ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Real-Time Semantic Stereo Matching

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Real-Time Semantic Stereo Matching / P. L. Dovesi, M. Poggi, L. Andraghetti, M. Martí, H. Kjellström, A. Pieropan, S. Mattoccia. - ELETTRONICO. - (2020), pp. 19987014.10780-19987014.10787. (Intervento presentato al convegno International Conference on Robotics and Automation (ICRA 2020) tenutosi a Paris, France nel 31 May 2020 - 31 August 2020) [10.1109/ICRA40945.2020.9196784].

Availability:
This version is available at: https://hdl.handle.net/11585/764255 since: 2020-07-05

Published:
DOI: http://doi.org/10.1109/ICRA40945.2020.9196784

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.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

03 May 2024

This is the final peer-reviewed accepted manuscript of:

P. L. Dovesi et al., "Real-Time Semantic Stereo Matching," 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 2020, pp. 10780-10787

The final published version is available online at ***https://doi.org/10.1109/ICRA40945.2020.9196784***

Rights / License:

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.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

Real-Time Semantic Stereo Matching

Pier Luigi Dovesi[1,2], Matteo Poggi[3], Lorenzo Andraghetti[1], Miquel Martí[1,2],
Hedvig Kjellström[2], Alessandro Pieropan[1], Stefano Mattoccia[3]

***Abstract*— Scene understanding is paramount in robotics, self-navigation, augmented reality, and many other fields. To fully accomplish this task, an autonomous agent has to infer the 3D structure of the sensed scene (to know where it looks at) and its content (to know what it sees). To tackle the two tasks, deep neural networks trained to infer semantic segmentation and depth from stereo images are often the preferred choices. Specifically, Semantic Stereo Matching can be tackled by either standalone models trained for the two tasks independently or joint end-to-end architectures. Nonetheless, as proposed so far, both solutions are inefficient because requiring two forward passes in the former case or due to the complexity of a single network in the latter, although jointly tackling both tasks is usually beneficial in terms of accuracy. In this paper, we propose a single compact and lightweight architecture for real-time semantic stereo matching. Our framework relies on coarse-to-fine estimations in a multi-stage fashion, allowing: i) very fast inference even on embedded devices, with marginal drops in accuracy, compared to state-of-the-art networks, ii) trade accuracy for speed, according to the specific application requirements. Experimental results on high-end GPUs as well as on an embedded Jetson TX2 confirm the superiority of semantic stereo matching compared to standalone tasks and highlight the versatility of our framework on any hardware and for any application.**

I. Introduction

In order to develop a fully autonomous system able to navigate in an unknown environment independently, scene understanding is essential. In particular, an intelligent agent needs to recognize objects in its surroundings and determine their 3D location before performing high-level reasoning concerning path planning, collision avoidance and other tasks. This requires addressing two problems: *depth estimation* and *semantic segmentation*. Among the techniques to infer depth, stereo vision has been around for a long time [1] since it is potentially accurate and efficient. In the few past years it has been heavily influenced by machine learning techniques. In contrast, semantic segmentation only recently emerged as an effectively addressable problem thanks to machine learning and the recent spread of deep learning.

In this paper, we refer to *Semantic Stereo Matching* as the combination of the two tasks aimed at understanding the surrounding environment sensed by a stereo camera. Nowadays, standalone networks trained for each of the two specific tasks represent the state-of-the-art. However, although modern deep architectures allow for easy integration of multiple tasks [2], top performing frameworks rarely exploit the possible synergies between the tasks. Indeed,

[1]Univrses AB
[2]KTH Royal Institute of Technology, Sweden.
[3]University of Bologna, Italy.

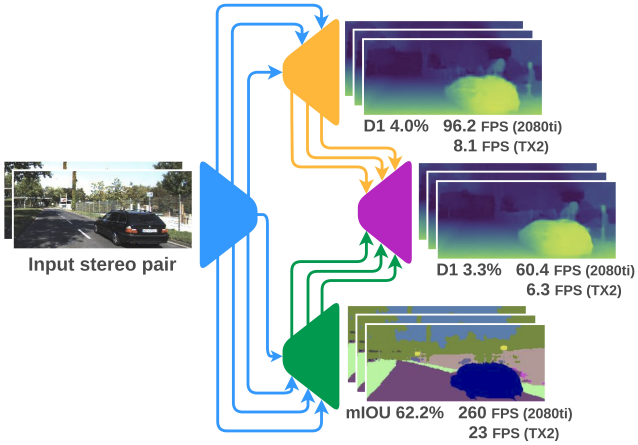

Fig. 1: RTS^{2}Net allows for fast and accurate semantic segmentation and disparity estimation, both on high-end GPUs and low power systems.

casting semantic stereo matching as a joint optimization of segmentation and disparity estimation yields mutual benefit to both tasks. For instance, depth estimation in challenging portions of the image corresponding to reflective surfaces can be improved by knowing that they belong to a car and thus to an object with defined 3D properties. On the other hand, depth awareness can help to reduce ambiguity when dealing, for instance, with the segmentation of vegetation and terrain. Several works in the literature support the synergy between semantic and depth inference [3]–[8] and more recently the first semantic stereo matching frameworks appeared [9], [10]. However, even if these first attempts confirm the effectiveness of such a paradigm, they are far from real-time performance even on power hungry high-end GPUs. In particular, they barely break the 1 FPS barrier, thus are not ready for deployment in real-world applications.

Purposely, in this paper, we propose a novel Real-Time Semantic Stereo Network (RTS^{2}Net) for jointly solving the two aforementioned tasks. It is designed to leverage the synergies between the two: it learns a common feature representation for both domains and employs separate decoders for estimating accurate semantic segmentation and disparity maps. Moreover, by designing a stack of multi-stage decoders, RTS^{2}Net produces coarse-to-fine estimations for the two tasks, enabling to i) keep low memory and runtime requirements for full inference and ii) further increasing the speed by early-stopping the model at coarse resolution [11], [12] according to the time/resource budget available

at deployment. Figure 1 sketches the RTS^{2}Net architecture, highlighting how from a shared representation (blue) our network can reason about both semantics (green) and disparity (yellow) and finally post-process early estimates together (purple) to improve depth accuracy. Thanks to its lightweight design, RTS^{2}Net can run at several FPS on an NVIDIA Jetson TX2 module with a power budget smaller than 15W, yet providing accurate results competitive with much more complex state-of-the-art networks. Moreover, by early-stopping the network, for instance, before the post-processing phase, we can increase speed with an acceptable decrease of accuracy. To the best of our knowledge, RTS^{2}Net represents the first real-time solution for joint semantic segmentation and stereo matching running seamlessly on high-end GPUs and low-power devices.

II. Related work

In this section, we review the literature concerning stereo matching, semantic segmentation and multi-task approaches combining depth and semantic.

Stereo matching. Before the deep learning era, stereo algorithms consisted of four well-defined steps [1]: i) cost computation, ii) cost aggregation, iii) disparity optimization/computation and iv) disparity refinement. Eventually, the very first attempts to leverage machine learning for stereo concerned confidence measures [13] or replacing some of the aforementioned steps in stereo with deep learning, for example learning a matching function by means of CNNs [14]–[16], improving optimization [17], [18] or refining disparity maps [19], [20].

End-to-end networks for stereo matching appeared simultaneously to the availability of synthetic data [21] and DispNetC was the first network introducing a custom correlation layer to encode similarities between pixels as features. Kendall *et al.* [22] designed GC-Net, a 3D network processing a cost volume built through features concatenation. Starting from these seminal works, two families of architectures were developed, respectively 2D and 3D networks. Frameworks belonging to the first class traditionally use one or multiple correlation layers [9], [23]–[28], while 3D networks build 4D volumes by means of concatenation [12], [29]–[31], features difference [32] or group-wise correlations [33]. Although most works focus on accuracy, some deployed compact architectures [12], [27], [32], [34] aimed at real-time performance. Finally, the guided stereo paradigm [35] combines end-to-end models with external depth cues to improve accuracy and generalization of both 2D and 3D architectures.

Semantic segmentation. The advent of deep learning moved semantic segmentation from hand-crafted features and classifiers, like Random Forests [36] or Support Vector Machines [37], to fully convolutional neural networks [38]. Architectures for semantic segmentation typically exploit contextual information according to five main strategies. The first consists of using multi-scale prediction models [39]–[42], making the same architecture process inputs at different scales so to extract features at different contextual levels. The second deploys traditional, encoder-decoder architectures [38], [43]–[45]. The third encodes long-range context information exploiting Conditional Random Fields either as a post-processing module [41] or as an integral part of the network [46]. The fourth uses spatial pyramid pooling to extract context information at different levels [41], [41], [47]. Finally, the fifth deploys *atrous*-convolutions to extract higher resolution features while keeping a large receptive field to capture long-range information [48], [49]. As for stereo, some recent works [50]–[54] focused on efficiency rather than on accuracy for semantic segmentation. Zhu *et al.* [55] recently proposed video prediction-based method to synthesize new training samples.

Multi-task frameworks. There exist approaches aimed at joint depth and semantic estimation, either from monocular images [3]–[8] or stereo images [9], [10]. In both cases, jointly learning depth and semantic segmentation enabled the improvement of each task. Nonetheless, stereo approaches are lagging far behind the real-time performance required by most practical applications.

III. Real-time semantic stereo network

In this section, we introduce our framework for semantic stereo matching. We start with a general overview of the proposed RTS^{2}Net, then focus on describing each component and their interactions.

A. Architecture Overview

In order to achieve high accuracy with limited execution time, the network design consists of a fully residual and pyramidal architecture [11], [12], [27]. As depicted in Figure 2, the network is divided into four distinct modules: shared encoder in blue, stereo disparity decoder in yellow, semantic decoder in green and synergy disparity refinement module in purple. For each block, we report the number of convolutional layers composing it and the number of features they output as multiple of a factor c, hyper-parameter of the network described in detail next. The network is designed to keep a symmetrical architecture between disparity regression and semantic segmentation in order to facilitate the exploitation of the shared parameters. Both segmentation and disparity are fully computed only at the lowest resolution and progressively refined through the higher resolution residual stages. The same design occurs for the final refinement module, processing the two outputs to improve the disparity estimation significantly. Indeed, even in this final stage, the full refined disparity is only computed at the lowest level and progressively upsampled together with the coarse disparity and semantic segmentation. This fully residual setup provides consistent advantages both at training-time, since early losses stabilize and accelerate this phase, and at testing-time since we can dynamically adjust the speed/accuracy trade-off, as discussed next.

B. Joint features extractor

As in most architectures, the earliest stage performs feature extraction from the input images. The shared encoder,

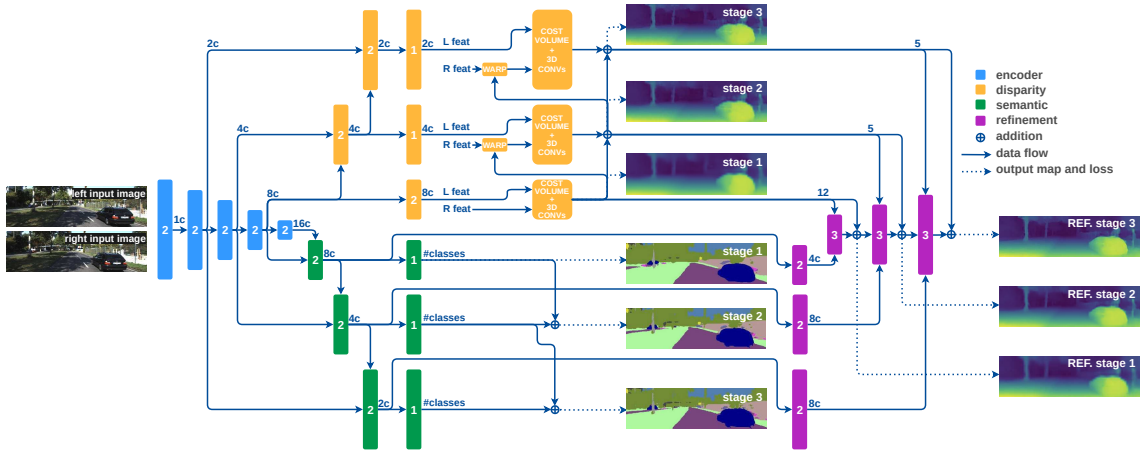

Fig. 2: RTS^{2}Net architecture overview. Features extracted from the input stereo pair (blue) are the common ground for stereo (yellow) and semantic (green) inference. Finally, the two outputs are combined (purple) for improved synergic disparity estimation. For each block, we report the number of convolutional layers composing it and the number of features they output, multiples of a factor c hyper-parameter of the network.

depicted in blue in Figure 2, is made of two initial 3×3 convolutions extracting c features and bringing the resolution to half, then followed by four blocks each one containing a 2×2 max-pooling operation and two 3×3 layers. The four respectively extract $2c$, $4c$, $8c$, $16c$ features while progressively halving the resolution, *i.e.* $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$ and $\frac{1}{32}$ respectively. Batch normalization and ReLU operations follow all convolutional layers. Features extracted by this module are processed by two subnetworks, in charge respectively of semantic segmentation and disparity estimation. This forces RTS^{2}Net to learn a general and enriched representation meaningful for both tasks. This design allows us for a dramatic reduction of the computational cost compared to much more complex encoders such as VGG [56], yet enabling accurate results. In particular, previous works [12] proved that a tiny amount of features, *i.e.* c=1, already enables for decent disparity estimation while significantly increasing the framerate. However, it is insufficient to learn a representation good enough for semantic segmentation too.

C. Disparity Network

Following the design of pyramidal networks [11], [12], [27], a stack of decoders is deployed to estimate coarse-to-fine disparity maps. This strategy allows us to keep computational efforts low as well as to manage the speed-accuracy trade-off dynamically, by performing three stages respectively at $\frac{1}{16}$, $\frac{1}{8}$ and $\frac{1}{4}$ resolution. These stages have been selected because the coarser resolution, *e.g.* at $\frac{1}{32}$, did not improve the results while running decoders at lower-res would significantly increase the runtime with negligible improvements on the final accuracy. Deploying the shared features computed by the feature extractor, task-specific embeddings are extracted at the three resolutions mentioned above, as shown by the yellow blocks in Figure 2.

At first, the disparity network takes the disparity features extracted at $\frac{1}{16}$ resolution and builds a distance-based cost volume by progressively shifting right features up to a maximum d_{max} range and subtracting them from left ones to directly obtain an approximation of matching costs. By building the volume at low resolution, a small d_{max} is enough to look for the entire disparity range at the original resolution. In particular, we choose $d_{max} = 12$, corresponding to 192 maximum disparity at full resolution. Then, the volume is regularized through three 3D *conv* blocks followed by batch normalization and ReLU, extracting respectively 16, 16 and 1 features. Finally the disparity map is obtained by means of a *soft-argmin* [22] operator. We kept the same amount of channels as in [12]. This first, coarse estimation is upsampled to $\frac{1}{16}$ and used to warp right disparity features towards left ones. At this stage, a new cost volume is built in order to find residual disparities and thus to obtain a more accurate disparity map. This time we assume $d_{max} = \pm 2$, *i.e.* ± 16 at full resolution (we look for both positive and negative residuals, since coarse disparities may be higher or lower than real values). Then we deploy a decoder with three 3D convolutions extracting 4, 4 and 1 features and a final *soft-argmin* layer as well. The residual disparity is summed to the upsampled estimation from $\frac{1}{16}$ resolution, and the resulting map is further upsampled to $\frac{1}{4}$ resolution for the final stage, identical to the previous, to improve further the disparity estimation. Finally, the result of the third stage is bilinearly upsampled from $\frac{1}{4}$ to full resolution.

D. Semantic Segmentation Network

The second subnetwork in charge of semantic segmentation follows the same coarse-to-fine design strategy for the reasons previously outlined as well as to balance the two branches (*i.e.* depth and segmentation) of the whole RTS^{2}Net network. Again, the shared features computed by the encoder are processed by additional 2D convolutions as in the disparity branch. Besides, $\frac{1}{32}$ features are also used to exploit a broader image context, crucial for semantic

segmentation. The semantic segmentation branch is made of three stages as well, as shown by the green blocks in Figure 2. Each stage produces per-pixel probability scores for each semantic class, defined according to the KITTI benchmark, at $\frac{1}{16}$, $\frac{1}{8}$ and $\frac{1}{4}$ as the disparity network does. As depicted in the figure, estimated probabilities are upsampled across the stages and summed using residual connections to the outputs of the same stage. These final probabilities allow to infer the semantic map at each stage through a *argmax* over the class scores.

E. Synergy Disparity Refinement module

The network described so far outputs standalone semantic and disparity maps, yet from a shared representation. The final module in RTS^{2}Net, namely *Synergy Disparity Refinement*, reverts this path by jointly processing the two task-specific estimates with a single module to refine the disparity regression leveraging semantic cues. A similar method has been successfully deployed by previous works [9], [10] with a simple, yet effective strategy consisting of a concatenation of the two embeddings into a *hybrid volume*.

We adapted this approach to the fully residual strategy followed both in the disparity network and in the semantic decoder. To achieve this, we perform a cascade of residual concatenations between semantic class probabilities and disparity volumes. The refinement module, in purple in Figure 2, performs three steps: 1) in order to limit computational time and balance the contributions in the hybrid volume, we compress the semantic embedding so to have dimensionality similar to the disparity cost volume, 2) we concatenate compressed semantic features with disparity volumes (reorganized so to have disparity dimension as channels) to form the hybrid volumes, in the second and third stage we also concatenate the upsampled previously computed refined disparity, 3) the hybrid volume is then processed through three 2D convolutional layers, producing disparity residuals summed up to the original, reorganized volumes on which the *soft-argmin* operator is applied.

F. Objective function

Summarizing the network outputs, we have 3 coarse disparities d_{st}, 3 semantic segmentation s_{st} and 3 refined disparities d^r_{st}, with stages $st \in [1, 2, 3]$ corresponding to the 3 different resolutions. Regarding the disparity regression, we employ smooth L1 losses $\mathcal{L}_{d_{st}}$ and $\mathcal{L}_{d^r_{st}}$ defined as

$$L1_{smooth} = \begin{cases} 0.5(d_i - \hat{d}_i)^2, & \text{if } |d_i - \hat{d}_i| < 1 \\ |d_i - \hat{d}_i| - 0.5, & \text{otherwise} \end{cases} \quad (1)$$

with d and $\hat{d}$ respectively the estimated and ground truth disparities, while $\mathcal{L}_{s_{st}}$ for semantic segmentation *multi class cross entropy*. All losses are averaged over the total amount of pixels. Since the outputs belong to different decoders and thus computed at different resolutions, we propose a double hierarchical loss weighing scheme:

$$\mathcal{L} = \sum_{st=1}^{3} \mathcal{W}_{st} \cdot (\mathcal{W}_d \cdot \mathcal{L}_{d_{st}} + \mathcal{W}_s \cdot \mathcal{L}_{s_{st}} + \mathcal{W}_{d^r} \cdot \mathcal{L}_{d^r_{st}}) \quad (2)$$

			Disparity	
Main dataset	epochs	KITTI epochs	EPE	D1-all%
Sceneflow	10	300	1.24	6.47
Sceneflow	40	800	1.18	6.28
CS (coarse→fine)	60→75	800	**1.14**	**5.75**

TABLE I: Different training schedules / train sets tested on KITTI 2015 validation split, with c=1 (AnyNet [12]).

where $\mathcal{L}$ is the overall objective function score, $\mathcal{W}_{st}$ are stage weights and $\mathcal{W}_d, \mathcal{W}_s, \mathcal{W}_{d^r}$ are task specific weights respectively for disparity, semantic and refined disparity. In our case W_{st} are respectively $\frac{1}{4}$, $\frac{1}{2}$ and 1 for first, second and final stages, while $\mathcal{W}_d$, $\mathcal{W}_s$, $\mathcal{W}_{d^r}$ are 1, 2 and 2. The segmentation cross-entropy is also weighted according to the class probability to alleviate the effect of unbalanced datasets [57]. Moreover, since we are working under a multi-task setup, we want to keep the impact of the segmentation independent to the choice of internal weighing schedule or class distribution. Therefore, we design the following weighing scheme:

$$\mathcal{W}_j = \frac{N}{\log(\mathcal{P}_j + k) \sum_{i=1}^{N} \frac{1}{\log(\mathcal{P}_i + k)}} \quad (3)$$

with W_j the weight of the j class, N the total number of classes, P a class probability and k a parameter that controls the variance of the class weights, set differently according to the dataset (*i.e.* , 1.12 for CityScapes [54] and 2 for KITTI 2015). Finally, in case of coarse semantic annotations [58], we re-weight the segmentation loss according to the percentage of unlabelled area left in the ground truth to obtain $\mathcal{L}_s*$

$$\mathcal{L}_s* = \mathcal{L}_s(1 + \gamma \cdot \frac{A_{unlab}}{A_{tot} - A_{unlab}}) \quad (4)$$

with γ set to 0.1, to achieve the best results, and A_{unlab}, A_{tot} respectively the unlabelled and total amounts of pixels.

IV. Experimental Results

In this section, we extensively evaluate the performance of RTS^{2}Net in terms of both accuracy and runtime. To compare different variants of our model and measure the impact of each of the design choices, we report quantitative results on a validation split sampled from the KITTI 2015 training split made of 40 images, using the remaining 160 for training. We report the End-Point-Error (EPE) and percentage of pixels with disparity error larger than 3 pixels and 5% of the ground truth (D1-all%) to evaluate the accuracy of estimated disparity maps. For both metrics: the lower, the better. For semantic segmentation, we compute the class mean Intersection Over Union (mIOU%) and the per-pixel accuracy (pAcc%). For both metrics: the higher, the better.

A. Training Schedule

Traditionally, end-to-end stereo networks are trained from scratch on the Freiburg SceneFlow dataset [21], an extensive collection of synthetic stereo images with dense ground truth disparities, before finetuning on the real, yet smaller

		Disparity		Semantic		Frame rate (FPS)	
Model	c	EPE	D1-all%	mIOU%	pAcc%	TX2	2080ti
AnyNet [12]	1	1.14	5.75	✗	✗	**10.4**	**96.8**
RTS^{2}Net	1	**1.12**	**5.57**	58.86	80.86	8.3	60.5
AnyNet [12]	4	0.96	4.22	✗	✗	**9.3**	**96.2**
RTS^{2}Net	4	**0.90**	**3.80**	60.93	89.77	7.4	60.5
AnyNet [12]	8	0.91	3.98	✗	✗	**8.1**	**96.2**
RTS^{2}Net	8	**0.84**	**3.33**	62.22	90.64	6.3	60.4
AnyNet [12]	16	0.87	3.52	✗	✗	**6.2**	**95.8**
RTS^{2}Net	16	**0.78**	**2.90**	67.41	92.92	4.5	60.4
AnyNet [12]	32	0.82	3.12	✗	✗	**3.5**	**64.1**
RTS^{2}Net	32	**0.74**	**2.62**	**69.62**	**93.57**	2.3	42.2

TABLE II: Impact of c on KITTI 2015 validation split.

target dataset such as KITTI 2015 [59]. However, since SceneFlow does only provide instance segmentation labels, it is not possible to train RTS^{2}Net for semantic segmentation on such imagery. Thus, we initialize our network on the CityScapes dataset [58] (CS), providing about 25K stereo pairs with disparity maps obtained employing Semi-Global Matching algorithm (SGM) [60] and semantic segmentation labels, for which 5K images are densely labeled and 20K coarsely. Although disparity ground truth maps are noisy, a proper training schedule on CS in place of the traditional SceneFlow dataset is more effective when moving to KITTI. Table I reports experiments supporting this strategy. We trained a variant of RTS^{2}Net by setting c=1 and removing both semantic and refinement networks, *i.e.* equivalent to the AnyNet architecture [12]. This way, we aim at measuring only the impact of the different training schedules on disparity estimation, excluding improvements introduced by model variants or multi-task learning that will be evaluated in the remainder. In all our experiments, we train on 256×512 crops with batch size 6. We use Adam as optimizer with betas 0.9 and 0.999 and set learning rate to $5e^{-4}$, kept constant on SceneFlow/CityScapes and halved every 200 epochs on KITTI. We can see how a more extended training on both SceneFlow and KITTI is beneficial compared to the scheduling proposed in [12], respectively extending from 10 to 40 and from 300 to 800 epochs. By replacing the SceneFlow pre-train with a multi-stage schedule on CS, 60 epochs on coarse ground truth followed by 75 on fine annotations, allows for better accuracy when followed by the same KITTI finetuning.

B. Model variants

As described in Section III-A, we designed most layers in RTS^{2}Net to extract features that are multiples of a basis factor c. For instance, by cutting off semantic and synergy modules and setting c to 1, we obtain the AnyNet architecture [12]. Although very fast for disparity inference alone, extracting so few features may lack at representing semantical information. To assess this, we train and evaluate variants of RTS^{2}Net by setting different c factors. Table II collects the outcome of these experiments conducted on both the AnyNet architecture, inferring disparity only, and our proposal, inferring disparity and semantic segmentation. All networks have been trained following the best schedule discussed in the previous section, *i.e.* 60 epochs on coarse CS, 75 on fine CS and 800 on KITTI.

	Disparity		Semantic		Frame rate (FPS)	
Networks	EPE	D1-all%	mIOU%	pAcc%	TX2	2080ti
Disp.	0.91	3.98	✗	✗	**8.1**	**96.2**
Disp. + Sem.	0.90	3.90	**64.21**	**91.56**	6.6	76.9
Disp. + Sem. + Ref.	0.91 (**0.84**)	3.91 (**3.33**)	62.22	90.64	6.3	60.4

TABLE III: Ablation study (c=8), KITTI 2015 validation split.

	Stage 1		Stage 2		Stage 3	
Model	FPS	D1-all%	FPS	D1-all%	FPS	D1-all%
AnyNet [12]	34.6	11.60	20.5	8.40	10.4	5.75
RTS^{2}Net (c=8)	17.2	8.00	10.9	4.70	6.3	3.33

TABLE IV: Anytime inference, KITTI 2015 validation split.

By setting c=1, we obtain the same AnyNet configuration reported in [12]. Choosing the same c on RTS^{2}Net allows for a moderate improvement on disparity estimation, as well as to obtain reasonable results in terms of semantic segmentation, at the cost of lower frame rate. By increasing c respectively to 4, 8, 16 and 32 we observe a consequent increase in accuracy on both tasks. In particular, passing from 1 to 32 allows for a significant improvement regarding semantic segmentation estimates, confirming that c=1 is insufficient for this purpose. Interestingly, the margin between RTS^{2}Net and AnyNet on disparity metrics gets larger by increasing the number of features. Indeed, EPE margin is 0.02, 0.06, 0.07, 0.09 and 0.12, while D1-all% margin is 0.18, 0.42, 0.65, 0.62 and 0.50. This highlights that increasing the number of features is much more beneficial when the model is trained jointly for semantic segmentation too, confirming this latter task to benefit more from a larger pool of features.

For practical applications, c=8 represents a good trade-off allowing for 6.3 FPS on Jetson TX2, *i.e.* about 160ms per inference. Most of the time is taken by the disparity subnetwork (120ms).

C. Impact of multi-task and synergy modules

We measure the contribution given by both the multi-task learning paradigm itself and the synergy refinement module specifically designed for RTS^{2}Net. Table III collects the results obtained by training ablated configuration of our model depicted in Figure 2, setting c=8. The training has been conducted on CS and KITTI, as described in the previous section. On top, the variant made of the features encoder (blue) and the disparity subnetwork (yellow). We can notice how, by simply adding the semantic network (green) and training for joint optimization of the two tasks, slightly increases the disparity accuracy respectively by 0.01 and 0.08 in terms of EPE and D1-all%. As expected, the best results are achieved by adding the synergy refinement module (purple), shown in brackets on the last row of the table together with EPE and D1-all% obtained from the disparity network without applying the refinement. From these results, as for semantic segmentation, we can notice that depth estimation marginally loses accuracy compared to the previous model (0.01 on both EPE and D1-all% and 1.99, 0.92 on mIOU% and pAcc%). Nonetheless, this

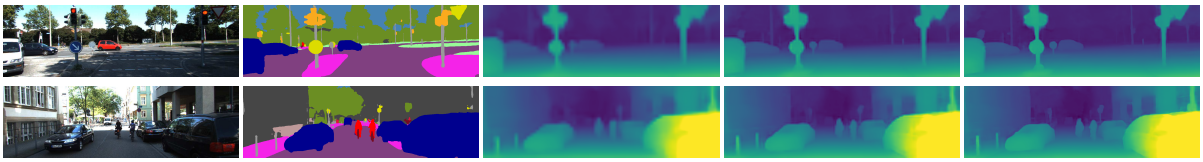

Fig. 3: Qualitative results on KITTI. From left to right: reference image, semantic and coarse to fine disparity maps.

Network	D1-bg%	D1-fg%	D1-all%	Runtime (s)
GANet [62]	1.48	3.46	1.81	1.80
HD^3 [28]	1.70	3.63	2.02	0.14
GWCNet [33]	1.74	3.93	2.11	0.32
SegStereo [9]	1.88	4.07	2.25	0.60
PSMNet [29]	1.86	4.62	2.32	0.41
RTS^2Net (ours)	3.09	5.91	3.56	0.02
DispNetC [21]	4.32	4.41	4.34	0.06
MADNet [27]	3.75	9.20	4.66	0.02
StereoNet [32]	4.30	7.45	4.83	0.02

TABLE V: Result on KITTI 2015 online benchmark (stereo).

configuration yields a more considerable improvement after refinement.

D. Anytime inference

RTS^2Net allows for trading accuracy for speed by early-stopping inference at an intermediate stage, a property shared with other architectures [11], [12]. Table IV compares the trade-off achieved respectively by AnyNet [12] and our architecture measured on the NVIDIA Jetson TX2. We focus on studying the impact on disparity estimation, since it represents the bottleneck in our system. First, we can notice how RTS^2Net at any stage runs roughly at half the frames per second, with ample margins in terms of improved accuracy. Moreover, we highlight in red the two configurations achieving the minimum frame rate compatible with the KITTI acquisition system (*i.e.* , 10 FPS [61]), respectively AnyNet Stage 3 and RTS^2Net Stage 2. In this setting, RTS^2Net runs slightly faster than AnyNet and achieves 1.05% reduction in terms of D1-all%, yet providing the additional semantic segmentation output making our framework the preferred choice for practical applications. Moreover, by paying a reasonable price in terms of speed RTS^2Net can further reduce the error rate compared to AnyNet by a total 2.42%.

E. Evaluation on KITTI online benchmark

We report the results achieved by submitting the maps produced by RTS^2Net on KITTI 2015 online benchmark. To this aim, we trained a model having c=32 to compete with state-of-the-art architectures, traditionally more complex, achieving 0.74, 2.62 in terms of EPE and D1-all% and 69.62, 93.57 on mIOU% and pAcc% on the validation split of Table II. We report runtimes on nVidia 2080ti.

Table V report a comparison between our model and published state-of-the-art architectures taken from the online stereo leaderboard, reporting the D1 metric on the background (D1-bg%), foreground (D1-fg%) and all (D1-all%) pixels. Unfortunately, results for AnyNet were not submitted by the authors to the online KITTI leaderboard.

Network	IoU class%	iIoU class%	IoU category%	iIoU category%	Runtime (s)
VideoProp-LabelRelax [55]	72.82	48.68	88.99	75.26	-
IfN-DomAdap-Seg [63]	59.50	30.28	81.57	61.91	1.00
SegStereo [9]	59.10	28.00	81.31	60.26	0.60
RTS^2Net (ours)	57.67	27.42	82.85	60.72	0.02 (0.008)
SDNet [64]	51.14	17.74	79.62	50.45	0.20
APMoE_seg_ROB [65]	47.96	17.86	78.11	49.17	0.20

TABLE VI: Result on KITTI 2015 segmentation benchmark.

Nonetheless, previous experimental results highlighted the superior accuracy of our proposal. From the table, we can notice how RTS^2Net results more accurate than state-of-the-art real-time frameworks MADNet [27] and StereoNet [32], confirming the effectiveness of jointly inferring semantic and disparity estimation. The gap with state-of-the-art architectures reported in the upper portion of Table V ranges between 1.2 and 1.7% on D1-all%, yet running 7 to 90× faster.

Table VI reports a comparison between RTS^2Net and published methods on the KITTI semantic segmentation online benchmark, highlighting semantic stereo frameworks in yellow. Regarding the execution time of RTS^2Net, we report it when regressing only semantic information (0.008s) and depth plus semantic (0.02s). Compared to SegStereo [9], our network performs slightly worse on class level, while being more accurate on categories and running about 30× faster. Moreover, it outperforms some competitors specifically trained for semantic segmentation only [64], [65].

F. Qualitative results

Figure 3 shows some qualitative examples of semantic segmentation and disparity maps generated by RTS^2Net. Finally, we refer the reader to the **supplementary material**[1] for qualitative results on a KITTI video sequence.

V. Conclusions

This paper proposed a fast and lightweight end-to-end deep network for scene understanding capable of jointly inferring depth and semantic segmentation exploiting their synergy. As reported in the exhaustive experimental results, this strategy compares favorably to the state-of-the-art in both tasks. Moreover, a peculiar pyramidal design strategy enables us to infer stereo and semantic segmentation in a fraction of the time required by other methods as well as to dynamically trade accuracy for speed according to the specific application requirements. To the best of our knowledge, our proposal is the first network enabling to simultaneously infer accurate depth and semantic segmentation suited for real-time applications, even on a low power budget deploying embedded devices like the NVIDIA Jetson TX2.

[1] https://www.youtube.com/watch?v=wbtQcWAbgo0

REFERENCES

[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *IJCV*, vol. 47, no. 1-3, pp. 7–42, 2002.

[2] R. Caruana, "Multitask learning," in *Learning to Learn*, 1998.

[3] L. Ladicky, J. Shi, and M. Pollefeys, "Pulling things out of perspective," in *CVPR*, 2014, pp. 89–96.

[4] A. Mousavian, H. Pirsiavash, and J. Košecká, "Joint semantic segmentation and depth estimation with deep convolutional networks," in *3DV*. IEEE, 2016, pp. 611–619.

[5] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille, "Towards unified depth and semantic prediction from a single image," in *CVPR*, 2015, pp. 2800–2809.

[6] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.

[7] P. Zama Ramirez, M. Poggi, F. Tosi, S. Mattoccia, and L. Di Stefano, "Geometry meets semantics for semi-supervised monocular depth estimation," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 298–313.

[8] F. Tosi, F. Aleotti, P. Zama Ramirez, M. Poggi, S. Salti, L. Di Stefano, and S. Mattoccia, "Distilled semantics for comprehensive scene understanding from videos," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[9] G. Yang, H. Zhao, J. Shi, Z. Deng, and J. Jia, "Segstereo: Exploiting semantic information for disparity estimation," in *ECCV*, 2018, pp. 636–651.

[10] J. Zhang, K. A. Skinner, R. Vasudevan, and M. Johnson-Roberson, "Dispsegnet: Leveraging semantics for end-to-end learning of disparity estimation from stereo imagery," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1162–1169, 2019.

[11] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia, "Towards real-time unsupervised monocular depth estimation on cpu," in *IEEE/JRS Conference on Intelligent Robots and Systems (IROS)*, 2018.

[12] Y. Wang, Z. Lai, G. Huang, B. H. Wang, L. van der Maaten, M. Campbell, and K. Q. Weinberger, "Anytime stereo image depth estimation on mobile devices," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2019.

[13] M. Poggi, F. Tosi, and S. Mattoccia, "Quantitative evaluation of confidence measures in a machine learning world," in *ICCV*, 2017, pp. 5228–5237.

[14] J. Žbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches." *Journal of Machine Learning Research*, vol. 17, no. 1-32, p. 2, 2016.

[15] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang, "A deep visual correspondence embedding model for stereo matching costs," in *ICCV*, 2015, pp. 972–980.

[16] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *CVPR*, 2016, pp. 5695–5703.

[17] A. Seki and M. Pollefeys, "Patch based confidence prediction for dense disparity map." in *BMVC*, vol. 2, no. 3, 2016, p. 4.

[18] ——, "Sgm-nets: Semi-global matching with neural networks," in *CVPR*, 2017, pp. 231–240.

[19] S. Gidaris and N. Komodakis, "Detect, replace, refine: Deep structured prediction for pixel wise labeling," in *CVPR*, 2017, pp. 5248–5257.

[20] K. Batsos and P. Mordohai, "Recresnet: A recurrent residual cnn architecture for disparity map enhancement," in *International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 238–247.

[21] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *CVPR*, 2016, pp. 4040–4048.

[22] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in *ICCV*, 2017, pp. 66–75.

[23] J. Pang, W. Sun, J. S. Ren, C. Yang, and Q. Yan, "Cascade residual learning: A two-stage convolutional neural network for stereo matching," in *ICCV*, 2017, pp. 887–895.

[24] Z. Liang, Y. Feng, Y. Guo, H. Liu, W. Chen, L. Qiao, L. Zhou, and J. Zhang, "Learning for disparity estimation through feature constancy," in *CVPR*, 2018, pp. 2811–2820.

[25] E. Ilg, T. Saikia, M. Keuper, and T. Brox, "Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[26] X. Song, X. Zhao, H. Hu, and L. Fang, "Edgestereo: A context integrated residual pyramid network for stereo matching," in *ACCV*, 2018.

[27] A. Tonioni, F. Tosi, M. Poggi, S. Mattoccia, and L. Di Stefano, "Real-time self-adaptive deep stereo," in *CVPR*, June 2019.

[28] Z. Yin, T. Darrell, and F. Yu, "Hierarchical discrete distribution decomposition for match density estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6044–6053.

[29] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *CVPR*, 2018, pp. 5410–5418.

[30] G.-Y. Nie, M.-M. Cheng, Y. Liu, Z. Liang, D.-P. Fan, Y. Liu, and Y. Wang, "Multi-level context ultra-aggregation for stereo matching," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[31] S. Tulyakov, A. Ivanov, and F. Fleuret, "Practical deep stereo (pds): Toward applications-friendly deep stereo matching," in *Advances in Neural Information Processing Systems*, 2018, pp. 5871–5881.

[32] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi, "Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction," in *ECCV*, 2018, pp. 573–590.

[33] X. Guo, K. Yang, W. Yang, X. Wang, and H. Li, "Group-wise correlation stereo network," in *CVPR*, 2019.

[34] F. Aleotti, M. Poggi, F. Tosi, and S. Mattoccia, "Learning end-to-end scene flow by distilling single tasks knowledge," in *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[35] M. Poggi, D. Pallotti, F. Tosi, and S. Mattoccia, "Guided stereo matching," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[36] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *CVPR*. IEEE, 2008, pp. 1–8.

[37] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *ICCV*. IEEE, 2009, pp. 670–677.

[38] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440.

[39] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *ICCV*, 2015, pp. 2650–2658.

[40] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *CVPR*, 2016, pp. 3640–3649.

[41] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *TPAMI*, vol. 40, no. 4, pp. 834–848, 2018.

[42] C. Liang-Chieh, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *ICLR*, 2015.

[43] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *TPAMI*, 2017.

[44] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[45] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation," *CVPR*, 2017.

[46] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *ICCV*, 2015, pp. 1529–1537.

[47] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017, pp. 2881–2890.

[48] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," *CoRR, abs/1703.06211*, vol. 1, no. 2, p. 3, 2017.

[49] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell, "Understanding convolution for semantic segmentation," *WACV*, 2018.

[50] R. P. Poudel, U. Bonde, S. Liwicki, and C. Zach, "Contextnet: Exploring context and detail for semantic segmentation in real-time," in *BMVC*, 2018.
[51] R. P. Poudel, S. Liwicki, and R. Cipolla, "Fast-scnn: fast semantic segmentation network," in *BMVC*, 2019.
[52] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
[53] D. Mazzini, "Guided upsampling network for real-time semantic segmentation," in *BMVC*, 2018.
[54] X. Chen, X. Lou, L. Bai, and J. Han, "Residual pyramid learning for single-shot semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, p. 111, 2019. [Online]. Available: http://dx.doi.org/10.1109/TITS.2019.2922252
[55] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. D. Newsam, A. Tao, and B. Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[57] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," 2016.
[58] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.
[59] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *CVPR*, 2015, pp. 3061–3070.
[60] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *PAMI*, vol. 30, no. 2, pp. 328–341, 2008.
[61] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *CVPR*, 2012.
[62] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr, "Ga-net: Guided aggregation net for end-to-end stereo matching," in *CVPR*, 2019.
[63] J.-A. Bolte, M. Kamp, A. Breuer, S. Homoceanu, P. Schlicht, F. Hger, D. Lipinski, and T. Fingscheidt, "Unsupervised Domain Adaptation to Improve Image Segmentation Quality Both in the Source and Target Domain," in *Proc. of CVPR - Workshops*, Long Beach, CA, USA, Jun. 2019.
[64] M. Ochs, A. Kretz, and R. Mester, "SDNet: Semantic guided depth estimation network," in *German Conference on Pattern Recognition (GCPR)*, 2019.
[65] S. Kong and C. Fowlkes, "Pixel-wise attentional gating for parsimonious pixel labeling," in *WACV*, 2019.