2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Markov Localisation using Heatmap Regression and Deep Convolutional Odometry

Oscar Mendez[1], Simon Hadfield[1], Richard Bowden[1]

***Abstract*— In the context of self-driving vehicles there is strong competition between approaches based on visual localisation and Light Detection And Ranging (LiDAR). While LiDAR provides important depth information, it is sparse in resolution and expensive. On the other hand, cameras are low-cost and recent developments in deep learning mean they can provide high localisation performance. However, several fundamental problems remain, particularly in the domain of uncertainty, where learning based approaches can be notoriously over-confident.**

Markov, or grid-based, localisation was an early solution to the localisation problem but fell out of favour due to its computational complexity. Representing the likelihood field as a grid (or volume) means there is a trade off between accuracy and memory size. Furthermore, it is necessary to perform expensive convolutions across the entire likelihood volume. Despite the benefit of simultaneously maintaining a likelihood for all possible locations, grid based approaches were superseded by more efficient particle filters and Monte Carlo sampling (MCL). However, MCL introduces its own problems e.g. particle deprivation.

Recent advances in deep learning hardware allow large likelihood volumes to be stored directly on the GPU, along with the hardware necessary to efficiently perform GPU-bound 3D convolutions and this obviates many of the disadvantages of grid based methods. In this work, we present a novel CNN-based localisation approach that can leverage modern deep learning hardware. By implementing a grid-based Markov localisation approach directly on the GPU, we create a hybrid Convolutional Neural Network (CNN) that can perform image-based localisation and odometry-based likelihood propagation within a single neural network. The resulting approach is capable of outperforming direct pose regression methods as well as state-of-the-art localisation systems.

I. Introduction

The reasoning that humans can localise using vision alone, has been used extensively to motivate machine localisation from visual sensors such as cameras. However, there has always been a significant gap in the performance obtained from vision compared to LiDAR and/or Global Positioning System (GPS). Recent advances in vision use Deep-Learning based localisation [13], [15] to bridge this gap by employing CNNs to regress the camera pose directly from images. The network learns an implicit mapping between scene appearance and location. However, the mapping cannot generalise beyond the training data and due to the one-to-one mapping, provides uni-modal estimates in the pose-likelihood space [11].

Most traditional sampling-based localisation approaches, such as Markov/Grid-based localisation or more modern Monte-Carlo Localisation (MCL), are based on the idea that maintaining multiple hypothesis is an important part of the localisation problem. This makes problems like global localisation (kidnapped robot) more stable. It also allows algorithms to deal with self-similarity in environments.

Grid based methods model the likelihood of every location in the map as a set of discrete states. The resolution of the grid therefore affects accuracy. This results in a trade-off between accuracy and memory where larger grids are slower to process. To solve this problem and scale to large spaces, MCL was proposed. MCL samples the space using a Particle Filter (PF). This makes the process computationally efficient, but such approaches suffer from particle depletion, non-uniform sampling, sample size tuning and poor parallelisation.

This work proposes a novel deep learning architecture that maps a single image into a pose-likelihood. The network incorporates a grid based markov localisation framework to estimate a robot's pose. To make this tractable and overcome the limitations that gave rise to MCL, we introduce a first-of-its-kind convolutional likelihood propagation approach that models each odometry update as a single call to a gpu-bound convolution operation as part of the neural network. This hybrid CNN allows us to leverage the advances of deep-learning hardware to make grid-based localisation tractable by developing a single CNN architecture that can perform pose regression, localisation and odometry based likelihood propagation, in a single network, efficiently and with one forward pass.

II. Related Work

One of the first implementations of a localisation[1] algorithm was Extended Kalman Filter (EKF) Localisation [23]. This approach suffered from many limitations, but the most fundamental of which is the fact that it assumes that the localisation likelihood is a uni-modal distribution. This is such a crucial limitation, that grid-based localisation [3], [14], [20] algorithms quickly replaced it as the state-of-the-art. However, the computational efficiency of grid-based localisation algorithms limited their application. As a response to this, MCL algorithms [7], [8] became the de-facto standard for localisation. More recently, approaches in autonomous agent localisation have leveraged advances in deep learning. One of the most common family of approaches is PoseNet [13] and its derivatives [1], [10], [16], [24]. Fundamentally, these approaches rely on sensor/pose pairs to train a CNN that

[1] Centre for Vision Speech and Signal Processing, University of Surrey, Guildford, UK {o.mendez, s.hadfield, r.bowden}@surrey.ac.uk
∗Funded by InnovateUK Autonomous Valet Parking Project Grant No 104273.

[1]While Simultaneous Localisation and Mapping (SLAM) can be used as a method to solve the localisation problem [4], [6], here we focus on approaches that explicitly tackle localisation within a known-environment

can regress poses given an input sensor measurement. Kendall *et al*. [13] first introduced the method of regressing pose from images by first encoding the images using an encoder network. This maps the image into a lower-dimensional latent space that can then be mapped to a 6-Degree of Freedom (DoF) pose using a series of fully connected layers. In subsequent work, Kendall [12] introduced a better geometric loss function that allowed faster convergence and performance and in [11] began to model the underlying Bayesian statistics of the localisation problem. The nature of the networks mean a uni-modal distribution for the pose estimate is provided *i.e*. the network regresses a single location for any given input. However, it is well understood that regardless of the sensor used, there will often be areas of self-similarity in the environment. In this work, we regress a *pose-likelihood heatmap* which provides multi-modal distributions across pose.

Sattler *et al*. [19] showed the limitations of PoseNet-like models. They demonstrated that models are only reliable at approximating an agent's pose at a series of *base poses* and do not generalise to unseen poses between these bases. They concluded that PoseNet-like models are only reliable for coarse pose estimation and this calls into question the reliability of learning-based methods that do not employ other sources of information. For example, temporal accumulation and odometry [1]. Yang *et al*. [25] combined a PoseNet-like architecture with a depth estimation network in order to estimate motion and uncertainty. However, they only model uncertainty in their depth estimates.

We combine modern deep learning with tried and tested approaches to motion propagation. Pöschmann *et al*. [18] and Mendez *et al*. [15] both combined deep-learning segmentation with MCL to provide a robust localisation approach. Similarly, Neubert *et al*. [17] used depth regression with MCL and Iterative Closest Point (ICP). Our pose-likelihood heatmaps are a CNN-based sensor model combined with an odometry source. However, we implement this as a single hybrid CNN with all operations performed on a single GPU in one forward pass making it extremely efficient.

Although superseded by MCL, Grid-based localisation has some important advantages e.g. not suffering from particle deprivation, robustness to "kidnapped robot", lack of expensive re-sampling operations and generally being well-suited to massive parallelisation. However, grid-based localisation approaches have traditionally struggled to maintain robust estimates of pose due to the computational complexity of estimating sensor and motion models for each cell in the grid. Coarse quantisation is typically employed to make the approach tractable. There are several methods to improve the performance of grid-based localisation, such as [2]. However, they rely on less frequent sensing and/or motion integration. We overcome these limitations by combining the pose estimation and motion model into a single neural network that can make efficient use of the GPU.

III. Methodology

We propose to leverage the advances of deep learning to make Markov Localisation not only tractable, but also gain state-of-the-art performance. We do this by introducing a novel hybrid CNN architecture that combines a feature encoder layer, a image-to-heatmap-feature bridge, a heatmap decoder layer with multi-level supervision and finally a convolutional odometry layer. Fig. 1 shows an overview of the architecture used for our hybrid CNN. Note, this entire pipeline resides on the GPU as a single network allowing both image regression, localisation and odometry updates to be done in a single forward pass.

A. Markov Localisation

Markov localisation, shown by the gray dotted line on fig. 1, operates by taking the state-space of the autonomous agent, given by $\mathbf{p}_t \in \mathbb{P}_t$ and discretising it into in a grid defined as

$$\dot{\mathbf{x}}_t^k \in \dot{\mathbb{X}}_t \tag{1}$$

where each $\dot{\mathbf{x}}_t^k$ is a cell in the grid $\dot{\mathbb{X}}_t$ at time t. The grid $\dot{\mathbb{X}}_t$ spans all possible states in the state-space $\mathbb{P}_t$. For a ground-based vehicle, it is sufficient to represent the state space of the vehicle as a 3-DoF vector $\dot{\mathbf{x}} = \langle x, y, \theta \rangle$. This means that the discretised grid $\dot{\mathbb{X}}_t$ is a 3 dimensional volume. This volume consists of $\langle x, y \rangle$ planes and θ slices. More explicitly, this volume consists of a tensor of size $[\Theta \times X \times Y]$, where each cell represents the likelihood that the robot's position lies within that cell's bounds. Under a Markov assumption, this likelihood can be defined as

$$\begin{aligned} &\Pr\left(\dot{\mathbf{x}}_t^k \middle| \mathbb{Z}_t, \mathbb{U}_t\right) = \\ &\Pr\left(z_t \middle| \dot{\mathbf{x}}_t^k\right) \Pr\left(\dot{\mathbf{x}}_t^k \middle| u_t, \dot{\mathbf{x}}_{t-1}^k\right) \Pr\left(\dot{\mathbf{x}}_{t-1}^k \middle| \mathbb{Z}_{t-1}, \mathbb{U}_{t-1}\right) \end{aligned} \tag{2}$$

which implies that the pose likelihood is conditioned on the sensor observations $z_t \in \mathbb{Z}_t$ and the odometry measurements $u_t \in \mathbb{U}_t$ and is fully described by the sensor model

$$\Pr\left(z_t \middle| \dot{\mathbf{x}}_t^k\right) \tag{3}$$

the motion model

$$\Pr\left(\dot{\mathbf{x}}_t^k \middle| u_t, \dot{\mathbf{x}}_{t-1}^k\right) \tag{4}$$

and the prior likelihood of the pose

$$\Pr\left(\dot{\mathbf{x}}_{t-1}^k \middle| \mathbb{Z}_{t-1}, \mathbb{U}_{t-1}\right) \tag{5}$$

which implies this measurement can be performed iteratively. In this work, we use a heatmap regression sensor model (III-B) to update the $[X \times Y]$ volume, while using convolutional odometry (III-C) for the orientation. We additionally perform a likelihood-volume-to-pose extraction, which consists of fitting a Gaussian distribution to the likelihood volume and reporting the mean.

B. Deep Likelihood Heatmap Regressor

One of the main reasons grid-based localisation was widely considered intractable was because the sensor model (equation 3) has to be estimated for every cell in the grid. Estimating this likelihood for sensors such as LiDAR, SoNAR or even RGB-D cameras can involve expensive ray-casting for each beam. This presents a practical issue when there are large areas

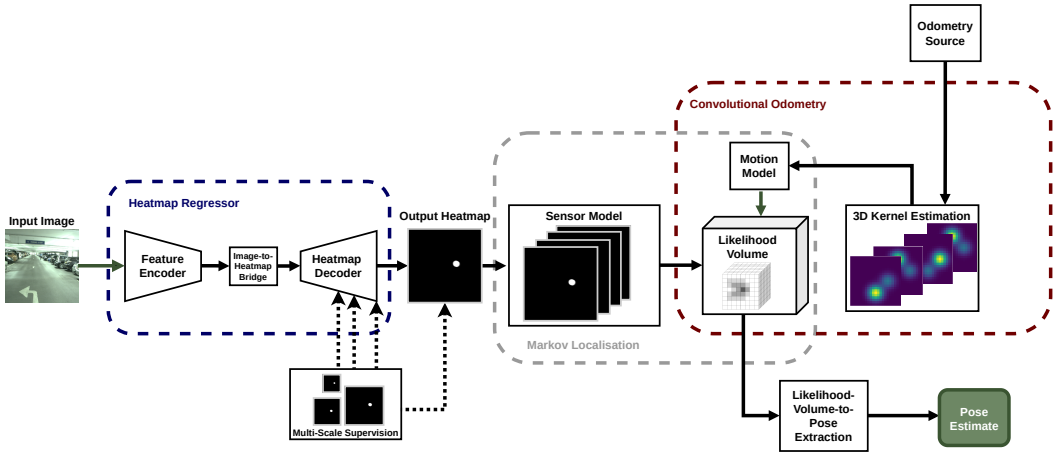

Fig. 1: Deep Markov Localisation: System Diagram.

of the grid that we are almost certain *not* to be the correct location. Instead, we propose to use a Fully Convolutional Network (FCN) trained for pose regression to replace the sensor model.

Using an FCN has several important advantages. Firstly, it allows a sensor model to be trained for any arbitrary sensor without the need for explicit mathematical derivation (although requiring training data). Secondly, the likelihood for every cell can be estimated simultaneously.

1) Feature Encoder Layer: To regress a likelihood for all cells, we use an encoder-decoder architecture. The architecture is shown in the blue dotted line on fig. 1. We use a ResNet encoder arm [9] similar to PoseNet [11] but rather than a fully connected layer to regress pose, we use the decoder arm of the network (discussed below) to force the network to learn a birds-eye-view likelihood map. We modify the ResNet by removing the fully connected layers and replace them with a convolutional image-to-heatmap bridge to the decoder. The whole network is trained end-to-end.

2) Heatmap Decoder Layer: The decoder is composed of a series of upsample blocks, which scale the image up by a factor of two. Each upsample block contains a deconvolution followed by two convolution blocks. The deconvolution is performed with a $[2 \times 2]$ kernel and a stride of 2. The convolution blocks consist of a $[3 \times 3]$ convolution with a stride of 1 followed by batch normalisation and ReLU. The output block additionally contains a final convolution layer with $[1 \times 1]$ kernel and a stride of 1 ensuring the desired number of output channels is achieved. The result is a map of size $[N \times M \times M]$ where M depends on the number of upsample blocks and N is the number of channels.

Intuitively, we could map this $[N \times M \times M]$ output volume directly to the grid we are localising in. In this mode, the volume would represent $[\Theta \times X \times Y]$ in each channel respectively. This ties the spatial resolution to the size of each channel, and the angular resolution to the number of channels. Traditional Markov localisation estimates the sensor model this way because the ray-casting operations need to be performed differently for each θ bin. However, this is not an optimal use of the network as it can directly estimate the probability of an $\dot{\mathbf{x}}_t^k$ cell without reasoning about the orientation. Furthermore, using the output volume this way would force the network to grow dramatically as the angular resolution increases. This not only increases the number of parameters, but actually defines a very complicated regression problem. Instead, we use each output channel as a "likelihood band" which allows us to treat the heatmap regression as a classification problem.

The likelihood volume represents $\Pr\left(z_t \middle| \dot{\mathbf{x}}_t^k\right)$ discretised into $[N \times M \times M]$ bins where N represents the number of likelihood bands, M denotes the x and y resolution and k spans each of the pixels on the XY plane across all orientations. More explicitly,

$$\Pr\left(z_t \middle| \dot{\mathbf{x}}_t^k\right) \propto \Pr\left(n_t \middle| \dot{\mathbf{x}}_t^k\right) \tag{6}$$

where $n_t \in N$ is the likelihood band of $\dot{\mathbf{x}}_t^k$ at time t. This discretisation of the probability space allows us to treat likelihood regression as a classification problem, where we classify each $\dot{\mathbf{x}}_t^k$ cell into a pose likelihood bin.

3) Multi-Scale Supervision: Using the regression-to-classification mapping defined in equation 6, we can define a cross entropy loss function (after softmax)

$$\mathcal{L}_c(n_t, \dot{\mathbf{x}}_t^k) = -\Pr\left(n_t \middle| \dot{\mathbf{x}}_t^k\right) \cdot \log\left(\mathcal{R}(I)_{x,y}\right), \tag{7}$$

where R is our heatmap regressor network, I is the input image and x, y denote the pixel location in the resulting heatmap. We additionally provide supervision in the form of an MSE loss, defined as

$$\mathcal{L}_m = \left|\left|\Pr\left(n_t \middle| \dot{\mathbf{x}}_t^k\right) - \mathcal{R}(I)_{x,y}\right|\right|_{L_2}. \tag{8}$$

The resulting loss function is defined as a weighted combination of these losses,

$$\mathcal{L} = \mathcal{L}_c + \omega\, \mathcal{L}_m. \tag{9}$$

where ω is a hyperparameter. Empirically, we have found that $\omega = 0.1$ ensures the network can produce accurate heatmaps with a smooth distribution around the correct cell.

To ensure a more robust loss, we perform this loss function over several different scales. Fig. 2 shows the proposed architecture, which consists of 4 encoder residual blocks (blue), 4 base heatmap decoder blocks (orange), followed by 4 output decoder blocks (orange + yellow). Note that the

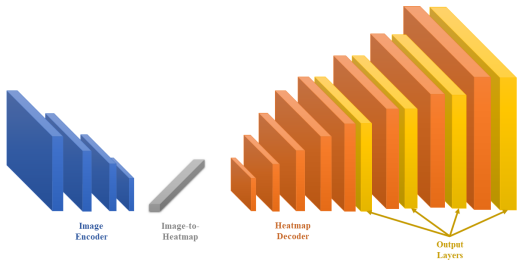

Fig. 2: Deep Markov Localisation: System Diagram.

yellow output layers do not feed into the upscaled orange blocks, but rather directly produce an output. Each of these output heatmaps can be directly supervised by a ground-truth heatmap. This multi-level supervision allows the network to learn a coarse-to-fine heatmap regression.

4) Sensor Model: In order to use this likelihood volume as a sensor model, it is necessary to map it back to our Markov localisation grid. To do so, we run a softmax operation along the probability bins, which ensures the sum of all likelihoods for a given $\dot{\mathbf{x}}_t^k$ cell sum to one. We then take the top $n < N$ slices and sum them to obtain a single pose likelihood for the XY plane which is then repeated for the θ bins, which we discuss in the following section.

C. Convolutional Odometry

The sensor model introduced in the previous section does not measure the likelihood of the agent's orientation θ. Instead, we use a novel convolution-based odometry layer as a motion model which estimates θ. Our model is efficient, so there is no need to artificially limit the update rate other than to guarantee at least one cell of displacement. This means that, assuming a non-holonomic agent, the motion model can propagate pose likelihoods in a manner that also selects the correct orientation.

Markov localisation relies on a motion model (equation 4) to propagate likelihoods into the correct areas of the grid. Normally, this is implemented as a shift according to the odometry measurement, followed by diffusion using convolution with a separable Gaussian. The kernel of this Gaussian is computed based on the odometry's uncertainty. For a 3-DoF likelihood grid this is a relatively expensive operation as it would require a set of 3 shifts, and a 3D convolution which would be prohibitively expensive on a CPU. However, we formulate the odometry kernel as a deep learning layer that enables us to perform an efficient operation on the GPU by mapping the 3 shifts and 3 convolutions into a singe 2D convolution kernel. By building our sensor and motion model as custom layers on a CNN, the entire framework can operate in a single forward pass on one GPU incredibly quickly. In order to estimate our odometry kernel for 2D convolution, we first look at how the odometry maps into a simple 3D convolution kernel. The odometry data from any non-holonomic 3-DoF vehicle can be decomposed as

$$\left(\delta\theta_t^1, \delta x_t, \delta\theta_t^2\right) = \boldsymbol{odometry\_model}\left(u_t\right) \tag{10}$$

where $\delta\theta_t^1$ is a rotation followed by a forward translation δx_t and a final rotation $\delta\theta_t^2$ [22]. Gaussian noise can be applied

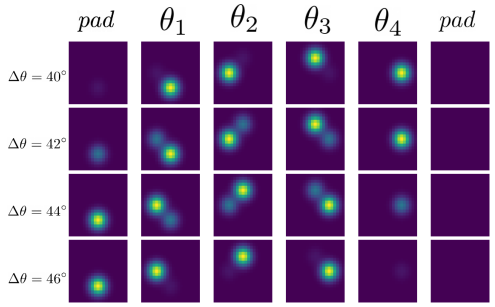

Fig. 3: Visualisation of an odometry kernel with pure forward movement. Each column represents a θ_c channel, while each row represents a different odometry measurement for the angle of the vehicle. Notice how each θ_c channel represents the forward motion differently. Similarly, as the rows go down the likelihoods shift between the different theta channels.

to each component independently, producing a new set of odometry estimates $\hat{\delta}\theta_t^1, \hat{\delta}x_t, \hat{\delta}\theta_t^2$.

In order to map this into a 3D kernel, we take this representation and map it into a vector as

$$\mathbf{s}_t = \begin{pmatrix} s_x \\ s_y \end{pmatrix} = \begin{pmatrix} \hat{\delta}x_t * cos(\hat{\delta}\theta_t^1) \\ \hat{\delta}x_t * sin(\hat{\delta}\theta_t^1) \end{pmatrix}. \tag{11}$$

However, this odometry is not directly applicable to every $\theta_c \in \Theta$ channel in our likelihood volume. In order to apply the odometry, we align it such that "forward" motion represents the orientation of the θ_c bin. This is done using a simple 2D rotation matrix defined as

$$\mathbf{R}_c = \begin{bmatrix} \cos(\theta_c) & -\sin(\theta_c) \\ \sin(\theta_c) & \cos(\theta_c) \end{bmatrix} \tag{12}$$

for every θ_c in the likelihood volume. Each of these matrices can then be multiplied with the linear component of the odometry vector

$$\mathbb{S}_t = \left\{ \begin{pmatrix} \mathbf{R}_c\, \mathbf{s}_t \\ \delta\theta_t^1 + \delta\theta_t^2 \end{pmatrix} \quad \forall\theta_c \in \Theta \right\} \tag{13}$$

in order to obtain a set of rotated odometries for each channel.

The set of rotated odometries are directly mapped into a kernel for convolution. However, it is first necessary map these odometries to the resolution of the likelihood grid, which can be done as $\mathbb{S}'_t = \mathbb{S}_t \odot \lambda$ where $\odot$ is element wise multiplication and $\lambda = \left\langle \frac{1}{r_x}, \frac{1}{r_y}, \frac{1}{r_\theta} \right\rangle$ is the resolution of the grid which has been tiled Θ times.

We are not guaranteed that the odometry will be larger than a single cell in the likelihood grid. For this reason, it is necessary to accumulate odometry measurements over time as

$$\mathbb{T}_t = \sum_i^t \mathbb{S}'_i - \mathbb{T}'_{t-1} \tag{14}$$

where $\mathbb{T}'_t = \mathbb{T}_t - \lfloor \mathbb{T}_t \rfloor$ and $\lfloor \mathbb{T}_t \rfloor$ is the odometry applied to the likelihood volume $\dot{\mathbb{X}}_t$.

Applying $\lfloor \mathbb{T}_t \rfloor$ to $\dot{\mathbb{X}}_t$ can be done by converting the rotated odometries into a series of 3D kernels for 2D convolution. The first step is to map each vector $\tau_t \in \lfloor \mathbb{T}_t \rfloor$ into a 3D

Scene Name	PoseNet [13]	PoseNet Bayesian [13]	PoseNet Spatial LSTM [24]	PoseNet Learn β [12]	PoseNet Geometric [12]	Heatmap Regressor (Us)
GreatCourt	-	-	-	7.00	6.83	**3.74**
KingsCollege	1.66	1.74	0.99	0.99	**0.88**	0.95
OldHospital	2.62	2.57	**1.51**	2.17	3.20	2.13
ShopFacade	1.41	1.25	1.18	1.05	0.88	**0.67**
StMarysChurch	2.45	2.11	1.52	1.49	1.57	**1.02**
Street	-	-	-	20.7	20.3	**10.21**

TABLE I: Median error (m) for Cambridge Landmarks [13]. PoseNet results are from [12], where '-' are not reported by authors.

kernel. To do this we take every element of τ_t and convert it into a discrete 1D Gaussian kernel of size $\langle k_\theta, k_x, k_y \rangle$. These kernels are then combined linearly to produce a 3D kernel $\mathbf{K}_c$ of size $\langle K_\theta, K_x, K_y \rangle$. This allows us to perform a 2D convolution on a subset of the likelihood grid for each 3D kernel. This subset consists of the channel the kernel was estimated for as well as the K_θ channels around it. In order to perform the operation as a single pass of a 2D convolution on GPU hardware, we stack these kernels into a single kernel. Each 3D kernel is offset so that it is centered on the θ_c channel it corresponds to, giving us a kernel with θ_c Gaussian distributions. Fig. 3 shows a visualisation of the odometry kernel when $\Theta = 4$ for a simple forward motion. Each column represents a θ_c channel (along with padding), while each row represents the rotated odometry for each angle bin. Notice how each θ_c channel represents the forward motion differently. Similarly, as the rows go down, the likelihoods shift between the different theta channels and respect the circular nature of θ by looping from the last bin to the first.

While it may seem like these are an expensive set of convolutions, it is important to remember that these are all performed directly on the GPU. More importantly, our Markov localisation approach is entirely GPU bound. Both our sensor and motion model are computed directly on the GPU with no need to ever retrieving the cost volume. This makes our approach both quick and efficient. In the following section, we will show that not only is our approach fast but it is also capable of producing state-of-the-art results for localisation.

IV. Results

Firstly, we validate the performance of our heatmap regressor by evaluating its performance on the well-established Cambridge Landmarks dataset [13]. Secondly, we evaluate on a vehicle navigating a multi-storey carpark.

All the experiments in this section are trained using the same standard architecture for the heatmap regression network: a ResNet50 as the encoder, with 8 decoder blocks to result in a regressed heatmap of $[256 \times 256]$ supervised at the last 4 blocks ($M = [32, 64, 128, 256]$). The learning rate was set to 0.0001 with a variable step learning rate and $\gamma = 0.5$.

A. Cambridge Landmarks

The dataset consists of 6 sequences of images captured at different landmarks across the city of Cambridge (UK) using a hand-held camera. Each sequence consists of anywhere between 300 and 6000 images which are then split between train and test sets. Ground-truth poses are estimated using Structure from Motion (SfM) [5] software. The sensor experiences 6-DoF motion during capture. Furthermore the SfM software does not guarantee that there is a well-defined ground plane. This makes the dataset inherently difficult for our approach, as we only localise in a 3-DoF space. Since most of the motion of the sensor is planar, we overcome this limitation by using an Single Value Decomposition (SVD) to regress a dominant plane and therefore the height of the sensor.

Table I shows a comparison against several pose-regression networks. We show the median pose error on all 3 spatial axis, as reported in [12]. It is important to note that this is an unfavourable scenario for our 3-DoF localisation, as our network does not directly regress the height of the sensor. We also do not estimate the orientation of the sensor, as the heatmap regressor should be able to cope with the appearance variation. Regardless, our heatmap regressor is able to outperform several state-of-the-art regression methods in 4 out of 6 scenarios. We believe there are several reasons for this. Firstly, a heatmap-based loss provides a more uniform supervision signal than a pose-based loss, as the heatmaps do not scale with the magnitude of the pose space. Secondly, and most importantly, while we do not strictly enforce multi-modal distributions with our losses, the network is capable of modelling them. This allows the network to predict uncertain poses without incurring a penalty. In the KingsCollege scenario, we are only outperformed by PoseNet Geometric [12], which makes use of an additional source of supervision: the 3D reconstruction points of the ground truth. For the OldHospital scenario, we are outperformed by Walch *et al*. [24] most likely because the self-repeating nature of the architecture is well-suited to the spatial LSTMs they employ.

B. Multi-Storey Carpark

One of the main advantages of our approach is the ability to generalise to self-similar environments. Car-parks are interesting environments for localisation, as they tend to be self-similar within each floor as well as across multiple floors. By their nature, accurate localisation within a multi-storey car-park requires a multi-modal distribution.

1) Data Capture: This dataset consists of a vehicle driving around a multi-storey car-park. The vehicle is equipped with 3 front-facing cameras and a 16-beam LiDAR. It traversed 12 floors, travelling a total of over $6,500m$ and an area of over $7000m^2$. The vehicle performed parking manoeuvres such as 3 point turns, bay parking, interacting with traffic, etc. The LiDAR is used to create ground-truth localisaton data. The vehicle was driven on two separate days, with two trajectories on the first day and one on the second. Of the three trajectories, trajectory 1 has 3207 images, 2 has 2860

Method	Trajectory 2 RMSE (m)	Mean (m)	Median (m)	Std. Dev. (m)	Trajectory 3 RMSE (m)	Mean (m)	Median (m)	Std. Dev. (m)
PoseNet [12]	9.31	5.26	2.66	7.68	14.09	6.75	3.21	12.37
PoseLSTM [24]	9.05	5.04	2.61	7.52	12.82	6.55	2.74	11.02
Raw Odometry	12.20	9.23	5.54	7.97	24.86	23.77	24.43	7.25
Ours (H)	9.74	4.21	**1.36**	8.78	16.65	7.14	**1.80**	15.04
Ours (H+O)	**6.48**	**3.81**	2.45	**5.24**	**6.99**	**3.88**	2.56	**5.82**

TABLE II: Average Trajectory Error: PoseNet, PoseLSTM, Ours (Odometry), Ours (Heatmap), Ours (Heatmap and Odometry). Top results are **Bold**, underline second.

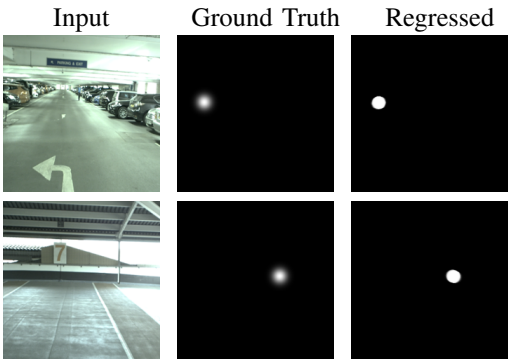

Fig. 4: Sample Images from the Multi-Storey Car Park dataset, along with the ground truth heatmap and output regression.

and 3 has 3922. We use trajectory 1 as training data, and reserve trajectories 2 and 3 for testing. The left column of fig. 4 shows sample images from the captured dataset. This dataset will be released upon publication of this work.

2) Heatmap Regression Training: To train our heatmap regressor we use the pose of the LiDAR along with a calibrated transformation between the sensors. The estimated pose is then projected to the ground plane, where the heatmaps can be estimated. The right column of Fig. 4 shows two example heatmaps used in training, as well as the regressed heatmaps.

The convolutional odometry consists of a $[72 \times 256 \times 256]$ likelihood volume, with a kernel of size $[15 \times 21 \times 21]$. Using this configuration on an AMD Threadripper 3960X with an Nvidia GeForce GTX 1080 Ti, the heatmap regressor takes $10.701ms$ with a standard deviation of $0.0377ms$ and the convolutional odometry layer takes $0.253ms$ with a standard deviation of $0.00538ms$. In order to simulate odometry, we use the ground-truth poses acquired from the LiDAR. For each successive pose, we add noise as described in [22] (forcing the trajectory to drift) and estimate a set of odometry measurements from the resulting noisy trajectory.

3) Quantitative Results: We evaluate the performance of our approach against PoseNet [12] and PoseLSTM [24], as both represent the state-of-the-art for camera pose regression. We compare using the Absolute Trajectory Error (ATE) as established by Sturm *et al.* [21], which accounts for orientation errors as part of the overall trajectory. In Table II it can be seen that our approach outperforms the state-of-the-art by a significant margin. This is because our approach has the ability to maintain multiple hypothesis of the pose as the vehicle moves through the car-park. By contrast, PoseNet and PoseLSTM are forced to relocalise the camera at every iteration. This results in paths that are unreliable. Since our approach keeps a likelihood field, failures in the heatmap regression do not directly result in jumps in the pose estimate, therefore smoothing our trajectory and reducing the ATE error. We additionally show the results of the raw odometry measurements and using our heatmap regressor only (with no motion model). As it can be seen, the odometry measurements are extremely noisy, resulting in a high ATE. The Heatmap Regressor alone outperforms competing approaches but despite the noisy odometry, its addition increase performance further.

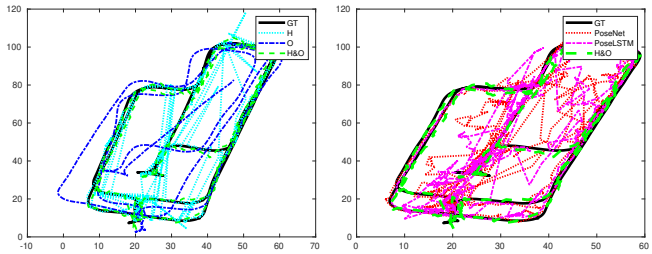

Fig. 5: Trajectories of Ground Truth (GT), PoseNet, PoseLSTM, Heatmap Regressor (H), Raw Odometry (O) and the Combined Heatmap Regressor & Convolutional Odometry (H&O). Note how PoseNet and PoseLSTM do not provide smooth trajectories.

4) Qualitative Results: In our experience, the numbers presented in tables I and II do not adequately convey the difference in the smoothness of the trajectories. Fig. 5 shows the resulting trajectories from PoseNet, PoseLSTM and our approach, as well as a comparison against the raw odometry and heatmap regressor alone. For clarity, this is done on the first three floors of the multi-storey carpark. As it can be seen, our approach is significantly smoother than both PoseNet and PoseLSTM. It can also be seen that our combined approach is smoother than the independent heatmap regressor, as well as more accurate than the raw odometry measurements. This is because our likelihood heatmap, convolutional odometry and likelihood grid work together to ensure that poor estimates do not cause jumps in the pose estimate. Additionally, the ability to represent a multi-modal distribution allows us to make quick corrections when the pose has been incorrectly estimated.

V. Conclusion

In summary, we have presented an approach that leverages advances in Deep-learning hardware to perform deep heatmap regression and convolutional odometry, in real time. Our work operates on commodity GPU hardware and leverages some of the important advances in tensor-based processing by performing all operations directly on the GPU, without the need to transfer data back to the CPU. Importantly, our approach capitalises on the important probabilistic properties of Markov localisation by exploiting modern parallel GPU technology.

REFERENCES

[1] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-aware learning of maps for camera localization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2616–2625, 2018.

[2] Wolfram Burgard, Armin B Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artificial intelligence*, 114(1-2):3–55, 1999.

[3] Wolfram Burgard, Dieter Fox, Daniel Hennig, and Timo Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *National Conference on Artificial Intelligence*, pages 896–901, 1996.

[4] Tim Caselitz, Bastian Steder, Michael Ruhnke, and Wolfram Burgard. Monocular camera localization in 3D LiDAR maps. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1926–1931. IEEE/RSJ, 2016.

[5] Wu Changchang. Towards Linear-Time Incremental Structure from Motion. In *International Conference on 3D Vision (3DV)*, pages 127–134, 2013.

[6] Hang Chu, Dong Ki Kim, and Tsuhan Chen. You are here: Mimicking the Human Thinking Process in Reading Floor-Plans. In *International Conference on Computer Vision (ICCV)*, pages 2210–2218, 2015.

[7] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte Carlo localization for mobile robots. In *International Conference on Robotics and Automation (ICRA)*, number May, pages 1322–1328, Detroit, 1999. IEEE.

[8] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 343–349, 1999.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[10] Joao F Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8476–8484, 2018.

[11] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016.

[12] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5974–5983, 2017.

[13] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2938–2946, 2015.

[14] Jong-Hwan Lim and Chul-Ung Kang. Grid-based localization of a mobile robot using sonar sensors. *Korean Society of Mechanical Engineers (KSME) International Journal*, 16(3):302–309, 2002.

[15] Oscar Mendez, Simon Hadfield, Nicolas Pugeault, and Richard Bowden. SeDAR - Semantic Detection and Ranging: Humans can localise without LiDAR, can robots? In *International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018. IEEE.

[16] Tayyab Naseer and Wolfram Burgard. Deep regression for monocular camera-based 6-dof global localization in outdoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1525–1530. IEEE, 2017.

[17] Peer Neubert, Stefan Schubert, and Peter Protzel. Sampling-based Methods for Visual Navigation in 3D Maps by Synthesizing Depth Images. In *International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[18] Johannes Poschmann, Peer Neubert, Stefan Schubert, and Peter Protzel. Synthesized Semantic Views for Mobile Robot Localization. In *European Conference on Mobile Robotics (ECMR)*, pages 403–408, Paris, 2017. IEEE.

[19] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3302–3312, 2019.

[20] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 95, pages 1080–1087, 1995.

[21] Jurgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580. IEEE/RSJ, 2012.

[22] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Robot Motion. In *Probabilistic Robotics*, chapter 5, pages 135–136. MIT Press, Cambridge, Massachussets, 2006.

[23] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. TheEKFLocalizationAlgorithm. In *Probabilistic Robotics*, chapter 4.5.2, pages 96–113. MIT Press, Cambridge, Massachussets, 2006.

[24] Florian Walch, Caner Hazirbas, Laura Leal-Taix, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using lstms for structured feature correlation. In *International Conference on Computer Vision (ICCV)*, October 2017.

[25] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1281–1292, 2020.