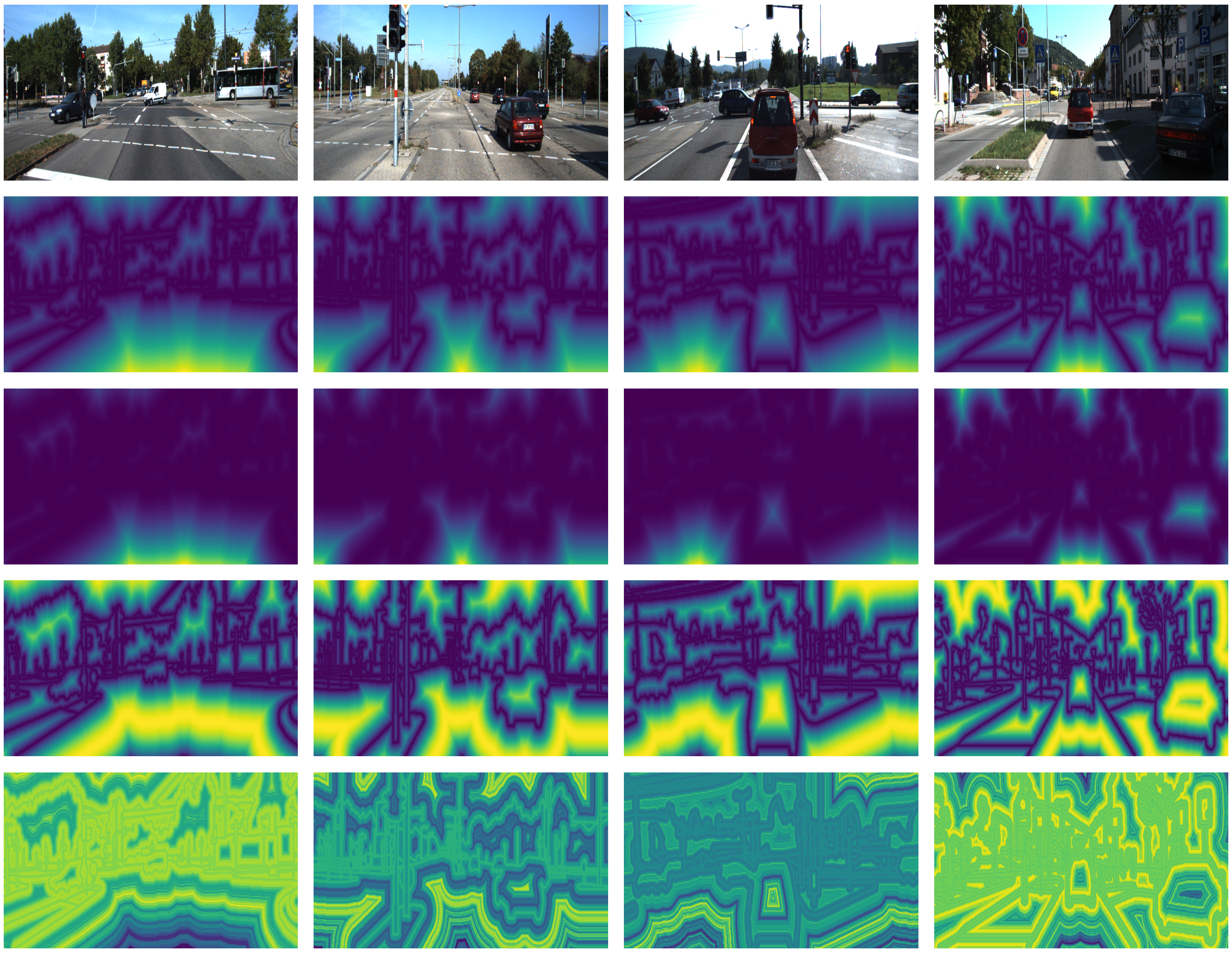


Figure F.2. Qualitative results of different functions of the distance transform. First row: $x \mapsto x$. Second row: $x \mapsto x^2$. Third row: $x \mapsto \sin(\pi x)$. Last row: first dimension of RW_3

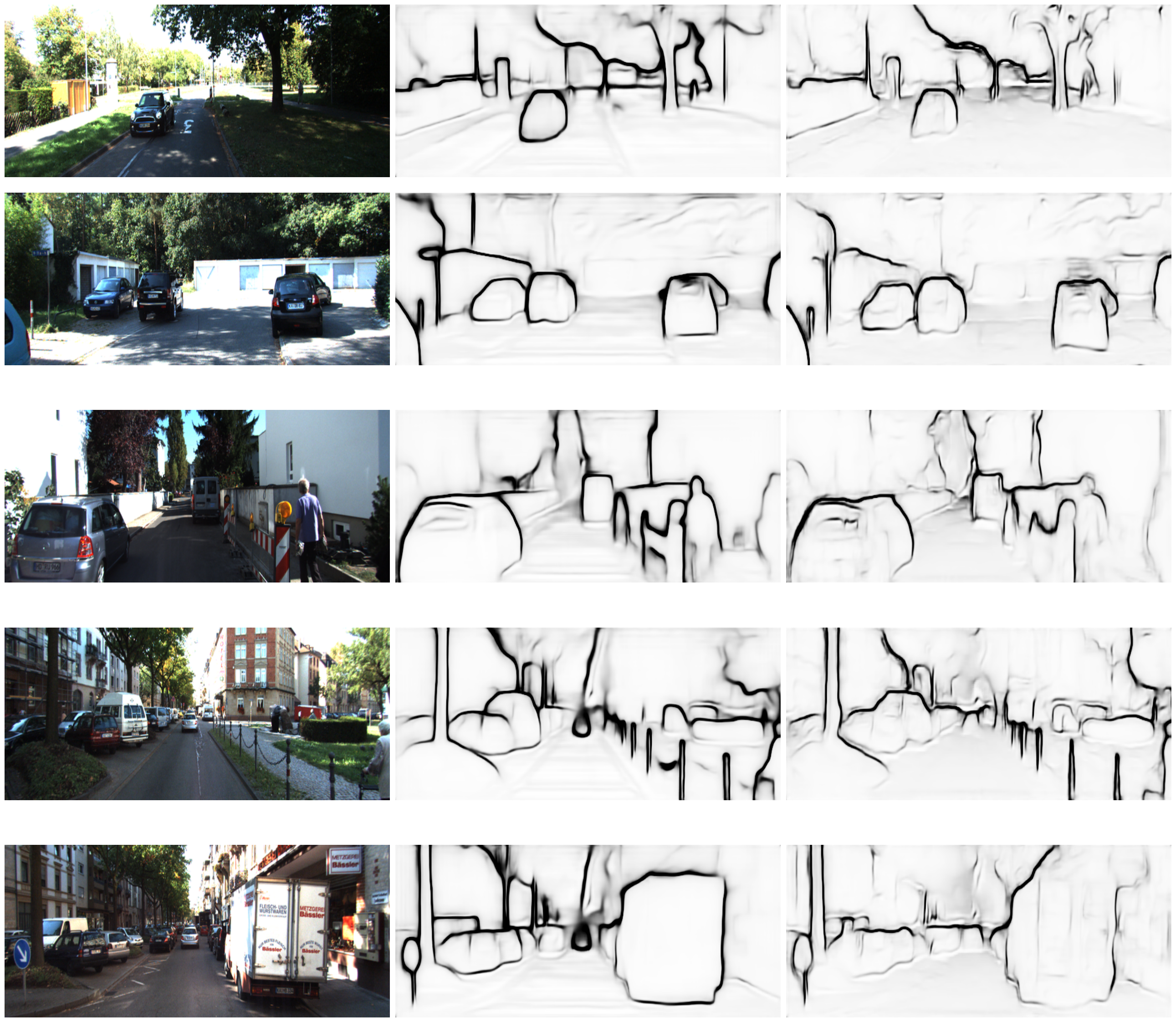


Figure F.3. Contour estimation results for our method (left) and Lego (right) are shown. The results presented here are before post-processing. The Lego method was re-implemented by us, with the rest of our approach remaining unchanged.

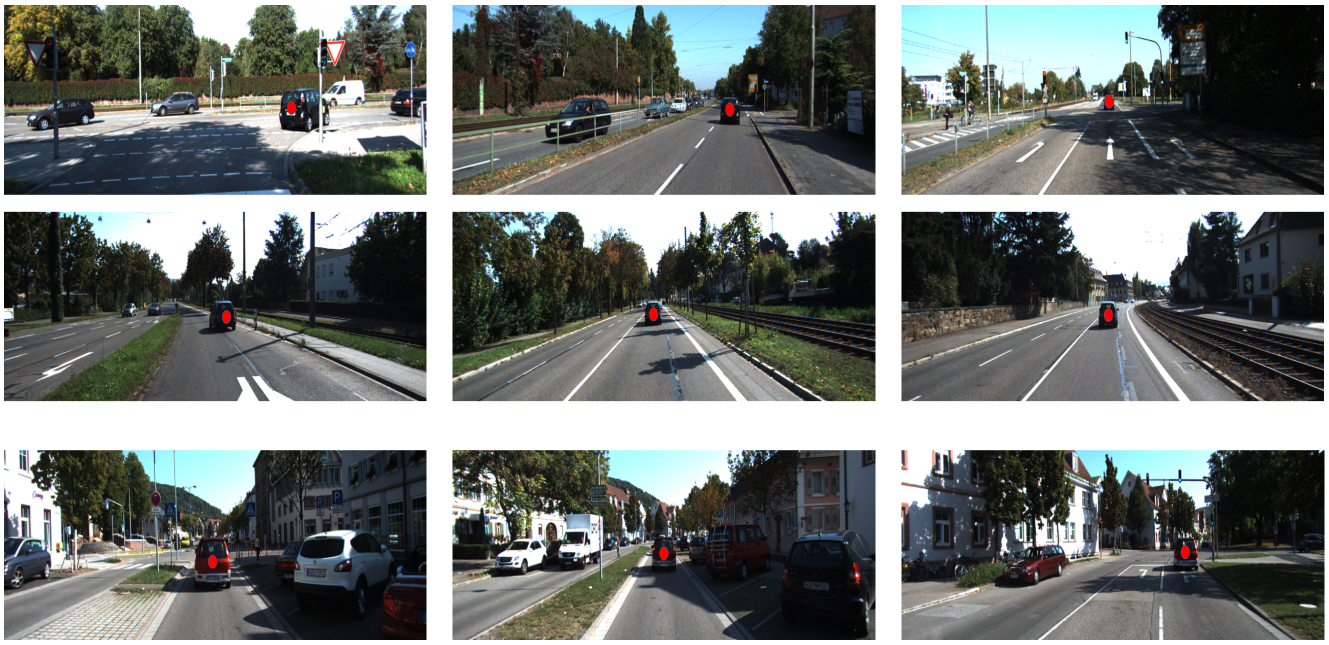


Figure F.4. Images of KITI MOTS[72]. Each row shows a red point on the object tracked along the sequence. First row is sequence 4, second row is sequence 10 and last row is sequence 11. Timeline is arranged from left to right. Red circle is the point tracked and considered for our experiments