Deep Convolutional Neural Networks for Camera Relocalization

Master's thesis defense - 13.12.2017

Maciej Marcin ŻURAD
`maciej.urad.001@student.uni.lu`

Supervisor: Prof. Dr.-Ing. Holger Voos
Reviewer: Prof. Dr. Christoph Schommer
Advisor: Dr. Miguel A. Olivares-Mendez

Faculty of Science, Technology and Communication
University of Luxembourg (FSTC)

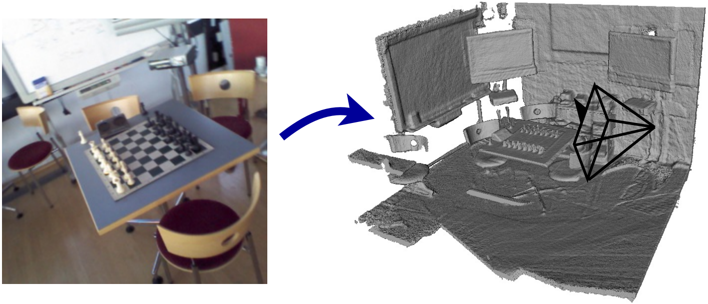
Outline

Introduction

Problem Statement

The Task

Camera relocalization, also known as image-based localization, is defined as the task of determining the location of a given image in an arbitrary coordinate frame.



Introduction

Problem Statement (continued)

Basic camera relocalization

Each image $\boldsymbol{x} \in I$ is processed independently in order to predict a 6-DOF pose $\boldsymbol{y} = (\boldsymbol{y}_{pos} \in \mathbb{R}^3, \boldsymbol{y}_{rot} \in SO(3)) = f(\boldsymbol{x})$.

Temporal camera relocalization

A sequence of N images $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}$ taken at a constant rate is processed jointly to predict: $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n)} = f(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)})$.

Introduction

Related Work

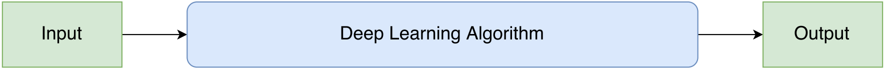
Approaches to camera relocalization

- **fiducial markers based localization**
  - markers usually not present in the environment
  - prone to blur, occlusion and illumination
- **sparse feature based localization**
  - rely on SIFT or ORB-like features
  - only work well in a controlled environment
  - do not scale with the spatial extent of the environment
  - computationally expensive
- **traditional machine learning methods**
- **deep learning methods**

Introduction

Machine Learning vs Deep Learning

Traditional Machine Learning pipeline



Deep Learning pipeline

Introduction

What is deep learning?

Deep Learning

- a class of Machine Learning methods
- focuses on representation learning
- employs deep neural networks (DNN)
- scales very well with the amount of training data
- successful on a variety of difficult problems

Introduction

Why deep learning for camera relocalization?

Deep Learning for Camera Relocalization

- end-to-end training without the need to hand-craft features
- constant space and time complexity at test-time
- camera intrinsics are not required
- robust to blur, occlusion, texture-less surfaces and varying lighting conditions

Outline

Deep Learning for Camera Relocalization

Camera Relocalization as Regression

Camera Relocalization as Regression

- Given a training dataset of image-pose pairs: $X_{train} = \{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}$
- We train a model $\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta})$, which regresses 6-DOF poses directly from image pixels
- The output prediction $\hat{\boldsymbol{y}} = (\hat{y}_{pos} \in \mathbb{R}^3,\ \hat{y}_{rot} \in \mathbb{R}^4)$ is composed of position and quaternion
- Models are composed of a pretrained CNN followed by a secondary regression model
- We use a multi-task loss function to learn position and attitude prediction at the same time

Deep Learning for Camera Relocalization

Related Work - Weighted loss function

Naive weighted loss, Kendall et al. [ICCV 2015]

$$\mathcal{L} = \mathcal{L}_{pos} + \beta \mathcal{L}_{rot}$$

$$\mathcal{L}_{pos} = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_p, \quad \mathcal{L}_{rot} = \left\| \boldsymbol{q} - \frac{\hat{\boldsymbol{q}}}{\|\hat{\boldsymbol{q}}\|} \right\|_p$$

- β is a hyper-parameter, which balances the importance between position and attitude loss
- Predicted quaternion $\hat{\boldsymbol{q}}$ is normalized to force it to a valid rotation in 3D space
- Searching for optimal β is expensive

Deep Learning for Camera Relocalization

Related Work - Homoscedastic uncertainty based loss function

Homoscedastic uncertainty loss, Kendall et al. [CVPR 2017]

- Homoscedastic uncertainty does not depend on the data and captures the uncertainty of the task itself.
- Loss for each task contains trainable parameter σ.

$$\mathcal{L}_\sigma = \mathcal{L}_{pos}\hat{\sigma}_{pos}^{-2} + \log\hat{\sigma}_{pos}^{2} \ + \ \mathcal{L}_{rot}\hat{\sigma}_{rot}^{-2} + \log\hat{\sigma}_{rot}^{2}$$

- For improved numerical stability, we learn $\hat{s} \leftarrow \log\hat{\sigma}^2$.

$$\mathcal{L}_\sigma = \mathcal{L}_{pos}\ e^{-\hat{s}_{pos}} + \hat{s}_{pos} \ + \ \mathcal{L}_{rot}\ e^{-\hat{s}_{rot}} + \hat{s}_{rot}$$

- We call this loss function Naive Homoscedastic (NH)

Deep Learning for Camera Relocalization

Proposed loss function - Quaternion error loss function

Quaternion error based loss, This work

- Difference between two rotation in 3D space is defined as: $\ominus : SO(3) \times SO(3) \mapsto \mathbb{R}^3$, which returns a vectorial difference $\boldsymbol{\theta} \in \mathbb{R}^3$ defined on quaternions as:

$$\boldsymbol{\theta} = \log\left(\boldsymbol{q}^{-1} \otimes \hat{\boldsymbol{q}}\right)$$

$$\log \boldsymbol{q} = \boldsymbol{q}_v \frac{\arctan(\|\boldsymbol{q}_v\|, q_w)}{\|\boldsymbol{q}_v\|} \approx \frac{\boldsymbol{q}_v}{q_w}\left(1 - \frac{\|\boldsymbol{q}_v\|^2}{3q_w^2}\right) \approx \boldsymbol{q}_v \xrightarrow[\theta \mapsto 0]{} 0$$

$$\mathcal{L}_{rot} = \|\log\left(\boldsymbol{q}^{-1} \otimes \hat{\boldsymbol{q}}\right)\|_p$$

- We combine this with homoscedastic uncertainty loss → Quaternion Error Homoscedastic (QEH)

Deep Learning for Camera Relocalization

Transfer Learning

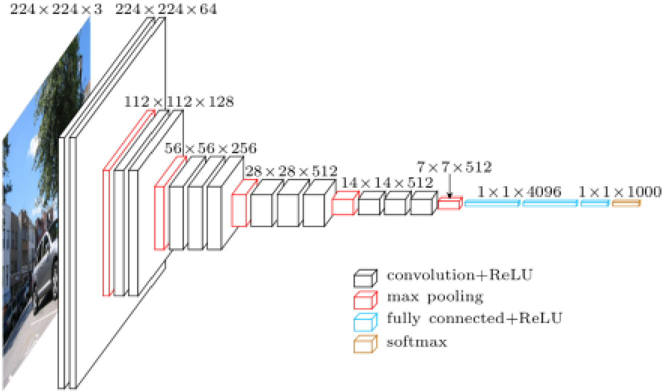
Transfer Learning

- We leverage Transfer Learning and compare performance on 4 different CNN models
- All FC layers and auxiliary branches are removed
- The output of a CNN is a feature vector passed in to a second model

	ImageNet	Places365	Hybrid1365	Non-frozen layers		Total params	Trainable params	Input size	Output size
GoogLeNet	×	×		last 3 Inception modules	$\sim 31.5\%$	~ 6M	$\sim$ 3.3M	$224 \times 224 \times 3$	1×1024
Inception ResNet V2	×			last 10 Inception ResNet blocks	$\sim 11.4\%$	$\sim$ 54.3M	$\sim$ 23.5M	$299 \times 299 \times 3$	1×1536
VGG16			×	last 3 Conv layers	$\sim 23\%$	$\sim$ 14.7M	$\sim$ 7.1M	$224 \times 224 \times 3$	1×512

Deep Learning for Camera Relocalization

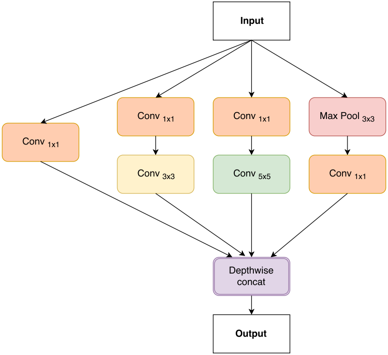
Transfer Learning - VGG16 architecture



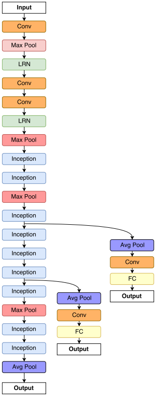
VGG16 architecture

Deep Learning for Camera Relocalization

Transfer Learning - GoogLeNet architecture



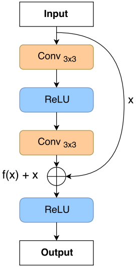
Inception module



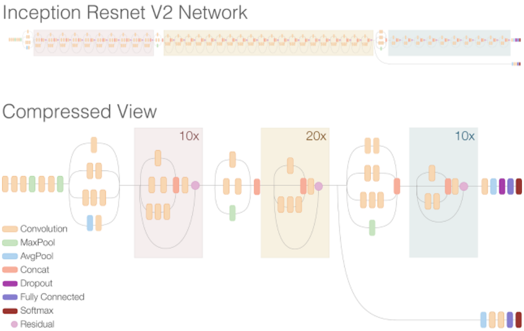
GoogLeNet architecture

Deep Learning for Camera Relocalization

Transfer Learning - Inception ResNet V2 architecture



Basic residual block



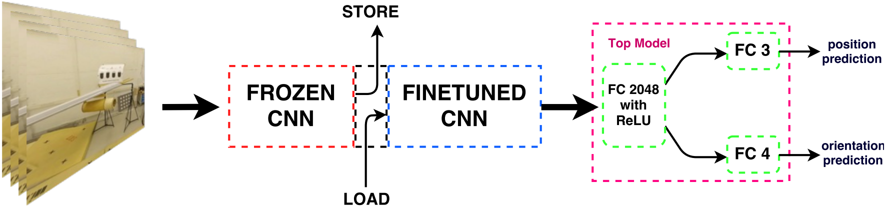
Inception ResNet V2 architecture

Deep Learning for Camera Relocalization

Regressor model

Regressor model

- Based on PoseNet, Kendall at al. [ICCV 2015]
- Images are fed through the frozen layers of the network and the output is stored
- Only the non-frozen layers from CNN are instantiated for training

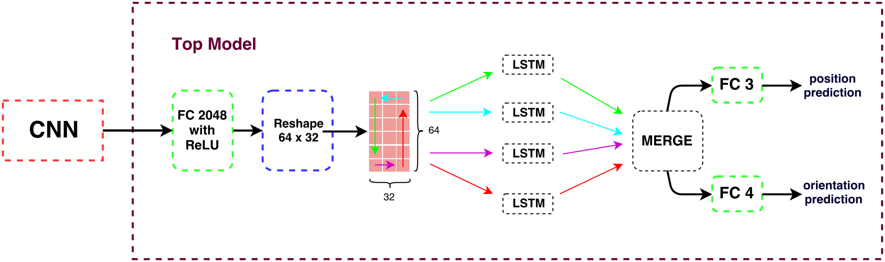


Deep Learning for Camera Relocalization

Spatial-LSTM model

Spatial-LSTM model

- Based on Walch et al. [ICCV 2017]
- Same as Regressor model, but intermediate spatial LSTMs are inserted for better structured feature correlation
- 4-way scanning of the reshaped feature vector

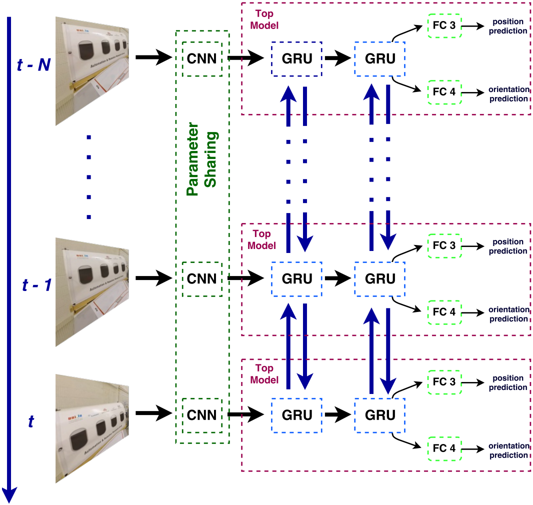


Deep Learning for Camera Relocalization

Temporal-GRU model

Temporal-GRU model

- Based on VidLoc, Clark et al. [CVPR 2017]
- Employs bidirectional GRUs instead of LSTMs
- Poses are jointly regressed from sequences of images



Outline

Introduction

Deep Learning for Camera Relocalization

Experimental evaluation

Conclusions

Experimental evaluation

Training methodology

Training methodology

- Implemented using Keras and TensorFlow[a]
- Trained on GeForce GTX 950 with 2GB of VRAM
- Random hyper-parameter search:
  - Adam optimizer with hyper-parameters: $\eta = 2 \times 10^{-4},\ \beta_1 = 0.9,\ \beta_2 = 0.999,\ \varepsilon = 1 \times 10^{-8}$
  - L2 regularization and Dropout set to 0
  - L1 norm used in loss functions
- Center-crops of images to fit the network input size

---

[a] Code available at `https://github.com/snt-robotics/camera_relocalization`

Experimental evaluation

Outline of experiments

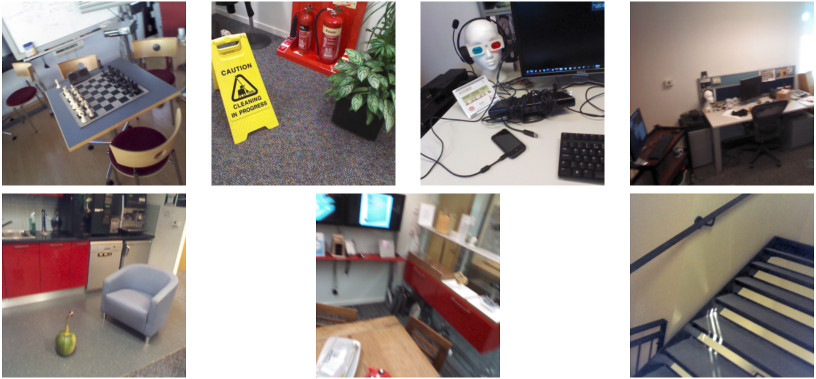
Outline of experiments

- 2 datasets: 7Scenes and **new** Airframe dataset
- 2 loss functions
  - Naive Homoscedastic (NH)
  - Quaternion Error Homoscedastic (QEH)
- 3 models: Regressor, Spatial-LSTM, Temporal-GRU
- 4 CNN models for feature extraction
  - GoogLeNet-ImageNet
  - GoogLeNet-Places365
  - InceptionResNetV2-ImageNet
  - VGG16-Hybrid1365

Experimental evaluation

7Scenes dataset

Example images from 7Scenes dataset



Chess **Fire** **Heads** **Office**

Pumpkin **Red kitchen** **Stairs**

Experimental evaluation

7Scenes dataset

7Scenes dataset summary

	# images		Spatial	# train
Scenes	Train	Test	Extent [m]	images per m^3
Chess	4000	2000	$3 \times 2 \times 1$	667
Fire	2000	2000	$2.5 \times 0.5 \times 1$	1600
Heads	1000	1000	$2 \times 0.5 \times 1$	1000
Office	6000	4000	$2.5 \times 2 \times 1.5$	800
Pumpkin	4000	2000	$2.5 \times 2 \times 1$	800
Red Kitchen	7000	5000	$4 \times 3 \times 1.5$	389
Stairs	2000	1000	$2.5 \times 2 \times 1.5$	267

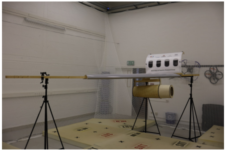
Experimental evaluation

Airframe dataset

Airframe dataset positions

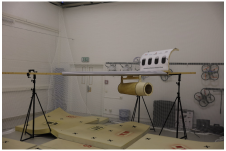
Position 1

$\boldsymbol{p} = (-0.405, 2.355, 1.621)$
$\boldsymbol{q} = (-0.010, 0.014, 0.870, -0.493)$



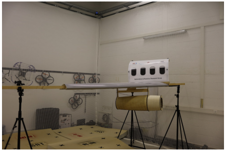
Position 2

$\boldsymbol{p} = (0.616, 2.980, 1.623)$
$\boldsymbol{q} = (0.013, -0.012, -0.707, 0.707)$



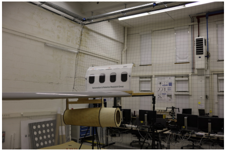
Position 3

$\boldsymbol{p} = (1.799, 1.165, 1.625)$
$\boldsymbol{q} = (-0.007, 0.022, 0.983, -0.182)$



Position 4

$\boldsymbol{p} = (0.801, -1.214, 1.625)$
$\boldsymbol{q} = (0.005, 0.017, 0.769, 0.639)$



Position 5

$\boldsymbol{p} = (0.798, -1.796, 1.628)$
$\boldsymbol{q} = (0.003, 0.018, 0.788, 0.615)$

Position 6

$\boldsymbol{p} = (2.262, -0.985, 1.629)$
$\boldsymbol{q} = (-0.005, 0.011, 0.972, 0.234)$

Experimental evaluation

7Scenes dataset

Airframe dataset summary

	airframe-mixed		**airframe-ind**	
	# images		# images	
	Train	Test	Train	Test
Position 1	3301	918	4219	0
Position 2	1451	1087	2538	0
Position 3	1733	809	2542	0
Position 4	1120	385	1505	0
Position 5	2002	670	2672	0
Position 6	3792	1628	0	5420
Sum	13399	5497	13476	5420
Total	18896			

Experimental evaluation

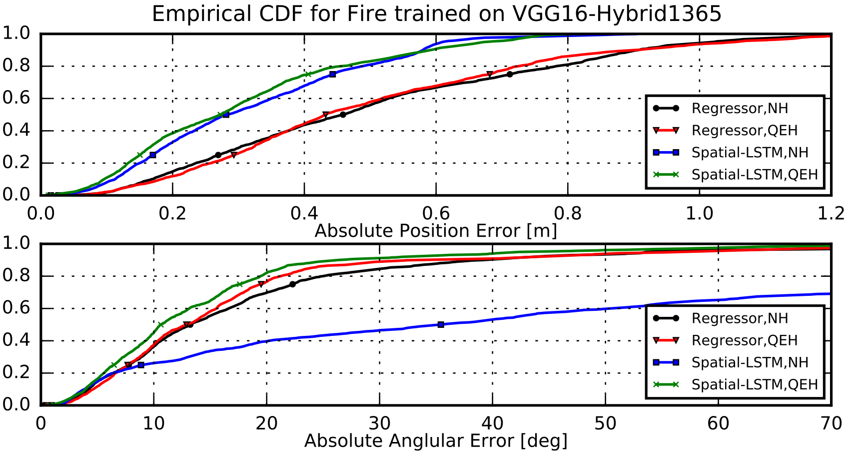
7Scenes results

Median performance on 7Scenes dataset

Data set	**Method**	**GoogLeNet-ImageNet**		**GoogLeNet-Places365**		**Inception-ResNet-V2**		**VGG16-Hybrid1365**	
		position	orientation	position	orientation	position	orientation	position	orientation
Chess	Spatial-LSTM,QEH	0.157m	6.95°	0.177m	6.41°	0.164m	7.36°	**0.148**m	**5.261**°
	Spatial-LSTM,NH	0.159m	9.85°	0.243m	8.21°	0.162m	72.07°	**0.137**m	**7.868**°
	Regressor,QEH	0.196m	7.21°	0.203m	6.53°	0.197m	7.78°	**0.188**m	**5.805**°
	Regressor,NH	**0.166**m	9.73°	0.183m	9.45°	0.209m	13.32°	0.197m	**8.157**°
Fire	Spatial-LSTM,QEH	0.325m	12.72°	0.331m	13.14°	0.344m	15.28°	**0.272**m	**10.62**°
	Spatial-LSTM,NH	0.342m	**15.49**°	0.346m	38.08°	0.333m	37.49°	**0.281**m	35.42°
	Regressor,QEH	**0.321**m	12.92°	0.362m	15.01°	0.354m	16.03°	0.432m	**12.88**°
	Regressor,NH	**0.319**m	38.05°	0.379m	35.96°	0.365m	40.16°	0.459m	**15.95**°
Stairs	Spatial-LSTM,QEH	0.365m	12.99°	0.392m	13.57°	0.345m	14.69°	**0.336**m	**11.79**°
	Spatial-LSTM,NH	0.363m	43.06°	0.390m	12.15°	0.350m	**11.92**°	**0.330**m	44.32°
	Regressor,QEH	0.424m	14.80°	0.406m	14.11°	**0.346**m	15.05°	0.388m	**13.12**°
	Regressor,NH	0.434m	**12.07**°	0.419m	40.68°	**0.361**m	11.70°	0.461m	13.28°

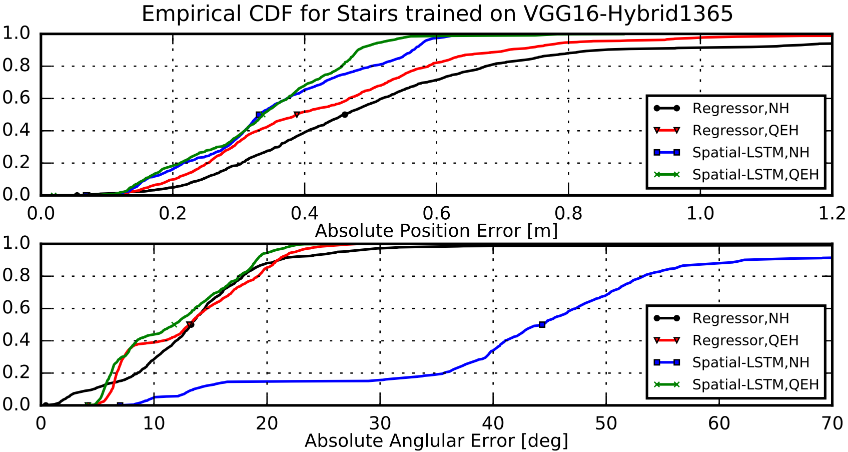
Experimental evaluation

Cumulative histograms



Experimental evaluation

Cumulative histograms



Experimental evaluation

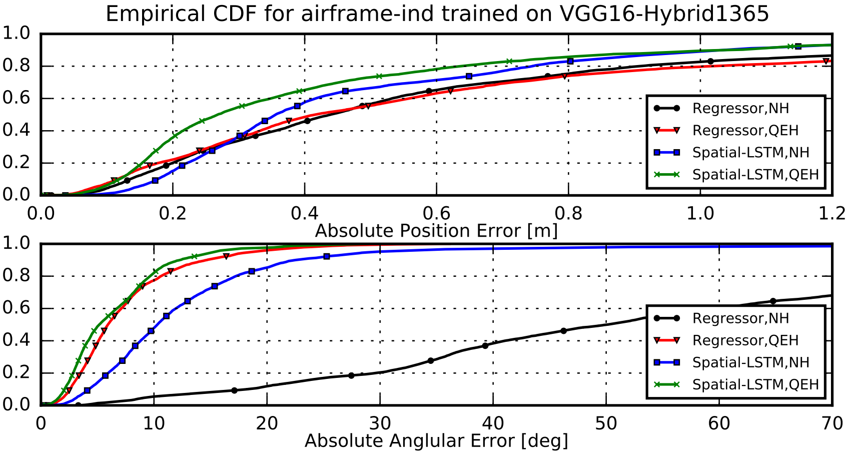
Airframe dataset

Median performance on Airframe dataset

Data set	**Method**	**GoogLeNet-ImageNet**		**GoogLeNet-Places365**		**Inception-ResNet-V2**		**VGG16-Hybrid1365**	
		position	orientation	position	orientation	position	orientation	position	orientation
airframe-mixed	Spatial-LSTM,QEH	0.193m	3.85°	0.279m	5.39°	0.196m	**3.76**°	**0.184**m	4.22°
	Spatial-LSTM,NH	0.259m	5.69°	0.292m	9.88°	0.273m	**5.13**°	**0.229**m	5.66°
	Regressor,QEH	0.264m	3.91°	0.350m	5.82°	**0.194**m	**3.37**°	0.229m	3.97°
	Regressor,NH	0.350m	6.75°	0.399m	12.40°	**0.265**m	**5.86**°	0.286m	7.06°
airframe-ind	Temporal-GRU,QEH	0.340m	**7.06**°	0.476m	9.67°	–	–	**0.247**m	7.67°
	Temporal-GRU,NH	0.437m	**8.68**°	0.691m	12.07°	–	–	**0.367**m	53.55°
	Spatial-LSTM,QEH	0.328m	7.39°	0.545m	10.41°	0.282m	5.28°	**0.268**m	**5.16**°
	Spatial-LSTM,NH	0.418m	13.44°	0.627m	14.97°	0.366m	**9.76**°	**0.356**m	10.29°
	Regressor,QEH	0.444m	7.95°	0.673m	10.78°	**0.287**m	**4.63**°	0.415m	5.92°
	Regressor,NH	0.708m	11.43°	0.906m	17.09°	**0.391**m	**6.78**°	0.439m	50.12°

Experimental evaluation

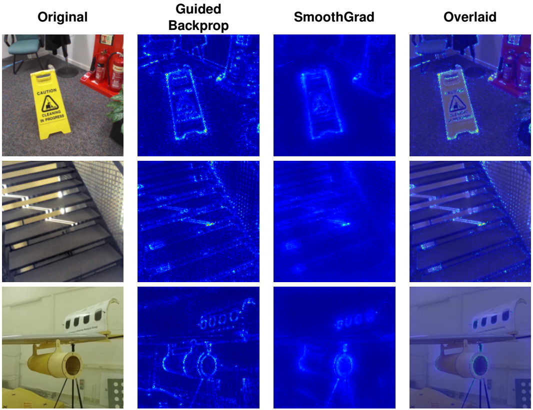
Cumulative histograms



Experimental evaluation

Saliency maps for 7Scenes and Airframe

Saliency maps [$\nabla_I f(I;\theta)$] for 7Scenes and airframe-ind datasets trained on Spatial-LSTM,QEH using VGG16-Hybrid1365



Experimental evaluation

State-of-the-Art comparison for 7Scenes dataset

State-of-the-Art comparison for 7Scenes

7Scenes	PoseNet[1] (β weight)	Walch et al[3] Spatial LSTM	PoseNet[2] Learn σ^2 Weight	This work Spatial-LSTM,QEH
Chess	0.32m, 6.60°	0.24m, 5.77°	0.14m, **4.50**°	**0.137**m, 7.868° *
Fire	0.47m, 14.0°	0.34m, 11.9°	**0.27**m, 11.8°	0.272m, **10.62**°
Heads	0.30m, 12.2°	0.21m, 13.7°	0.18m, 12.1°	**0.164**m, 14.79°
Office	0.48m, 7.24°	0.30m, 8.08°	**0.20**m, **5.77**°	0.212m, 7.83°
Pumpkin	0.49m, 8.12°	0.33m, 7.00°	**0.25**m, **4.82**°	0.264m, 18.33°
Red Kitchen	0.58m, 8.34°	0.37m, 8.83°	**0.24**m, **5.52**°	0.291m, 7.04°
Stairs	0.48m, 13.1°	0.40m, 13.7°	0.37m, **10.6**°	**0.336**m, 11.79°

* Naive-Homoscedastic (NH)

[1] Kendall et al. [ICCV 2015] - *PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization*

[2] Kendall et al. [CVPR 2017] - *Geometric Loss Functions for Camera Pose Regression with Deep Learning*

[3] Walch et al. [ICCV 2017] - *Image-based localization using LSTMs for structured feature correlation*

Outline

Faculty of Sciences, Technology and Communication

Conclusions

Conclusions

- We achieved competitive and sometimes outperforming results while using significantly less computation power
- VGG16-Hybrid1365 is the best choice for the CNN
- The novel quaternion homoscedastic (QEH) loss function vastly improves position and orientation prediction
- The new Airframe dataset is very challenging, but has interesting applications in Robotics
- Temporal models require a lot of computational power and data, although have much higher potential compared to standard models

Future work

Faculty of Sciences, Technology and Communication

Future work

- Obtaining a measure of uncertainty together with pose prediction
- Finetuning the whole network as opposed to just a part of it
- Further investigation of temporal GRU models

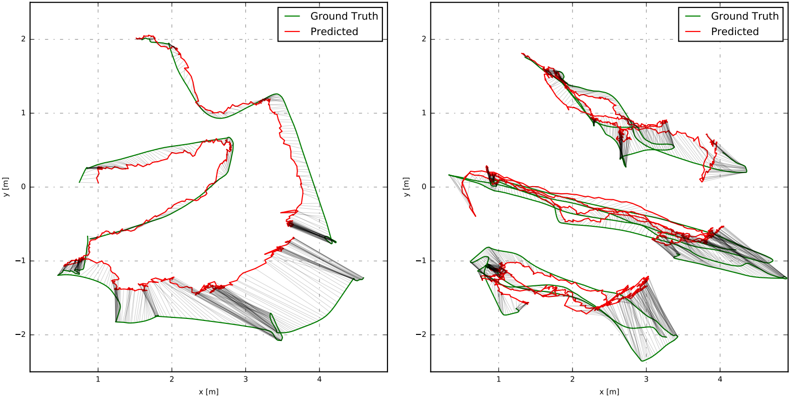
Thank you for your attention

Outline

Experimental evaluation

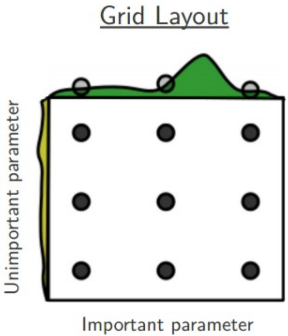
Predicted trajectories

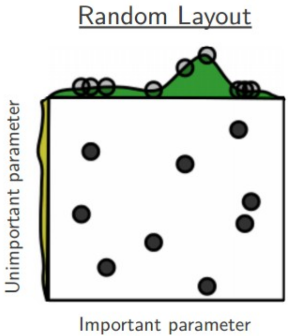
Top-down view of predicted trajectories on **airframe-ind** dataset using **Spatial-LSTM,QEH** with VGG16-Hybrid1365



Background

Hyperparameter Search

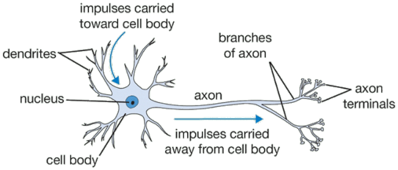




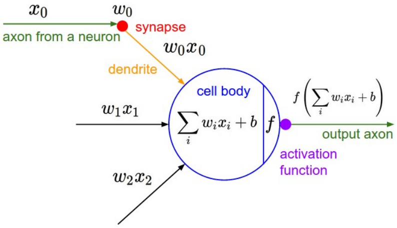
Grid Search vs Random Search, Bergstra et al. [NIPS 2011]

Background

Artificial Neural Networks



(a) Biological neuron



(b) Mathematical model

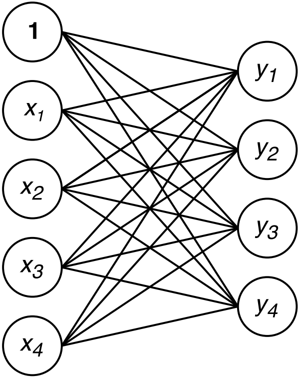
Background

Artificial Neural Networks

Fully-connected Layer (FC)

- simplest neural network
- activation is usually ReLU: $f(x) = max(x, 0)$
- matrix multiplication followed by an element-wise activation:

$$\boldsymbol{y} = f(\boldsymbol{Wx}) = f(\sum_{k=1}^{m} w_{ik} x_{ki} + b_i)$$



Background

Training

Stochastic Gradient Descent

```
Require: Training dataset $T = \{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), (\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}), \ldots, (\boldsymbol{x}^{(N)}, \boldsymbol{y}^{(N)})\}$
Require: Neural network as function $f(\boldsymbol{x}; \boldsymbol{\theta}_t)$ with $\boldsymbol{\theta}_0$ as initial parameters
Require: Loss function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{y})$
Require: Learning rate $\eta$ and batch-size $m$
$t \Leftarrow 1$
while stopping criterion not met do
    Shuffle $T$
    $i \Leftarrow 1$
    while $i \leq N$ do
        Compute network output for $j = i, \ldots, max(i+m-1, N)$: $\hat{\boldsymbol{y}}^{(j)} = f(\boldsymbol{x}^{(j)}; \boldsymbol{\theta}_t)$
        Compute gradient estimate: $\hat{\boldsymbol{g}} \Leftarrow \frac{1}{m} \sum^{j} \nabla_{\boldsymbol{\theta}_t} \mathcal{L}(\hat{\boldsymbol{y}}^{(j)}, \boldsymbol{y}^{(j)})$
        Update network parameters: $\boldsymbol{\theta}_t \Leftarrow \boldsymbol{\theta}_{t-1} - \eta \hat{\boldsymbol{g}}$
        $t \Leftarrow t+1$
        $i \Leftarrow i+m$
    end while
end while
```

Background

Training (continued)

Adam optimizer

- per-parameter adaptive with a leaky counter $\boldsymbol{v}_t$
- momentum $\boldsymbol{m}_t$ helps smooth noisy gradient
- bias correction for first few updates due to 0 initialization
- very good convergence without the need for heavy finetuning of η

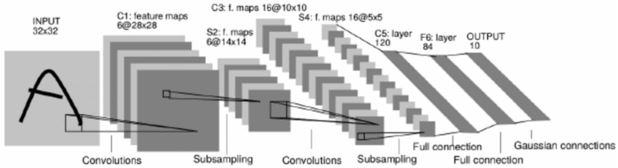
$$
\begin{aligned}
\boldsymbol{m}_t &= \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\hat{\boldsymbol{g}} \\
\boldsymbol{v}_t &= \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\hat{\boldsymbol{g}} \odot \hat{\boldsymbol{g}} \\
\hat{\boldsymbol{m}}_t &= \frac{\boldsymbol{m}_t}{1 - \beta_1^t} \\
\hat{\boldsymbol{v}}_t &= \frac{\boldsymbol{v}_t}{1 - \beta_2^t} \\
\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \eta \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \varepsilon}
\end{aligned}
$$

Background

Convolutional Neural Networks (CNN)

Convolutional Neural Network

- excellent for processing data with grid-like topology e.g. images
- parameter sharing allows for processing high-dimensional data
- composed of convolution (conv) and pooling (pool) layers

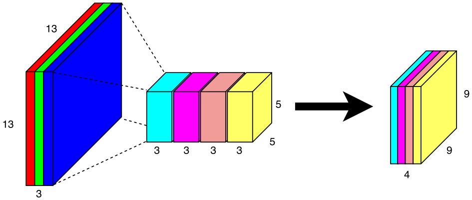


Background

Convolution Layer

Convolution Layer

- accepts input with size $x \times y \times d$ and outputs $x_o \times y_o \times n$ size
- performs convolutions on the input tensor for each filter
- resulting *activation maps* are stacked along depth dimension
- hyper-parameters are: # filters (n), filter size (f_x,f_y), strides (s_x, s_y) and padding (p_x, p_y)



$$x_o = \frac{x - f_x + 2p_x}{s_x} + 1$$

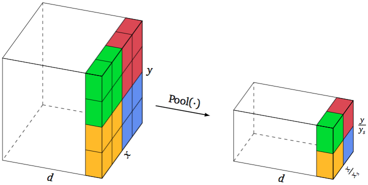
$$y_o = \frac{y - f_y + 2p_y}{s_y} + 1$$

Background

Pooling Layers

Pooling Layer

- accepts input with size $x \times y \times d$ and outputs $x_o \times y_o \times d$ size
- reduces spatial dimensionality
- the reduction operation is usually *max* or *average*
- hyper-parameters are: type of operation, window size (w_x, w_y), strides (s_x, s_y)



$$x_o = \frac{x - w_x}{s_x} + 1$$

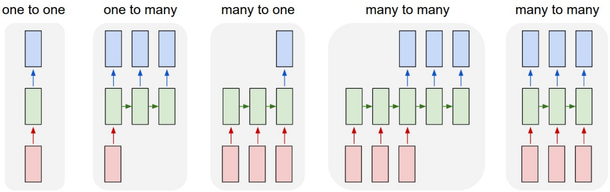
$$y_o = \frac{y - w_y}{s_y} + 1$$

Background

Recurrent Neural Networks (RNN)

Recurrent Neural Networks

- offer persistence of information between time-steps using loops
- overcome the limitation of processing fixed-sized inputs
- more challenging to train than regular neural networks
- turing-complete



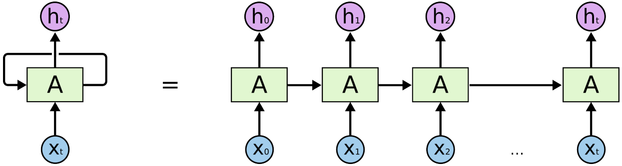
Background

Vanilla RNN

Vanilla RNN

- simplest RNN
- vanishing and exploding gradient problem
- unable to learn long-term dependencies

$$\boldsymbol{h}_t = tanh(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xh}\boldsymbol{x}_t + \boldsymbol{b})$$



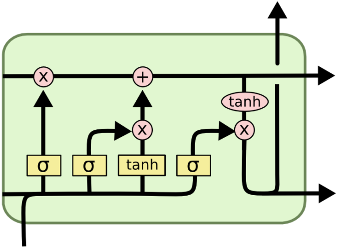
Background

Long-short Term Memory

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f x_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f)$$
$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i x_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i)$$
$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o x_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o)$$
$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot tanh(\boldsymbol{W}_c x_t + \boldsymbol{U}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c)$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot tanh(\boldsymbol{c}_t)$$

Long-short Term Memory (LSTM)

- much better at learning long-term dependencies
- gating mechanism allows for uninterrupted gradient flow
- adds cell state, which acts as memory
- 4 times more parameters than Vanilla RNN

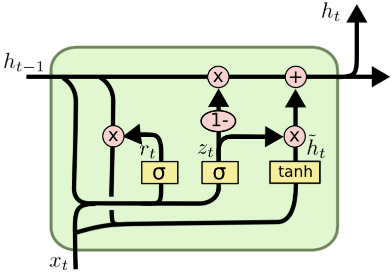


Background

Gated Recurrent Unit

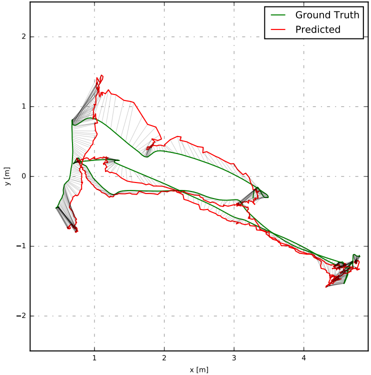
Gated Recurrent Unit (GRU)

- variant of LSTM
- only 2 gates: reset r_t and update z_t
- 2 times less parameters than LSTM
- does not need as much data and time for training



Experimental evaluation

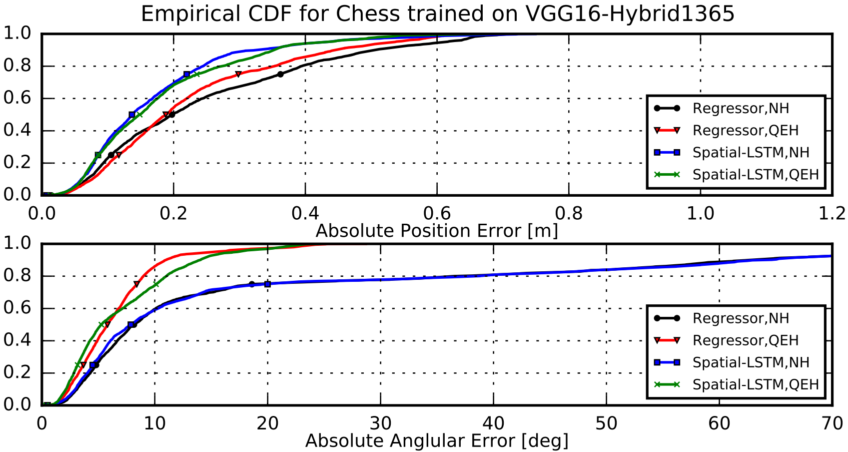
Trajectories on airframe-mixed - Position 6



Top-down view of predicted trajectory on **airframe-mixed** dataset using **Spatial-LSTM,QEH** with VGG16-Hybrid1365

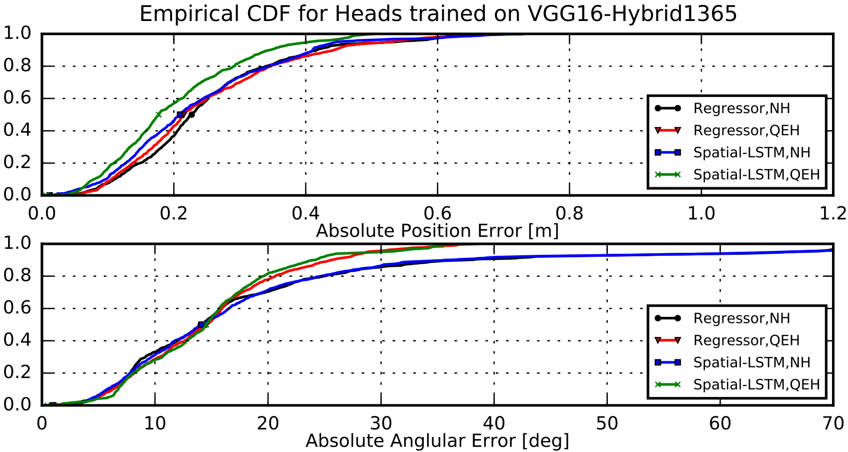
Experimental evaluation

Cumulative histograms



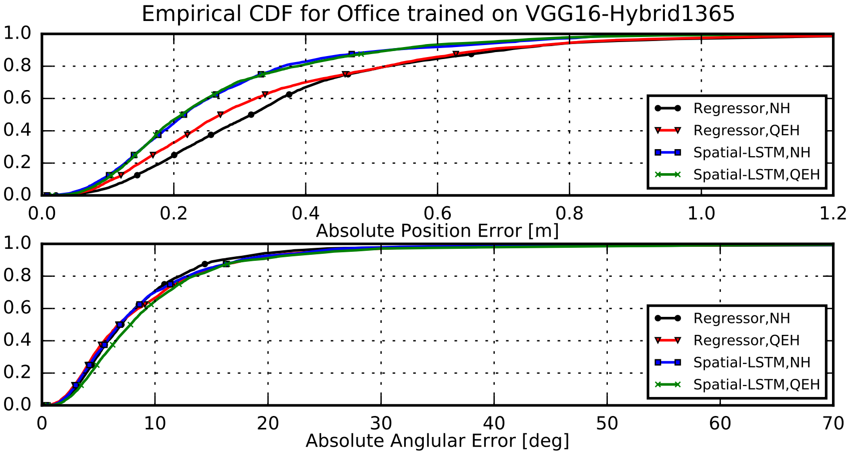
Experimental evaluation

Cumulative histograms



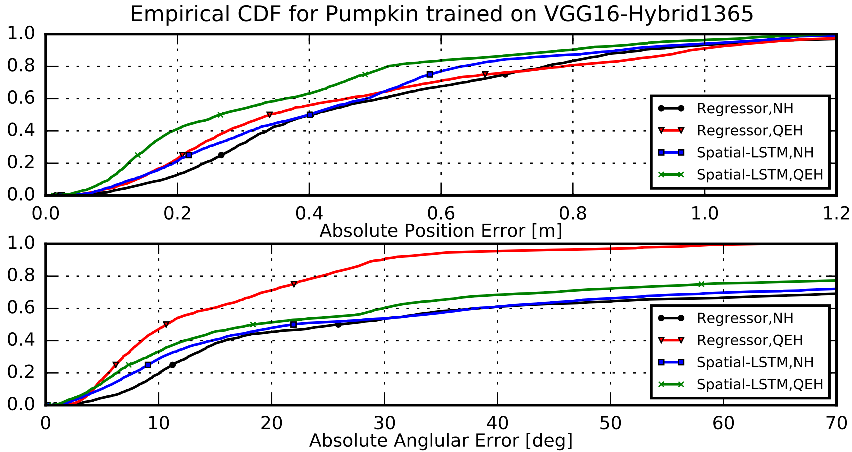
Experimental evaluation

Cumulative histograms



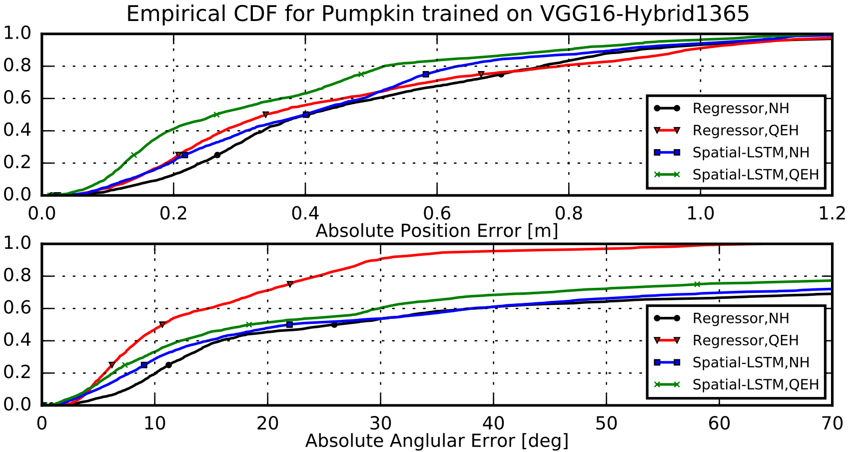
Experimental evaluation

Cumulative histograms



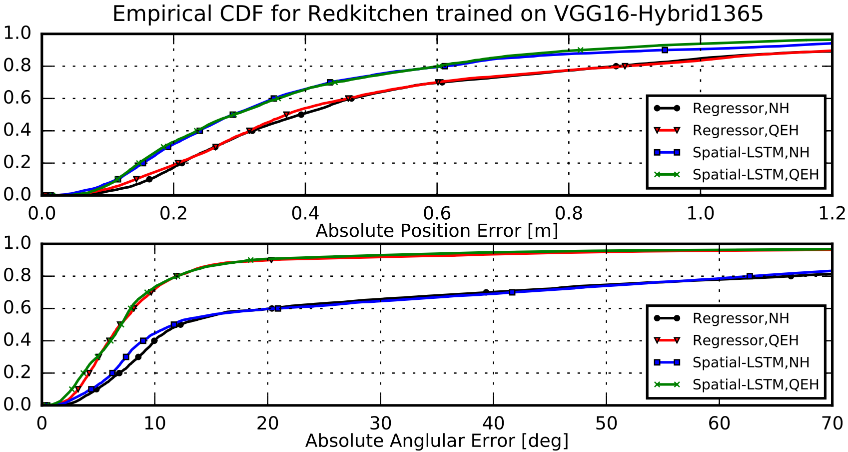
Experimental evaluation

Cumulative histograms



Experimental evaluation

Cumulative histograms



Experimental evaluation

Airframe dataset results with Temporal-GRU model

Airframe summary results with Temporal-GRU model

Data set	**Method**	**median**		**mae**		**max**		**std**	
		pos	orien	pos	orien	pos	orien	pos	orien
GoogLeNet ImageNet	Temporal-GRU,QEH	0.340	**7.059**	0.485	8.512	2.698	33.96	0.412	5.382
	Temporal-GRU,NH	0.437	8.68	0.565	11.79	3.426	174.1	0.434	11.90
GoogLeNet Places365	Temporal-GRU,QEH	0.476	9.67	0.567	11.90	2.100	49.00	0.345	7.76
	Temporal-GRU,NH	0.691	12.07	0.805	18.98	2.763	178	0.451	21.37
VGG16 Hybrid1365	Temporal-GRU,QEH	**0.247**	7.67	**0.378**	**8.48**	**1.592**	**32.40**	0.310	**5.00**
	Temporal-GRU,NH	0.367	53.55	0.443	62.86	1.701	179.8	**0.271**	37.28