This is the final peer-reviewed accepted manuscript of:


The final published version is available online at: https://doi.org/10.1109/ICCV51070.2023.00848

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.
Learning Depth Estimation for Transparent and Mirror Surfaces

Alex Costanzino*  Pierluigi Zama Ramirez*  Matteo Poggi*
Fabio Tosi  Stefano Mattoccia  Luigi Di Stefano
CVLAB, Department of Computer Science and Engineering (DISI)
University of Bologna, Italy
{alex.costanzino, pierluigi.zama, m.poggi, fabio.tosi5}@unibo.it
https://cvlab-unibo.github.io/Depth4ToM/

Abstract

Inferring the depth of transparent or mirror (ToM) surfaces represents a hard challenge for either sensors, algorithms, or deep networks. We propose a simple pipeline for learning to estimate depth properly for such surfaces with neural networks, without requiring any ground-truth annotation. We unveil how to obtain reliable pseudo labels by in-painting ToM objects in images and processing them with a monocular depth estimation model. These labels can be used to fine-tune existing monocular or stereo networks, to let them learn how to deal with ToM surfaces. Experimental results on the Booster dataset show the dramatic improvements enabled by our remarkably simple proposal.

1. Introduction

In our daily lives, we often interact with several objects of various appearances. Among them are those made of transparent or mirror surfaces (ToM), ranging from the glass windows of buildings to the reflective surfaces of cars and appliances. These might represent a hard challenge for an autonomous agent leveraging computer vision to operate in unknown environments. Specifically, among the many tasks involved in Spatial AI, accurately estimating depth information on these surfaces remains a challenging problem for both computer vision algorithms and deep networks [64], yet necessary for proper interaction with the environment in robotic, autonomous navigation, picking, and other application fields. This difficulty arises because ToM surfaces introduce misleading visual information about scene geometry, which makes depth estimation challenging not only for computer vision systems but even for humans — e.g., we might not distinguish the presence of a glass door in front of us due to its transparency. On the one hand, the definition of depth itself might appear ambiguous in such cases:

is depth the distance to the scene behind the glass door or to the door itself? Nonetheless, from a practical point of view, we argue that the actual definition depends on the task itself — e.g., a mobile robot should definitely be aware of the presence of the glass door. On the other hand, as humans can deal with this through experience, depth sensing techniques based on deep learning, e.g., monocular [38, 37] or stereo [26, 22] networks, hold the potential to address this challenge given sufficient training data [64].

Unfortunately, light reflection and refraction over ToM

*These authors contributed equally to this work.
surfaces violate also the working principles of most active depth sensors, such as Time-of-Flight (ToF) cameras or devices projecting structured-light patterns. This has two practical consequences: i) it makes active sensors unsuited to deal with ToM objects in real-world applications, and ii) prevents the use of these sensors for collecting and annotating data to train deep neural networks to deal with ToM objects. As evidence of this, very few datasets featuring transparent objects provide ground-truth depth annotations, which have been obtained through very intensive human intervention [64], graphical engines [40], or based on the availability of CAD models [5] for ToM objects.

In short, accurately perceiving the presence (and depth) of ToM objects represents an open challenge for both sensing technologies and deep learning frameworks. Purposely, this paper proposes a simple yet effective strategy for obtaining training data and, thereby, dramatically boosting the accuracy of learning-based depth estimation frameworks dealing with ToM surfaces. Driven by the observation that ToM objects alone are responsible for misleading recent monocular networks such as AdaBins [2], DPT [37], and MiDaS [38], which would otherwise generalize well to most unseen environments, we argue that replacing them with equivalent, yet opaque objects would allow restoring an environment layout in which such networks could accurately estimate the depth of the scene. To this end, we mask ToM objects in images by in-painting them with arbitrary uniform colors. Then, we employ a monocular depth network to generate a virtual depth map out of the modified image. By repeating this process on a variety of images featuring ToM objects, we can easily and effectively annotate a dataset and then use it to train the same monocular network used to distill labels, which will now process the not-in-painted images. As a result, the trained monocular network will learn to handle ToM objects, producing consistent depth even in their presence.

Our main contributions can be resumed as follows:

- We propose a simple yet very effective strategy to deal with ToM objects. We trick a monocular depth estimation network by replacing ToM objects with virtually textured ones, inducing it to hallucinate their depths.
- We introduce a processing pipeline for fine-tuning a monocular depth estimation network to deal with ToM objects. Our pipeline exploits the network itself to generate virtual depth annotations and requires only segmentation masks delineating ToM objects – either human-made or predicted by other networks [54, 60] – thus getting rid of the need for any depth annotations.
- We show how our strategy can be extended to other depth estimation settings, such as stereo matching. Our experiments on the Booster dataset [64] prove how monocular and stereo networks dramatically improve their prediction on ToM objects after being fine-tuned according to our methodology.

Fig. 1 highlights some specific regions where monocular (top) and stereo (bottom) models struggle (middle column), and how they learn to handle ToM surfaces thanks to our strategy (rightmost column).

2. Related Work

Monocular Depth Estimation. Early methods used CNNs for pixel-level regression [10, 11]. More recent approaches such as AdaBins [2], DPT [37], and MiDaS [38] use adaptive bins and vision transformers for depth regression and leverage large-scale depth training by mixing multiple datasets. Self-supervised methods use view synthesis for image reconstruction, where predicted depth is combined with known or estimated camera pose to establish correspondences between adjacent images, exploiting either stereo pairs [10, 11] or monocular videos [70, 12]. Recent works aim to improve the robustness of the photometric loss based on SSIM and L1 [68] by incorporating photometric uncertainty [57, 35], feature descriptors [66, 44, 46], 3D geometric constraints [29], proxy supervision [52, 49], optical flow [62, 50], or adversarial losses [1, 34]. Others propose architecture changes as in [69, 14, 33, 18, 13]. Except for some works that address non-Lambertian depth estimation using depth completion approaches and sparse depth measurements from active sensors [8, 40], to the best of our knowledge, we are not aware of any previous single-view depth estimation network that can handle ToM surfaces.

Stereo Matching. Traditional algorithms [42] utilize handcrafted features to estimate a disparity map [63, 17, 59, 58, 24, 47, 21, 3]. Then, deep learning methods replaced traditional matching cost computation, as demonstrated in [65], and, eventually, end-to-end approaches became the most effective solution for disparity estimation. These networks can be mainly categorized into 2D and 3D architectures, with the former adopting an encoder-decoder design [30, 32, 25, 39, 45, 56, 61, 48] and the latter building a feature cost volume from extracted features on the image pair [19, 4, 20, 67, 6, 7, 9, 55, 51, 16, 43]. A thorough review of these works can be found in [36]. Recent papers exploit iterative refinement paradigms [26, 22] or rely on Vision Transformers [23, 15]. However, due to its inherently ill-posed nature, dealing with non-Lambertian surfaces, such as ToM objects, remains a very challenging problem for any kind of existing stereo approach.

Non-Lambertian Object Perception. Due to the relevance of dealing with ToM objects, some recent datasets focus on them. Trans10K [54] and MSD [60] consist of over 10 000 and 4 000 real in-the-wild images of transparent objects and mirrors, respectively. Both datasets provide manually annotated segmentations of ToM materials, though
none of them provide depth labels. Others provide depth annotations: ClearPose [5] includes over 350,000 labeled real-world RGB-D frames of 63 household objects. ClearGrasp [40] consists of over 50,000 synthetic RGB-D images of transparent objects, as well as real-world test benchmark with 286 RGB-D images. In addition, Booster [64] focuses on stereo matching, providing high-resolution depth labels and stereo pairs acquired in indoor scenes with specular and transparent surfaces. TOD [28] contains 15 transparent objects, labeled with relevant 3D keypoints, comprising 48,000 stereo and RGBD images. StereOBJ-1M [27] also deals with stereo vision, but focuses on pose estimation for ToM objects and do not provides depth ground truths. Obtaining depth labels for these kinds of datasets is expensive, challenging, and time-consuming since it requires either CAD models for ToM objects [5], painting such objects in the scene [40, 64, 28] or relies on a complex multi-camera setup [53]. In contrast, our proposal effectively sidesteps these challenges, by demonstrating that monocular and stereo networks can learn to deal with these objects in the absence of depth annotations.

Therefore, by virtually replacing these objects with textured artifacts that resemble their very same shapes, the monocular model may be possibly tricked and induced into estimating the depth of an opaque object, ideally placed at the very same spot in the scene. This methodology can be realized by delineating ToM objects, through manual annotations or a segmentation network, masking them from the image and then in-painting virtual textures within the masked areas. On the one hand, since a proper detection of ToM objects is crucial to our methodology, manual labeling indisputably results in the most accurate choice, though it comes with significant annotation costs. On the other hand, relying on a segmentation network would alleviate this cost: one would need some initial human annotations for training, but this would then allow to segment a large number of images for free. Unfortunately, the overall effectiveness of our methodology would be inevitably affected by the accuracy of the trained segmentation model. However, we reckon that annotating images with segmentation masks requires, definitely, a vastly lower effort compared to depth annotation [64, 27]. Hence, we settled on exploring both the aforementioned approaches.

3. Method

Our goal is to generate depth annotations for images featuring ToM objects in a cheap and scalable manner. This allows for training deep networks to properly estimate their depth as the distance of the closest surface in front of the camera, rather than the distance of the scene content refracted/reflected through it. Our strategy is simple yet dramatically effective and relies on the availability of recent pre-trained monocular depth estimation models [38, 37], which are capable of strong generalization across a variety of scenes though struggling to deal with ToM surfaces. Based on the above state of affairs, we argue that ToM objects are often the sole elements harming the reliability of recent pre-trained monocular depth estimation networks.

Figure 2. Monocular Distillation pipeline. Given an RGB and a segmentation mask, we in-paint pixels belonging to transparent and mirror surfaces with a random uniform color and process these augmented images with a pre-trained monocular network. The obtained virtual depths are aggregated to obtain a pseudo-labeled dataset for fine-tuning the network itself.
works. Our experiments will highlight that the former strategy results ineffective, while we achieve a large boost in accuracy by fine-tuning depth models with our approach.

In the remainder, we describe our methodology to deal with ToM objects. Given a dataset of images $\mathcal{I}$, our pipeline sketched in Fig. 2 builds as follows: i) surface labeling, ii) in-painting and distillation, and iii) fine-tuning of the depth network on virtual labels. Additionally, we show how it can be revised to fine-tune also deep stereo networks.

**Surface Labeling.** For any image $I_k \in \mathcal{I}$, we produce a segmentation mask $M_k$ classifying each pixel $p$ as

\[
M_k(p) = \begin{cases} 
1 & \text{if } I_k(p) \in \text{ToM surfaces} \\
0 & \text{Otherwise} 
\end{cases}
\]

by labeling pixels as either 1 or 0 if they belong to a ToM surface or not, respectively. Such a segmentation mask can be obtained either through manual annotation or by means of a segmentation network $\Theta$ as $M_k = \Theta(I_k)$.

**In-painting and Distillation.** Given an image $I_k$ and its corresponding segmentation mask $M_k$, we generate an augmented image $I_k^*$ applying an in-painting operation to replace the pixels belonging to ToM objects with a color $c$:

\[
\tilde{I}_k(p) = \begin{cases} 
c & \text{if } M_k(p) = 1 \\
I_k & \text{otherwise} 
\end{cases}
\]

Then, a virtual depth $\tilde{D}_k$ for image $I_k$ is obtained by forwarding $\tilde{I}_k$ to a monocular depth network $\Psi$ as $\tilde{D}_k = \Psi(\tilde{I}_k)$. Colors are randomly sampled for every single frame $I_k$. However, depending on the image content, certain colors might result ineffective and increase the scene ambiguity – e.g., by in-painting white pixels into a transparent object located in front of a white wall. To discourage these occurrences, we sample a set of $N$ custom colors $c_i, i \in [0, N-1]$, and in-paint $\tilde{I}_k$ using each of these custom colors, so as to generate a set of $N$ augmented images $\tilde{I}_k^i$. Then, we obtain the final, Virtual Depth $\tilde{D}_k$ by computing the per-pixel median between the $N$ depth maps

\[
\tilde{D}_k = \text{med}\left\{\Psi(\tilde{I}_k^i), i \in [0, N-1]\right\}
\]

As depicted in Fig. 3, in some cases, the in-painted color might be similar to the background – e.g., the transparent object disappears when a single gray mask is used – while it is visible by aggregating multi-color in-painting.

**Fine-Tuning on Virtual Labels.** The steps outlined so far allow for labeling a dataset $\mathcal{I}$ with virtual depth labels that are not influenced by the ambiguities of ToM objects. Then, our newly annotated dataset can be used to train or fine-tune a depth estimation network, thereby enabling it to handle the aforementioned difficult objects robustly. Specifically, during training, the original images $I_k$ are forwarded to the network, and the predicted depth $\hat{D}_k$ is optimized with respect to the distilled virtual ground-truth map $\tilde{D}_k$ obtained from in-painted images. This simple pipeline can dramatically improve the accuracy of monocular depth estimation networks when dealing with ToM objects, as we will show in our experiments.

**Extension to Deep Stereo.** Our pipeline can be adapted to fine-tune deep stereo models as well, as shown in Fig. 4. Again, we argue that state-of-the-art stereo architectures [26, 22] already expose outstanding generalization capabilities while struggling with ToM objects, due to the task of matching pixels belonging to non-Lambertian surfaces being inherently ambiguous. Consequently, we exploit a monocular depth estimation network to obtain virtual depth annotations solely for these objects. Given a dataset $\mathcal{S}$ consisting of stereo pairs $(L_k, R_k)$, we distill virtual depth labels $\tilde{D}_k^s$ from $L_k$ and triangulate them into disparities $d_k^s$ according to extrinsic parameters of the stereo rig. Then,
we predict a Base disparity map $d_k$ by forwarding $(L_k, R_k)$ to the stereo network we aim to fine-tune. Eventually, we replace the disparities for ToM objects with those from $d_k$ according to $M_k$, this latter produced over $L_k$ this time. Formally, this operation namely Merging is, defined as:

$$d_k(p) = \begin{cases} 
    d_k(p) & \text{if } M_k(p) = 0 \\
    \alpha_k d_k(p) + \beta_k & \text{otherwise}
\end{cases}$$

with $\alpha_k, \beta_k$ being scale and shift factors, as monocular predictions are up to an unknown scale factor. Following [38], $\alpha_k, \beta_k$ are estimated through Least Square Estimation (LSE) regression over $d_k$ for pixels not belonging to any ToM object, i.e., having $M_k(p) = 0$:

$$\arg \min_{\alpha, \beta} \sum_{p | M_k(p) = 0} \left( \alpha d_k(p) + \beta - d_k(p) \right)^2$$

### 4. Experimental Settings

**Implementation Details.** We employ MiDaS [38] and DPT [37] as our monocular networks using the official pretrained weights, given their excellent in-the-wild generalization performance. To fine-tune them, we iterate for 20 epochs with batch size 2 and a learning rate of $10^{-7}$ with exponential decay with gamma 0.95. We use random color and brightness and random horizontal flip augmentations. We pad/crop and resize images to match the pre-training resolution, i.e., 384 pixels for the long or short side, preserving aspect ratio with mirror pad or square crop, for MiDaS or DPT, respectively. We normalize images as the original networks do. Regarding stereo networks, we employ RAFT [26] and CREStereo [22], using the official pre-trained weights, since they achieve the top rankings in the Middlebury dataset [41] among published methods. To fine-tune them, we run 20 epochs, with batch size 2, fixed learning rate $10^{-5}$. Following [64], we randomly resize images to half or quarter of the original dataset resolution, randomly crop to $456 \times 884$ and $448 \times 880$ for RAFT and CREStereo respectively, and further random scale images and disparities by a factor $\in [0.9, 1.1]$. We assume 22 and 10 iterations during training for RAFT-Stereo and CREStereo, respectively. During testing, we run 32 and 20 iterations. When creating virtual labels with our masking strategy, we fix the random color of seed sampling to 0.

**Datasets.** Among the datasets, we selected Trans10K [54], MSD [60], and Booster [64] as they focus on ToM surfaces and contain images acquired in many realistic environments. Trans10K contains 5,003, 1,003, 4,431 images for the training, validation, and test set, respectively, featuring common transparent objects and stuff. It provides segmentation masks with pixels categorized into 12 different classes that we collapse into 2 – ToM (classes 1 to 11) or not. MSD contains 3,066, and 958 images and binary segmentation masks for the training and test set, respectively, featuring mirrors. Booster contains 228, and 191 images for training and testing, respectively. The dataset provides disparity and segmentation maps for the training set, where the segmentation maps are categorized into 4 classes, which we group into 2 – classes 2-3 into “ToM” category, classes 0-1 into “Other” materials. We fine-tune on Trans10K and MSD for monocular models and on the Booster training split for stereo networks, without using any depth ground truths.

**Evaluation Protocol.** We evaluate the accuracy of the monocular networks using several metrics, including absolute error relative to the ground-truth value (ABS Rel.), the percentage of pixels having the maximum between the prediction/ground-truth and ground-truth/prediction ratios lower than a threshold ($\delta_i$, with $i$ being 1.05, 1.10, 1.15, 1.20, and 1.25), the mean absolute error (MAE) and Root Mean Squared Error (RMSE). Additionally, we evaluate stereo networks using the metrics defined in Booster [64], i.e. bad-2, bad-4, bad-6, bad-8, MAE, RMSE. Results are reported on all valid pixels (All) or for those belonging to either ToM or other objects, in order to assess the impact of our strategy on the different kinds of surfaces. For any metrics considered for stereo networks, the lower, the better – annotated with \( \downarrow \) in tables. The same applies to metrics used for monocular networks except for $\delta_i$, resulting in the higher, the better – with \( \uparrow \) being reported in tables. As the predictions by monocular networks are up to an unknown scale factor, we rescale them according to the LSE criterion [38] defined in Eq. 5, using all valid pixels here. Monocu-
MAE Abs. Rel RMSE
70.68 0.06 83.06
91.08 88.83 0.07 118.23
89.17 0.07 119.58
92.48 118.84 0.11 141.27
64.93 0.05 85.93
72.45 0.06 94.07
75.35 0.06 100.68
96.83 94.60 90.14 79.46 56.89
126.71 0.12 145.69
93.66 0.09 104.80
122.33 0.12 140.31
86.30 82.96 72.84 61.04 37.91
64.70 0.05 85.57
93.34
113.14 0.10 136.28
92.69 89.49 81.12 63.62 57.95
118.84 0.11 141.27
70.68 0.06 85.93

Experiments
In this section, we report the outcome of our findings concerning both monocular and stereo depth estimation.

5.1. Monocular Depth Estimation.
Number of In-Paintings. We investigate the quality of the virtual depth labels by varying N. When using N = 1 we generate a single in-painted image that is forwarded to the monocular network, while with N = 5 we generate virtual depths from 5 masked images with different colors, which are then aggregated by selecting the pixel-wise depth median. In Tab. 1, we report the accuracy of depth maps produced by the two strategies, together with those of the Base architectures, i.e. without applying any in-painting strategy. Firstly, with both MiDaS and DPT, both in-painting strategies obtain virtual depths that are much more accurate for ToM regions w.r.t. the Base architecture. Secondly, N=5 maps yield slightly better results in most metrics, especially when looking at DPT performance. We ascribe it to the higher robustness of the second strategy. For the remaining experiments, we fix N = 5 as we did not observe any further improvement with larger values.

Fine-tuning Results (GT Segmentation). In Tab. 2, we report results on the Booster train set, obtained after fine-tuning MiDaS and DPT on all available data from Trans10K and MSD. In the Base row, we report the results of the network using the officially released weights without any further training, and we compare with those in row Ft. Virtual Depth, i.e., the results of our method. We notice that the accuracy on All pixels is improved with our approach. In particular, we achieve a significant boost in performances for ToM surfaces, of 4.37, 5.72, 8.97, 11.12 and 11.57%, 28.67mm, 0.03%, 35.51mm for MiDaS [38], and 3.91, 7.19, 11.25, 16.8 and 16.97%, 42.46mm, 0.04%, 53.255mm for DPT [38] in the δ1, δ2, δ10, δ11, δ15, δ10, δ0.5, MAE, Abs.Rel, and RMSE, respectively. We highlight that, after fine-tuning, the accuracy on ToM is only slightly worse than on Other. Moreover, class Other metrics are also slightly

<table>
<thead>
<tr>
<th>Category</th>
<th>Method</th>
<th>δ &lt; 1.25 %</th>
<th>δ &lt; 1.20 %</th>
<th>δ &lt; 1.15 %</th>
<th>δ &lt; 1.10 %</th>
<th>δ &lt; 1.05 %</th>
<th>MAE (mm)</th>
<th>Abs. Rel (%)</th>
<th>RMSE (mm)</th>
<th>MAE (mm)</th>
<th>Abs. Rel (%)</th>
<th>RMSE (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Base</td>
<td>94.56</td>
<td>91.72</td>
<td>85.68</td>
<td>74.00</td>
<td>50.12</td>
<td>90.82</td>
<td>0.07</td>
<td>120.31</td>
<td>96.79</td>
<td>94.45</td>
<td>89.71</td>
</tr>
<tr>
<td>All</td>
<td>Ft. Virtual Depth</td>
<td>94.09</td>
<td>91.28</td>
<td>85.29</td>
<td>73.55</td>
<td>49.14</td>
<td>93.34</td>
<td>0.07</td>
<td>124.54</td>
<td>89.67</td>
<td>86.84</td>
<td>81.99</td>
</tr>
<tr>
<td>All</td>
<td>Ft. Virtual Depth (Proxy)</td>
<td>95.07</td>
<td>92.31</td>
<td>86.39</td>
<td>75.20</td>
<td>50.79</td>
<td>88.83</td>
<td>0.07</td>
<td>118.23</td>
<td>97.99</td>
<td>96.65</td>
<td>93.55</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Method</th>
<th>δ &lt; 1.25 %</th>
<th>δ &lt; 1.20 %</th>
<th>δ &lt; 1.15 %</th>
<th>δ &lt; 1.10 %</th>
<th>δ &lt; 1.05 %</th>
<th>MAE (mm)</th>
<th>Abs. Rel (%)</th>
<th>RMSE (mm)</th>
<th>MAE (mm)</th>
<th>Abs. Rel (%)</th>
<th>RMSE (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Base</td>
<td>89.44</td>
<td>83.40</td>
<td>72.71</td>
<td>59.63</td>
<td>36.28</td>
<td>122.33</td>
<td>0.12</td>
<td>140.31</td>
<td>92.77</td>
<td>87.77</td>
<td>80.98</td>
</tr>
<tr>
<td>All</td>
<td>Ft. Virtual Depth</td>
<td>66.30</td>
<td>82.96</td>
<td>72.84</td>
<td>61.04</td>
<td>37.91</td>
<td>126.71</td>
<td>0.12</td>
<td>143.09</td>
<td>92.69</td>
<td>89.49</td>
<td>81.12</td>
</tr>
<tr>
<td>All</td>
<td>Ft. Virtual Depth (Proxy)</td>
<td>91.81</td>
<td>89.12</td>
<td>83.68</td>
<td>70.75</td>
<td>47.85</td>
<td>93.66</td>
<td>0.09</td>
<td>104.00</td>
<td>70.68</td>
<td>0.06</td>
<td>85.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Method</th>
<th>δ &lt; 1.25 %</th>
<th>δ &lt; 1.20 %</th>
<th>δ &lt; 1.15 %</th>
<th>δ &lt; 1.10 %</th>
<th>δ &lt; 1.05 %</th>
<th>MAE (mm)</th>
<th>Abs. Rel (%)</th>
<th>RMSE (mm)</th>
<th>MAE (mm)</th>
<th>Abs. Rel (%)</th>
<th>RMSE (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToM</td>
<td>Base</td>
<td>71.87</td>
<td>85.57</td>
<td>74.01</td>
<td>50.28</td>
<td>91.08</td>
<td>0.07</td>
<td>119.86</td>
<td>97.10</td>
<td>94.88</td>
<td>90.08</td>
<td>79.76</td>
</tr>
<tr>
<td>ToM</td>
<td>Ft. Base</td>
<td>94.27</td>
<td>91.49</td>
<td>85.87</td>
<td>73.77</td>
<td>49.38</td>
<td>92.48</td>
<td>0.07</td>
<td>122.46</td>
<td>97.15</td>
<td>95.02</td>
<td>90.97</td>
</tr>
<tr>
<td>ToM</td>
<td>Ft. Virtual Depth (GT)</td>
<td>95.31</td>
<td>92.53</td>
<td>86.80</td>
<td>75.43</td>
<td>50.46</td>
<td>89.57</td>
<td>0.07</td>
<td>118.23</td>
<td>97.00</td>
<td>96.64</td>
<td>93.52</td>
</tr>
</tbody>
</table>
better, probably because of the enhanced features extracted by the network, which has a better understanding of the scene context. Finally, we have reported in *Ft. Base* the fine-tuning results obtained by self-training the networks on their own predictions without any in-painting strategy. As expected, without the appropriate virtual depth labels, the networks cannot effectively learn from the new dataset, yielding results comparable to the *Base* architecture. Experiments on additional datasets are in the supplement.

### Fine-tuning Results (Proxy Segmentation)

Even though obtaining semantic labels is cheaper than collecting depth ground truths, using the predictions of a segmentation network as proxy semantic annotations would accelerate the dataset collection process. Thus, we investigate the impact of replacing manually annotated masks in our pipeline with the predictions of Trans2Seg [54] and MirrorNet [60], pre-trained on the training set of Trans10K and MSD, respectively, on the unseen test set of each dataset. We use weights made available by the authors. For a fair comparison, we also re-train again the models exploiting GT segmentations only on the test sets of the two datasets. Tab. 3 highlights that both models, using either GT - *Ft. Virtual Depth (GT)* - or proxy segmentations - *Ft. Virtual Depth (Proxy)*, achieve much more accurate results compared to the *Base* network. Interestingly, the two networks yield comparable results in the class *Other*, while the one using GTs is slightly better than the other in the class *ToM*, yet still comparable.

Finally, in row *Virtual Depth (Proxy)*, we report the results of our in-painting methodology (i.e., without fine-tuning) but coloring pixels according to the proxy segmentations. We note that performances are even worse than the *Base* method. Indeed, the segmentation network struggles to generalize to the Booster dataset, making the depth model incapable of estimating the correct values, e.g., due to some overextended in-painted ToM areas, as shown in Fig. 5.

<table>
<thead>
<tr>
<th>Category</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToM</td>
<td>Base</td>
</tr>
<tr>
<td>ToM</td>
<td>Ft. Base</td>
</tr>
<tr>
<td>ToM</td>
<td>Ft. Virtual Depth</td>
</tr>
<tr>
<td>ToM</td>
<td>Ft. Merged</td>
</tr>
<tr>
<td>Other</td>
<td>Base</td>
</tr>
<tr>
<td>Other</td>
<td>Ft. Base</td>
</tr>
<tr>
<td>Other</td>
<td>Ft. Virtual Depth</td>
</tr>
<tr>
<td>Other</td>
<td>Ft. Merged</td>
</tr>
</tbody>
</table>

### Table 5. Stereo networks fine-tuning – ground truth segmentation.

Results on Booster test set at quarter resolution. All models start from the official weights [26, 22] and are fine-tuned according to different strategies. Best results in **bold**.

![Image](312x325 to 387x381)

**Figure 6. Qualitative comparison – disparity virtual labels.** On top: left RGB image, ground truth segmentation mask, and virtual disparities by CREStereo processing masked stereo pairs. At bottom: depth by DPT on the in-painted left image, disparities by CREStereo, and their final merged labels.

---

9250
5.2. Stereo Depth Estimation

Virtual Disparity Generation Alternatives. We inquire about two main alternatives to generate virtual disparities: i) Virtual Disparity: masking both left and right images according to material segmentation masks – as materials annotations are provided for the left image only, we warp it over the right image according to ground-truth disparity – and then processing the two with the stereo network we are going to fine-tune similar to Monocular networks, ii) Merged: merging disparity labels produced by the stereo model itself with those generated by original DPT weights [37], as detailed in Eq. 4. Although the former might appear as the natural extension of our proposal from the monocular to the stereo case, we will demonstrate its ineffectiveness.

Fine-tuning Results (GT Segmentation). Tab. 5 collects the results obtained by fine-tuning RAFT-Stereo and CREStereo through our technique. From top to bottom, we report the results achieved by the original models (Base) as well as the instances fine-tuned on their own predictions (Ft. Base) or on pseudo labels obtained according to the two strategies (Ft. Virtual Depth, Ft. Merged).

Not surprisingly, fine-tuning the networks on their own predictions is harmful (RAFT-Stereo) or scarcely effective (CREStereo). Applying the first of the two strategies sketched before yields just a negligible improvement over the original models on ToM classes. This evidence confirms that our pipeline designed for the monocular case cannot naïvely be extended to the stereo case by in-painting the two images since masking ToM objects with constant colors does not ease matching – on the contrary, it introduces textureless regions, which are likely to be labeled as planar surfaces by stereo models. Conversely, the second strategy consistently improves the predictions with both RAFT-Stereo and CREStereo. In particular, the former achieves 9.23, 13.83, 15.62, and 16.69% absolute reductions on bad-2, bad-4, bad-6, and bad-8, respectively, as well 7.62 and 9.13 reductions on MAE and RMSE on ToM regions. CREStereo obtains 14.93, 17.13, 17.48, and 17.54% on bad metrics, and 7.40 and 8.91 reductions on MAE and RMSE. Moreover, the accuracy over Other pixels is also improved, although with minor margins. Fig. 6 provides a qualitative comparison between the labels obtained by the two strategies. The former produces a planar surface for the mirror completely misaligned with respect to the wall, whereas the latter combines the virtual depth labels from DPT on masked images with disparity labels at best.

Fine-tuning Results (Proxy Segmentation). Finally, we replace the manually annotated segmentation masks with those predicted by Trans2Seg and MirrorNet and then distill virtual disparities for fine-tuning both stereo networks. As pointed out before, both Trans2Seg and MirrorNet have not been trained on Booster. Thus, Merging produces significant differences with respect to the use of manually annotated masks, as shown in Fig. 7. Nevertheless, Tab. 6 shows that our pipeline improves the performance of both RAFT-Stereo and CREStereo on ToM objects, even in the case of extremely noisy proxy semantic annotations. More precisely, CREStereo seems to benefit more from the Proxy segmentation configuration than RAFT-Stereo. Indeed, on the one hand, we can notice how RAFT-Stereo improves on ToM regions at the expense of accuracy on other pixels when using Proxy segmentations. This yields,
on All pixels, an increase in the bad-2 and bad-4 error rates, whereas bad-6, bad-8, MAE, and RMSE remain lower. On the other hand, CREStereo seems capable of exploiting fine-tuning much better, yielding more accurate results on any metric with both Proxy or GT masks. This outcome proves that our pipeline is effective for fine-tuning stereo models even without manually annotated masks. Nonetheless, segmenting images through human labeling unleashes its full potential, whose cost is much lower compared to that that would be required to annotate depths.

5.3. Qualitative Results

To conclude, Fig. 8 shows the effect of the fine-tuning carried out according to our proposal, with two examples for monocular (top) and stereo (bottom) networks from the Booster train and test sets respectively. We highlight how MiDaS, DPT, RAFT-Stereo, and CREStereo learn to deal with ToM surfaces either when relying on proxy segmentation masks provided by neural networks or accurately annotated by humans. More qualitative examples are reported in the supplementary material.

6. Conclusion

We have proposed an effective methodology for training depth estimation networks to deal with transparent and mirror surfaces. By in-painting these surfaces on RGB images, we can quickly annotate a dataset with virtual depth labels, that can be used to fine-tune both monocular and stereo networks, with outstanding results. A promising future direction would be to extend our technique to instance segmentation masks to get better virtual depth maps in the presence of multiple ToM objects in the same scene.
References


for disparity estimation through feature constancy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2


[37] Thanks to the OpenCV developers for their excellent and freely available library. OpenCV, 2019. 1, 2, 3, 4, 5, 6, 7, 8, 9


[40] Fabio Tosi, Filippo Aleotti, Pierluigi Zama Ramirez, Matteo Poggi, Samuele Salti, Luigi Di Stefano, and Stefano Mat-


[60] Xin Yang, Haiyang Mei, Ke Xu, Xiaopeng Wei, Baozai Yin, and Ryson W.H. Lau. Where is my mirror? In The IEEE International Conference on Computer Vision (ICCV), October 2019. 2, 5, 7
