

# Learning to see in 3D



Telmo Pessoa Pires

telmo@apple.com

 @thetruetelmo

# Disclaimer

Opinions here are my own, not my employer's.

Simplifications and omissions were made: this is an extensive topic.

Any mistakes are mine.

# Overview

1. Introduction
2. Data sources
3. Models
4. Moving objects (the most interesting part!)
5. Main Takeaways

# About me

- Senior Machine Learning Researcher/Engineer @ Apple
- Before:
  - AI Research Resident @ Google
  - AI Researcher @ Unbabel
  - Researcher @ IT
  - Student @ IST

# The problem

There are many variants:

- Stereo depth estimation (multi image)
- Monocular depth estimation (single image)
- Infilling depth from other sources (LiDAR, parallax)

In this talk, I will focus on monocular depth estimation.

# Why is this useful

- Augmented reality



Image from [1].

# Why is this useful

- Augmented reality
- 3D reconstruction

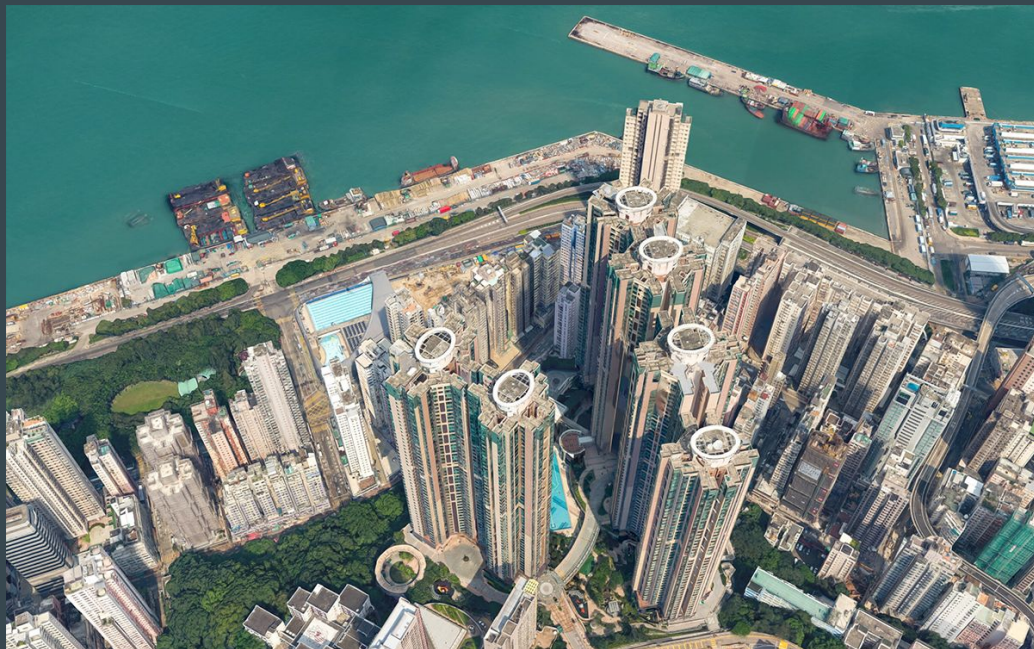


Image from [2].

# Why is this useful

- Augmented reality
- 3D reconstruction
- Visual effects: object insertion, anaglyphs, video inpainting.



Image from [3].



# Why is this useful

- Augmented reality
- 3D reconstruction
- Visual effects: object insertion, anaglyphs, video inpainting.
- View synthesis (e.g. Ken Burns effect)



Image from [4].

# Monocular depth

There are plenty of classical techniques for depth estimation, particularly in the multi-image scenario.

Can we teach a neural network to capture cues in the scene for monocular depth estimation?

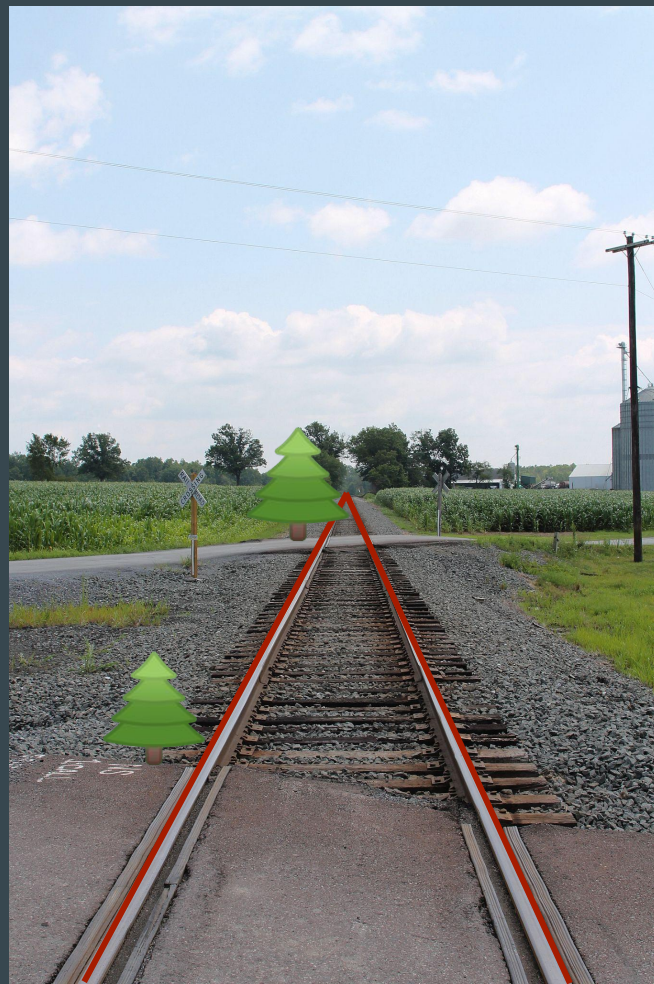


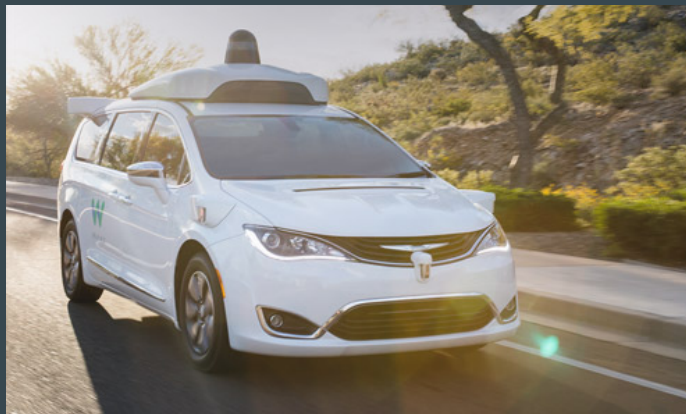
Image from [5].

# Data

# Lidar, sonar, radar

Special hardware that estimates depth by measuring the round trip time of a signal.

Some public datasets: waymo, kitty, etc

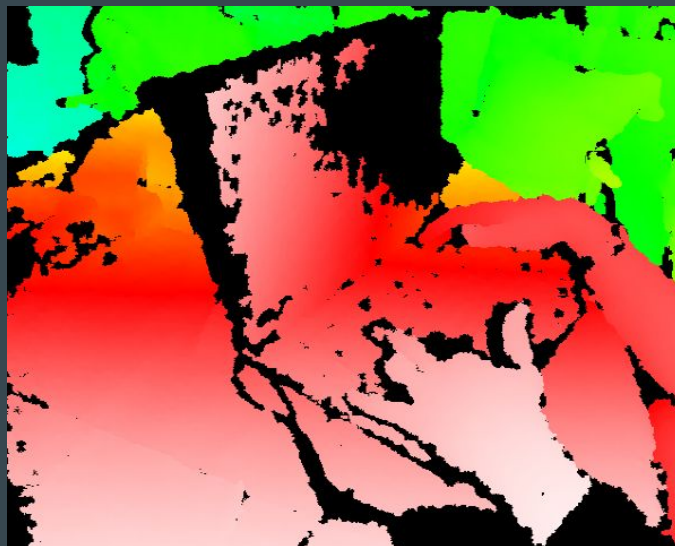


Images from [6].

# Kinect and other depth cameras

Some (small) publicly available datasets: TUM.

Cons: time consuming, expensive hardware.



Images from [7].



# Stereo data



Images from [8].

Distance between cameras

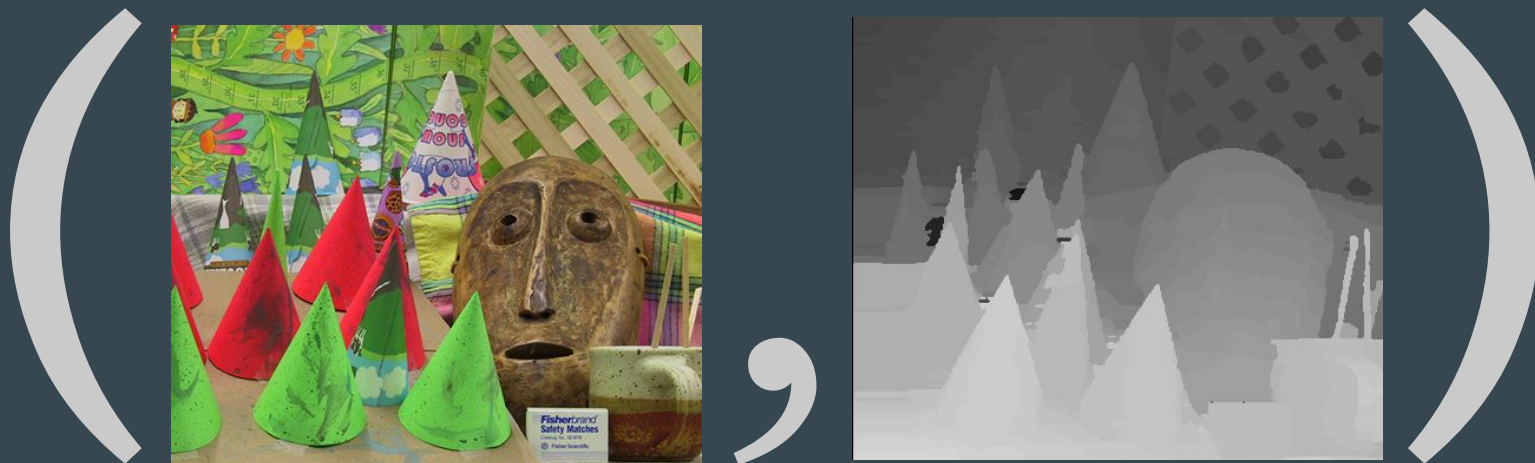
$$Z = \frac{d}{\Delta x} f$$

Focal length



# Training

Using any of the previous techniques, you should get a dataset like:

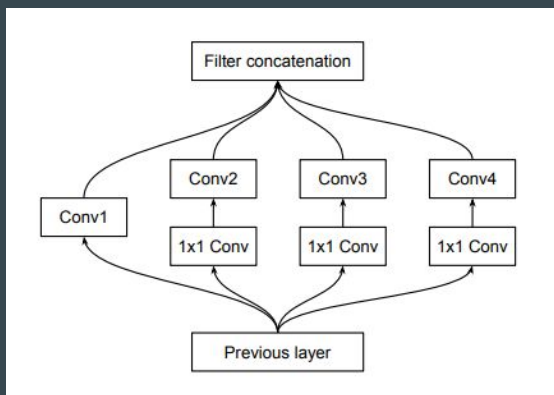
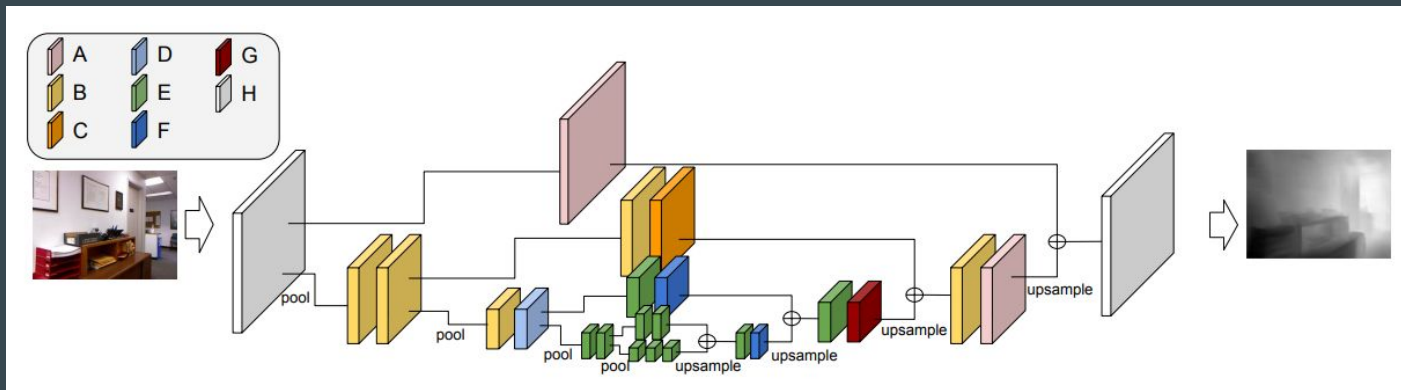


Images from [8].

# Models



# Hourglass network



Block Id	A	B	C	D	E	F	G
#In/#Out	128/64	128/128	128/128	128/256	256/256	256/256	256/128
Inter Dim	64	32	64	32	32	64	32
Conv1	1x1	1x1	1x1	1x1	1x1	1x1	1x1
Conv2	3x3	3x3	3x3	3x3	3x3	3x3	3x3
Conv3	7x7	5x5	7x7	5x5	5x5	7x7	5x5
Conv4	11x11	7x7	11x11	7x7	7x7	11x11	7x7

Images from [9].

# L2 Loss

Straightforward regression problem, right? Not exactly...

A naive loss function would be something like:

$$error = (\hat{y} - y)^2$$

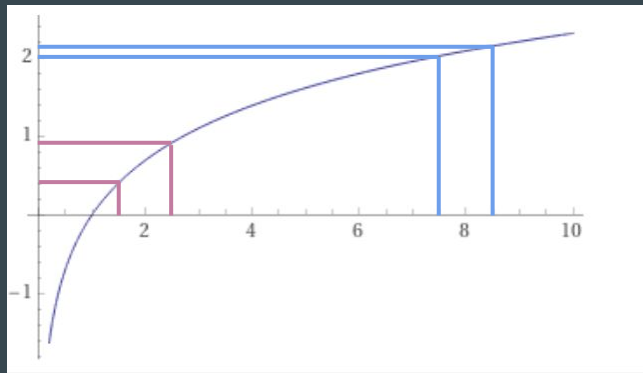
Not all errors are alike: the distance to the camera matters. A 1 m error can be big or small depending on how far the object is!

# Log L2 Loss

Solution? Work in the log domain!

$$error = (\log \hat{y} - \log y)^2$$

When  $y$  is large, (small) errors matter less!



# DORN Network

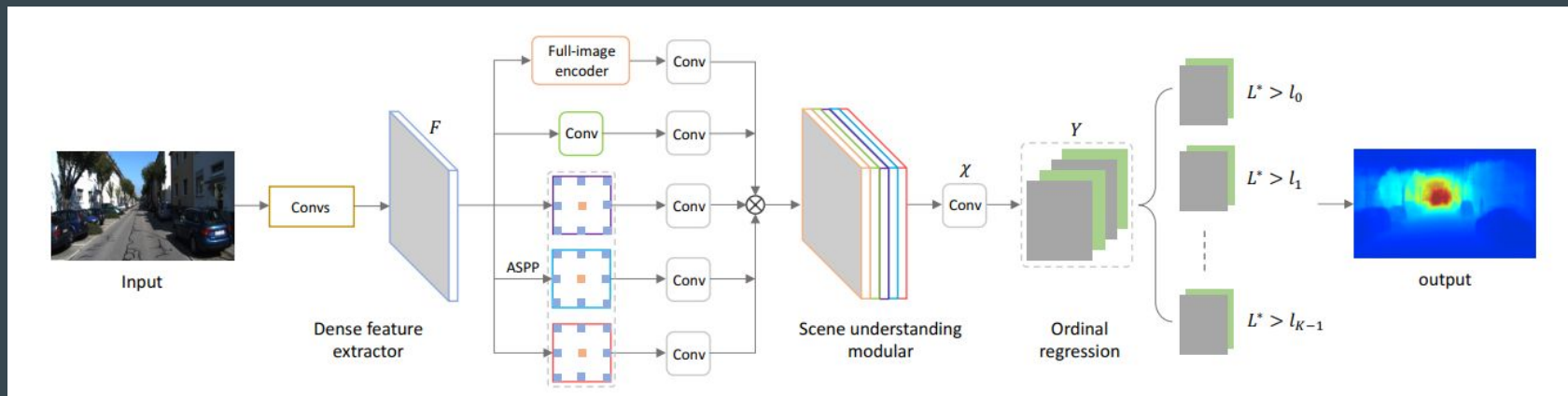


Image from [10].

# Classification with DORN

In general, neural networks tend to work better for classification, and depth estimation is no exception.

When discretizing depth into buckets, it is a good idea to do it in the log domain:

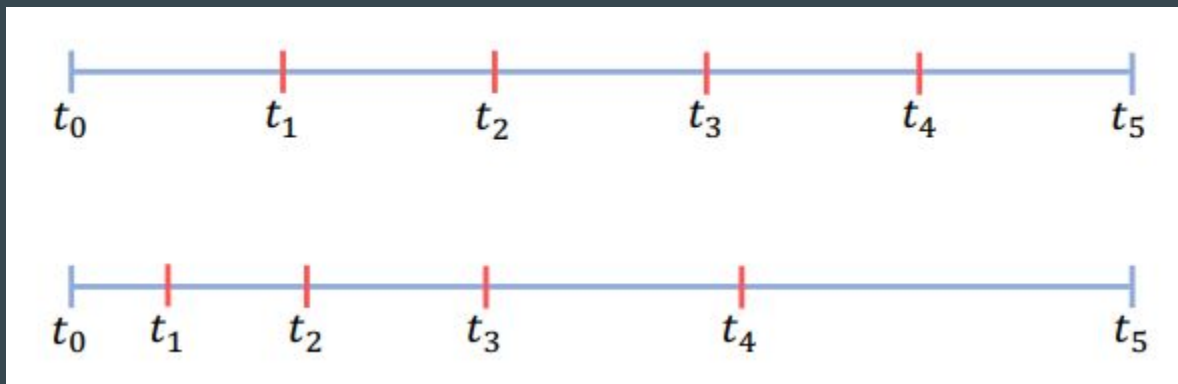


Image from [10].

# Ordinal Regression

In regular classification, the classes are independent of each other.

This is not always true: sometimes there is intrinsic order between classes.

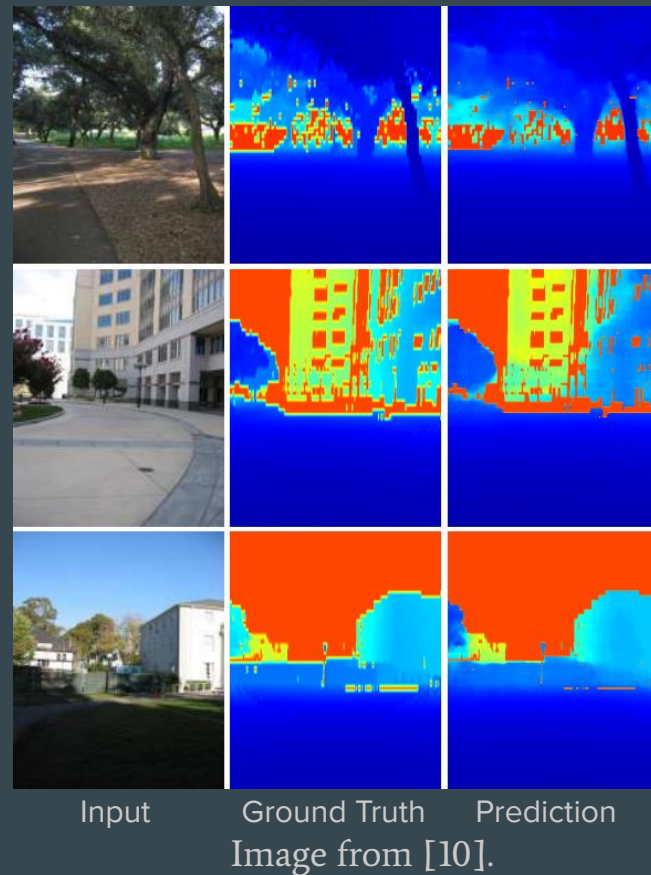
Intuition: it does not make sense for a model's top 2 predictions to be [25-30]m and [0-1]m.

Ordinal regression fixes this: the model will not make a confident prediction of a class (say, [25-30]m) if it is not confident of the lower classes.

# Some results



Image from [9].



# Moving objects



# Why is it hard?

Data is hard to come by.

- Lidar driving data has people, but it's mostly outdoors.
- Stereo techniques don't work.
- Depth cameras are expensive.

# How do we see?

We have two eyes: our brain composes a 3D image from two images.

Also, visual cues (vanishing lines) and motion parallax.

We can think of it as triangulation.

It also works with a single (but moving) camera!

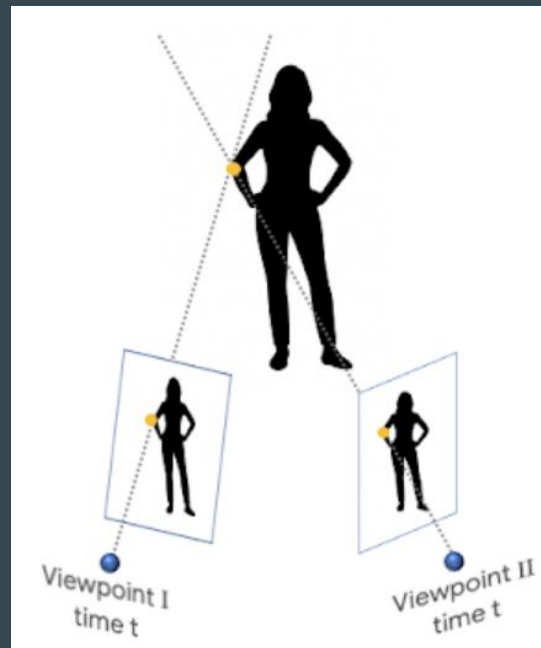


Image from [3].

# Handling moving objects

Triangulation between different timesteps doesn't work!

So since stereo methods are out of question, we need to use other datasets.

Or are they?

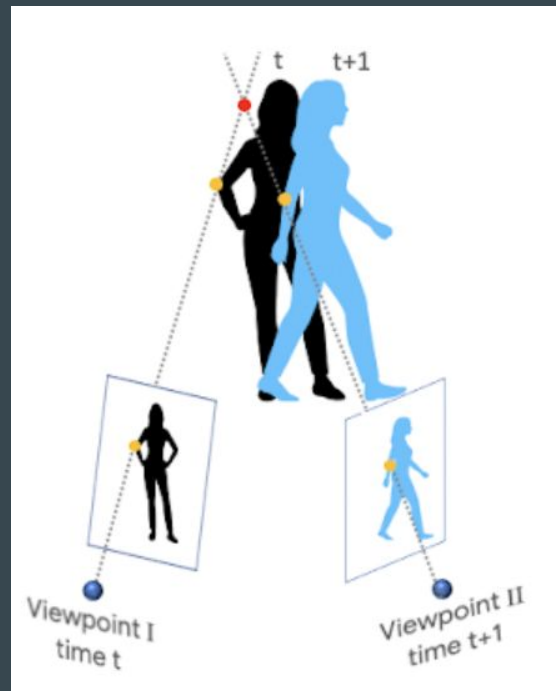


Image from [3].

# The Mannequin Challenge



**Idea:** use classical CV techniques to estimate ground truth depth to train neural network!

# Step 1: Compute camera positions and orientations

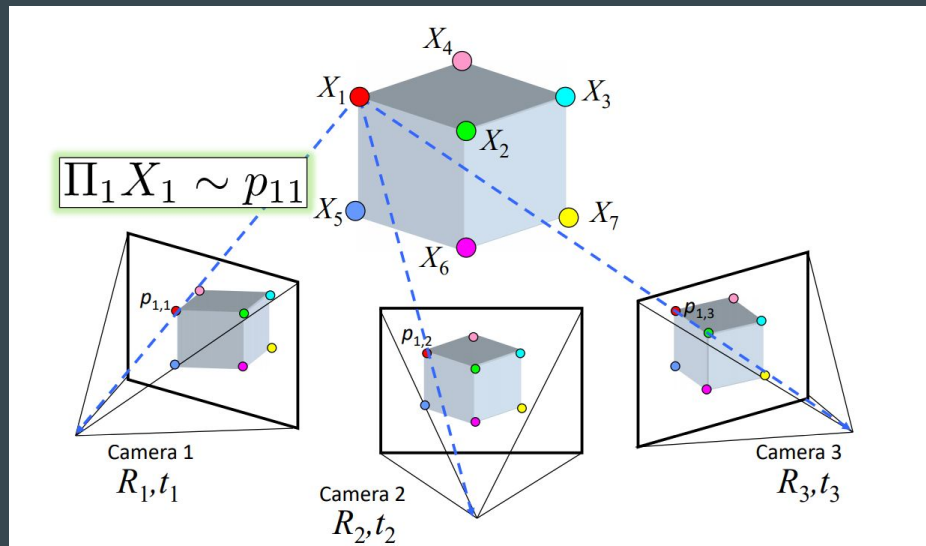


Image from [12].

If we know the positions of the camera, we can compute the 3D position of the points of interest.

If we know the positions (in space and in the images) of the points of interest, we can compute the positions of the camera.

Solution? Iterate.

# Structure from Motion

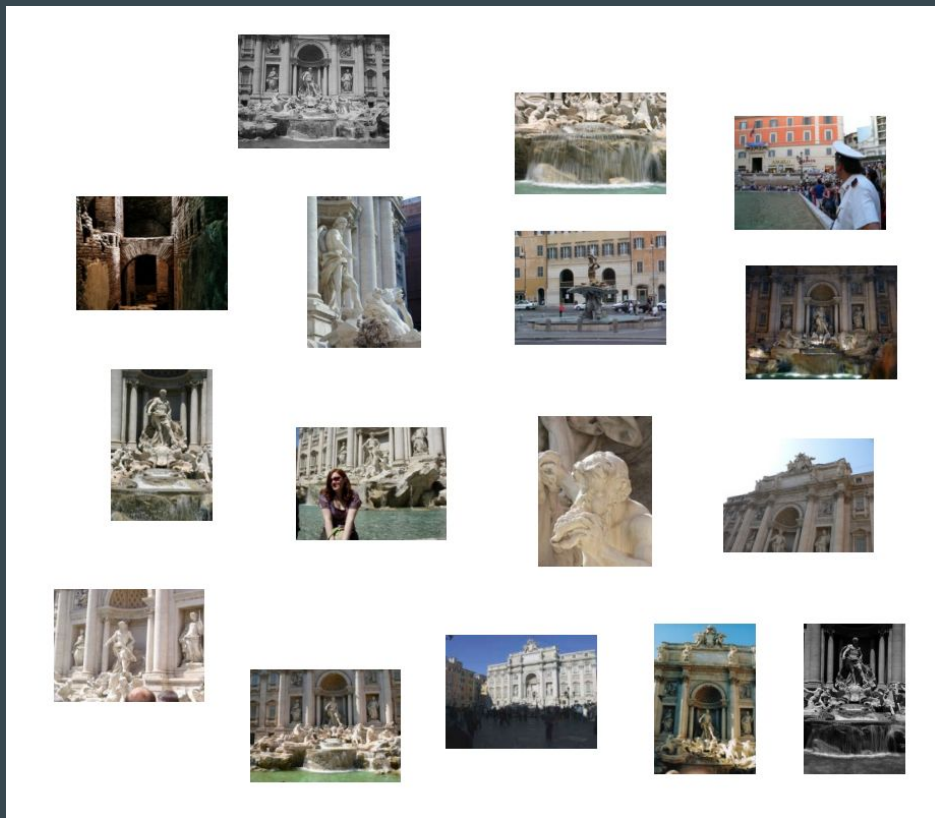


Image from [12].

# Structure from Motion



Image from [12].

# Structure from Motion

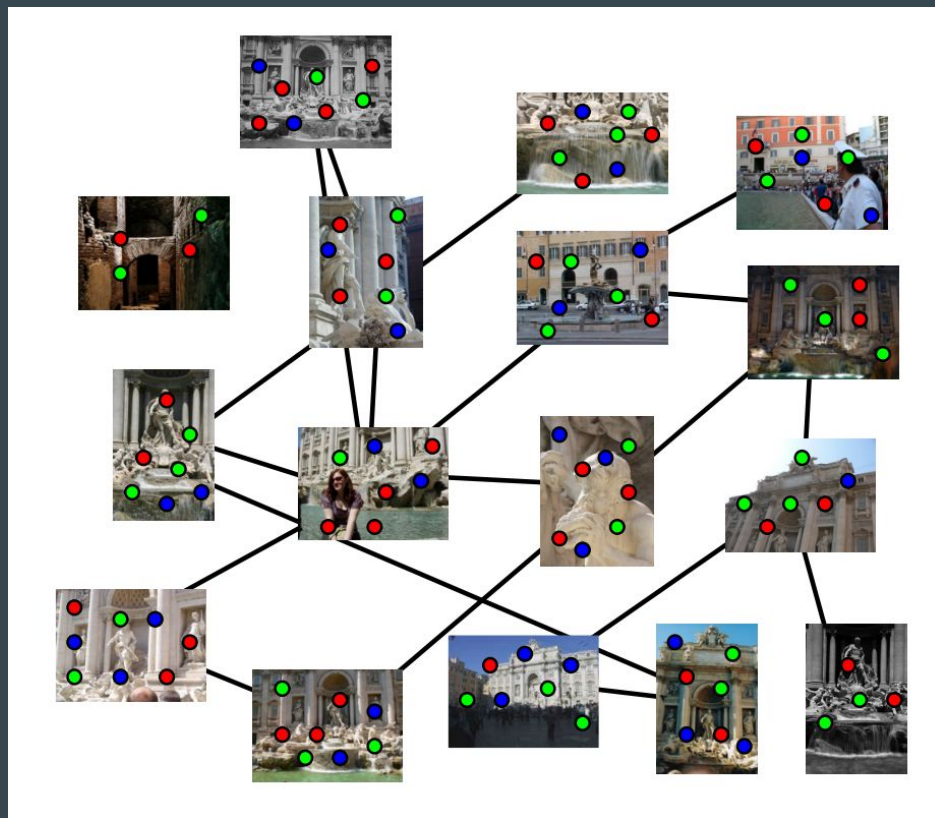


Image from [12].



# Structure from Motion

Orientation and 3D coordinates of camera  $j$

3D coordinates of point  $i$

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^m \sum_{j=1}^n \underbrace{w_{ij}}_{\substack{\text{indicator variable:} \\ \text{is point } i \text{ visible in image } j?}} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)}_{\substack{\text{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\text{observed} \\ \text{image location}}} \right\|^2$$

Image from [12].

# One last detail



Image from [13].

The train on the right is miniature.

SfM can't distinguish this from real train (we need information about the cameras for that).

Since we can't fix this, the 3D coordinates we get are correct up to a scaling factor. (And so will be our final depths!)

## Step 2: Multi-view Stereo



Images from [8].

At this point we have dense depth estimates.

Distance between cameras

$$Z = \frac{d}{\Delta x} f$$

Focal length



## Step 3: Filtering and Cleaning

Due to motion blur, shadows, reflections, etc, MVS depths will not always be accurate:

1. Filter depths that are inconsistent with motion parallax depth
2. Filter sequences with high camera distortion
3. Filter short sequences
4. Manual filtering of point clouds

# Training data examples



Image adapted from [14].

# Loss explained

$$\mathcal{L} = \mathcal{L}_{MSE} + \lambda_1 \cdot \mathcal{L}_{grad} + \lambda_2 \cdot \mathcal{L}_{smoothness}$$

## Scale invariant loss

Forces the model to predict up to a scaling factor.

$$\mathcal{L}_{grad} = \frac{1}{n} \sum_k \sum_i (|\nabla_x R_i^k| + |\nabla_y R_i^k|)$$
$$R_i = \log \hat{y}_i - \log y_i$$

## (Scale invariant) Gradient matching loss

Encourages ground truth and prediction gradients to match: sharper boundaries.

$$\mathcal{L}_{smoothness} = \sum_i e^{-|\nabla^2 I_i|} |\nabla^2 \log \hat{y}_i|$$

## Edge-aware smoothness loss

Encourages smooth predictions in flat image regions, and does nothing near edges.

# Scale Invariant MSE Loss explained

Key insight:  $y_i = s \cdot \tilde{y}_i \Rightarrow \log y_i = \log s + \log \tilde{y}_i$

↓      ↓  
scale   dimensionless  
         depth

Same as regular log-domain loss

$$\mathcal{L}_{MSE} = \frac{1}{n} \sum_i \left( \left( \log \hat{y}_i - \frac{1}{n} \sum_j \log \hat{y}_j \right) - \left( \log y_i - \frac{1}{n} \sum_j \log y_j \right) \right)^2$$

Average (log) depth, cancels the scale

# Results

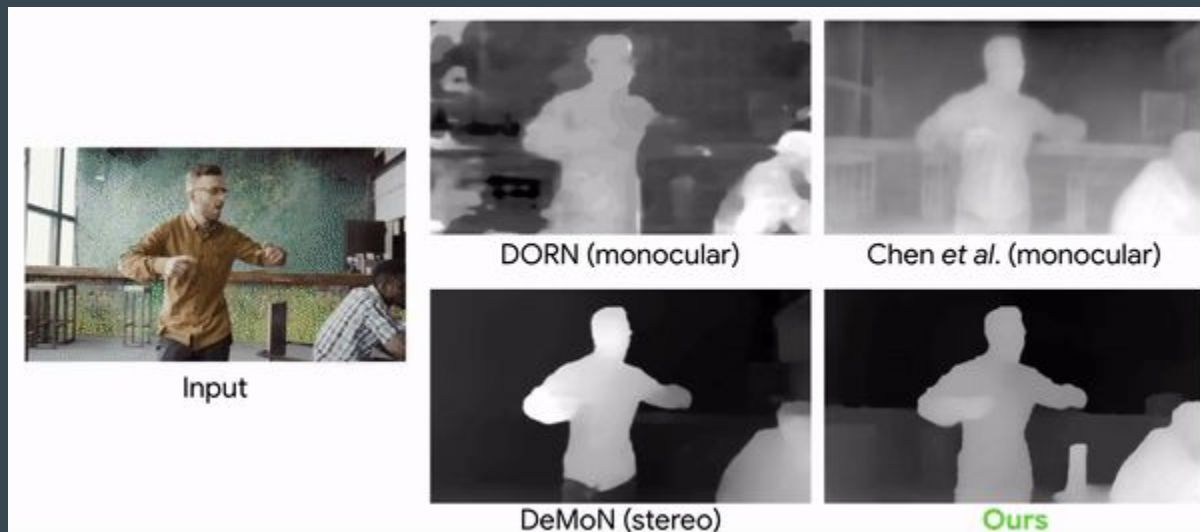


Image from [3].



# Results



Input Ground Truth Our Model +Parallax

Image adapted from [14].

# Some applications



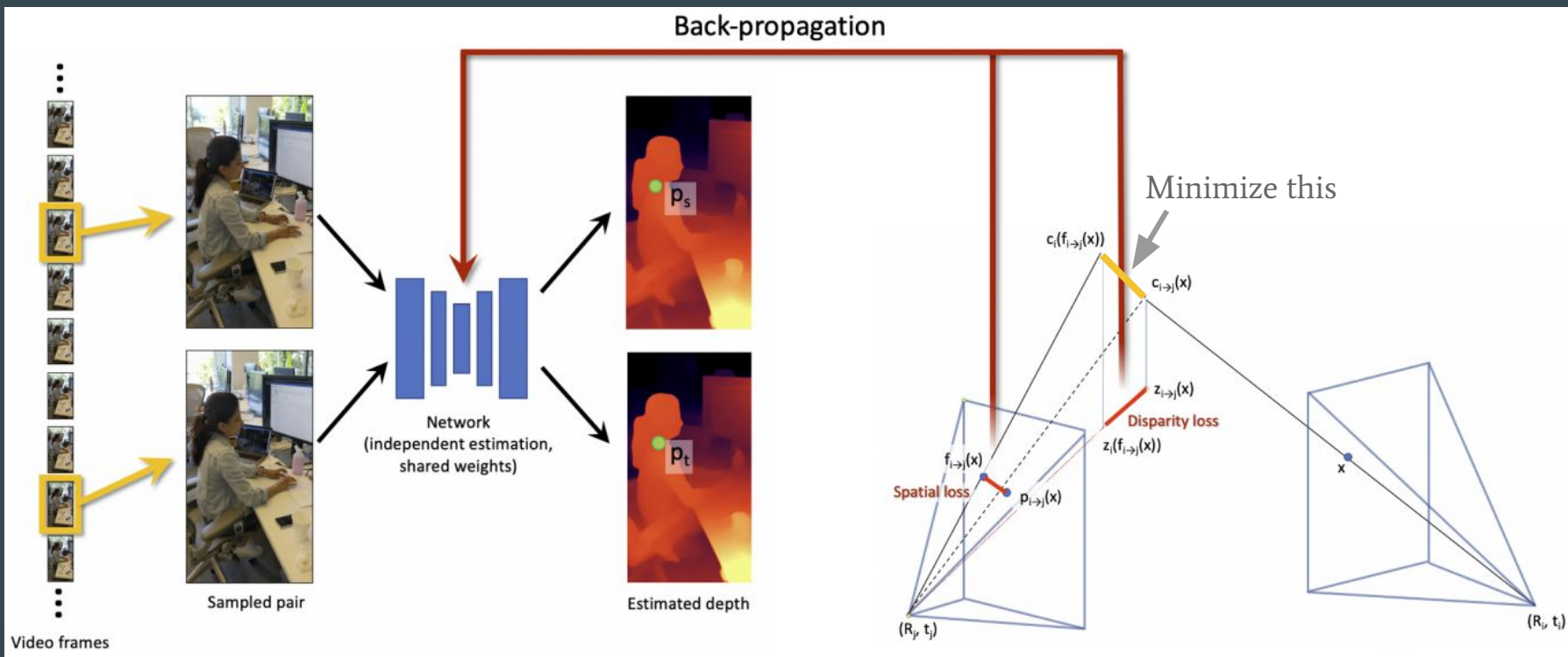
Image from [3].

# Some applications



Image from [3].

# Bonus: consistency loss





## Bonus: consistency loss

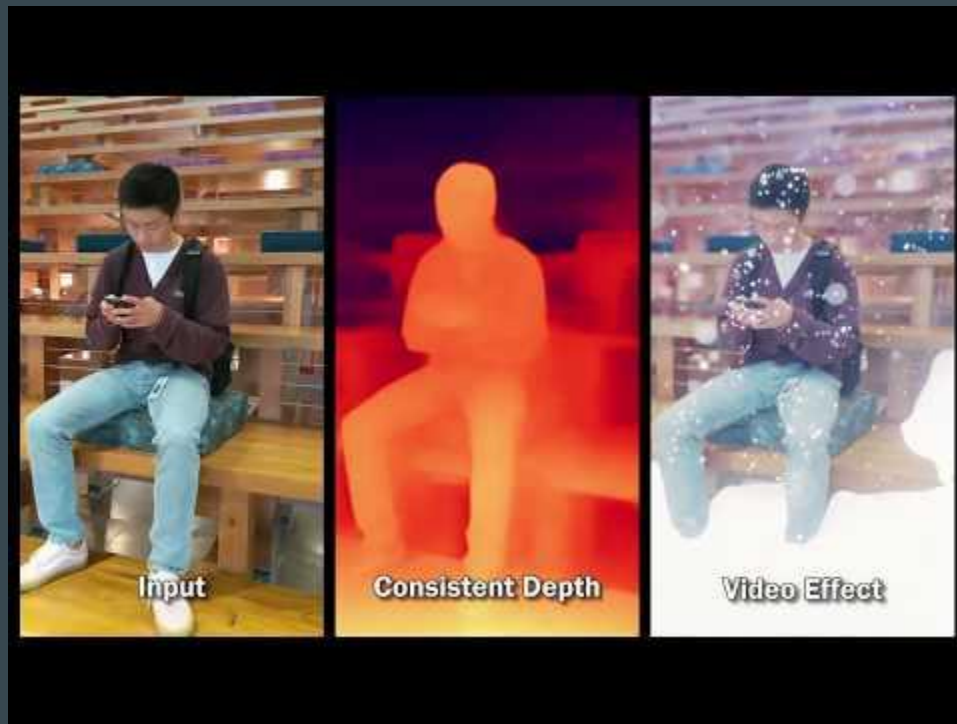


Image from [3].

# Main takeaways


Depth estimation is a useful and fun.

But, more importantly:

- Being creative about data can help you solve problems.
- Data cleaning is at least as important as modeling.
- Domain knowledge is important: tailored losses can get better results.

# Q&A

telmo@apple.com

 @thetruetelmo

# References

[1] IKEA AR App: <https://thespaces.com/ikea-place-app/>

[2] Google Earth's 3D Imagery:  
<https://blog.google/products/earth/google-earths-incredible-3d-imagery-explained/>

[3] Mannequin Challenge Google Blog:  
<https://ai.googleblog.com/2019/05/moving-camera-moving-people-deep.html>

[4] The Beetles Ken Burns effect:  
<https://waxy.org/2019/11/turning-photos-into-2-5d-parallax-animations-with-machine-learning>

[5] Vanishing Point Wikipedia: [https://en.wikipedia.org/wiki/Vanishing\\_point](https://en.wikipedia.org/wiki/Vanishing_point)

[6] Waymo Open Dataset: <https://waymo.com/open/data/perception/>

[7] Kinect Wikipedia: <https://en.wikipedia.org/wiki/Kinect>



# References

[8] “Segmentation-based adaptive support for accurate stereo correspondence”

[http://vision.deis.unibo.it/~smatt/stereo\\_segment\\_support.html](http://vision.deis.unibo.it/~smatt/stereo_segment_support.html)

[9] “Single-Image Depth Perception in the Wild”: <https://arxiv.org/abs/1604.03901>

[10] “Deep Ordinal Regression Network for Monocular Depth Estimation”:

<https://arxiv.org/abs/1806.02446>

[11] Portugal Mannequin Challenge: <https://www.youtube.com/watch?v=lSeGvnYU348>

[12] Structure from Motion Lecture Notes, Cornell U.

[https://www.cs.cornell.edu/courses/cs5670/2021sp/lectures/lec18\\_sfm\\_for\\_web.pdf](https://www.cs.cornell.edu/courses/cs5670/2021sp/lectures/lec18_sfm_for_web.pdf)

[13] Miniature train image:

<https://www.wallpaperflare.com/orange-black-and-gray-train-toy-miniature-trains-and-containers-toy-set-wallpaper-qry>

# References

- [14] “Learning the Depths of Moving People by Watching Frozen People”:  
[https://www.cs.cornell.edu/~zl548/images/Learning\\_the\\_Depths\\_of\\_Moving\\_People\\_by\\_Watching\\_Frozen\\_People\\_TPAMI.pdf](https://www.cs.cornell.edu/~zl548/images/Learning_the_Depths_of_Moving_People_by_Watching_Frozen_People_TPAMI.pdf)
- [15] Mannequin Model Video: [https://www.youtube.com/watch?v=fj\\_fK74y5\\_0](https://www.youtube.com/watch?v=fj_fK74y5_0)
- [16] “Consistent Video Depth Estimation”: <https://arxiv.org/abs/2004.15021>
- [17] Consistent Video Depth Estimation Video: <https://www.youtube.com/watch?v=5Tia2oblJAg>
- [18] “Deep Learning-Based Monocular Depth Estimation Methods-A State-of-the-Art Review”:  
<https://pubmed.ncbi.nlm.nih.gov/32316336/>
- [19] “uDepth: Real-time 3D Depth Sensing on the Pixel 4”:  
<https://ai.googleblog.com/2020/04/udepth-real-time-3d-depth-sensing-on.html>