Heteroskedastic Geospatial Tracking with Distributed Camera Networks

Abstract

Visual object tracking has seen significant progress in recent years. However, the vast majority of this work focuses on tracking objects within the image plane of a single camera and ignores the uncertainty associated with predicted object locations. In this work, we focus on the geospatial object tracking problem using data from a distributed camera network. The goal is to predict an object's track in geospatial coordinates along with uncertainty over the object's location while respecting communication constraints that prohibit centralizing raw image data. We present a novel single-object geospatial tracking data set that includes high-accuracy ground truth object locations and video data from a network of four cameras. We present a modeling framework for addressing this task including a novel backbone model and explore how uncertainty calibration and fine-tuning through a differentiable tracker affect performance.

1 INTRODUCTION

Deep neural network models have enabled remarkable advances in visual object tracking performance across a wide range of scenarios over the last decade [Marvasti-Zadeh et al., 2021]. The task of visual object tracking involves detecting and localizing objects as they move through a scene. The vast majority of prior work on visual object tracking considers video data from a single camera only and poses the problem as tracking objects within the image plane of the camera. Further, most approaches to tracking provide outputs as a sequence of bounding boxes that lack a representation of uncertainty over the locations of object.

In this work, we focus on the geospatial object tracking problem using a distributed camera network. The goal is to predict an object's track in geospatial coordinates along with uncertainty over the location while accounting for communication constraints that prohibit centralizing raw image data. This problem is motivated by the proliferation of Internet of Things (IoT) devices with local sensing, compute and wireless communication capabilities. The geospatial object tracking problem has important applications in smart cities such as traffic monitoring and pedestrian safety that require knowledge of where objects are in real-world map coordinates [Datta and Sharma, 2017].

Importantly, the continuous deployment of camera and other sensor networks in real-world environments introduces multiple challenges due to resource constraints and changing environmental conditions [Pereira et al., 2020]. Due to resource constraints the set of available cameras may not cover the entire environment of interest resulting in observability gaps. Objects may occlude each other from some vantage points but not others as they move through the environment. Lighting and weather changes will also effect some or all cameras at different times. As a result, it is essential to develop models and systems that can refelect the uncertainty in an object's location in a meaningful way taking into account local observability, occlusions, and environmental effects. These aspects are often overlooked in traditional tracking benchmarks that assume access to high-quality video data [Dendorfer et al., 2020].

To facilitate research on this problem, we collected a novel single-object tracking data set that includes high-quality ground truth geospatial object locations collected at 100 samples per second using a motion capture system. These data are combined with video from four camera nodes with different partially overlapping views of the environment. The object tracked is a remote controlled vehicle. The dataset spans multiple scenarios including an open environment and an environment with occluding structures, as well as a normal and low lighting scenarios. This data set and associated processing algorithms will be released along with the paper.

We present a modeling framework for addressing this task

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.

that augments neural network object detection backbones with adapters that translate within image-plane features into geospatial coordinates with associated uncertainty. We refer to such as model as a heteroskedastic geospatial detector (HGD). To address communication constraints, each HGD is restricted to operate over data available locally at a singe camera node. To solve the tracking problem, the low-dimensional probabilistic outputs of a set of independent HGDs are centralized and fused together using a multi-observation Kalman filter model [Kalman, 1960].

We experiment with both existing and custom backbones within the HGDs. Further, we explore the effects of geospatial detection calibration to improve the quality of the distributions provided by individual HGDs to the Kalman filter. Finally, we experiment with backpropagating loss through the Kalman filter and in to individual HGD adapters as a form of fine tuning and assess its impact on tracking performance.

Our results show that re-calibrating the output of the HGD models using an affine transformation of their raw covariance outputs can significantly improve log likelihood of true object locations. We show that fine tuning by backpropagating through the Kalman filter further improves performance for most HGD models.

Finally, we show that our custom HGD backbone can provide performance on par with ResNet50 and DETR backbones under the normal lighting condition while simultaneously providing significant improvements in run time latency. Interestingly, we show that in the low-light condition, the DETR-based model has remarkably more robust performance than the other models considered.

The rest of the paper is organized as follows: Section 2 discusses prior work on tracking. Section 3 presents the data set including the data collection process and subsequent data post-processing. Section 4 outlines our modeling framework. Section 5 presents our experimental protocol and evaluation metrics. Section 6 presents results. We conclude with a discussion in Section 7.

2 RELATED WORK

There is considerable prior work applying deep learning to visual object detection and tracking problems. Marvasti-Zadeh et al. [2021] present a survey paper that discusses many of these approaches in depth. In this section we highlight a handful of papers that are of particular relevance to our work, noting that we focus on the multi-view geospatial tracking problem and not the traditional single-view, within-image-plane tracking problem.

Model Architectures: Convolutional neural networks Krizhevsky et al. [2017] provide the backbone in most deep learning object detection and tracking models. In this work, we experiment with a ResNet50 backbone as an example of this class of approach [He et al., 2016]. More recently, vision transformer-based models have shown promising performance on image classification [Liu et al., 2021], object detection [Carion et al., 2020], and within-image-plane object tracking [Meinhardt et al., 2022, Zeng et al., 2022, Yan et al., 2021]. The attention mechanism used in the transformer architecture of Vaswani et al. [2017] has proven to be a powerful tool for reasoning about track identity through time. We consider backbone models based on two transformer architectures.

Multi-view Computer Vision: The task of fusing data from multiple cameras to solve computer vision problems has a long history. One popular application is the use of multiple views to enable 3D object detection in autonomous driving systems, as seen in recent works such as DETR3D [Wang et al., 2022] and Chen et al. [2017]. While some approaches predict object location in an overhead "bird's-eye-view" space, such as Can et al. [2021], these approaches still rely on predicting bounding boxes and do not account for object location uncertainty. More recently, the work of [Li et al., 2022] introduced a novel transformer architecture to fuse multiple camera views to make "bird's-eye-view" predictions. However, this method assumes that camera data can be centralized at no cost and thus relies on early fusion while we focus on the communication constrained distributed camera network setting.

Probabilistic Tracking: Many early approaches to tracking leverage probabilistic models [Pérez et al., 2002]. The Kalman filter is a commonly used approach due to its simplicity [Kalman, 1960]. For example, the Simple Online and Realtime Tracking (SORT) approach of Bewley et al. [2016] converts bounding boxes predicted from an object detector into a track using a Kalman Filter. However, this approach does not make probabilistic predictions for tracked objects and operates within the image plane of a single camera. Danelljan et al. [2020] pose the tracking problem as a probabilistic regression that minimizes KL divergence. However, unlike our approach they consider ground-truth locations that are represented as bounding boxes where as we represent distributions over the centroid of a tracked object.

3 GEOSPATIAL TRACKING DATASET

In this section we present the data collection process and data pre-processing methods used to create our data set.

Data Collection Infrastructure: The data set was collected using RGB video data obtained from a network of four camera nodes. Each camera node included a 1080×1920 resolution ZED 2i camera and a Jetson Xavier NX providing local video compression and storage. The data collection experiments were performed in an indoor motion capture environment measuring 5×7 meters in size. The motion

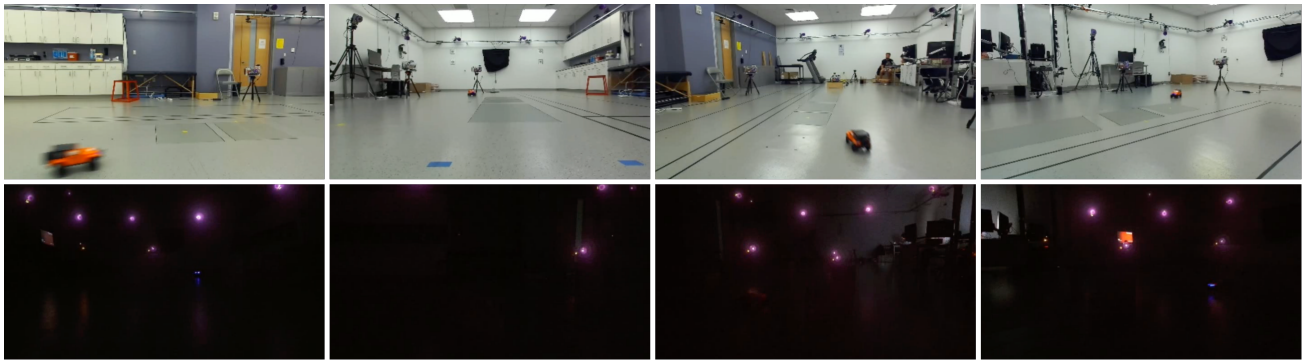


Figure 1: Example data from each of the four cameras in the normal lighting (top) and low lighting scenarios (bottom).

capture system provides high-quality ground truth location and orientation data for tracked objects at a sampling rate of 100Hz. We used a single remote control vehicle (an orange truck) as the tracked object in all data collection experiments. This object is 15×30 cm in size.

The four camera nodes were located on the four sides of the rectangular motion capture environment looking inwards. The ZED camera has a 2.1mm lens, producing a 120-degree field of view. The nodes were positioned such that their fields of view overlapped while no single node captured the whole environment. This allows the tracked object to be out of the field of view of different camera nodes at different times. The locations and orientations of the camera nodes remained constant throughout all data collection experiments.

Tracking Scenarios: Data were collected under two different scenarios. In the first scenario the environment is open (no occluding objects are present) and fully lit. In the second data collection scenario, the environment is minimally lit and occluding objects are also present. While the tracked object has onboard illumination (headlights) that make tracking it plausible under low lighting, the second scenario is expected to be significantly more difficult than the first scenario. Examples from both scenarios are displayed in Figure 1. Each data scenario has a total length of 5 minutes. We split each scenario such that 2.5 minutes are used for training, 30 seconds are used for validation, and 2 minutes are used for testing. Video data are recorded at 15 FPS.

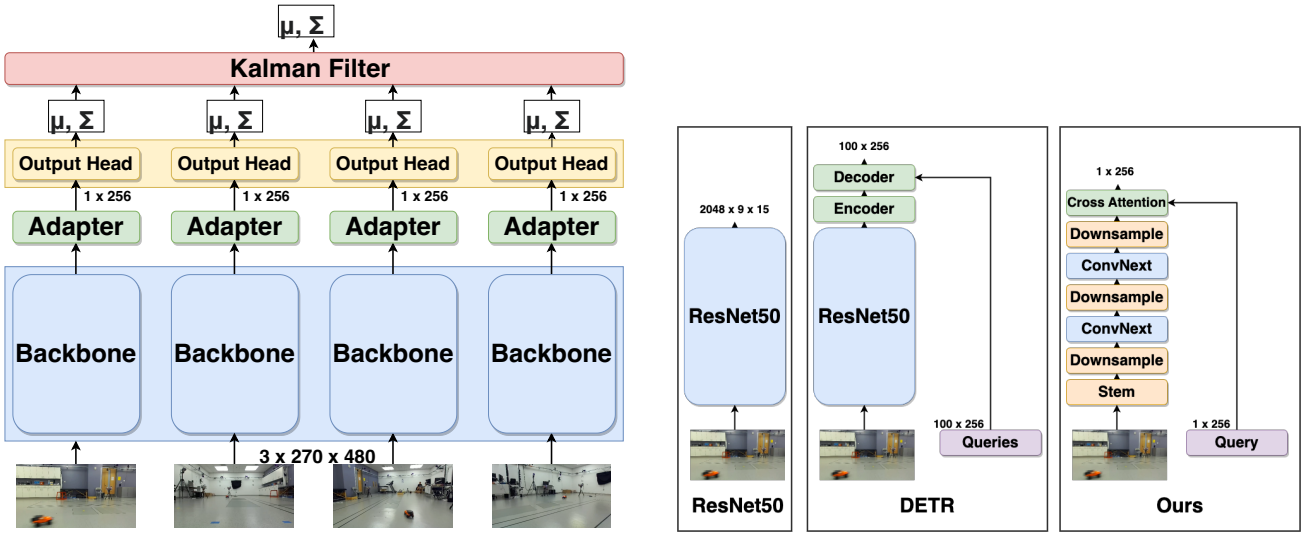
Data Pre-processing: We construct a multi-view tracking data set as a sequence of instances where each instance contains four images (one from each of the four camera nodes) and the corresponding ground truth object location. Given that the cameras are operating independently on the four nodes, we deal with video stream synchronization disparities by initializing a buffer for each video stream as well as for the the ground truth object location stream output by the motion capture system. We update these buffers as the data arrives from each source. We create individual instances in the tracking sequence by taking a snapshot of all buffers at a rate of 20 frames per second. Before learning models, we downsample the image data by dividing each side length by a factor of 4. This results in input images that have size 270×480. No further data augmentation is applied.

4 MODELS FOR HETEROSKEDASTIC GEOSPATIAL TRACKING

In this section we present our proposed heteroskedastic geospatial tracking model framework. The framework consists of two primary components: a set of heteroskedastic geospatial detection (HGD) models and a geospatial tracker (GST) based on a multi-observation Kalman filter.

Overview: An HGD model takes an image from a single view as input and predicts a full covariance normal distribution over the location of the object in geospatial coordinates. Specifically, an HGD model outputs a 2-dimensional mean object location μ and a 2×2 object location covariance matrix Σ. There is one HGD model for each view and all HGD models operate independently. The GST component then filters a set of HGD model outputs in order to generate smoother trajectories that fuse the predictions from each view taking into account their associated object location uncertainties. A block diagram illustrating this framework is presented in Figure 2a.

HGD Models: An HGD model is further divided into three sub-components: a backbone, an adapter, and an output head. The backbone processes the input images in order to generate general higher-level features. The adapter is a small feed-forward network which converts the features into a single 256-dimensional vector which we treat as a latent encoding of the object's position in geospatial coordinates. Finally, the output head converts this encoding into the final low-dimensional geospatial object location mean and covariance representation using an additional feed-forward network. We share the parameters of the backbone and output head across all views. We instantiate a separate adapter for each view. This allows the general backbone features to



(a) (b)

Figure 2: (a) General model architecture. Layers with rectangular background denote model components which are shared across views. (b) Backbone model block diagrams.

be translated into view-specific features.

HGD Backbones: We provide results using two baseline backbone models: ResNet50 [He et al., 2016] and DETR [Carion et al., 2020] as well as a custom backbone. A visual description of the three backbone models is shown in Figure 2b.

ResNet50 is a popular CNN model that was originally designed for image classification [He et al., 2016]. We tap a pretrained model at the last feature layer. This model generates a feature map with height and width downscaled by a factor of 32 relative to the input. The feature map has 2048 channels. The HGD adapter converts the feature map to 256 channels via a feed forward network and then flattens out the spatial dimensions using a linear projection.

DETR is a recent object detector which performs detection end-to-end using transformers [Carion et al., 2020]. It consists of a ResNet50 model followed by a transformer encoder-decoder pair. Using self-attention and cross-attention the model learns a set of 100 query embeddings which act as latent encodings of the predictions. We use the output of the last decoder layer which has shape 100×256. The adapter applies a feed forward network and then applies a linear layer to project the 100 query embeddings to a single prediction (i.e. 1×256). We note that DETR was originally trained for within-image-plane bounding box object detection. Therefore, the adapter has the additional duty of translating this information into the geospatial coordinate space.

Lastly, we introduce an additional custom backbone model for this task. The general design is inspired by DETR, while being much more lightweight. We start with a custom CNN feature extractor. First we copy the stem of a ResNet50 model, which consists of the first convolution layer of the model and a max pooling layer. This stem was trained on ImageNet and is left frozen throughout training and evaluation. We then interleave ConvNext blocks and downsampling operations. The ConvNext block [Liu et al., 2022] is a simple residual CNN block consisting of a depthwise and pointwise convolutional layer. Downsampling is done via convolutional layers with stride 2. In total we apply 3 downsampling layers and 2 ConvNext layers. Along with the stem, this results in just 8 convolutional layers. We maintain a feature dimension of 256 throughout the CNN and output a feature map with height and width downscaled by a factor of 32 relative to the input image.

Associated with this backbone is a single query embedding with 256 dimensions. It is treated as a vector of free parameters. We take the output of the CNN stage and flatten the spatial dimensions into a sequence of pixel values. We then apply cross attention between the query embedding and this pixel sequence. This has the effect of compressing the visual information from the CNN input into a single 256 dimensional vector. This is then fed into an adapter layer.

We present a runtime analysis of these three models in Table 1. We report only the latency of the backbone models without taking into account the latency of the Kalman filter, which is constant for each model and trivial compared to the latency of the deep models. The latencies reported are the time taken to generate a predicted distribution for every view. We see that our custom model is 6 times faster than DETR and 2.5 faster than ResNet50. Run time latencies were measured using an NVIDIA 3080TI GPU.

Table 1: Runtime Results

Model	Latency (ms)	FPS
ResNet50	21.23	47.12
DETR	51.02	19.60
Ours	8.05	124.25

HGD Output Head: The HGD output head is responsible for predicting the mean μ and covariance matrix Σ of the Gaussian distribution that represents the location of the object. Given a 256-dimensional encoding, it outputs five values. The first two values determine the mean location μ. In our experiments, we apply a sigmoid activation to these values and then scale them by 500 and 700 so that they correspond to a location in centimeters within the tracking environment's $5\times$ 7 meter area.

The remaining three output values are used to predict the covariance matrix Σ. We apply a softplus function to two of the values which are taken to be the diagonal elements of Σ. The remaining value is then taken to be the off-diagonal element. Finally, we multiply this intermediate matrix with its transpose to ensure that Σ is positive definite. For numerical stability reasons, we add the identity matrix I to the output. This prevents Σ from collapsing to 0 during learning.

GST Model: For the GST model, we use a constant-velocity multi-observation Kalman filter-based tracker Kalman [1960], Baisa [2020]. A Kalman filter-based tracker is a dynamic Bayesian network model with Gaussian distributed latent state. In the constant velocity Kalman tracker, these latent variables represent the location and velocity of the object being tracked. A detection is modeled as a noisy observation given the latent variables in the tracker. Posterior inference is used to update the tracker's latent state given a detection.

In a multi-observation Kalman filter, any number of simultaneous detections can be used to update the latent state of the tracker under the assumption that they are all mutually conditionally independent given the latent state of the tracker. The posterior inference process fuses multiple simultaneous observations while automatically putting more weight on the detections that have lower associated uncertainty. An example of this effect can be seen in Figure 3.

HGD and GST Training: We train the HGD models to minimize of the negative log likelihood of the ground-truth object positions under the distribution output by the model. All views are trained simultaneously with a separate loss contribution for each view. All models are trained for 50 epochs. We use the AdamW optimizer [Loshchilov and Hutter, 2019] with an initial learning rate of 10^{-4} and weight decay of 10^{-4}. At the 40th epoch, the learning rate is divided by 10. We train on 8 GPUs each with 4 samples per batch for an effective batch size of 32. Batches are chosen randomly across the training sequence as temporal information is not used during training of the HGD models. The gradient is clipped such that it's L2 norm is less than 0.1. We start from a ResNet 50 model pretrained on ImageNet [Deng et al., 2009] and a DETR model pretrained on the COCO detection dataset [Lin et al., 2014]. Parameters of the baseline models are fine-tuned during training of the detector.

In the experiments where we backprop through the Kalman Filter, a batch size of 1 is used on each GPU for an effective batch size of 8. We use time aligned sequences of length 100 for this portion of training. When training with the Kalman Filter we use detectors that are already trained on the task. The backbones are frozen during the Kalman Filter training while the paramaeters of the Kalman Filter, output head and adapters are fine-tuned. All learning hyperparameters are the same for GST and HGD training except for the training schedule, which is decreased in length by a factor of 10 (we train for 5 epochs and reduce learning rate at 4th epoch).

5 EVALUATION METRICS

In this section we present metrics for evaluating the performance of models that solve the heteroskedastic geospatial object tracking problem.

Negative Log Likelihood (NLL): Since any model that solves the heteroskedastic geospatial object tracking problem must output a probability distribution over the locations of tracked objects by definition, evaluating the quality of such a model's output using a likelihood-based evaluation metric is natural. In this work, all tracking models output Gaussian distributions over object location via the distribution over the Kalman tracker's latent state. We specifically evaluate the negative log likelihood of the Kalman tracker's marginal distribution over the object's location. Note that the individual heteroskedastic detection models also output probabilistic detections, and thus their performance can also be evaluated using negative log likelihood.

Object Probability Mass (OPM): Within image plane tracking problems typically use object bounding boxes as the ground-truth objection location representation and within image place tracking models typically also output bounding boxes. This leads applying an Intersection-over-Union (IoU) metric to true and predicted bounding boxes as a natural performance metric.

Under the assumption that it is sufficient for the probability distribution of an object's location to concentrate within the geospatial extent of the object, we can derive a related metric for the heteroskedastic geospatial tracking problem by computing the predicted probability mass that falls within the ground truth extent of an object.

Given a predicted distribution with mean μ and covariance

Table 2: Baseline tracking results

	Normal Light				Low Light			
Backbone	NLL	OPM	DetPr	LocA	NLL	OPM	DetPr	LocA
ResNet50	6.598	0.959	0.959	0.997	19.675	0.469	0.470	0.900
DETR	7.149	0.985	0.987	0.990	6.834	0.917	0.919	0.978
Ours	8.624	0.958	0.960	0.993	35.338	0.644	0.646	0.947

Σ along with the ground truth location, orientation, and extent of the object being tracked, we compute the OPM using a Monte Carlo estimate of the underlying integral that defines the desired probability mass. We sample 1000 points from the predicted distribution and check what percentage of them fall inside the rectangle defining the object's physical extent. A visual depiction of the application of this metric is shown in Figure 1 in the supplemental material.

Tracking Metrics: Like IoU, OPM has the useful propriety of being in the range $[0, 1]$, with 1 being a perfect score. This allows us to use OPM as a drop-in replacement for IoU when using evaluation code for tracking metrics. In particular, we use evaluation code for the Higher Order Tracking Accuracy (HOTA) suite of tracking metrics [Luiten et al., 2021].

In the HOTA metrics, a bijective mapping between predicted and ground-truth objects is computed. The number of true positives ($|TP|$) is the number of matched pairs. The number of unmatched predictions is $|FP|$. The number of unmatched ground-truth objects is $|FN|$. A match is only considered valid if the similarity score (e.g. IoU) is greater than a given threshold α. We can then compute the detection recall and precision ($DetRe$ and $DetPr$) for a single α value as shown below in Equation 1 and 2. These values are computed for a range of α values and the mean is reported.

$$DetPr_\alpha = \frac{|TP|}{|TP| + |FN|} \tag{1}$$

$$DetRe_\alpha = \frac{|TP|}{|TP| + |FP|} \tag{2}$$

In our data set, there is one ground-truth object and prediction per time step. Therefore, the only kind of error that can occur is that the detection results in a similarity score less than the threshold α. This results in an unmatched ground truth object and an unmatched prediction, so $|FN| = |FP|$. This implies that $DetPr = DetRe$. We therefore chose to report $DetPr$ only. We reinterpret this metric to be the percent of timesteps with OMP greater than α, averaged across a range of α values. That is, it measures how often the tracker is on track.

We additionally report the location accuracy (LocA) metric from the HOTA suite. LocA can be defined as shown below in Equation 3 where c is a valid ground-truth/prediction pair. Put simply, $LocA_\alpha$ is the average similarity score for pairs that are within the α threshold. That is, it measures the tracker's localization performance when it is on track. We again note that an individual heteroskedastic geospatial detection model can also be used to derive these metrics in the single object tracking case.

$$LocA_\alpha = \frac{1}{|TP_\alpha|} \sum_{c \in TP_\alpha} OPM(c) \tag{3}$$

6 RESULTS

In this section we report the results of tracking experiments. We use the two data scenarios described in Section 3 (normal light vs. low light), the three backbone models described in Section 4 (ResNet50, DETR, and our custom model), and the evaluation metrics described in Section 5 (NLL, OPM, DetPr, LocA). The supplemental material includes a table of all results shown.

Experiment 1 - Baseline Performance: We present baseline performance results for both data scenarios in Table 2. These results evaluate the performance of the Kalman filter-based geospatial tracking (GST) model coupled with heteroskedastic geospatial detection (HGD) models based on each backbone. We label each model by the backbone used as the other components are identical across models.

In terms of NLL, we see that results are fairly mixed apart from the result that the DETR backbone achieves much better NLL in the low light setting. In terms of OPM and the tracking metrics, we see that the DETR backbone outperforms the other backbones in both settings. It shows especially strong performance relative to the other models in the visually challenging low light setting. One possible explanation for this is that DETR is uniquely able to fuse global information by applying self-attention between all pixels in the final feature map. In the easier data scenario with normal light, we see that our more efficient model is competitive with the ResNet50 baseline in terms of OPM and tracking metrics.

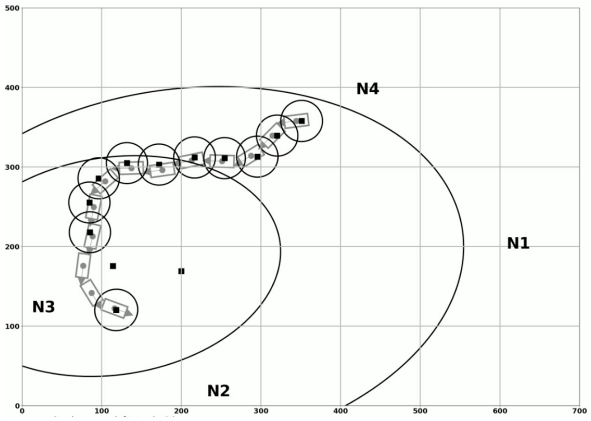
Experiment 2 - Effect of Kalman Filtering: One benefit of using a late fusion approach is that each HGD model can function independently without the need of the centralized fusion process provided by the Kalman Filter. In Tables 3 and 4 we compare the average performance of the four HGD models compared to the performance of the baseline Kalman Filter model.

Table 3: Normal Light, Detectors vs. Tracker

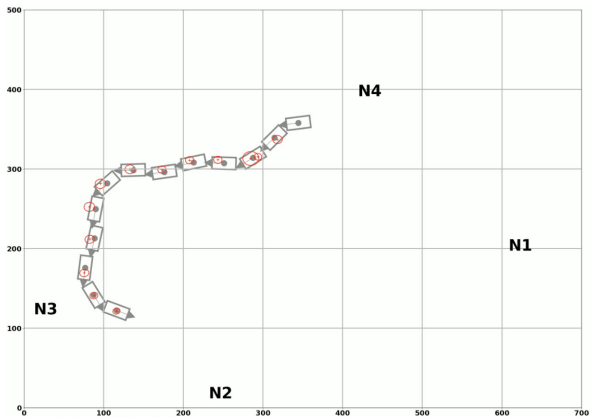
	NLL		OPM	
Backbone	Detector	Tracker	Detector	Tracker
ResNet50	6.361	6.598	0.803	0.959
DETR	5.288	7.149	0.819	0.985
Ours	7.873	8.624	0.701	0.958

Table 4: Low Light, Detectors vs. Tracker

	NLL		OPM	
Backbone	Detector	Tracker	Detector	Tracker
ResNet50	11.329	19.675	0.197	0.469
DETR	7.484	6.834	0.595	0.917
Ours	17.489	35.338	0.308	0.644



(a) Example trajectory from single view detector.



(b) Example trajectory from multi-view tracker.

Figure 3: Example trajectories. The gray rectangles denote the location of the object being tracked with an arrow indicating the object's current heading. The location of the camera nodes are labeled as N1 to N4. The left-hand plot shows detections from the custom backbone using data from node N2. The black squares show the predicted mean location with an associated 95% confidence ellipse. The right-hand plot shows the same track with multi-observation Kalman Filter output shown in red. We see that for two time points the single-view detector predicts a mean that is off track with high uncertainty. This occurs when the object is not viewable by node N2. However, the tracker remains on tracker as it fuses data from the other nodes that can see the object and have much higher confidence.

We can see that using the tracker leads to much better performance on the OPM metric. This suggests that the individual HGD models are producing higher uncertainty and larger covariance than the Kalman tracker output. After applying the tracker, the uncertainty is reduced and the covariance is significantly smaller. This can be seen in Figure 3. This results in a higher OPM score as the OPM metric is highly sensitive to the scaling of the covariance matrix. However, we see that the average NLL across the four HGD models per backbone is actually better than the NLL provided by the Kalman tracker in all but one case. This suggests that from a predictive likelihood standpoint, the baseline Kalman tracker is actually under representing uncertainty. We address this issue in the next experiments.

Experiment 3 - Post-Hoc Model Recalibration: As noted in the previous section, the end-to-end output of the Kalman tracker under represents uncertainty in a predictive log likelihood sense. As a first approach to addressing this issue, we consider post-hoc recalibration of the output of the individual HGD models. Specifically, we consider an affine transformation of the covariance matrix output by an HGD model: $\Sigma' = a\Sigma + bI$. This transformation inflates the covariance matrix for parameters $a > 1$ and $b > 0$.

To select the values of a and b we perform a grid search over $a \in [0.05 \ldots 10]$ and $b \in [0 \ldots 500]$. We assess the performance of each re-scaled distribution on a validation set and select the a, b which minimize validation set NLL.

The results of this calibration experiment are shown in Table 5. We report the NLL of the Kalman Tracker using uncalibrated and calibrated input from the HGD models. Comparing the calibrated models to the uncalibrated models, we see consistent improvements in terms of NLL. We see particularly large improvements in the visually challenging low light scenario. On the easier normal light scenario, we see that our customized model slightly out performs the DETR baseline. However the ResNet50 baseline is still the strongest model in this scenario.

We further generate histograms of the NLL values with and without calibration across all time steps in the test data set. These results are displayed for all backbones in Figure 4. We can see that the effect of calibration is to increase the minimum NLL somewhat for all models, while significantly reducing the upper tail of the NLL distribution.

Table 5: Calibration Results (NLL).

	Normal Light		Low Light	
Backbone	Uncalib.	Calib.	Uncalib.	Calib.
ResNet50	6.598	4.537	19.675	7.200
DETR	7.149	5.465	6.834	4.803
Ours	8.624	5.364	35.338	7.419

Table 6: Results with/without Kalman Filter tuning (NLL)

	Normal Light		Low Light	
Backbone	without	with	without	with
ResNet50	4.537	4.510	7.200	7.360
DETR	5.465	5.980	4.803	5.002
Ours	5.364	4.973	7.419	7.240

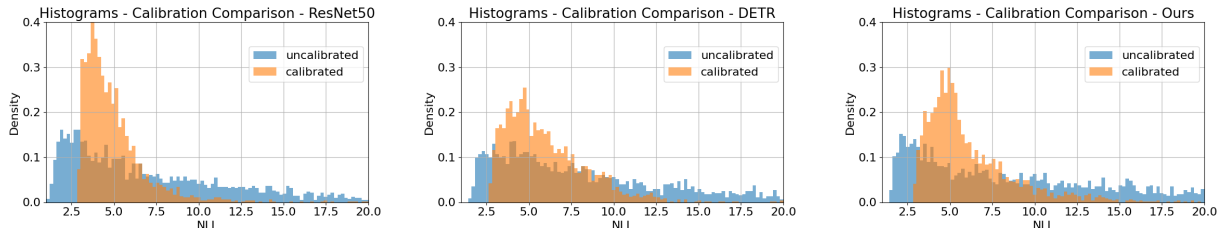


Figure 4: NLL Histograms pre- and post-calibration. Plots are generated using all test set data cases.

Overall we find that this form of calibration is an easy way to improve the NLL of the tracker. This is convenient as it incurs almost no additional runtime latency compared to the complexity of executing the HGD backbones.

Experiment 4 - Kalman Tracker Fine Tuning: Inference in the Kalman tracker is fully differentiable. It is thus possible to learn while backpropagating the Kalman filter's NLL through both the parameters of the Kalman filter and the parameters of individual HGD models. We refer to this approach as Kalman filter fine tuning.

In our implementation of the constant velocity Kalman tracker, the standard deviation of acceleration parameter is the only learnable parameter. However, this parameter is important as it determines how much the Kalman filter's uncertainty over its internal latent variables increases between observations, which has implications both for the uncertainty in the tracker's output and the degree of smoothness in the output track.

Further, Kalman filter fine tuning allows the output heads and adapters inside the HGD models to adjust themselves to the tracking scenario (for computational complexity reasons, we do not consider full end-to-end training of the backbones). We present results for this setting in Table 6. The results reported are using calibrated HGD models as described in the previous section. The reported NLLs are computed with respect to the Kalman tracker's output distribution.

We see that applying Kalman filter fine tuning results in an improvement in out custom model on both data scenarios. Surprisingly, the DETR baseline sees worse NLL performance on both data settings. After calibrating and training through the Kalman Filter, our custom model is able to achieve better NLL score than DETR in the normal light scenario at six times lower latency. It is also able to close to within 0.5 nats of the performance achieved by the ResNet50 backbone in the normal light scenario at 2.5 times lower latency. Lastly, despite the fact that the DETR-based model does not improve in performance under Kalman filter fine tuning, it remains the best performing model in the low light scenario by a wide margin.

7 CONCLUSIONS

In this paper, we focus on the geospatial object tracking problem using data from a distributed camera network. The goal is to predict an object's track in geospatial coordinates along with uncertainty over the object's location. We have presented a novel single-object geospatial tracking data set to support this work that includes high-accuracy ground truth object locations and video data from a network of four cameras captured under normal and low light conditions.

We have presented a modeling framework for addressing this task that leverages a multi-observation Kalman filter-based tracker in conjunction with a set of independent heteroskedastic geospatial detection models. This framework is specifically motivated by a communication constrained version of the tracking problem where it is too expensive to centralize raw image data and thus late fusion of low dimensional representations is required. We present a custom detection backbone with significantly reduced prediction latency relative to state-of-the-art models and show that it can achieve strong performance in the normal lighting setting when tuned specifically to optimize tracking performance via a combination of post-hoc recalibration and Kalman filter based fine tuning. Our result show that a version of the proposed framework using the DETR backbone achieves superior performance in the low light setting, a finding that requires further exploration.

References

Nathanael L Baisa. Derivation of a constant velocity motion model for visual tracking. *arXiv preprint arXiv:2005.00844*, 2020.

Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, pages 3464–3468, 2016.

Yigit Baran Can, Alexander Liniger, Danda Pani Paudel, and Luc Van Gool. Structured bird's-eye-view traffic scene understanding from onboard images. In *ICCV*, pages 15661–15670, October 2021.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229. Springer, 2020.

Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, pages 1907–1915, 2017.

Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *CVPR*, June 2020.

Parul Datta and Bhisham Sharma. A survey on iot architectures, protocols, security and smart city based applications. In *2017 8th international conference on computing, communication and networking technologies (ICCCNT)*, pages 1–5. IEEE, 2017.

P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv:2003.09003[cs]*, March 2020. arXiv: 2003.09003.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *ECCV*, 2022.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CVPR*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International journal of computer vision*, 129:548–578, 2021.

Seyed Mojtaba Marvasti-Zadeh, Li Cheng, Hossein Ghanei-Yakhdan, and Shohreh Kasaei. Deep learning for visual tracking: A comprehensive survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *CVPR*, pages 8844–8854, 2022.

Felisberto Pereira, Ricardo Correia, Pedro Pinho, Sérgio I Lopes, and Nuno Borges Carvalho. Challenges in resource-constrained iot devices: Energy and communication as critical success factors for future iot deployment. *Sensors*, 20(22):6420, 2020.

Patrick Pérez, Carine Hue, Jaco Vermaak, and Michel Gangnet. Color-based probabilistic tracking. In *ECCV*, pages 661–675, 2002.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, pages 180–191. PMLR, 2022.

Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *ICCV*, pages 10448–10457, 2021.

Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer. In *ECCV*, pages 659–675, 2022.