007-定位-Towards Real-Time Unsupervised Monocular Depth Estimation on CPU

序号：007
名称：Towards Real-Time Unsupervised Monocular Depth Estimation on CPU
在CPU上进行实时无监督单目深度估计
作者：Matteo Poggi1, Filippo Aleotti2, Fabio Tosi1, Stefano Mattoccia1
文献类型：IROS2018 会议
年份：2018
关键词：单目相机、深度估计、CPU
源码网站：https: //github.com/mattpoggi/pydnet
整理日期：2019年5月15日
论文链接：007-Towards Real-Time Unsupervised Monocular Depth Estimation on CPU.pdf

主要解决问题

单个图像的无监督深度估计是一种非常有吸引力的技术，在机器人，自主导航，增强现实等方面具有多种意义。深度学习可以解决这个问题，但是网络架构非常复杂。 因此，仅通过利用耗电量大的GPU可以实现实时性能，所述GPU不允许在以低功率约束为特征的应用领域中推断深度图。
在本文中，我们提出了一种新颖的架构，能够使用从单个输入图像中提取的特征金字塔，在CPU甚至是嵌入式系统上快速推断出精确的深度图。
目前是第一个提出在CPU上进行无监督单目深度估计的方法。

解决思路

提出一个compact CNN（参数数量少）框架结构，使得每次的内存使用小于150MB，且在例如Raspberry Pi 3的嵌入式系统上能以2fps得到深度图，在标准CPU上以几十fps得到深度图。

方法特点：参数更少、内存占用量更小、运行时间更少。

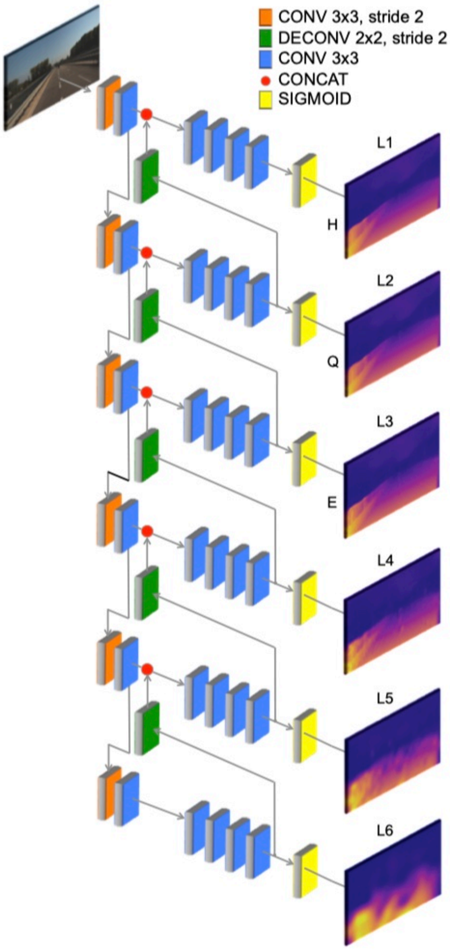


Fig. 2: PyD-Net architecture. A pyramid of features is extracted from the input image and at each level a shallow network infers depth at that resolution. Processed features are then up-sampled to the above level to refine estimation, up to the highest one.

将深度预测看做出图像重建问题，使用无监督学习方法来训练PyD-Net（本文提出的网络）。
训练未标记的点对是必要的:对于每个样本，通过神经网络对左侧的图像帧进行处理，得到反深度图(即，视差图)相对于左、右图像。利用这些映射将两幅输入图像相互扭曲，并利用重构误差作为反向传播的监控信号。

核心知识点

1、卷积神经网络CNN
https://blog.csdn.net/jiaoyangwm/article/details/80011656

程序功能分块说明

1、特征金字塔的提取
输入特征的提取由12个卷积层组成的小型编码架构完成。每两层一组，第一层步长为2，第二层步长为1，用3 *3核函数进行卷积，激活函数ReLU中的α=0.2。总共6组，从L1~L6，每组的图像分辨率分别从原图的1/2到1/64，每组提取的特征数为16，32，64，96，128，192。

2、深度解码和下采样
L6层使用4层卷积层的深度解码器来提取特征，分别提取到96，64，32和8个特征地图。这一层的输出主要有两个目的：

- 提取当前分辨率的深度图像，使用sigmoid函数
- 进行2*2且步长为2的反向卷积，并连接到上一层

其他层类似操作：反卷积---连接到上一层

3、训练loss函数

We train PyD-Net to estimate depth at each resolution deploying a multi-scale loss function as sum of different contributions computed at scales $s \in [1..6]$

$$\mathcal{L}_s = \alpha_{ap}(\mathcal{L}^l_{ap} + \mathcal{L}^r_{ap}) + \alpha_{ds}(\mathcal{L}^l_{ds} + \mathcal{L}^r_{ds}) + \alpha_{lr}(\mathcal{L}^l_{lr} + \mathcal{L}^r_{lr}) \quad (1)$$

The loss signal computed at each level of the pyramid is a weighted sum of three contributions computed on left and right images and predictions as in [2]. The first term represents the reconstruction error $\mathcal{L}_{ap}$, measuring the difference between the original image I^l and the warped one $\tilde{I}^l$ by means of SSIM [28] and L1 difference.

$$\mathcal{L}^l_{ap} = \frac{1}{N}\sum_{i,j} \alpha \frac{1 - SSIM(I^l_{i,j}, \tilde{I}^l_{i,j})}{2} + (1-\alpha)||(I^l_{i,j}, \tilde{I}^l_{i,j})|| \quad (2)$$

The disparity smoothness term, $\mathcal{L}_{ds}$, discourages depth discontinuities according to L1 penalty unless a gradient δI occurs on the image.

$$\mathcal{L}^l_{ds} = \frac{1}{N}\sum_{i,j} |\delta_x d^l_{i,j}| e^{-||\delta_x I^l_{i,j}||} + |\delta_y d_{i,j}| e^{-||\delta_y I^l_{ij}||} \quad (3)$$

The third and last term includes the left-right consistency check, a well-known cue from traditional stereo algorithms [9], enforcing coherence between predicted left d^l and right d^r depth maps.

$$\mathcal{L}^l_{lr} = \frac{1}{N}\sum_{i,j} |d^l_{i,j} - d^r_{i,j+d^l_{i,j}}| \quad (4)$$

The three terms are also computed for right image predictions, as shown in Equation 1. As in [2], the right input image and predicted output are used only at training time, while at testing time our framework works as a monocular depth estimator.

[2] requires 20 hours for 50 epochs. The weights for our loss terms are always set to $\alpha_{ap} = 1$ and $\alpha_{lr} = 1$, while left-right consistency weight is set to $\alpha_{ds} = 0.1/r$, being r the down-sampling factor at each resolution layer as suggested in [2]. The inferred maps are multiplied by $0.3\times$ image width,

具体实验结果

将3D点云投影到2D图后进行实验，得到的结果与原始深度信息比较。

		Lower is better				Higher is better			
Method	Training dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Params.
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	54.2M
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	40.0M
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	34.2M
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	31.6M
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	1.9M
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	1.9M
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	16.8M
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	

TABLE I: Evaluation on KITTI [1] using the split of Eigen et al. [4]. For training, K refers to KITTI dataset, CS+K means training on CityScapes [27] followed by fine-tuning on KITTI as outlined in [2]. On top and middle of the table evaluation of all existing methods trained on K, at the bottom evaluation of unsupervised methods trained on CS+K. We report results for PyD-Net with two training configurations.

	Power	250+ [W]	91+ [W]	3.5 [W]
Model	Res.	Titan X	i7-6700K	Raspberry Pi 3
Godard et al. [2]	F	0.035 s	0.67 s	10.21 s
Godard et al. [2]	H	0.030 s	0.59 s	8.14 s
PyD-Net	H	0.020 s	0.12 s	1.72 s
Godard et al. [2]	Q	0.028 s	0.54 s	6.72 s
PyD-Net	Q	0.011 s	0.05 s	0.82 s
Godard et al. [2]	E	0.027 s	0.47 s	5.23 s
PyD-Net	E	0.008 s	0.03 s	0.45 s

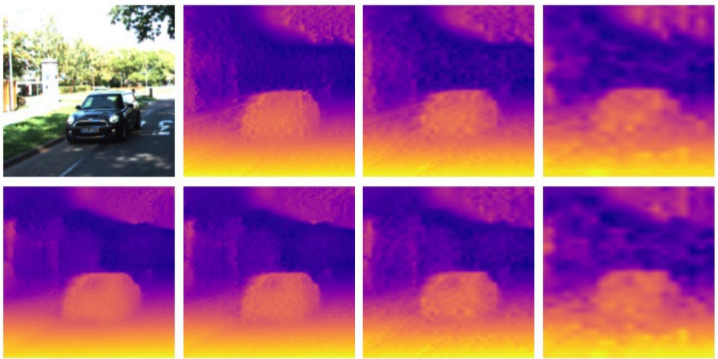
TABLE II: Runtime analysis. We report for PyD-Net and [2] the average runtime required to process the same KITTI image with 3 heterogeneous architectures at Full, Half, Quarter and Eight resolution. The measured power consumption for the Raspberry Pi 3 concerns the whole system plus a Logitech HD C310 USB camera while for CPU and GPU it concerns only such devices.

We report single forward time at full(F), half(H), quarter(Q) and eight(E) resolution.

		Lower is better				Higher is better		
Method	Res.	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 0.125$	$\delta < 0.125^2$	$\delta < 0.125^3$
Godard et al. [2]	F	0.124	1.076	5.311	0.219	0.847	0.942	0.973
Godard et al. [2]	H	0.126	1.051	5.347	0.222	0.843	0.940	0.972
PyD-Net (50)	H	0.148	1.316	5.929	0.244	0.800	0.925	0.967
PyD-Net (200)	H	0.146	1.291	5.907	0.245	0.801	0.926	0.967
Godard et al. [2]	Q	0.132	1.091	5.632	0.231	0.830	0.935	0.970
PyD-Net (50)	Q	0.152	1.342	6.185	0.252	0.789	0.920	0.964
PyD-Net (200)	Q	0.148	1.285	6.146	0.252	0.787	0.919	0.965
Godard et al. [2]	E	0.160	1.601	7.121	0.270	0.773	0.909	0.958
PyD-Net (50)	E	0.169	1.659	7.161	0.280	0.751	0.901	0.954
PyD-Net (200)	E	0.167	1.643	7.222	0.282	0.747	0.898	0.953

TABLE III: Comparison between [2] and PyD-Net at different resolutions. All models were trained on CS+K datasets and results are not post-processed to achieve maximum speed. As for Table I, we report results for PyD-Net with two training configurations.

在精确度略有下降的前提下，CPU运行的时间还是符合实时要求的。



a) b) c) d)

Fig. 3: Qualitative comparison on a portion of a KITTI image between PyD-net (top) and Godard et al. [2] (bottom) respectively at F, H, Q and E resolution. Detailed timing analysis at each scale is reported in Table II.

存在的问题

改进的思路

将PyD-Net应用到专为计算机视觉应用而设计的嵌入式设备中，从而使得在硬低功耗约束的应用（如无人机、可穿戴和辅助系统等）中实现实时单目深度估计。