Thesis

Enhancement of consistent depth estimation approach for monocular videos

,Presented By
Mohamed Nasser

, Supervisor:
Evgeny Pavlovsky
Nikolay Tolstokulakov

December 22, 2020

Content

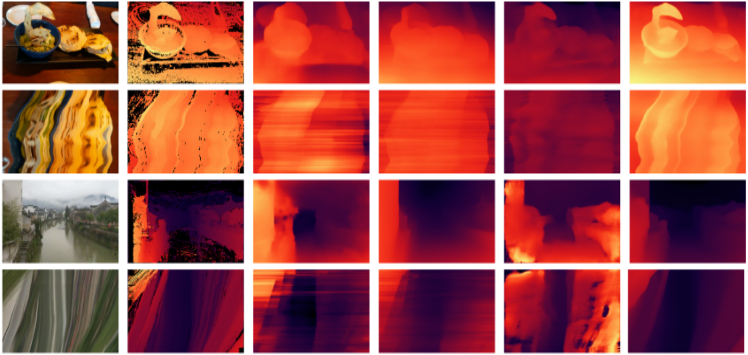
- Introduction
- Approach
- Challenges and limitations
- Our goal
- Enhancement
- Current progress
- Plan for the future work

Introduction

- 3D scene reconstruction from image sequences has been an active research topic in both the robotics and computer vision communities for over a decade.
- Depth perception is an essential step to tackle real-world problems such as robotics and autonomous driving
- reconstructing dense, geometrically consistent depth for all pixels in a monocular video.

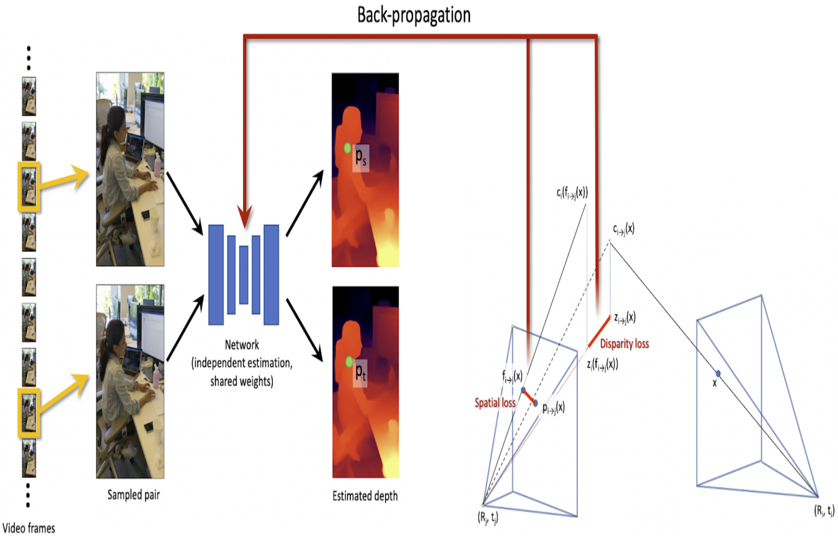
Approach

- Consistent Video Depth Estimation
- 1.Pre-processing
- 2-Test-time training on input video



(a) Input (b) COLMAP (c)Mannequin (d) MiDaS-v2 (e) NeuralRGBD (f) Ours

Approach



Back-propagation
Video frames
Sampled pair
Network
(independent estimation,
shared weights)
p_s
p_t
Estimated depth
$c_i(f_{i\to j}(x))$
$c_{i\to j}(x)$
$z_{i\to j}(x)$
Disparity loss
$z_j(f_{i\to j}(x))$
$f_{i\to j}(x)$
Spatial loss
$p_{i\to j}(x)$
x
(R_j, t_j)
(R_i, t_i)

approach

- Let x be a 2D pixel coordinate in frame i. The flow-displaced point

$$f_{i\to j}(x) = x + F_{i\to j}(x).$$

$$c_i(x) = D_i(x) K_i^{-1} \tilde{x},$$

$$c_{i\to j}(x) = R_j^{T}\left(R_i c_i(x) + \tilde{t}_i - \tilde{t}_j\right),$$

$$p_{i\to j}(x) = \pi\left(K_j c_{i\to j}(x)\right),$$

$$\mathcal{L}^{disparity}_{i\to j}(x) = u_i \left| z^{-1}_{i\to j}(x) - z_j^{-1}\left(f_{i\to j}(x)\right)\right|,$$

$$\mathcal{L}^{spatial}_{i\to j}(x) = \left\| p_{i\to j}(x) - f_{i\to j}(x)\right\|_2,$$

- (1) the TUM dataset (2) the ScanNet dataset (3) the KITTI 2015 datasets
- Evaluation metrics.

	Static			Dynamic	
	E_s (%) ↓	E_d (%) ↓	E_p ↓	E_s (%) ↓	E_p ↓
WSVD [2019a]	4.13	19.12	11.90	4.10	17.46
NeuralRGBD [2019]	1.86	15.25	11.33	1.30	18.62
Mannequin [2019]	3.88	13.22	12.05	2.38	18.16
MiDaS-v2 [2019]	3.14	10.14	11.74	2.83	15.76
COLMAP [2016]	1.02	6.19	-	1.47	-
Ours	**0.44**	**2.12**	**10.09**	**0.40**	**14.44**

Challenges and limitations

- Colmap : to estimate the camera pose from a monocular video
- Dynamic motion : the method supports videos containing moderate object motion. It breaks for extreme object motion.
- Speed : As they extract geometric constraints using all the frames in a video, they do not support online processing.

Our goal

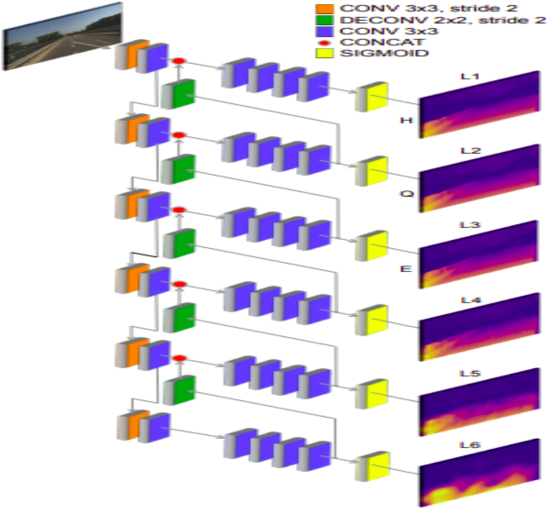
- reduce the time
- without reducing the accuracy
- without effecting on Consistent video depth estimation

Enhancement

- change depth estimation from a single color image model
- networks (PyDNet, DSNet and FastDepth potentially fulfil these requirements
- PyDNet : compact CNN, enabling accuracy comparable to state-of-the-art, with very limited memory footprint at test time (i.e., ¡ 150 MB) -
- this model runs in real-time on standard CPUs .

Enhancement

- PyD-Net architecture.



Enhancement

- implemented PyD-Net in TensorFlow and for experiments , deployed a pyramid with 6 levels
- train the network for 50 epochs on batches of 8 images resized to 512 256,30 thousand images from KITTI raw data
- provide results training PyD-Net for 200 epochs
- training on CityScapes followed by fine-tuning on KITTI

Enhancement

		Lower is better				Higher is better		
Method	Training dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen et al. [4]	K	0.203^{5}	1.548^{4}	6.307^{4}	0.282^{5}	0.702^{4}	0.890^{5}	0.958^{5}
Liu et al. [5]	K	0.201^{4}	1.584^{5}	6.471^{5}	0.273^{4}	0.680^{5}	0.898^{4}	0.967^{1}
Zhou et al. [6]	K	0.208^{6}	1.768^{6}	6.856^{6}	0.283^{6}	0.678^{6}	0.885^{6}	0.957^{6}
Godard et al. [2]	K	0.148^{1}	1.344^{1}	5.927^{1}	0.247^{1}	0.803^{1}	0.922^{1}	0.964^{2}
PyD-Net (50)	K	0.163^{3}	1.399^{3}	6.253^{3}	0.262^{3}	0.759^{3}	0.911^{3}	0.961^{4}
PyD-Net (200)	K	0.153^{2}	1.363^{2}	6.030^{2}	0.252^{2}	0.789^{2}	0.918^{2}	0.963^{3}
Garg et al. [19] cap 50m	K	0.169^{4}	1.080^{4}	5.104^{4}	0.273^{4}	0.740^{4}	0.904^{4}	0.962^{4}
Godard et al. [2] cap 50m	K	0.140^{1}	0.976^{1}	4.471^{1}	0.232^{1}	0.818^{1}	0.931^{2}	0.969^{2}
PyD-Net (50) cap 50m	K	0.155^{3}	1.045^{3}	4.776^{3}	0.247^{3}	0.774^{3}	0.921^{3}	0.967^{3}
PyD-Net (200) cap 50m	K	0.145^{2}	1.014^{2}	4.608^{2}	0.227^{2}	0.813^{2}	0.934^{1}	0.972^{1}
Zhou et al. [6]	CS+K	0.198^{4}	1.836^{4}	6.565^{4}	0.275^{4}	0.718^{4}	0.901^{4}	0.960^{4}
Godard et al. [2]	CS+K	0.124^{1}	1.076^{1}	5.311^{1}	0.219^{1}	0.847^{1}	0.942^{1}	0.973^{1}
PyD-Net (50)	CS+K	0.148^{3}	1.316^{3}	5.929^{3}	0.244^{2}	0.800^{3}	0.925^{3}	0.967^{2}
PyD-Net (200)	CS+K	0.146^{2}	1.291^{2}	5.907^{2}	0.245^{3}	0.801^{2}	0.926^{2}	0.967^{2}

current progress

future plan

- train and apply PyD-Net architecture.(January)
- New Results and compare it
- Publication

- Thank you.