Foveated Attention for Neural Nets

Chittesh Thavamani

CMU-CS-22-129

August 2022

Thesis Committee:
Deva Ramanan (Chair)
Deepak Pathak

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements
for the degree of Master of Science.

This work was supported by the CMU Argo AI Center for Autonomous Vehicle Research.

Keywords: Attention, Fovea, Retina, Foveation, Adaptive Downsampling, Dynamic Neural Networks, Object Detection, Autonomous Navigation, Warp Inversion, Streaming Perception, 3D Object Detection, Semantic Segmentation, COCO, Argoverse, Cityscapes, NuScenes, BDD100K, Faster-RCNN, PSP-Net, RetinaNet, YOLOF, FCOS3D

For my loving parents and super uber cool brother

Abstract

Efficient processing of high-res video streams is safety-critical for many robotics applications such as autonomous driving. To maintain real-time performance, many practical systems downsample the video stream. But this can hurt downstream tasks such as (small) object detection. Instead, we take inspiration from biological vision systems that allocate more foveal “pixels" to salient parts of the scene. We introduce FOVEA, an approach for intelligent downsampling that ensures salient image regions remain “magnified" in the downsampled output. Given a high-res image, FOVEA applies a differentiable resampling layer that outputs a small fixed-size image canvas, which is then processed with an object detector, whose output is then differentiably backward mapped onto the original image size. In order to maintain overall efficiency, FOVEA makes use of cheap and readily available saliency cues, including dataset-specific spatial priors or temporal priors computed from recent object predictions. On the autonomous driving datasets Argoverse-HD and BDD100K, our proposed method boosts the detection AP over standard Faster-RCNN, both with and without finetuning. Without any noticeable increase in compute, we improve accuracy on small objects by over 2x without degrading performance on large objects. Finally, FOVEA sets a new record for streaming AP (from 17.8 to 23.0 on a GTX 1080 Ti GPU), a metric designed to capture both accuracy and latency. However, FOVEA is designed specifically for 2D object detection. To generalize to arbitrary spatial tasks, in our followup work, we "learn to zoom" in on the input image, compute spatial features, and then "unzoom" to revert any deformations (LZU). To enable efficient and differentiable unzooming, we approximate the zooming warp with a piecewise bilinear mapping that is invertible. LZU can be applied to any task with spatial input and any model with spatial features, and we demonstrate this versatility by evaluating on a variety of tasks and datasets: *object detection* on Argoverse-HD and a synthetic video COCO, *semantic segmentation* on Cityscapes, and *RGB-based 3D detection* on NuScenes. Interestingly, we observe boosts in performance even when high-resolution sensor data is unavailable, implying that LZU can be used to "learn to upsample" as well.

Acknowledgments

First of all, I want to thank to my amazing mentors Mengtian Li and Professor Deva Ramanan. Also, this work was supported by the CMU Argo AI Center for Autonomous Vehicle Research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Safety-critical robotic agents such as self-driving cars make use of an enormous suite of high-resolution perceptual sensors, with the goal of minimizing blind spots, maximizing perception range, and ensuring redundancy [4, 6, 49]. We argue that "over-sensed" perception platforms provide unique challenges for vision algorithms since those visual sensors must rapidly consume sensor streams while continuously reporting back the state of the world. While numerous techniques exist to make a particular model run fast, such as quantization [54], model compression [10], and inference optimization [42], at the end of the day, simple approaches that subsample sensor data (both spatially by frame downsampling and temporally by frame dropping) are still most effective for meeting latency constraints [30]. However, subsampling clearly throws away information, negating the goals of high-resolution sensing in the first place! This status quo calls for novel vision algorithms.

To address this challenge, we take inspiration from the human visual system; biological vision makes fundamental use of *attentional* processing. While current sensing stacks make use of regular grid sampling, the human vision system in the periphery has a much lower resolution than in the center (fovea), due to the pooling of information from retinal receptors by retinal ganglion cells. Such variable resolution is commonly known as foveal vision [26].

In this thesis, first, we propose FOVEAted image magnification (FOVEA) for object detection, which retains high resolution for objects of interest while maintaining a small canvas size. We exploit the sparsity of detection datasets – objects of interest usually only cover a portion of the image. *The key idea is to resample such that background pixels can make room for salient pixels of interest.* The input images are downsampled and warped such that salient areas in the warped image have higher resolutions. While image warping has been explored for image classification [22, 43] and regression [43], major challenges remain when applying such methods to detailed spatial prediction tasks such as object detection. First, processing warped images will produce warped spatial predictions (bounding box coordinates). We make use of differentiable backward maps to unwarp spatial predictions back to the original space. Second, it is hard to efficiently identify salient regions; in the worst case, a saliency network tuned for object detection may be as expensive as the downstream detection network itself, thereby eliminating any win from downsampling. In our case, we make use of cheap and readily available saliency cues, either in the form of dataset-specific spatial priors (i.e., small objects tend to exist near a fixed horizon) or temporal priors (small objects tend to lie nearby small object predictions from previous frames).

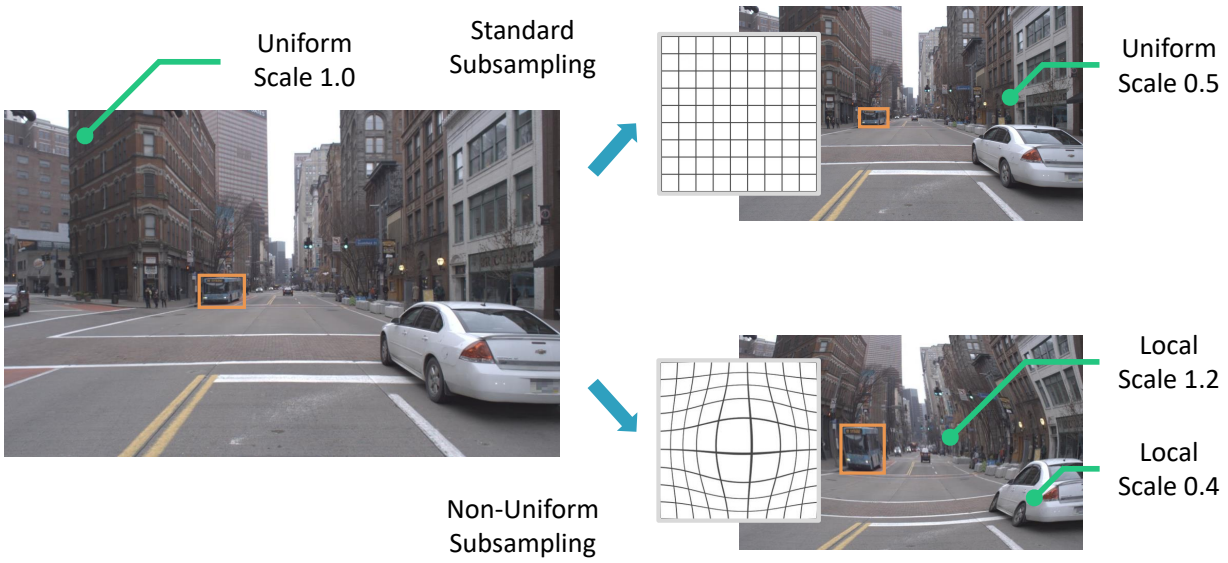


Figure 1.1: Standard image downsampling (top right) limits the capability of the object detector to find small objects. In this thesis, we propose an attentional warping method FOVEA (bottom right) that enlarges salient objects in the image while maintaining a small input resolution. Challenges arise when warping also alters the output labels (*e.g.*, bounding boxes).

Third, previous image warps (tuned for image classification tasks) can produce cropped image outputs. Since objects can appear near the image boundary, we introduce anti-cropping constraints on the warping.

We validate our approach on two self-driving datasets for 2D object detection: Argoverse-HD [30] and BDD100K [60]. First, we show that FOVEA can improve the performance of off-the-shelf detectors (Faster R-CNN [45]). Next, we finetune detectors with differentiable image warping and backward label mapping, further boosting performance. In both cases, small objects improve by more than 2x in average precision (AP). Finally, we evaluate FOVEA under streaming perception metrics designed to capture both accuracy and latency [30], producing state-of-the-art results.

Then, we generalize our method to arbitrary tasks with spatial labels. This is trickier, but has been accomplished in followup works for semantic segmentation (LDS [23]) and object detection (FOVEA [51]). LDS [23] does not unzoom during learning, and so defines losses in the warped space. This requires additional regularization that may not apply to non pixel-dense tasks such as detection. FOVEA [51] *does* train 2D detectors by unzooming bounding boxes, but this is a special purpose solution that avoids computing an inverse, making it inapplicable to tasks such as semantic segmentation. Notwithstanding these otherwise elegant solutions, there does not appear to be a general solution for learning to intelligently downsample across spatial tasks.

Our primary contribution is a general framework in which we "zoom" into an input image, process the zoomed image, and then *un*zoom the output back with an inverse warp. "Learning to zoom and unzoom" (LZU) can be applied to *any* network that makes use of 2D spatial features

[1]According to the reference implementation of [23], the forward "zoom" takes 3.4ms, and the inference-time inversion takes 70.126s. We suspect that inversion is not optimized, so we provide a lower bound on the latency.

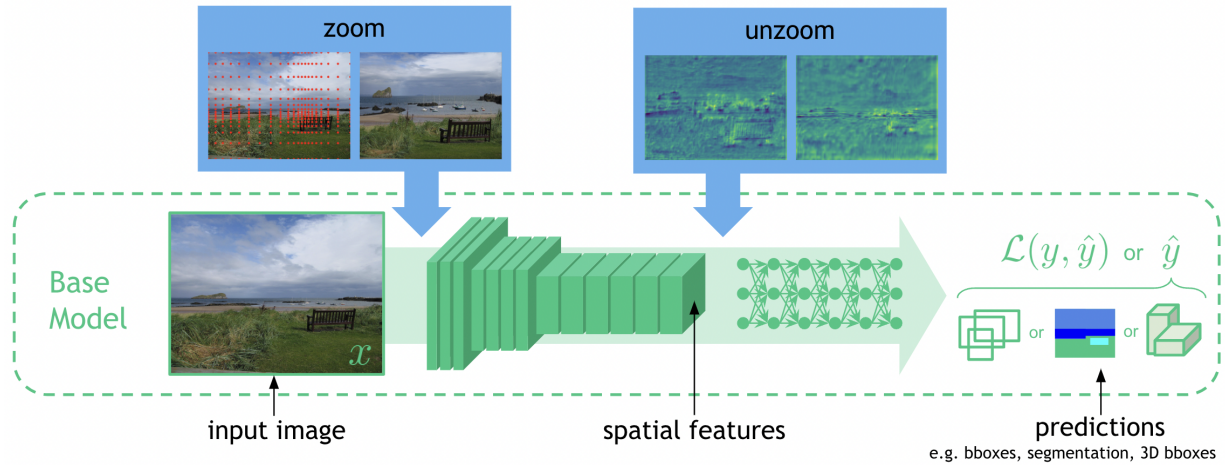


Figure 1.2: LZU is characterized by "zooming" the input image, computing spatial features, then "unzooming" to revert spatial deformations. LZU can be applied to any task and model that makes use of internal 2D features to process 2D inputs. We show visual examples of output tasks including 2D detection, semantic segmentation, and 3D detection from monocular images.

to process 2D spatial inputs (Fig. 1.2) *with no adjustments to the network or loss*. To unzoom, we approximate the zooming warp using a piecewise bilinear mapping. This allows efficient and differentiable computation of the forward and inverse warps.

To demonstrate the generalizability of LZU, we explore a variety of tasks: *object detection* with RetinaNet [36] on Argoverse-HD [29] and a synthetic video COCO [34], *semantic segmentation* with PSPNet[61] on Cityscapes [13], and *RGB-based 3D detection* with FCOS3D [57] on NuScenes [5]. In our experiments, to maintain favorable accuracy-latency tradeoffs, we use cheap sources of saliency (as in [51]) when determining where to zoom. On each task and dataset, LZU increases performance over uniform downsampling and prior works with minimal additional latency, as shown in Table 1.1.

Interestingly, for both 2D and 3D object detection, we also see performance boosts even when processing low resolution input data. While prior works focus on performance improvements via intelligent downsampling [43, 51], our results show that LZU can also improve performance by intelligently *up*sampling (suggesting that current networks struggle to remain scale invariant for small objects, a well-known observation in the detection community [34]).

Table 1.1: Summary of task-agnostic generalization of LZU. For each of the below tasks, we evaluate LZU and compare to a uniform downsampling baseline and prior work. We also train a soft uniform downsampling "upper bound" at a higher input resolution. For segmentation, our baseline's performance (55.4) varies from the reported performance in [23] (54). Details on the choice of dataset, model, and input resolution are provided in Tables 6.1, 6.3, and 6.4.

	2D Object Detection		Semantic Segmentation		3D Detection	
Method	Acc (mAP)	Lat (ms)	Acc (mIoU)	Lat (ms)	Acc (NDS)	Lat (ms)
Baseline	22.6	45.8	54 / 55.4	27.0	27.5	58.3
Prior Work	24.9 [51]	47.7	54 $\rightarrow$ 55 [23]	$\geq$30.4 [1]	–	–
LZU	25.3	49.7	55.4 $\rightarrow$ 56.8	29.3	29.3	58.7
Upper Bound	29.5	68.5	55.4 $\rightarrow$ 65.3	42.5	31.2	87.9

Chapter 2

Related Work

In this section, we outline related works for both our methods FOVEA and LZU.

2.1 Related Works for FOVEA

We split related works for FOVEA into three sections. The first elaborates on the well-established problem of object detection in computer vision. The second describes a class of approaches dealing with online perception, and the last outlines methods in computer vision utilizing visual attention to achieve better performance.

2.1.1 Object detection

Object detection is one of the most fundamental problems in computer vision. Many methods have pushed the state-of-the-art in detection accuracy [7, 17, 35, 41, 45], and many others aim for improving the efficiency of the detectors [3, 38, 44, 50]. The introduction of fully convolution processing [48] and spatial pyramid pooling [18] have allowed us to process the input image in its original size and shape. However, it is still a common practice to downsample the input image for efficiency purposes. Efficiency becomes a more prominent issue when people move to the video domain. In video object detection, the focus has been on how to make use of temporal information to reduce the number of detectors invoked [39, 62, 63]. These methods work well on simple datasets like ImageNet VID [47], but might be unsuitable for the self-driving car senarios, where multiple new objects appear at almost every frame. Furthermore, those methods are usually designed to work in the offline fashion, *i.e.*, allowing access to future frames. Detection methods are the building blocks of our framework, and our proposed approach is largely agnostic to any particular detector.

2.1.2 Online/streaming perception

In the online setting, the algorithm must work without future knowledge. [33] proposes the Temporal Shift Module that enables video understanding through channel shifting and in the online setting, the shifting is restricted to be uni-directional. [2] proposes a multi-object tracking

method that takes input previous frame detection as addition proposals for the current frame. FOVEA also takes previous frame detection as input, but we use that to guide image warping. Streaming accuracy [30] is a recently proposed metric that evaluates the output of a perception algorithm at all time instants, forcing the algorithm to consider the amount of streaming data that must be ignored while computation is occuring. [30] demonstrates that streaming object detection accuracy can be significantly improved by tuning the input frame resolution and framerate. In this work, we demonstrate that adaptive attentional processing is an orthogonal dimension for improving streaming performance.

2.1.3 Adaptive visual attention

Attentional processing has been well studied in the vision community, and it appears in different forms [14, 21, 24, 32, 37, 55]. Specially in this thesis, we focus on dynamic resolutions. For image classification, [53] designs an algorithm to select high-resolution patches, assuming each patch is associated with a data acquisition cost. [40] applies non-uniform downsampling to semantic segmentation and relies on the network to learn both the forward and backward mapping, whose consistency is not guaranteed. For object detection, a dynamic zoom-in algorithm is proposed that processes high-resolution patches sequentially [16]. However, sequential execution might not meet latency requirements for real-time applications. Most similar to our work, [43] proposes an adaptive image sampling strategy that allocates more pixels for salient areas, allowing a better downstream task performance. But the method only works for image classification and regression, where the output is agnostic to the input transformation.

2.2 Related Work for LZU

We split related work for LZU into two sections. The first discusses the broad class of methods aiming to improve efficiency by paying "attention" to specific image regions. The second delves into works like LZU that accomplish this by differentiably resampling.

2.2.1 Spatial Attentional Processing

By design, convolutional neural networks (CNNs) pay equal "attention" (perform the same computations) to all regions of the image. In many cases, this is suboptimal, and much work has gone into developing attentional methods that resolve this inefficiency.

One such method is Dynamic Convolutions [55], an alternative to vanilla convolutions that uses sparse convolutions to selectively compute outputs at only the salient regions. Similarly gated convolutions are used in [25, 59]. Notably, these all use "hard" attention in that the saliency is binary, and non-salient regions are ignored completely.

Deformable Convolutions [14, 64] provides a softer implementation of spatial attention by learning per pixel offsets when applying convolutions, allowing each output pixel to attend adaptively to pixels in the input image. SegBlocks [56] also provides a softer attention mechanism by splitting the image into blocks and training a lightweight reinforcement learning policy to determine whether each block should be processed at a high or low resolution. This is akin to LZU,

which also has variable resolution, albeit in a more continuous manner. LZU is also generalizable to tasks in which it's infeasible to "stitch" together outputs from different blocks of the image (e.g. in detection where an object can span multiple blocks).

2.2.2 Spatial Attention via Differentiable Image Resampling

Spatial Transformer Networks [22] introduces a differentiable method to resample an image. They originally propose this to invert changes in appearance due to viewpoint, thereby enforcing better pose invariance.

Learning to Zoom (LZ) [43] later adapts this resampling operation to "zoom" on salient image regions, acting as a spatial attention mechanism. Their key contribution is a transformation parameterized by a saliency map such that regions with higher saliency are more densely sampled. However, this deforms the image, requiring the task to have non-spatial labels.

Followup works [23, 51] adapt LZ downsampling to detection and semantic segmentation. For object detection, FOVEA [51] exploits the fact that image resampling is implemented via an inverse mapping to map predicted bounding boxes back into the original image space. This allows all processing to be done in the downsampled space and the final bounding box regression loss to be computed in the original space. However, when there are intermediate losses, as is the case with two-stage detectors containing region proposal networks (RPNs) [45], this requires more complex modifications to the usual delta loss formulation, due to the irreversibility of the inverse mapping. For semantic segmentation, Jin et al. [23] applies LZ downsampling to both the input image and the ground truth and computes the loss in the downsampled space. This is elegant and model-agnostic but leads to misalignment between the training objective and the desired evaluation metric. In the extreme case, the model learns degenerate warps that sample "easy" parts of the image to reduce the training loss, and to address this, they introduce additional regularization on the downsampler. Independently, [40] handcraft an energy minimization formulation to sample more densely at semantic boundaries.

In terms of warping and unwarping, the closest approach to ours is Dense Transformer Networks [27], which also inverts deformations introduced by nonuniform resampling. However, their warping formulation is not saliency-based, which makes it hard to work with spatial or temporal priors and also makes it time-consuming to produce the warping parameters. Additionally, they only show results for semantic segmentation, whereas we show that our formulation generalizes across spatial vision tasks.

Chapter 3

FOVEA [51]

Assume we are given a training set of image-label pairs (I, L). We wish to learn a nonlinear deep predictor f that produces a low loss $\mathcal{L}(f(I), L)$. Inspired by past work [22, 43], we observe that certain labeling tasks can be performed more effectively by warping/resampling the input image. However, when the label L itself is spatially defined (e.g., bounding box coordinates or semantic pixel labels), the label itself may need to be warped, or alternatively, the output of the deep predictor may need to be inverse-warped.

In this section, we first introduce the saliency-guided spatial transform from related work as the foundation of our method. Next, we introduce our solutions to address the challenges in image warping for object detection. An overview of FOVEA, our method, is shown in Fig 3.1.

3.1 Background: Saliency-Guided Spatial Transform

The seminal work of spatial transformer networks (STN) introduces a differentiable warping layer for input images and feature maps [22]. It was later extended to incorporate a saliency map to guide the warping [43]. Here we provide implementation details that are crucial to FOVEA. Please refer to the original papers [22, 43] for more details.

A 2D transformation can be written as:

$$\mathcal{T} : (x, y) \rightarrow (x', y'), \tag{3.1}$$

where (x, y) and (x', y') are the input and output coordinates. Since image pixels are usually discrete, interpolation is required to sample values at non-integral coordinates. An image warp $\mathcal{W}_\mathcal{T}$ takes input an image I, samples the pixel values according to the given transformation $\mathcal{T}$, and outputs the warped image I':

$$I'(\mathcal{T}(x, y)) = I(x, y) \tag{3.2}$$

Naive forward warping of discrete pixel locations from input I can result in non-integral target pixel positions that need to be "splatted" onto the pixel grid of I, which can produce artifacts such as holes. Instead, image warps are routinely implemented via a *backward map* [1]: iterate over each output pixel grid location, compute its *inverse mapping* $\mathcal{T}^{-1}$ to find its corresponding input

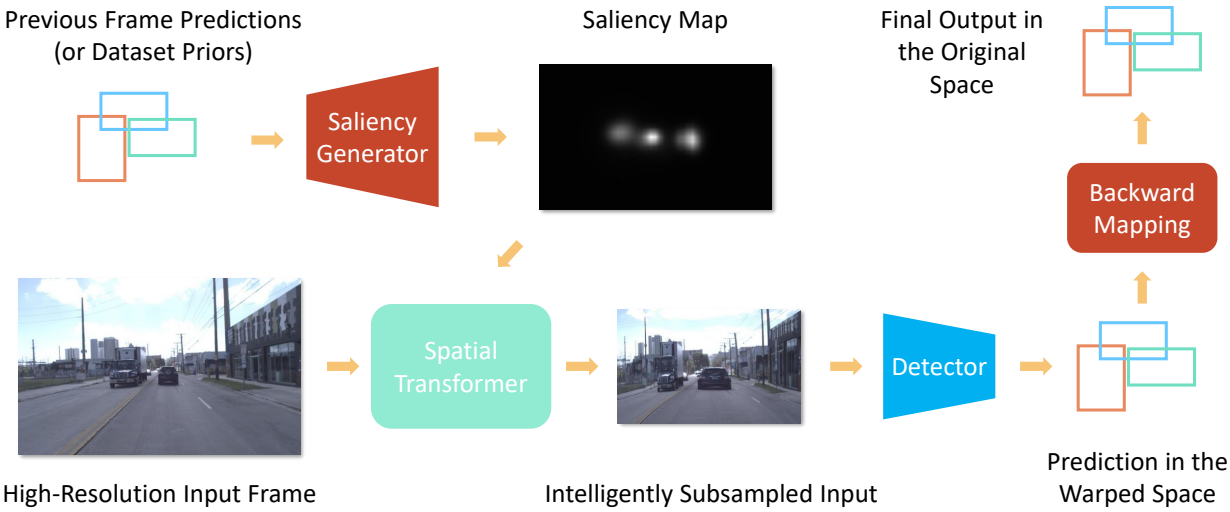


Figure 3.1: FOVEA for object detection. Given bounding box predictions from the previous frame (if the input are videos) or a collection of all the ground truth bounding boxes in the training set, the saliency generator creates a saliency map and that is fed into the spatial transformer (adapted from [22, 43]) to downsample the high-resolution input frame while magnifying salient regions. Then we feed the downsampled input into a regular object detector, and it produces bounding box output in the warped space, which is then converted back to the original image space as the final output.

coordinates (which may be non-integral), and bilinearly interpolate its color from neighboring input pixel grid points:

$$I'(x, y) = I(\mathcal{T}^{-1}(x, y)) \tag{3.3}$$

In other words, the implementation of $\mathcal{W}_\mathcal{T}$ only requires the knowledge of the inverse transformation $\mathcal{T}^{-1}$. The pixel iteration can be replaced with a batch operation by using a grid generator and apply the transformation $\mathcal{T}^{-1}$ over the entire grid.

STN uses a differentiable formulation of $\mathcal{T}_\theta^{-1}$ (parameterized by θ) and an ensuing bilinear grid sampler, which is differentiable and parameter-free. [43] proposes a special form of $\mathcal{T}^{-1}$ parameterized by a saliency map S: $\mathcal{T}_\theta^{-1} = \mathcal{T}_S^{-1}$. This transform has a convolution form and is therefore fast, using the intuition that each input pixel (x, y) attracts samples from the original image with a force $S(x, y)$, leading to more sampling at salient regions. *We point out that both [22] and [43] ignore the effect of warping on the output label space and skip the modeling of the forward transform $\mathcal{T}$, which (we will show) is required for unwarping certain label types.*

3.2 Image Warping for Object Detection

In this section, we first explain our high-level inference formulation, then our specific form of the warping, and in the end some adjustments for training the task network.

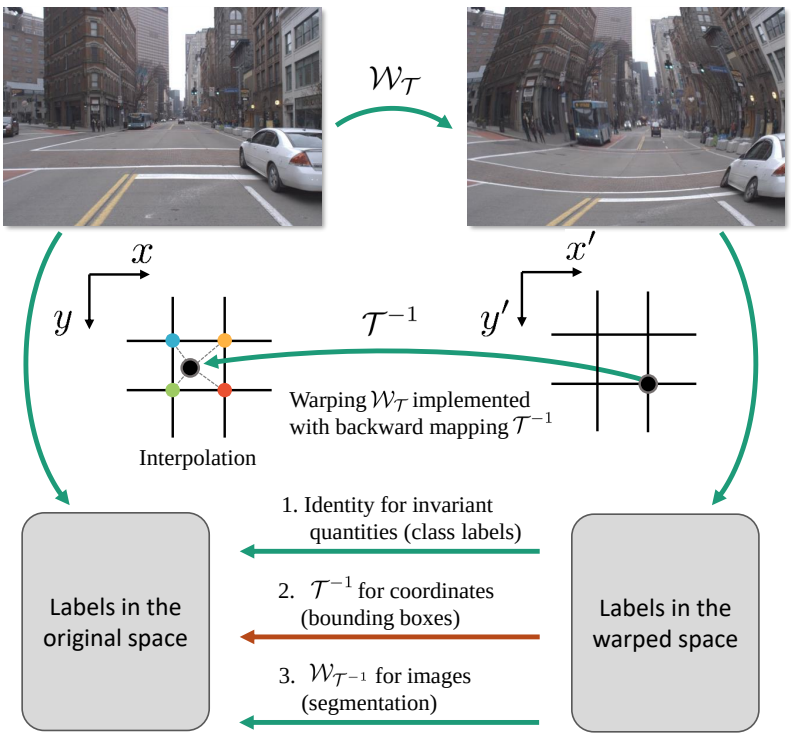


Figure 3.2: Image warps $\mathcal{W}_\mathcal{T}$ are commonly implemented via a backward map $\mathcal{T}^{-1}$ followed by (bilinear) interpolation of nearby source pixel grid values, since forward mapping $\mathcal{T}$ can result in target pixel positions that do not lie on the pixel grid (not shown). Though image warping is an extensively studied topic (notably by [22, 43] in the context of differentiable neural warps), its effect on labels is less explored because much prior art focuses on global labels invariant to warps (e.g. an image class label). We explore warping for spatial prediction tasks whose output must be transformed back into the original image space to generate consistent output. Interestingly, transforming pixel-level labels with warp $\mathcal{W}_{\mathcal{T}^{-1}}$ requires inverting $\mathcal{T}^{-1}$, which can be difficult depending on its parameterization [1]. For FOVEA, we focus on transforming pixel *coordinates* of bounding boxes, which requires only the already-computed backward map $\mathcal{T}^{-1}$ (the red arrow).

3.2.1 Inference formulation

We visually lay out the space of image and label warps in Fig 3.2. Recent methods for differentiable image warping assume labels are invariant under the warping (the first pathway in Fig 3.2). For object detection, however, image warping clearly warps bounding box outputs. To produce consistent outputs (e.g., for computing bounding box losses during learning), these warped outputs need to transformed back into the original space (the second pathway in Fig 3.2). Quite conveniently, because standard image warping is implemented via the backward map $\mathcal{T}^{-1}$, the backward map is already computed in-network and so can be directly applied to the pixel coordinates of the predicted bounding box. The complete procedure for our approach $\hat{f}$ can be written as $\hat{f}(I, \mathcal{T}) = \mathcal{T}^{-1}(f(\mathcal{W}_\mathcal{T}(I)))$. where $f(\cdot)$ is the nonlinear function that returns bounding box coordinates of predicted detections. Importantly, this convenience doesn't exist when warping pixel-level *values*; *e.g.*, when warping a segmentation mask back to the original image input space

(the third pathway in Fig 3.2). Here, one needs to invert $\mathcal{T}^{-1}$ to explicitly compute the forward warp $\mathcal{T}$.

3.2.2 Warping formulation

We adopt the saliency-guided warping formulation from [43]:

$$\mathcal{T}_x^{-1}(x,y) = \frac{\int_{x',y'} S(x',y')k((x,y),(x',y'))x'}{\int_{x',y'} S(x',y')k((x,y),(x',y'))}, \tag{3.4}$$

$$\mathcal{T}_y^{-1}(x,y) = \frac{\int_{x',y'} S(x',y')k((x,y),(x',y'))y'}{\int_{x',y'} S(x',y')k((x,y),(x',y'))}, \tag{3.5}$$

where k is a distance kernel (we use a Gaussian kernel in our experiments). However, in this general form, axis-aligned bounding boxes might have different connotations in the original and warped space. To ensure axis-alignment is preserved during the mapping, we restrict the warping to be separable along the two dimensions, *i.e.*, $\mathcal{T}^{-1}(x,y) = (\mathcal{T}_x^{-1}(x), \mathcal{T}_y^{-1}(y))$. For each dimension, we adapt the previous formulation to 1D:

$$\mathcal{T}_x^{-1}(x) = \frac{\int_{x'} S_x(x')k(x',x)x'}{\int_{x'} S_x(x')k(x,x')}, \tag{3.6}$$

$$\mathcal{T}_y^{-1}(y) = \frac{\int_{y'} S_y(y')k(y',y)y'}{\int_{y'} S_y(y')k(y,y')}. \tag{3.7}$$

We call this formulation *separable* and the general form *nonseparable*. Note that the nonseparable formulation has a 2D saliency map parameter, whereas the separable formulation has two 1D saliency maps, one for each axis. Fig 3.3 shows an example of each type of warp.

One nice property of $\mathcal{T}^{-1}$ is that it is differentiable and thus can be trained with backpropagation. One limitation though is that its inverse $\mathcal{T}$ doesn't have a closed-form solution, nor does its derivative. The absence of $\mathcal{T}$ is not ideal, and we propose some workaround as shown in the following subsection.

3.2.3 Anti-Cropping Constraint

We find the convolution form of saliency-guided spatial transform tends to crop the images, which might be acceptable for image classification where a large margin exists around the border. However, any cropping in object detection creates a chance to miss objects. We solve this by using reflect padding on the saliency map while applying the attraction kernel in Eq 3.6. This introduces symmetries about each of the edges of the saliency map, eliminating all horizontal offsets along vertical image edges and vice versa. Thus cropping is impossible under this formulation. A 1D illustration is shown in Fig 3.4 to explain the problem and the solution.

3.2.4 Training formulation

Once we have the inference formulation, training is also straightforward as we require the loss $\mathcal{L}$ to be computed in the original space: $\mathcal{L}(\mathcal{Q}(f(\mathcal{W}_\mathcal{T}(I)), L)$, where $\mathcal{Q}$ is the label-type-specific

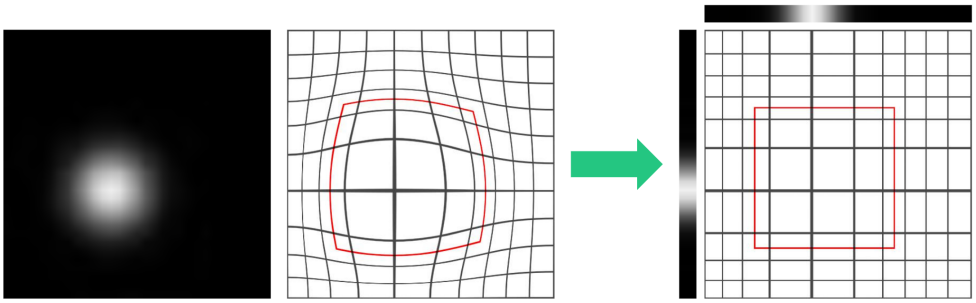


Figure 3.3: By restricting the general class of warps (**left**) to be separable (**right**), we ensure that bounding boxes in the warped image (examples outlined in red) remain axis-aligned. We demonstrate that such regularization (surprisingly) improves performance, even though doing so theoretically restricts the range of expressible warps (details in Sec 4.1.2).

backward mapping as shown in Fig 3.2, and in our case, $\mathcal{Q} = \mathcal{T}^{-1}$. Note that $\mathcal{W}_{\mathcal{T}}$, f and $\mathcal{T}^{-1}$ are all differentiable. While inference itself does not require the knowledge of $\mathcal{T}$, it is not the case for training detectors with region proposal networks (RPN) [45]. When training RPNs [45], the regression targets are the deltas between the anchors and the ground truth, and the deltas are later used in RoI Pooling/Align [18, 20]. The former should be computed in the original space (the ground truth is in the original space), while the latter is in the warped space (RoI Pooling/Align is over the warped image). This implies that the deltas need first to be learned in the original space, applied to the bounding box, and then mapped to the warped space using $\mathcal{T}$ for RoI Pooling/Align. But as discussed before, $\mathcal{T}$ cannot be easily computed. As a workaround, we omit the delta encoding and adopt Generalized IoU (GIoU) loss [46] to account for the lost stability. The main idea of GIoU is to better reflect the similarity of predicted and ground truth bounding boxes in cases of zero intersection; this has been shown to improve results.

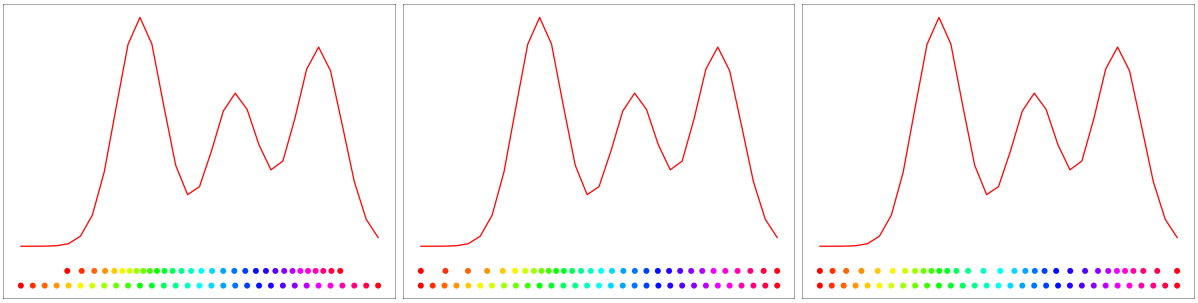
3.3 KDE Saliency Generator

Prior work [22, 43] trains a saliency network to generate saliency maps, which we explore as a baseline in Sec 4.1. Because saliency maps for object detection appear hard to learn, we explore cheap alternatives for saliency map construction: dataset-level priors over object locations or temporal priors extracted from previous frame's predictions. Both priors can be operationalized with an approach that converts bounding boxes to a saliency map.

Intuitively, we build a saliency map by "overlaying" boxes on top of one another via non-parametric kernel density estimation (KDE). More precisely, given a set of bounding boxes B with centers c_i, heights h_i and widths w_i, we model the saliency map S_B as a sum of normal distributions:

$$S_B^{a,b} = \frac{1}{K^2} + a \sum_{(c_i,w_i,h_i)\in B} \mathcal{N}\left(c_i, b \begin{bmatrix} w_i & 0 \\ 0 & h_i \end{bmatrix}\right) \tag{3.8}$$

where a and b are hyperparameters for amplitude and bandwidth, respectively, and K is the size of the attraction kernel k in Eq 3.6. Adding the small constant is done to prevent extreme warps. We



(a) Default, $\sigma \approx 5.5$ (b) Anti-crop, $\sigma \approx 5.5$ (c) Anti-crop, $\sigma \approx 1.7$

Figure 3.4: Saliency-guided transform illustrated in 1D. The red curve is a saliency map S. The bottom row of dots are the output points (at uniform intervals), and the top row of dots are the locations where we've sampled each output point from the original "image", as computed by applying $\mathcal{T}_S^{-1}$ to the output points. (a) The default transform can be understood as a weighted average over the output points and thus ignores points with near zero weights such as those at the boundaries. (b) Note the effects of introducing anti-crop reflect padding, and (c) how decreasing the std dev σ of the attraction kernel k results in more local warping around each peak (better for multimodal saliency distributions).

then normalize the 2D saliency map such that it sums to 1 and marginalize along the two axes if using the separable formulation[1]. As laid out in the previous section, this is then used to generate the image transformation $\mathcal{T}_S^{-1}$ according to Eq 3.6. Ensuring that each kernel is locally normalized produces our desired behavior; we'll have high saliency for pixels covered by objects, and even higher saliency for pixels covered by small objects (that have their Gaussian mass focused on a smaller object size).

We can apply S_B to the set of all bounding boxes in the training set to obtain a dataset-wide prior (denoted as S_D), or apply it to the previous frame's predictions to obtain a image-specific temporal prior (denoted as S_I). The former encodes dataset-level spatial priors such as small objects appearing near the horizon (Fig 4.2). The latter encodes a form of temporal contextual priming, allocating pixel samples to previously seen objects (with a default of uniform saliency for the first frame). We also experiment with a weighted combination of both: $S_C = \alpha \cdot S_I + (1-\alpha) \cdot S_D$. All of the above saliency generators are differentiable, so the final task loss can be used to learn hyperparameters a, b, α.

[1]When using the separable formulation, we could instead skip the intermediate 2D saliency map representation. However, we opt not to, because the intermediate 2D saliency map produces more interpretable visualizations, and the difference in runtime is negligible.

Chapter 4

FOVEA Experiments

We first compare FOVEA to naive downsampling on autonomous driving datasets such as Argoverse-HD. Next, we use streaming perception metrics to show that the accuracy gain is worth the additional cost in latency. Finally, we present results on BDD100K, showing the generalization of FOVEA. We include additional results, diagnostic experiments, and implementation details in the appendix.

4.1 Object Detection for Autonomous Navigation

Argoverse-HD [30] is an object detection dataset for autonomous vehicles. Noteably, it contains high framerate (30 FPS) data and annotations. As is standard practice, we adopt AP for evaluation. We also report *end-to-end* latency (including image preprocessing, network inference, and bounding box postprocessing) measured on a single GTX 1080 Ti GPU. The image resolution for this dataset is 1920×1200, much larger than COCO's, which is capped at 640. Since all models used are fully convolutional, we run them with different input scales, denoted by ratios to the native resolution, *e.g.*, 0.5x means an input resolution of 960×600.

4.1.1 Baseline and Setup

The baseline we compare to throughout our experiments is Faster RCNN [45] with a ResNet-50 backbone [19] plus FPN [35]. The default input scale for both the baseline and our method is 0.5x. For the baseline, however, we additionally train and test at 0.75x and 1x scales, to derive a sense of the latency-accuracy tradeoff using this model. Our contribution is orthogonal to the choice of the baseline detector and we obtain similar results with other detectors including RetinaNet [36] and YOLOF [9].

Notably, Argoverse-HD's training set only contains *pseudo ground truth* (at the time of experimentation) generated by running high-performing detector HTC [7] in the offline setting. For all experiments, unless otherwise stated, we train on the train split with pseudo ground truth annotations, and evaluate on the val split with real annotations. Additional measures are taken to prevent overfitting to biased annotations. We finetune COCO pretrained models on Argoverse-HD for only 3 epochs (*i.e.*, early stopping). We use momentum SGD with a batch size of 8, a learning

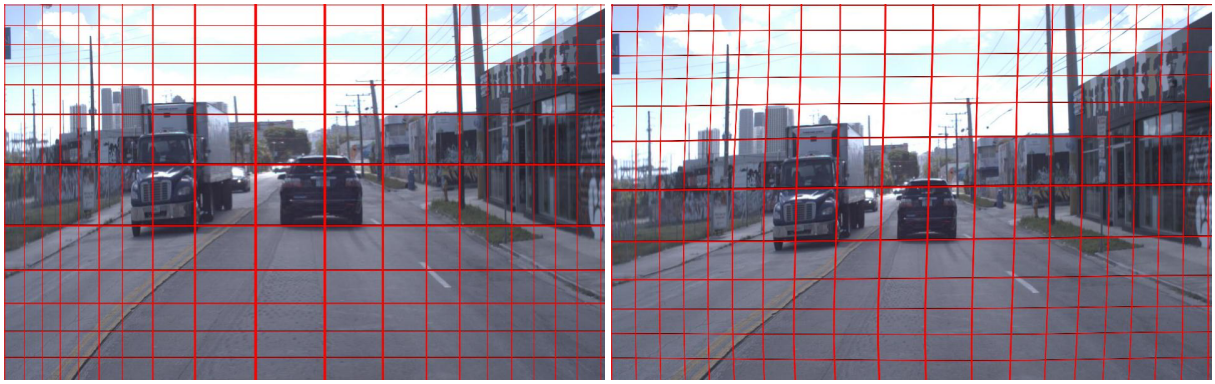


Figure 4.1: The learned direct separable (**left**) and nonseparable (**right**) dataset-wide warps. Despite the vastly greater flexibility of nonseparable warps, the learned warp is almost separable anyway.

rate of 0.02, 0.9 momentum, 10^{-4} weight decay, and a step-wise linear learning rate decay for this short schedule [31]. Also, when training detectors with warped input, we apply our modifications to RPN and the loss function as discussed in Sec 3.2.

4.1.2 Learned Saliency

Our first control experiment does not make use of bounding box KDE priors, but rather directly learns a *global, dataset-wide* saliency map $S(x, y)$ via backprop. We directly learn both separable and nonseparable saliency maps in Tab 4.1.

We find that both separable and nonseparable warps significantly improve overall AP over the baseline, owing to the boosted performance on small objects. However, there is also a small decrease in AP on large objects. Interestingly, even though nonseparable warps are more flexible, the learned solutions look nearly separable (Fig 4.1) but perform worse, indicating overfitting. Therefore, going forward, we focus on separable warps in our experiments.

Following [43], we also learn a "saliency network" that maps each input image to its saliency map via a ResNet-18 backbone [19]. In this sense, the learned saliency map would adapt to each image. However, we find that this approach very unstable for object detection. From our experiments, even with a small learning rate of 10^{-5} on the saliency network, the model learns a degeneracy in which an extreme warp leads to no proposals being matched with ground truth bounding boxes in the RoI bounding box head, leading to a regression loss of 0.

4.1.3 KDE Saliency Generator

This section makes use of the KDE construction in Sec 3.3 to generate saliency maps. We first manually tune the amplitude a and bandwidth b to obtain desired magnifications. We find that an amplitude $a = 1$ and a bandwidth $b = 64$ works the best, paired with an attraction kernel of std. dev. of about 17.8% the image height, which allows for more local warps as illustrated in Fig 3.4. We finetune our models using the same configuration as the baseline, the only difference being the added bounding box and saliency-guided spatial transformation layer. We learn S_D using all

bounding boxes from the training set and for simplicity, learn S_I with jittered ground-truth boxes from the current frame (though at test-time it always uses predictions from the previous frame). We set $\alpha = 0.5$ for S_C.

We then learn hyperparameters a and b through backpropagation, since our KDE formulation is differentiable. We initialize parameters a' and b' to 0, under the construction that $a = |1 + a'| + 0.1, b = 64 \cdot |1 + b'| + 0.1$. The learning rate of a' and b' is set to 10^{-4} with zero weight decay. Other than this, we train the learned KDE (LKDE) model with the same configuration as the baseline. We implement the S_I formulation.

All results are shown in Table 4.1. Even without finetuning our detector, using a simple fixed dataset-wide warp S_D, we find significant improvements in AP. As we migrate to temporal priors with finetuning, we see even more improvement. As in the learned saliency case, these improvements in overall AP are due to large boosts in AP_S, outweighing the small decreases in AP_L. Combining our saliency signals (S_C) doesn't help, because in our case, it seems that the temporal signal is strictly stronger than the dataset-wide signal. Perhaps if we had an alternate source of saliency like a map overlay, combining saliencies could help. Our best method overall is LKDE, which learned optimal values $a = 1.07, b = 71.6$. Learning a nonseparable saliency performs better than our hand-constructed dataset-wide warp S_D; however, they're both outperformed by S_I. Importantly, our LKDE not only significantly improves AP_S, but also improves *all* other accuracy measures, suggesting that our method does not need to tradeoff accuracy of large objects for that of small objects. Finally, we note that our increased performance comes at the cost of only about 2 ms in latency.

Argoverse-HD before finetuning

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	person	mbike	tffclight	bike	bus	stop	car	truck	Latency (ms)
Baseline	21.5	35.8	22.3	2.8	22.4	**50.6**	20.8	9.1	13.9	7.1	48.0	16.1	37.2	20.2	49.4 ± 1.0
KDE (S_D)	23.3	40.0	22.9	5.4	25.5	48.9	20.9	13.7	12.2	9.3	**50.6**	20.1	40.0	19.5	52.0 ± 1.0
KDE (S_I)	**24.1**	**40.7**	**24.3**	**8.5**	24.5	48.3	**23.0**	**17.7**	**15.1**	**10.0**	49.5	17.5	**41.0**	19.4	51.2 ± 0.7
KDE (S_C)	24.0	40.5	**24.3**	7.4	**26.0**	48.2	22.5	14.9	14.0	9.5	49.7	**20.6**	**41.0**	**19.9**	52.0 ± 1.2
Upp. Bound (0.75x)	27.6	45.1	28.2	7.9	30.8	51.9	29.7	14.3	21.5	6.6	54.4	25.6	44.7	23.7	86.9 ± 1.6
Upp. Bound (1.0x)	32.7	51.9	34.3	14.4	35.6	51.8	33.7	21.1	33.1	5.7	57.2	36.7	49.5	24.6	133.9 ± 2.2

Argoverse-HD after finetuning

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	person	mbike	tffclight	bike	bus	stop	car	truck	Latency (ms)
Baseline	24.2	38.9	26.1	4.9	29.0	50.9	22.8	7.5	23.3	5.9	44.6	19.3	43.7	26.6	50.9 ± 0.9
Learned Sep.	27.2	44.8	28.3	**12.2**	29.1	46.6	24.2	14.0	22.6	7.7	39.5	**31.8**	50.0	27.8	51.5 ± 1.0
Learned Nonsep.	25.9	42.9	26.5	10.0	28.4	48.5	25.2	11.9	20.9	7.1	39.5	25.1	49.4	28.1	50.0 ± 0.8
KDE (S_D)	26.7	43.3	27.8	8.2	29.7	54.1	25.4	13.5	22.0	8.0	**45.9**	21.3	48.1	29.3	50.8 ± 1.2
KDE (S_I)	28.0	45.5	**29.2**	10.4	**31.0**	**54.5**	27.3	16.9	**24.3**	**9.0**	44.5	23.2	**50.5**	28.4	52.2 ± 0.9
KDE (S_C)	27.2	44.7	28.4	9.1	30.9	53.6	27.4	14.5	23.0	7.0	44.8	21.9	49.9	**29.5**	52.1 ± 0.9
LKDE (S_I)	**28.1**	**45.9**	28.9	10.3	30.9	54.1	**27.5**	**17.9**	23.6	8.1	45.4	23.1	50.2	28.7	50.5 ± 0.8
Upp. Bound (0.75x)	29.2	47.6	31.1	11.6	32.1	53.3	29.6	12.7	30.8	7.9	44.1	29.8	48.8	30.1	87.0 ± 1.4
Upper Bound (1.0x)	33.3	53.9	35.0	16.8	34.8	53.6	33.1	20.9	38.7	6.7	44.7	36.7	52.7	32.7	135.0 ± 1.6

Table 4.1: Results before and after finetuning on Argoverse-HD. Without retraining, processing warped images (KDE S_I, top table) improves overall AP by 2.6 points and triples AP_S. Even larger gains can be observed after finetuning, making our final solution (LKDE S_I) performing close to the 0.75x upper bound. Please refer to the text for a more detailed discussion.

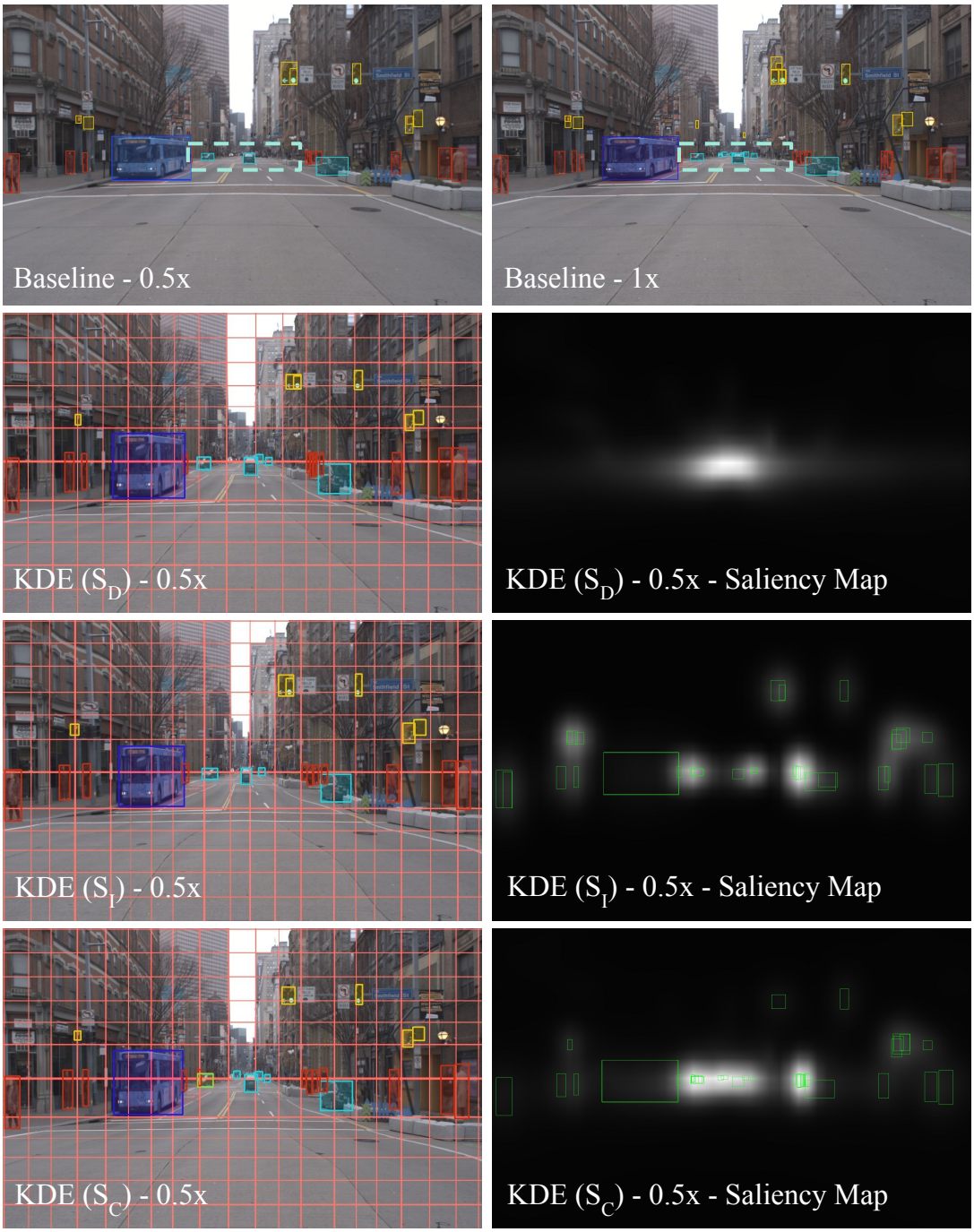


Figure 4.2: Qualitative results for FOVEA after finetuning on Argoverse-HD. The cars in the distance (in the dotted boxes), undetected at 0.5x scale, are detected at 1x scale, and partially detected by our methods. Different rows show the variations within FOVEA based on the source of attention.

ID	Method	AP	AP_S	AP_M	AP_L
1	Prior art [30]	17.8	3.2	16.3	33.3
2	+ Better implementation	19.3	4.1	18.3	34.9
3	+ Train with pseudo GT	21.2	3.7	**23.9**	43.8
4	2 + Ours (S_I)	19.3	5.2	18.5	39.0
5	3 + Ours (S_I)	**23.0**	**7.0**	23.7	**44.9**

Table 4.2: Streaming evaluation in the full-stack (with forecasting) setting on Argoverse-HD. We show that FOVEA significantly improves previous state-of-the-art by 5.2, in which 1.5 is from better implementation, 1.9 is from making use of pseudo ground truth and 1.8 is from our proposed KDE warping.

4.2 Streaming Accuracy for Cost-Performance Evaluation

Streaming accuracy is a metric that coherently integrates latency into standard accuracy evaluation and therefore is able to *quantitatively measure the accuracy-latency tradeoff* for embodied perception [30]. Such a setup is achieved by having the benchmark stream the data to the algorithm in real-time and query for the state of the world at all time instants. One of their key observations is that by the algorithm finishes processing, the world has around changed and therefore proper temporal scheduling and forecasting methods should be used to compensate for this latency. Here we adopt their evaluation protocol for our cost-performance analysis. In our case of streaming object detection, the streaming accuracy refers to *streaming AP*. We use the same GPU (GTX 1080 Ti) and their public available codebase for a fair comparison with their proposed solution. Their proposed solution includes a scale-tuned detector (Faster R-CNN), dynamic scheduler (shrinking-tail) and Kalman Filter forecastor. Our experiments focus on improving the detector and we keep the scheduler and forecastor fixed.

Tab 4.2 presents our evaluation under the full-stack setting. We see that FOVEA greatly improves the previous state-of-the-art. The improvement first comes from a faster and slightly more accurate implementation of the baseline. Note that under streaming perception, a faster algorithm while maintaining the same offline accuracy translates to an algorithm with higher streaming accuracy. The second improvement is due to training on pseudo ground truth (discussed in Sec 4.1.1). Importantly, our KDE image warping further boosts the streaming accuracy significantly *on top of* these improvements. Overall, these results suggest that *image warping is a cost-efficient way to improve accuracy*.

4.3 Cross-Dataset Generalization

Our experiments so far are all conducted on the Argoverse-HD dataset. In this section, we cross-validate FOVEA on another autonomous driving dataset BDD100K [60]. Note that BDD100K and Argoverse-HD are collected in different cities. For simplicity, we only test out *off-the-shelf*

ID	Method	AP	AP_S	AP_M	AP_L
1	Baseline (0.5x)	15.1	1.0	10.6	**39.0**
2	Ours S_D (0.5x)	13.7	1.3	10.0	34.7
3	Ours S_I (0.5x)	**16.4**	**2.1**	**12.8**	38.6
4	Baseline (0.75x)	19.7	3.0	16.1	**44.2**
5	Ours S_D (0.75x)	18.2	3.4	15.4	40.0
6	Ours S_I (0.75x)	**20.1**	**5.2**	**17.0**	42.5
7	Upper bound (1.0x)	22.6	5.7	20.1	45.7

Table 4.3: Cross-dataset generalization to BDD100K [60]. Rows 2 & 5 are saliency computed on the Argoverse-HD training set, as expected, they fail to generalize to a novel dataset. Despite operating at a larger temporal stride (5 FPS vs 30 FPS), our proposed image-adaptive KDE warping generalizes to a novel dataset (row 3 & 6). Note that here the image native resolution is smaller at 1280×720.

generalization without any finetuning. We experiment on the validation split of the MOT2020 subset, which contains 200 videos with 2D bounding boxes annotated at 5 FPS (40K frames in total). Also, we only evaluate on common classes between BDD100K and Argoverse-HD: person, bicycle, car, motorcycle, bus, and truck. The results are summarized in Tab 4.3, which demonstrate the generalization capability of FOVEA.

Chapter 5

LZU [52]

5.1 Background for LZU

Before diving into our second approach LZU, since LZU is a generalization of previous works [23, 43, 51], we include this section as a condensed explanation of prerequisite formulations critical to understanding LZU.

5.1.1 Image Resampling

Suppose we want to resample an input image $\mathbf{I}(\mathbf{x})$ to produce an output image $\mathbf{I}'(\mathbf{x})$, both indexed by spatial coordinates $\mathbf{x} \in [0,1]^2$. Resampling is typically implemented via the *inverse* map $\mathcal{T} : [0,1]^2 \to [0,1]^2$ [1]. For each output coordinate, the inverse map computes the source location from which to "steal" the pixel value, i.e. $\mathbf{I}'(\mathbf{x}) = \mathbf{I}(\mathcal{T}(\mathbf{x}))$. In practice, we are given a discretized input image $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$ and are interested in computing a discretized output $\mathbf{I}' \in \mathbb{R}^{H' \times W' \times C}$. Formally, we compute $\mathbf{I}'(\mathbf{x})$ at grid points $\mathbf{x} \in \mathrm{Grid}(H', W')$, where $\mathrm{Grid}(H, W) := \mathrm{Grid}(H) \times \mathrm{Grid}(W)$ and $\mathrm{Grid}(D) := \{\frac{d-1}{D-1} : d \in [D]\}$. However, $\mathcal{T}(\mathbf{x})$ may return non-integer pixel locations at which the exact value of $\mathbf{I}$ is unknown. In such cases, we use bilinear interpolation to compute $\mathbf{I}(\mathcal{T}(\mathbf{x}))$. As proven in [22], such image resampling is differentiable with respect to $\mathcal{T}$ and $\mathbf{I}$.

5.1.2 Saliency-Guided Downsampling

When using nonuniform downsampling for information retention, it is useful to parameterize $\mathcal{T}$ with a saliency map $\mathbf{S}(\mathbf{x})$ representing the desired sample rate at each spatial location $\mathbf{x} \in [0,1]^2$ [43]. Recasens et al. [43] go on to approximate this behavior by having each sample coordinate $\mathcal{T}(\mathbf{x})$ be "attracted" to nearby areas $\mathbf{x}'$ with high saliency $\mathbf{S}(\mathbf{x}')$ downweighted according to a distance kernel $k(\mathbf{x}, \mathbf{x}')$, as illustrated in Figure 5.1. Concretely,

$$\mathcal{T}_{\mathrm{LZ}}(\mathbf{x}) = \left(\frac{\int_{\mathbf{x}'} \mathbf{S}(\mathbf{x}') k(\mathbf{x}, \mathbf{x}') \mathbf{x}'_x \, d\mathbf{x}'}{\int_{\mathbf{x}'} \mathbf{S}(\mathbf{x}') k(\mathbf{x}, \mathbf{x}') \, d\mathbf{x}'}, \frac{\int_{\mathbf{x}'} \mathbf{S}(\mathbf{x}') k(\mathbf{x}, \mathbf{x}') \mathbf{x}'_y \, d\mathbf{x}'}{\int_{\mathbf{x}'} \mathbf{S}(\mathbf{x}') k(\mathbf{x}, \mathbf{x}') \, d\mathbf{x}'} \right). \tag{5.1}$$

[51] proposes *anti-cropping* and *separable* variants of this downsampler. The anti-cropping variant $\mathcal{T}_{\mathrm{LZ,ac}}$ uses reflect padding on $\mathbf{S}(\mathbf{x}')$ when computing $\mathcal{T}_{\mathrm{LZ}}(\mathbf{x})$. This induces symmetries

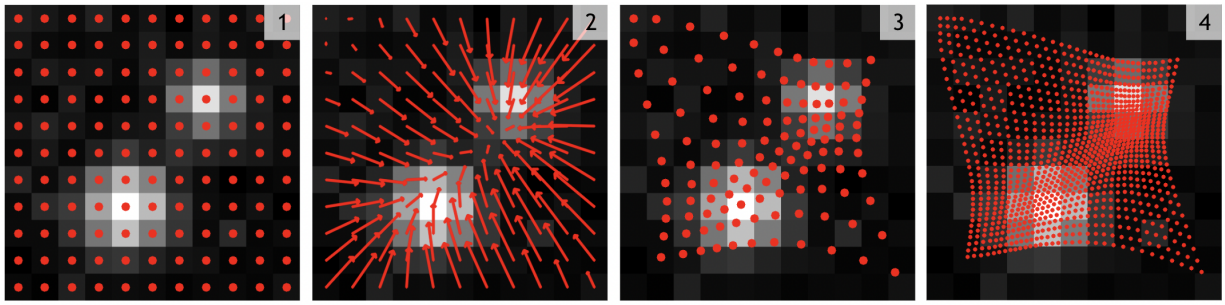


Figure 5.1: Illustration of $\mathcal{T}_{\mathrm{LZ}}$ [43]. Suppose we have a saliency map $\mathbf{S} \in \mathbb{R}^{h \times w}$ (visualized in the background) and want a warped image of size $H' \times W'$. (1) We start with a uniform grid of sample locations $\mathrm{Grid}(h, w)$. (2) Grid points are "attracted" to nearby areas with high saliency. (3) Applying this "force" yields $\mathcal{T}_{\mathrm{LZ}}[\mathrm{Grid}(h, w)]$. (4) Finally, we upsample bilinearly to get $\widetilde{\mathcal{T}}_{\mathrm{LZ}}[\mathrm{Grid}(H', W')]$.

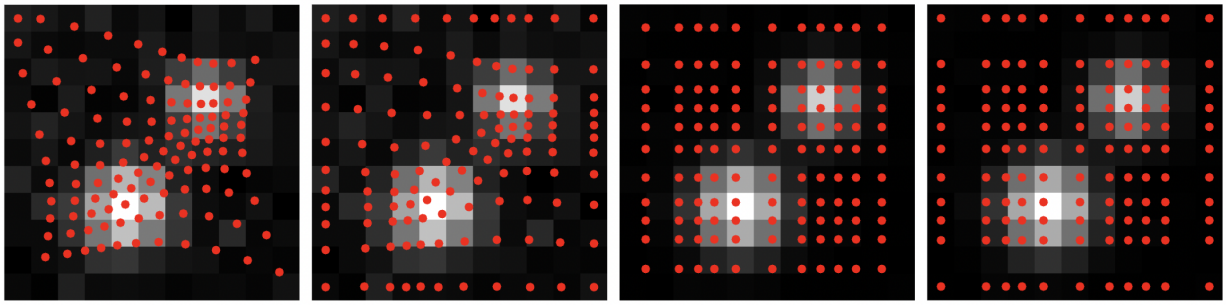


Figure 5.2: Examples of the anti-cropping (ac) and separable (sep) variants of $\mathcal{T}_{\mathrm{LZ}}$ from [51].

about each edge, leading to zero displacement in the sample coordinates along each edge and thus preventing cropping. The separable variant marginalizes the saliency map $\mathbf{S}(\mathbf{x})$ into two 1D saliency maps $\mathbf{S}_x(x)$ and $\mathbf{S}_y(y)$, and replaces the kernel $k(\mathbf{x}, \mathbf{x}')$ with a two 1D kernels k_x and k_y (although generally $k_x = k_y$). The resulting inverse mapping is given by $\mathcal{T}_{\mathrm{LZ,sep}}(\mathbf{x}) = (\mathcal{T}_{\mathrm{LZ,sep,x}}(\mathbf{x}_x), \mathcal{T}_{\mathrm{LZ,sep,y}}(\mathbf{x}_y))$ for

$$\mathcal{T}_{\mathrm{LZ,sep,x}}(x) = \frac{\int_{x'} \mathbf{S}_x(x') k_x(x, x') x' \, dx'}{\int_{x'} \mathbf{S}_x(x') k_x(x, x') \, dx'} \quad \text{and} \quad \mathcal{T}_{\mathrm{LZ,sep,y}}(y) = \frac{\int_{y'} \mathbf{S}_y(y') k_y(y, y') y' \, dy'}{\int_{y'} \mathbf{S}_y(y') k_y(y, y') \, dy'}. \tag{5.2}$$

This preserves axis-alignment of rectangles, which is crucial to object detection where object locations are specified via corners. We refer to the above method and all variants as *LZ downsamplers*, after the pioneering work "Learning to Zoom" [43]. Examples of each variant are shown in Figure 5.2.

5.2 LZU

We begin by discussing our general technique for warp inversion. Then, we discuss the LZU framework and how we apply warp inversion to efficiently "unzoom".

5.2.1 Generalized Warp Inversion

Suppose we have a inverse mapping $\mathcal{T} : [0,1]^2 \to [0,1]^2$. Our primary technical innovation is an approximation of the forward mapping $\mathcal{T}^{-1} : [0,1]^2 \to [0,1]^2$, even in cases where $\mathcal{T}$ has no closed-form inverse. We do this by approximating $\mathcal{T}$ as a piecewise tiling of simpler invertible transforms.

In practice, given a discretized input image $\mathbf{I} \in \mathbb{R}^{H \times W \times d}$, to produce a warped image $\mathbf{I}' \in \mathbb{R}^{H' \times W' \times d}$, we evaluate $\mathcal{T}[\mathrm{Grid}(H', W')]$ to determine for each grid point where to sample the input image. Because $\mathcal{T}$ is difficult to invert, let us approximate it with another map $\widetilde{\mathcal{T}}$ that is designed for easy inversion. Given a grid rectangle $R(i,j) = [\frac{i-1}{H'-1}, \frac{i}{H'-1}] \times [\frac{j-1}{W'-1}, \frac{j}{W'-1}]$ for $i \in [H']$ and $j \in [W']$, we define $\widetilde{\mathcal{T}}$ within $R(i,j)$ by bilinearly interpolating the values at the corners, making $\widetilde{\mathcal{T}}$ a continuous piecewise bilinear map. More precisely, letting $\mathrm{BilinearTransformation}_{ij}$ be the ij-th bilinear map, the overall approximation is given by

$$\widetilde{\mathcal{T}}(\mathbf{x}) = \mathrm{BilinearTransformation}_{ij}(\mathbf{x}) \qquad \text{for } \mathbf{x} \in R(i,j). \tag{5.3}$$

Furthermore, so long as $\widetilde{\mathcal{T}}$ is injective (if each of the bilinear maps is nondegenerate/invertible and no two bilinear maps overlap), we are guaranteed a well-defined left inverse $\widetilde{\mathcal{T}}^{-1} : [0,1]^2 \to [0,1]^2$ given by

$$\widetilde{\mathcal{T}}^{-1}(\mathbf{x}) = \begin{cases} \mathrm{BilinearTransformation}_{ij}^{-1}(\mathbf{x}) & \text{if } \mathbf{x} \in \widetilde{\mathcal{T}}[R(i,j)] \\ 0 & \text{else} \end{cases}. \tag{5.4}$$

Eq 5.4 is **efficient** to compute, since determining if $\mathbf{x} \in \widetilde{\mathcal{T}}[R(i,j)]$ simply involves checking if $\mathbf{x}$ is in a quadrilateral and computing the inverse of a bilinear map amounts to solving a quadratic equation [58]. This efficiency is crucial to maintaining favorable accuracy-latency tradeoffs. $\widetilde{\mathcal{T}}^{-1}$ is guaranteed to be **differentiable** with respect to $\mathcal{T}$, since for each $\mathbf{x} \in \widetilde{\mathcal{T}}[R(i,j)]$, the inverse bilinear map $\mathrm{BilinearTransformation}_{ij}^{-1}$ can be written as a quadratic function of the four corners of tile ij (see supplement for exact expression). This allows gradients to flow back into $\mathcal{T}$, letting us learn the parameters of the warp.

In the case of LZ warps, $\mathcal{T}_{\mathrm{LZ}}$ has no closed form inverse to the best of our knowledge. Because $\mathcal{T}_{\mathrm{LZ}}[\mathrm{Grid}(H', W')]$ has no foldovers [43], $\widetilde{\mathcal{T}}_{\mathrm{LZ}}$ must be injective, implying its inverse $\widetilde{\mathcal{T}}_{\mathrm{LZ}}^{-1}$ is well-defined.

5.2.2 Learning to Zoom and Unzoom

In the Learning to Zoom and Unzoom (LZU) framework, we use existing LZ downsamplers (see Section 5.1.2) to "zoom" in on the input image, compute spatial features, and then use our warp inversion formulation to "unzoom" and revert any deformations in the feature map, as shown in Figure 1.2. This framework is applicable to all tasks with 2D spatial input and all models with some intermediate 2D spatial representation.

To maintain favorable accuracy-latency tradeoffs, we make several optimizations to our forward and inverse warps. As with previous works [23, 43, 51], instead of computing $\mathcal{T}_{\mathrm{LZ}}[\mathrm{Grid}(H', W')]$, we compute $\mathcal{T}_{\mathrm{LZ}}[\mathrm{Grid}(h, w)]$ for smaller $h \ll H'$ and $w \ll W'$ and bilinearly upsample this to get $\widetilde{\mathcal{T}}_{\mathrm{LZ}}[\mathrm{Grid}(H', W')]$. This also reduces the complexity of computing the inverse, by reducing the number of cases in our piecewise bilinear map from $H' \cdot W'$ to $h \cdot w$.

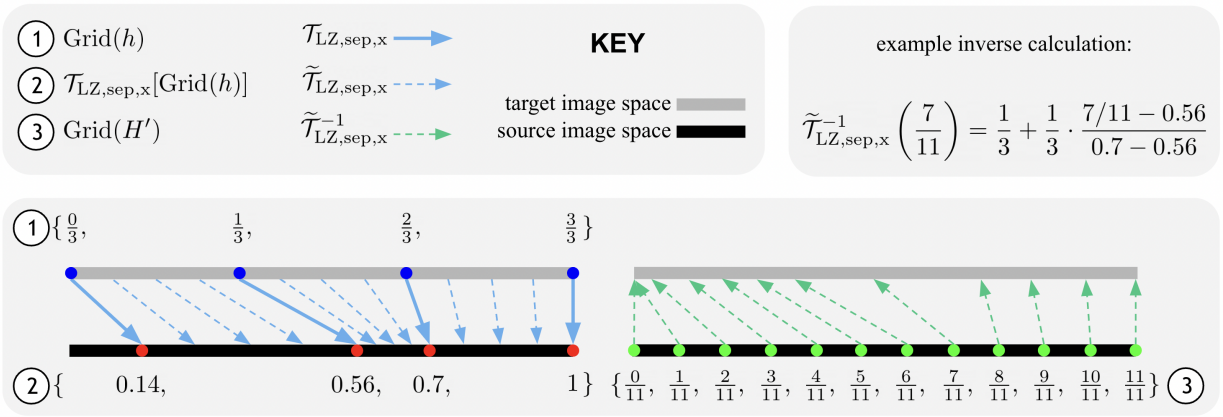


Figure 5.3: Inverting each axis of a separable warp. LZU first evaluates the forward warp $\mathcal{T}_{\text{LZ,sep,x}}$ (solid blue arrows) at a uniform grid of target locations (blue points). The resulting source locations are shown as red points. LZU then approximates the warp in between these samples via a *linear* transform; this piecewise linear map is $\widetilde{\mathcal{T}}_{\text{LZ,sep,x}}$ (dotted blue arrows). To evaluate the inverse $\widetilde{\mathcal{T}}^{-1}_{\text{LZ,sep,x}}$ (dotted green arrows), we must determine for each green point which red points it falls between and invert the corresponding linear transform. An example is shown in the top-right.

We explore efficient implementations of both separable and nonseparable warp inversion, but we find experimentally that nonseparable warps perform no better than separable warps for a strictly higher latency cost, so we use separable warps for our experiments. Details for efficiently inverting nonseparable warps are given in the supplementary. For separable warps $\mathcal{T}_{\text{LZ,sep}}$, we invert each axis separately and take the Cartesian Product:

$$\widetilde{\mathcal{T}}^{-1}_{\text{LZ,sep}}[\text{Grid}(H', W')] = \widetilde{\mathcal{T}}^{-1}_{\text{LZ,sep,x}}[\text{Grid}(H')] \times \widetilde{\mathcal{T}}^{-1}_{\text{LZ,sep,y}}[\text{Grid}(W')]. \tag{5.5}$$

This further reduces our problem from inverting a piecewise bilinear map with $h \cdot w$ pieces to inverting two piecewise *linear* maps with h and w pieces each. Figure 5.3 visualizes how to invert each axis.

When unwarping after feature pyramid networks [35], we may have to evaluate the inverse $\widetilde{\mathcal{T}}^{-1}_{\text{LZ}}$ at multiple resolutions $\text{Grid}(H', W')$, $\text{Grid}(H'/2, W'/2)$, etc. In practice, we evaluate $\widetilde{\mathcal{T}}^{-1}_{\text{LZ}}[\text{Grid}(H', W')]$ and then approximate the inverse at lower resolutions via bilinear downsampling.

Finally, as introduced in [51], we can also use a fixed warp to exploit dataset-wide spatial priors, such as how objects are concentrated around the horizon in many autonomous driving datasets. This allows us to cache forward and inverse warps, greatly reducing additional latency.

Chapter 6

LZU Experiments

First, we compare LZU to naive uniform downsampling and previous works on the tasks of 2D object detection and semantic segmentation. We include ablations to evaluate the effectiveness of training techniques and measure the importance of information retention. Then, we evaluate LZU on RGB-based 3D object detection, a task which no previous works have applied "zooming" to. A summary of our results is given in Table 1.1. We perform all timing experiments with a batch size of 1 on a single RTX 2080 Ti GPU.

6.1 2D Object Detection

In all detection experiments, we use RetinaNet [36] with a ResNet-50 backbone [19] and FPN [35] for the base model and the standard AP metric for evaluation. We split into sections by dataset.

6.1.1 Argoverse-HD

First, we evaluate LZU on Argoverse-HD [29], an object detection dataset for autonomous driving with high resolution 1920×1200 videos. We run all experiments at 0.5x scale. For our baseline, we train and test vanilla RetinaNet with uniform downsampling. We also compare to FOVEA [51], a previous work that applies LZ warping to detection by unwarping bounding boxes. Finally, we train another uniform downsampling model at 0.75x scale, to get a sense of the usual accuracy-latency tradeoff when varying the input resolution. We use the same hyperparameters and setup as in [51]. We report baseline latency as the time taken for image preprocessing and inference.

Our LZU models "unzoom" the feature map at each level after the FPN [35]. We adopt the low-cost saliency generators introduced in [51] — a "fixed" saliency map exploiting dataset-wide spatial priors, and an "adaptive" saliency map exploiting temporal priors by zooming in on detections from the previous frame. When training the adaptive version, we simulate previous detections by jittering the ground truth for the first two epochs. For the last epoch, we jitter *detections* on the current frame to better simulate previous detections; we call this "cascaded" saliency. We use a learning rate of 0.01 and keep all other training settings identical to the baseline. Latencies are reported relative to the baseline by timing only the additional operations (the "zoom" and "unzoom").

Table 6.1: 2D object detection results of RetinaNet [36] on Argoverse-HD [29]. LZU (3, 4) consistently outperforms the uniform downsampling baseline (1) and prior work (2), with minimal additional latency. Latencies for (2-6) are reported as a delta with respect to the baseline. Note that the accuracy-latency tradeoff achieved by LZU is far better than the usual tradeoff associated with varying the input resolution (1, 7). We also run several ablations to analyze why LZU is effective. We find that using cascaded saliency (see Section 6.1) is crucial when training adaptive LZU (5). We also find that LZU helps *even in the absence of high-resolution inputs* (6). This suggests that simply allocating more pixels to small objects (without retaining extra information) can help performance.

ID	Scale	Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	Latency (ms)
1	0.5x	Uniform	22.6	38.7	21.7	3.7	22.1	53.1	45.8
2	0.5x	FOVEA [51]	24.9	40.3	**25.3**	**7.1**	**27.7**	50.6	+1.9
3	0.5x	LZU, fixed	25.2	42.1	24.8	5.5	26.7	51.8	**+0.7**
4	0.5x	LZU, adaptive	**25.3**	**43.0**	24.6	6.1	25.9	**52.6**	+3.9
5	0.5x	4 w/o cascaded saliency	22.8	39.3	22.3	5.1	22.7	48.9	+3.9
6	0.5x	4 w/o high-res inputs	24.0	41.7	23.0	5.4	24.5	51.2	+3.9
7	0.75x	Uniform	29.5	48.4	29.6	9.1	32.4	55.1	68.5

Results are given in Table 6.1. We outperform both the uniform sampling baseline and FOVEA while incurring an additional latency of < 4ms. Our boosts are due to better performance on small and medium objects, with a small drop in performance on large objects.

We also perform several ablations on the adaptive LZU model. We confirm that using cascaded saliency during the last epoch is crucial. We also test a version of LZU with no access to high resolution inputs. This is implemented by first uniformly downsampling input images to 0.5x scale and then "zooming" in on the downsampled image. Even in this case, there is a 1.4 AP improvement over the baseline, showing that LZU can used even in the absence of high-resolution sensor data.

6.1.2 Synthetic Video COCO

Then, we evaluate on COCO [34], a standard object detection dataset with 80 categories and smaller images with side length at most 640 pixels and variable aspect ratios. We run experiments at $\geq$0.5x scale, by scaling the longer side to 320 pixels, and at $\geq$1x scale, by scaling the longer side to 640 pixels. For our baseline, again we train and test vanilla RetinaNet with uniform downsampling. No prior works have applied "zooming" to COCO, so we only compare to this naive baseline. we train from an ImageNet-pretrained backbone for 12 epochs using a batch size of 16, a step learning schedule of 0.1 at epochs 8 and 11, a learning rate of 0.01. All other settings are the same as with Argoverse-HD.

Again, our LZU model "unzooms" feature maps at each level after the FPN [35]. Unlike Argoverse-HD [29], which has a strong dataset-wide prior of small objects near the center, the

Table 6.2: 2D object detection results of RetinaNet on Synthetic Video COCO. Since COCO [34] is not a video dataset, we simulate detections on the previous frame by jittering ground truths from the current frame. This assumes almost perfect knowledge on where to zoom. LZU outperforms uniform downsampling at both resolutions. Importantly, at the 640×640 resolution, LZU is "learning" to adaptively upsample (by up to 3x on smaller images)!

Resolution	Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
320×320 ($\geq$0.5x)	Uniform	24.6	39.5	**25.7**	**3.5**	26.6	43.9
	LZU, adaptive	**24.9**	**41.1**	25.6	3.4	**27.7**	**44.5**
640×640 ($\geq$1x)	Uniform	33.0	51.1	34.9	14.0	37.5	47.8
	LZU, adaptive	**33.4**	**53.9**	34.9	**15.3**	**38.6**	**48.0**

distribution of small objects in COCO is more arbitrary. So we choose to use an adaptive saliency map based on previous detections, as in [51]. However, since COCO is not a video dataset, we simulate previous detections by jittering ground truths from the current frame. Essentially, this assumes we have almost perfect knowledge on where to zoom. We train and test with a jitter of $\mathcal{N}(0, 10)$ pixels in the x and y directions. All other training settings are identical to the baseline.

Results are given in Table 6.2. LZU improves on the uniform downsampling baseline at both resolutions. Performance increases come mostly at lower bounding box overlap thresholds (AP_{50} as opposed to AP_{75}), suggesting that LZU is good at detecting the presence of objects but struggles more at precisely localizing them. Interestingly, we see performance improvements even in the *upsampling* regime of 640×640 ($\geq$ 1x) scale. In this regime, the "zoom" simply allocates more pixels and computation to salient regions (without extra information retention), and it seems even this can prompt performance gains.

6.2 Semantic Segmentation

For our semantic segmentation experiments, we compare to previous works [40] and [23], so we adopt their setup. We test the PSPNet [61] model with a ResNet-50 backbone [19] and FPN [35] on Cityscapes [13]. Cityscapes is an urban scene dataset with high resolution 1024×2048 images and 19 classes. We perform our experiments at several image scales (64×64, 128×128, and 256×256), taken by resizing a centered square crop of the input image. Our simple baseline trains and tests PSPNet with uniform downsampling. We train for 80K iterations with the photometric distortion augmentation (random adjustments in brightness, contrast, saturation, and hue), a batch size of 16, momentum SGD with a learning rate of 0.01, momentum of 0.9, weight decay of 5e-4, and a polynomial learning rate schedule with power 0.9. We evaluate every 10K iterations, and report the best performance, to account for overfitting.

For our LZU model, we unzoom spatial features after the FPN and use a fixed saliency map. Inspired by the idea of zooming on semantic boundaries in [40], we generate our fixed saliency by averaging the ground truth semantic boundaries over the train set. All training settings are identical to the baseline. We use the mIoU metric and report our results in Table 6.3. Since

Table 6.3: Semantic segmentation results of PSPNet [61] on Cityscapes [13], in mIoU. We test at three input resolutions and report results relative to the uniform downsampling baseline. At each resolution, LZU outperforms uniform downsampling. At 256×256, we outperform prior works. At 64×64 and 128×128, we are worse than Optimal Edge [40] and LDS [23], perhaps because "unzooming" features at such small scales is more destructive. We also posit the performance losses from such aggressive downsampling factors (across all methods) may be too impractical for deployment, and so focus on the 256×256 downsampling regime.

Method	Downsampled Resolution			
	64×64	128×128	256×256	512×512
Uniform (theirs)	29	40	54	–
Uniform (ours)	25.8	41.4	55.3	65.3
Optimal Edge [40]	32 (+10.3%)	43 (+7.5%)	54 (+0.0%)	–
LDS [23]	35 (**+20.7%**)	45 (**+12.5%**)	55 (+1.9%)	–
LZU, fixed	26.6 (+3.1%)	43.7 (+5.6%)	56.9 (**+2.8%**)	–

our baseline results are slightly different than reported in previous works [40] and [23], we compare results using a percent change relative to the corresponding baseline. We find increased performance over the baseline at all three scales, and at 256×256, we beat both previous works with only 2.3ms of additional latency.

6.3 RGB-based 3D Object Detection

Finally, we evaluate LZU on RGB-based 3D object detection. To the best of our knowledge, no previous work has applied LZ downsampling to this task. We use FCOS3D [57], a fully convolutional model for monocular 3D detection, with a ResNet-50 backbone [19] and FPN [35]. For our dataset, we use NuScenes [5], an autonomous driving dataset with multi-view 1600×900 RGB images for 1000 scenes and 3D bounding box annotations for 10 object classes. As is standard practice for NuScenes, we use the NuScenes Detection Score (NDS) metric, which is a combination of the usual mAP and measures of translation error, scale error, orientation error, velocity error, and attribute error (mATE, mASE, mAOE, mAVE, mAAE). We run all experiments at 0.5x scale, at an input resolution of 800×450. Our baseline trains and tests FCOS3D with uniform downsampling. We train for 12 epochs with a batch size of 16 with default parameters as in [11].

For our LZU model, again we unzoom post-FPN features and use a fixed saliency map. Inspired by FOVEA [51], our fixed saliency is generated by using KDE on the set of projected bounding boxes in the image space. All training settings are identical to the baseline. Results are given in Table 6.4. We close the gap to the 1x "upper bound" by almost half with just 0.4ms of additional latency.

Table 6.4: 3D object detection results of FCOS3D [57] on NuScenes [5]. Methods (1-3) are trained at tested at 0.5x input scale, and latencies for (2, 3) are reported relative to the baseline (1). Intuitively, size is an important cue for depth, and image deformations would stifle this signal. Suprisingly, even with deformations, LZU (2) improves upon the uniform downsampling baseline (1) in all categories and closes roughly half the gap to the 1x upper bound (4)! As with 2D detection, we see improvements even in the absence of high-resolution input data (3).

ID	Method	NDS	mAP	mATE	mASE	mAOE	mAVE	mAAE	Latency (ms)
1	Uniform	27.5	17.5	90.1	28.8	75.5	131.6	17.8	58.3
2	LZU, fixed	29.3	20.1	88.9	28.3	73.9	130.6	16.7	+0.4
3	2 w/o high-res inputs	29.2	19.9	89.1	28.3	73.1	128.5	17.3	+0.4
4	Uniform (1x)	31.2	22.4	84.2	27.4	70.9	129.6	17.4	87.9

Chapter 7

Conclusion

In our thesis, we propose two methods for foveated attention for neural nets: FOVEA and LZU. FOVEA is a highly efficient attentional model for object detection. Our model magnifies regions likely to contain objects, making use of top-down saliency priors learned from a dataset or from temporal context. To do so, we make use of differentiable image warping that ensures bounding box predictions can be mapped back to the original image space. The proposed approach significantly improves over the baselines on Argoverse-HD and BDD100K. For future work, it would be natural to make use of trajectory forecasting models to provide even more accurate saliency maps for online processing.

Then, we generalize this method across tasks with LZU, a simple attentional framework consisting of "zooming" in on the input image, computing spatial features, and "unzooming" to invert any deformations. To unzoom, we approximate the forward warp as a piecewise bilinear mapping and invert each piece. LZU is highly general and can be applied to any task with 2D spatial input and any model with 2D spatial features. We demonstrate the versatility of LZU empirically on a variety of tasks and datasets, including RGB-based 3D detection which has never been done before. We also show that LZU may even be used when high-resolution sensor data is unavailable. For future work, we can consider alternatives to the "unzoom" formulation that are perhaps less destructive than simple resampling of features.

Broader impact. Our work focuses on increasing the efficiency and accuracy of flagship vision tasks (detection, segmentation, 3D understanding) with high-resolution imagery. We share the same potential harms of the underlying tasks, but our approach may increase privacy concerns as identifiable information may be easier to decode at higher resolutions (e.g., facial identities or license plates). Because our approach agnostic to the underlying model, it is reproducible with minimal changes to existing codebases. As such, code is provided in the supplement. We include a full checklist of concerns in the supplement.

Chapter 8

Appendix

8.1 FOVEA

8.1.1 The Role of Explicit Backward Label Mapping

Related work either focus on tasks with labels invariant to warping like image classification or gaze estimation [22, 43] (discussed in Sec 3.1), or expect an implicit backward mapping to be learned through black-box end-to-end training [40] (discussed in Sec 2). In this section, we suggest that the implicit backward label mapping approach is not feasible for object detection. To this end, we train and test our KDE methods minus any bounding box unwarping. Specifically, we no longer unwarp bounding boxes when computing loss during training and when outputting final detections during testing. Instead, we expect the model to output detections in the original image space.

Due to instability, additional measures are taken to make it end-to-end trainable. First, we train with a decreased learning rate of 1e-4. Second, we train with and without adding ground truth bounding boxes to RoI proposals. The main KDE experiments do not add ground truth to RoI proposals, because there is no way of warping bounding boxes into the warped image space (the implementation of $\mathcal{T}$ does not exist). We additionally try setting this option here, because it would help the RoI head converge quicker, under the expectation that the RPN should output proposals in the original space. All other training settings are identical to the baseline setup (Sec 4.1.1) .

Results are shown in Tab 8.1. The overall AP is single-digit under all of these configurations, demonstrating the difficulty of implicitly learning the backward label mapping. This is likely due to the fact that our model is pretrained on COCO [34], so it has learned to localize objects based on their exact locations in the image, and finetuning on Argoverse-HD is not enough to “unlearn" this behavior and learn the backward label mapping. Another factor is that in the S_I and S_C cases, each image is warped differently, making the task of learning the backwards label mapping even more challenging. We suspect that training from scratch with a larger dataset like COCO and using the warp parameters (e.g. the saliency map) as input may produce better results. However, this only reinforces the appeal of our method due to ease of implementation and cross-warp generalizability (we can avoid having to train a new model for each warping mechanism).

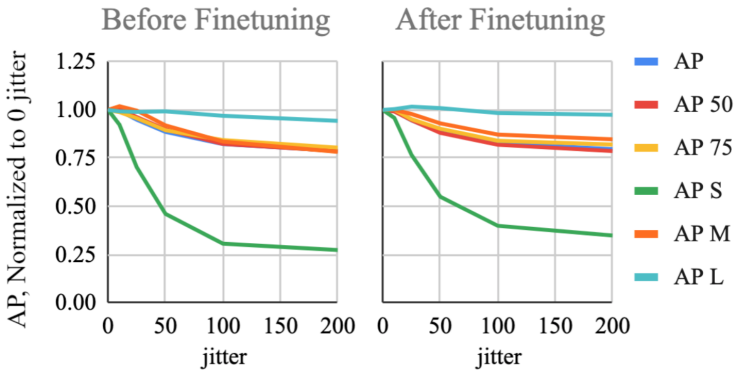


Figure 8.1: Plots showing the effect of motion (jitter) on AP using the KDE S_I formulation. Results have been normalized according to the AP at 0 jitter. As is intuitive, motion affects AP_S the most and AP_L the least. After finetuning (with an artificial jitter of 50), we see that the model reacts less adversely to jitter, indicating that our regularization has helped.

8.1.2 Sensitivity to Quality of Previous-Frame Detections

Two of our methods, S_I and S_C are dependent on the accuracy of the previous-frame detections. In this section, we analyze the sensitivity of such a dependency through a soft upper bound on S_I and S_C, which is generated using the current frame's ground truth annotations in place of detections from the previous frame. This soft upper bound is a perfect saliency map, up to the amplitude and bandwidth hyperparameters. Note that this is only a change in the testing configuration.

We report results in Tab 8.1. We see a significant boost in accuracy in all cases. Notably, the finetuned KDE S_I model at 0.5x scale achieves an AP of 29.6, outperforming the baseline's accuracy of 29.2 at 0.75x scale.

8.1.3 Sensitivity to Inter-Frame Motion

Having noted that the S_I and S_C formulations are sensitive to the accuracy of the previous-frame detections, in this section, we further test its robustness to motion between frames. We use ground truth bounding boxes (rather than detections) from the previous frame in order to isolate the effect of motion on accuracy. We introduce a jitter parameter j and translate each of the ground truth bounding boxes in the x and y directions by values sampled from $\mathcal{U}(-j, j)$. The translation values are in pixels in reference to the original image size of 1920×1200. As in Sec 8.1.2, this is a purely testing-time change. Also note that the upper bound experiments in Sec 8.1.2 follows by setting $j = 0$. We test only on S_I and report the full results in Tab 8.1. We also plot summarized results and discuss observations in 8.1.

8.1.4 FOVEA Beyond Faster R-CNN

In the main text and other sections of the appendix, we conduct our experiment based on Faster R-CNN. However, our proposed warping-for-detection framework is agnostic to specific detectors. To show this, we test our methods on RetinaNet [36], a popular single-stage object detector, and

Argoverse-HD before finetuning

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	person	mbike	tffclight	bike	bus	stop	car	truck
Main Results (copied from the main text for comparison)														
Baseline	21.5	35.8	22.3	2.8	22.4	**50.6**	20.8	9.1	13.9	7.1	48.0	16.1	37.2	20.2
KDE (S_D)	23.3	40.0	22.9	5.4	25.5	48.9	20.9	13.7	12.2	9.3	**50.6**	20.1	40.0	19.5
KDE (S_I)	**24.1**	**40.7**	**24.3**	**8.5**	24.5	48.3	**23.0**	**17.7**	**15.1**	**10.0**	49.5	17.5	**41.0**	19.4
KDE (S_C)	24.0	40.5	**24.3**	7.4	**26.0**	48.2	22.5	14.9	14.0	9.5	49.7	**20.6**	**41.0**	**19.9**
Upp. Bound (0.75x)	27.6	45.1	28.2	7.9	30.8	51.9	29.7	14.3	21.5	6.6	54.4	25.6	44.7	23.7
Upp. Bound (1x)	32.7	51.9	34.3	14.4	35.6	51.8	33.7	21.1	33.1	5.7	57.2	36.7	49.5	24.6
Without an Explicit Backward Label Mapping (Sec 8.1.1)														
KDE (S_D)	5.4	14.2	3.7	0.0	0.9	20.7	3.2	0.4	1.2	0.8	27.9	0.0	5.3	4.2
KDE (S_I)	6.1	15.6	4.0	0.2	0.8	20.3	2.3	0.6	0.7	1.8	30.8	0.0	7.0	5.4
KDE (S_C)	6.0	15.9	3.8	0.1	0.9	21.9	3.0	0.6	0.9	1.5	30.2	0.0	6.7	5.2
Upper Bound with Ground Truth Saliency (Sec 8.1.2)														
KDE (S_I)	25.4	42.6	25.6	9.1	26.2	49.5	25.3	17.4	16.8	10.1	49.4	23.4	41.7	19.4
KDE (S_C)	24.5	41.7	24.6	7.5	26.8	48.8	23.6	14.5	15.2	9.7	49.7	22.6	41.3	19.8
Sensitivity to Inter-Frame Motion (Sec 8.1.3)														
KDE (S_I), $j = 10$	25.3	42.9	25.3	8.4	26.7	49.1	25.0	16.4	16.2	10.1	48.8	25.0	41.8	19.5
KDE (S_I), $j = 25$	24.1	41.0	24.5	6.4	26.1	49.0	24.0	12.6	15.2	9.0	48.5	22.9	41.1	19.6
KDE (S_I), $j = 50$	22.5	38.3	22.9	4.2	24.1	49.1	21.9	9.9	14.4	8.2	48.4	18.5	39.0	19.7
KDE (S_I), $j = 100$	20.9	35.1	21.6	2.8	21.9	48.0	20.1	7.1	14.0	6.8	47.8	15.3	36.7	19.1
KDE (S_I), $j = 200$	20.0	33.5	20.6	2.5	20.5	46.7	19.2	6.0	13.4	6.2	46.7	14.3	35.5	18.5

Argoverse-HD after finetuning

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	person	mbike	tffclight	bike	bus	stop	car	truck
Main Results (copied from the main text for comparison)														
Baseline	24.2	38.9	26.1	4.9	29.0	50.9	22.8	7.5	23.3	5.9	44.6	19.3	43.7	26.6
Learned Sep.	27.2	44.8	28.3	**12.2**	29.1	46.6	24.2	14.0	22.6	7.7	39.5	**31.8**	50.0	27.8
Learned Nonsep.	25.9	42.9	26.5	10.0	28.4	48.5	25.2	11.9	20.9	7.1	39.5	25.1	49.4	28.1
KDE (S_D)	26.7	43.3	27.8	8.2	29.7	54.1	25.4	13.5	22.0	8.0	**45.9**	21.3	48.1	29.3
KDE (S_I)	28.0	45.5	**29.2**	10.4	**31.0**	**54.5**	27.3	16.9	**24.3**	**9.0**	44.5	23.2	**50.5**	28.4
KDE (S_C)	27.2	44.7	28.4	9.1	30.9	53.6	27.4	14.5	23.0	7.0	44.8	21.9	49.9	**29.5**
LKDE (S_I)	**28.1**	**45.9**	28.9	10.3	30.9	54.1	**27.5**	**17.9**	23.6	8.1	45.4	23.1	50.2	28.7
Upp. Bound (0.75x)	29.2	47.6	31.1	11.6	32.1	53.3	29.6	12.7	30.8	7.9	44.1	29.8	48.8	30.1
Upp. Bound (1x)	33.3	53.9	35.0	16.8	34.8	53.6	33.1	20.9	38.7	6.7	44.7	36.7	52.7	32.7
Without an Explicit Backward Label Mapping (Sec 8.1.1)														
KDE (S_D), no RoI GT	2.1	2.6	2.5	0.0	0.0	4.0	0.6	0.0	0.0	0.6	14.8	0.0	0.0	0.9
KDE (S_D)	1.8	2.7	1.9	0.0	0.0	3.2	0.6	0.0	0.0	0.0	13.3	0.0	0.1	0.6
KDE (S_I), no RoI GT	2.5	3.0	2.9	0.0	0.1	4.3	0.7	0.0	0.0	0.6	17.0	0.9	0.0	0.9
KDE (S_I)	2.0	2.8	2.4	0.0	0.0	3.7	0.6	0.0	0.0	0.0	14.8	0.0	0.3	0.5
Upper Bound with Ground Truth Saliency (Sec 8.1.2)														
KDE (S_I)	29.6	48.7	30.7	12.0	32.8	54.4	28.3	16.3	27.7	9.9	43.9	30.6	50.9	28.8
KDE (S_C)	27.8	45.5	28.8	9.6	31.7	53.4	27.5	13.9	24.7	6.5	44.5	25.1	50.2	29.6
Sensitivity to Inter-Frame Motion (Sec 8.1.3)														
KDE (S_I), $j = 10$	29.4	48.3	30.7	11.5	32.8	54.6	27.9	15.9	27.2	9.7	43.7	31.1	50.6	28.7
KDE (S_I), $j = 25$	28.0	46.1	29.2	9.2	32.1	55.3	26.4	13.9	25.9	9.3	43.9	26.8	49.2	28.7
KDE (S_I), $j = 50$	26.2	42.9	27.7	6.6	30.5	54.9	24.1	12.1	24.9	8.6	44.1	21.8	46.2	27.9
KDE (S_I), $j = 100$	24.5	39.9	25.8	4.8	28.6	53.5	22.3	10.2	23.5	7.6	43.5	17.7	43.9	27.1
KDE (S_I), $j = 200$	23.6	38.3	25.2	4.2	27.8	53.0	21.4	8.6	22.8	7.4	42.9	16.6	42.7	26.6

Table 8.1: Additional Diagnostics Experiments on Argoverse-HD.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet, Before Finetuning on Argoverse-HD						
Baseline (0.5x)	18.5	29.7	18.6	1.3	17.2	48.8
KDE (S_I)	18.5	31.2	17.9	4.5	16.8	44.9
Upp. Bound (0.75x)	24.8	38.8	25.5	4.5	28.7	52.0
RetinaNet, After Finetuning on Argoverse-HD						
Baseline (0.5x)	22.6	38.9	21.4	4.0	22.0	53.1
KDE (S_I)	24.9	40.3	25.3	7.1	27.7	50.6
Upp. Bound (0.75x)	29.9	48.6	30.1	9.7	32.5	54.2
YOLOF, Before Finetuning on Argoverse-HD						
Baseline (0.5x)	15.0	25.4	14.3	0.6	11.0	46.0
KDE (S_I)	16.8	29.0	16.0	0.9	14.0	46.4
Upp. Bound (0.75x)	21.6	35.5	22.3	2.3	22.2	52.7
YOLOF, After Finetuning on Argoverse-HD						
Baseline (0.5x)	18.4	30.5	18.3	1.4	16.5	47.9
KDE (S_I)	21.3	36.7	20.2	3.5	21.8	49.7
Upp. Bound (0.75x)	25.1	41.3	25.3	4.7	27.6	54.1

Table 8.2: Experiments with RetinaNet [36] and YOLOF [9]. We follow the same setup as the experiment with Faster R-CNN. The top quarter suggests that unlike Faster R-CNN, RetinaNet does not work off-the-shelf with our KDE warping. However, the second quarter suggests similar performance boosts as with Faster R-CNN can be gained after finetuning on Argoverse-HD. Interestingly, for YOLOF, our method boosts AP in all categories – small, medium, and large – even with off-the-shelf weights.

on YOLOF [9], a recent YOLO variant that avoids bells and whistles and long training schedules (up to 8x for ImageNet and 11x for COCO compared to standard schedules for YOLOv4 [3]).

For both these detectors, we test baselines at 0.5x and 0.75x scales both before and after finetuning. We then compare these results against our KDE S_I method at 0.5x scale. We use a learning rate of 0.01 for the RetinaNet KDE S_I model and 0.005 for the RetinaNet baselines. All other training settings for RetinaNet are identical to the Faster-RCNN baseline. For YOLOF, we use a learning rate of 0.012 and keep all other settings true to the original paper. Results are presented in Tab 8.2.

8.1.5 Comparison Against Additional Baselines

There are other approaches that make use of image warping or patch-wise zoom for visual understanding. The first noticeable work [43], explained extensively in the main text, warps the input image for tasks that have labels invariant to warping. The second noticeable work [16] employs reinforcement learning (RL) to decide which patches to zoom in for high-resolution

processing. In this section, we attempt to compare our FOVEA with these two approaches.

Our method builds upon spatial transformer networks [22, 43] and we have already compared against [43] sporadically in the main text. Here provides a summary of all the differences (see Tab 8.3). A naive approach might directly penalize the discrepancy between the output of the (warped) network and the unwarped ground-truth in an attempt to implicitly learn the inverse mapping, but this results in abysmal performance (dropping 28.1 to 2.5 AP, discussed in Sec 8.1.1). To solve this issue, in Sec 3.1, we note that [22, 43] actually learn a backward map $\mathcal{T}^{-1}$ instead of a forward one $\mathcal{T}$. This allows us to add a backward-map layer that transforms bounding box coordinates back to the original space via $\mathcal{T}^{-1}$, dramatically improving accuracy. A second significant difference with [22, 43] is our focus on attention-for-efficiency. If the effort required to determine where to attend is more than the effort to run the raw detector, attentional processing can be inefficient (see the next paragraph). [43] introduces a lightweight saliency network to produce a heatmap for where to attend; however, this model does not extend to object detection, perhaps because it requires the larger capacity of a detection network (see Sec 4.1.1). Instead, we replace this feedforward network with an essentially *zero*-cost saliency map constructed via a simple but effective global spatial prior (computed offline) or temporal prior (computed from previous frame's detections). Next, we propose a technique to prevent cropping during warping (via reflection padding, as shown in Fig 3.4), which also boosts performance by a noticeable amount. Finally, as stated in the training formulation in Sec 3.2, it *doesn't even make sense* to train a standard RPN-based detector with warped input due to choice of delta encoding (which normally helps stabilize training). We must remove this standard encoding and use GIoU to compensate for the lost stability during training.

Method	AP
FOVEA (Ours full)	28.1
w/o Explicit backward mapping	2.5
w/o KDE saliency (using saliency net as in [43])	Doesn't train
w/o Anti-crop regularization	26.9
w/o direct RPN box encoding	N/A

Table 8.3: Summary of Key Modifications in FOVEA.

Next, we attempt to compare against this RL-based zoom method [16] using our baseline detector (public implementation from mmdetection [8]) on their Caltech Pedestrian Dataset [15]. However, while their full-scale 800×600 Faster R-CNN detector reportedly takes 304ms, our implementation is *dramatically* faster (44ms), consistent with the literature for modern implementations and GPUs. This changes the conclusions of that work because full-scale processing is now faster than coarse plus zoomed-in processing (taking 28ms and 25ms respectively), even assuming a zero-runtime RL module (44ms $<$ 28ms + 25ms).

8.1.6 Additional Visualizations

Please refer to Fig 8.2 and 8.3 for additional qualitative results of our method.

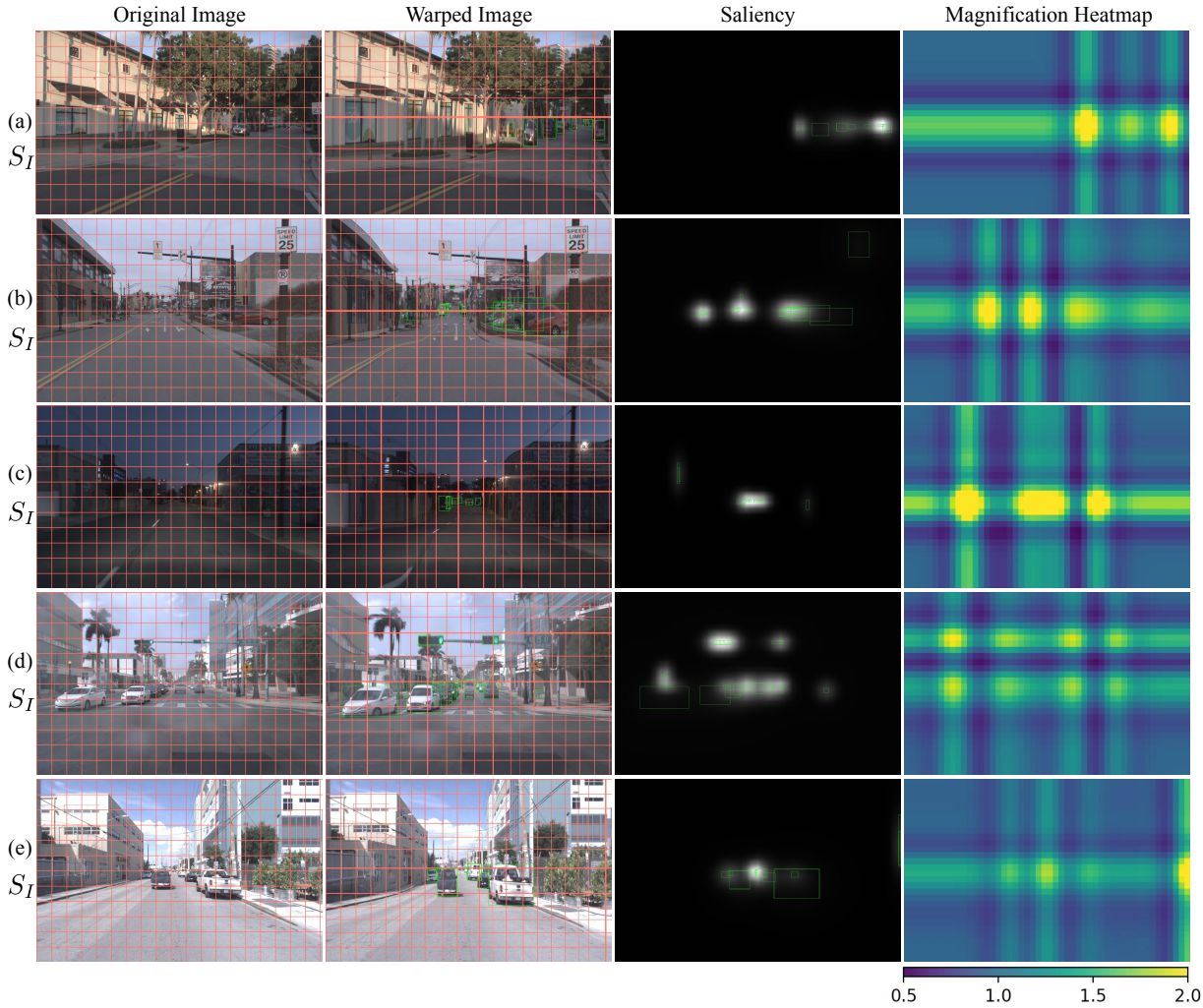


Figure 8.2: Additional examples of the S_I KDE warping method. Bounding boxes on the saliency map denote previous frame detections, and bounding boxes on the warped image denote current frame detections. The magnification heatmap depicts the amount of magnification at different regions of the warped image. (a) is an example of S_I correctly adapting to an off-center horizon. (b) shows a multimodal saliency distribution, leading to a multimodal magnification in the x direction. (c) is another example of S_I correctly magnifying small objects in the horizon. (d) is a failure case in which duplicate detections of the traffic lights in the previous frame leads to more magnification than desired along that horizontal strip. One solution to this could be to weight our KDE kernels by the confidence of the detection. (e) is another failure case of S_I, in which a small clipped detection along the right edge leads to extreme magnification in that region. One general issue we observe is that the regions immediately adjacent to magnified regions are often contracted. This is visible in the magnification heatmaps as the blue shadows around magnified regions. This is a byproduct of the dropoff in attraction effect of the local attraction kernel. Perhaps using non-Gaussian kernels can mitigate this issue.

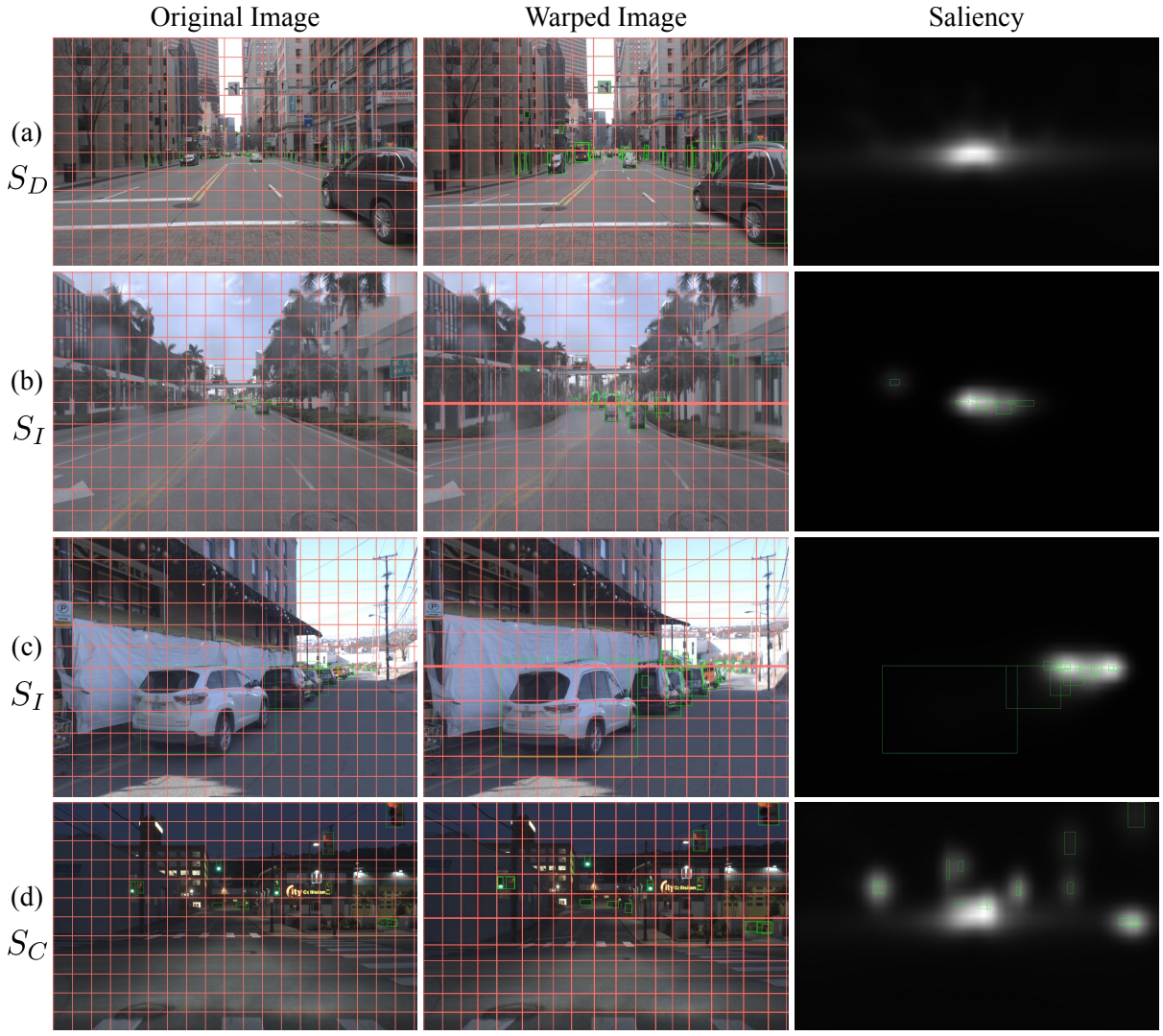


Figure 8.3: Examples of KDE warp computed from bounding boxes, extracted from a training dataset (S_D) or the previous frame's detections (S_I, S_C). We visualize predicted bounding boxes in the warped image. Recall that large objects won't be visible in the saliency due to their large variance from Eq 3.8. (a) S_D magnifies the horizon (b) S_I magnifies the center of the image, similar to S_D (c) S_I adapts to magnify the mid-right region (d) S_C's saliency combines the temporal and spatial biases.

ID	Method	AP	AP_S	AP_M	AP_L
1	Prior art [30]	13.0	1.1	9.2	26.6
2	+ Better implementation	14.4	1.9	11.5	**27.9**
3	+ Train with pseudo GT	15.7	3.0	14.8	27.1
4	2 + Ours (S_I)	15.7	4.7	12.8	26.8
5	3 + Ours (S_I)	**17.1**	**5.5**	**15.1**	27.6

Table 8.4: Streaming evaluation in the detection-only setting. First, we are able to improve over previous state-of-the-art through better implementation (row 2) and training with pseudo ground truth (row 3). Second, our proposed KDE warping further boosts the streaming accuracy (row 4-5).

8.1.7 Detection-Only Streaming Evaluation

In Sec 4.2 of the main text, we provide the full-stack evaluation for streaming detection. Here we provide the detection-only evaluation for completeness in Tab 8.4. This setting only allows detection and scheduling, and thus isolating the contribution of tracking and forecasting. We observe similar trend as in the full-stack setting in Tab 4.2.

8.1.8 Additional Implementation Details

In this section, we provide additional details necessary to reproduce the results in the main text.

For the learned separable model from Sec 4.1.2 , we use two arrays of length 31 to model saliency along the x and y dimensions, and during training, we blur the image with a 47×47 Gaussian filter in the first epoch, a trick introduced in [43] to force the model to zoom. For the learned nonseparable model, we use an 11×11 saliency grid, and we blur the image with a 31×31 filter in the first epoch. We use an attraction kernel k with a standard deviation of 5.5 for both versions. Additionally, we multiply the learning rate and weight decay of saliency parameters by 0.5 in the first epoch and 0.2 in the last two epochs, for stability. We find that we don't need anti-crop regularization here, because learning a fixed warp tends to behave nicely.

For each of our KDE methods, we use arrays of length 31 and 51 to model saliency in the vertical and horizontal directions, respectively. This is chosen to match the aspect ratio of the original input image and thereby preserve the vertical and horizontal "forces" exerted by the attraction kernel.

For the baseline detector, we adopt the Faster R-CNN implementation of mmdetection 2.7 [8]. All our experiments are conducted in an environment with PyTorch 1.6, CUDA 10.2 and cuDNN 7.6.5. For streaming evaluation, we mention a performance boost due to better implementation in Tab 8.4 & Tab 4.2, and the changes are mainly adopting newer versions of mmdetection and cuDNN compared to the solution in [30] (switching from a smooth L1 loss to L1 loss for the regression part and code optimization).

8.2 LZU

8.2.1 Bilinear Transformations

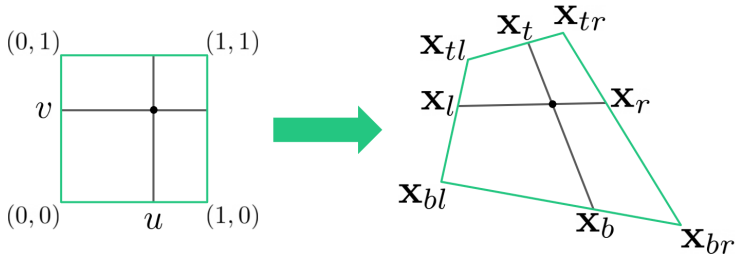


Figure 8.4: Geometric interpretation of bilinear transformations. Suppose we have such a transformation from the unit square to an arbitrary quadrilateral. Given coordinates (u, v) in the unit square, if we draw lines in the quadrilateral such that $u = \frac{|\mathbf{x}_b\mathbf{x}_{bl}|}{|\mathbf{x}_{br}\mathbf{x}_{bl}|} = \frac{|\mathbf{x}_t\mathbf{x}_{tl}|}{|\mathbf{x}_{tr}\mathbf{x}_{tl}|}$ and $v = \frac{|\mathbf{x}_l\mathbf{x}_{bl}|}{|\mathbf{x}_{tl}\mathbf{x}_{bl}|} = \frac{|\mathbf{x}_r\mathbf{x}_{br}|}{|\mathbf{x}_{tr}\mathbf{x}_{br}|}$, they will intersect at $\text{BilinearTransformation}(u, v)$.

Our construction in Section 4.1 assumes prior knowledge of bilinear transformations. Bilinear transformations have actually been widely studied in the context of computer graphics [58]. Here, for unfamiliar readers, we outline its definition and inverse formulation, as it appears in Eq 3, 4.

For simplicity, consider a bilinear transformation that maps the unit square to the quadrilateral with corners $\mathbf{x}_{bl}, \mathbf{x}_{br}, \mathbf{x}_{tl}, \mathbf{x}_{tr}$. True to its name, the transformation is defined *within* the square via bilinear interpolation:

$$\text{BilinearTransformation}(u, v) = \mathbf{x}_{bl} + (\mathbf{x}_{br} - \mathbf{x}_{bl})u + (\mathbf{x}_{tl} - \mathbf{x}_{bl})v + (\mathbf{x}_{tr} - \mathbf{x}_{br} - \mathbf{x}_{tl} + \mathbf{x}_{bl})uv \quad (8.1)$$

Interestingly, this transformation also has a geometric interpretation, shown in Figure 8.4.

Now, consider the inverse of this mapping. Given a point (x, y) in the quadrilateral, we want to find the point (u, v) in the unit square that maps to it. A full derivation is given in [58], but if we define the following scalars

$$(a_0, b_0) = \mathbf{x}_{bl} \quad (8.2)$$
$$(a_1, b_1) = \mathbf{x}_{br} - \mathbf{x}_{bl} \quad (8.3)$$
$$(a_2, b_2) = \mathbf{x}_{tl} - \mathbf{x}_{bl} \quad (8.4)$$
$$(a_3, b_3) = \mathbf{x}_{tr} - \mathbf{x}_{br} - \mathbf{x}_{tl} + \mathbf{x}_{bl} \quad (8.5)$$
$$c_0 = a_1(b_0 - y) + b_1(x - a_0) \quad (8.6)$$
$$c_1 = a_3(b_0 - y) + b_3(x - a_0) + a_2b_1 - a_2b_1 \quad (8.7)$$
$$c_2 = a_3b_2 - a_2b_3, \quad (8.8)$$

then the solution (u, v) must satisfy

$$c_2v^2 + c_1x + c_0 = 0 \quad (8.9)$$

and

$$u = \frac{x - a_0 - a_2v}{a_1 + a_3v}. \quad (8.10)$$

Applying the quadratic formula on Eq 8.9, we can solve for v. Then, we can substitute into Eq 8.10 to find u. Given a point (x, y) in the quadrilateral, this will produce exactly one pair of solutions (u, v) in the unit square (there may be extraneous solutions with u or v negative or greater than 1).

Although these results assume a mapping from the unit square, they extend naturally to our use case. Recall from Section 4.1 that $\text{BilinearTransformation}_{ij}$ maps $R(i,j) = [\frac{i-1}{H'-1}, \frac{i}{H'-1}] \times [\frac{j-1}{W'-1}, \frac{j}{W'-1}]$ to the quadrilateral formed by evaluating $\mathcal{T}$ at the corners. We can apply all previous results, simply by normalizing the coordinates within $R(i,j)$.

8.2.2 Efficient Inversion of Nonseparable Warps

In Section 4.2, we detail how to efficiently invert separable zooms $\widetilde{\mathcal{T}}_{\text{LZ,sep}}$. To invert nonseparable zooms $\widetilde{\mathcal{T}}_{\text{LZ}}$, it no longer suffices to invert each axis. We must instead reason in the full 2D space.

Suppose we have a nonseparable zoom $\mathcal{T}_{\text{LZ}}$. We compute $\mathcal{T}_{\text{LZ}}[\text{Grid}(h, w)]$ for small h, w and use this to approximate the forward zoom as $\widetilde{\mathcal{T}}_{\text{LZ}}$, an $(h-1) \times (w-1)$ piecewise tiling of bilinear maps. Now, to unzoom to a desired output resolution of $H'' \times W''$, we must evaluate $\widetilde{\mathcal{T}}_{\text{LZ}}[\text{Grid}(H'', W'')]$. That is, for each $\mathbf{x} \in \text{Grid}(H'', W'')$, we must determine which of the $(h-1)(w-1)$ quadrilateral pieces it falls in and apply the corresponding inverse bilinear map. Recall from Section 8.2.1 that applying an inverse bilinear map amounts to solving a quadratic.

In our actual implementation, we parallelize operations as much as possible. For the ij-th tile, instead of first determining which points $\mathbf{x} \in \text{Grid}(H'', W'')$ are inside of it and then applying the ij-th inverse bilinear map, we implement it the other way around. We consider a set of candidate interior points, apply the ij-th inverse bilinear map to all of them, and keep only those with a valid solution. The candidate points are those falling inside the axis-aligned rectangle enclosing that tile. The full procedure is described in Algorithm 1 and visualized in Figure 8.5.

Our implementation takes about 12.6ms to invert a nonseparable warp with $(h, w) = (31, 51)$ and an output shape $(H'', W'') = (600, 960)$, as in our Argoverse-HD [29] experiments. While this is not fast enough to support favorable accuracy-latency tradeoffs,

8.2.3 Qualitative Results

We provide an array of qualitative results from our experiments in Figure 8.6.

8.2.4 Implementation Details

Our experiments are implemented using open-source libraries MMDetection [8], MMSegmentation [12], and MMDetection3D [11], all released under the Apache 2.0 License. We use GeForce RTX 2080 Ti's for training and training, which takes at most 5 GPU-days for any given model, but the precise amount varies by model and task. We perform all timing experiments with a batch size of 1 on a single GPU.

Algorithm 1 Inverting nonseparable zooms $\mathcal{T}_{\mathrm{LZ}}$.
In practice, we optimize this code as follows. We vectorize the loop on line 13. We also fix B_{ij} to be the max size over choices of (i, j), allowing us to implement the loop on line 4 using batch-processing.

1: ▷ See Section 8.2.2 for the algorithm setup/meaning of variables
2: **function** UNZOOM($\mathcal{T}_{\mathrm{LZ}}[\mathrm{Grid}(h, w)], (H'', W'')$)
3: Initialize $\mathcal{T}_{\mathrm{LZ}}^{-1}(\mathbf{x}) = (0, 0)$ for all $\mathbf{x} \in \mathrm{Grid}(H'', W'')$
4: **for** $(i, j) \in [h-1] \times [w-1]$ **do**
5: ▷ corners of $R(i, j)$
6: $\mathbf{x}'_{tl}, \mathbf{x}'_{tr}, \mathbf{x}'_{bl}, \mathbf{x}'_{br} = \left(\frac{i-1}{h-1}, \frac{j-1}{w-1}\right), \left(\frac{i-1}{h-1}, \frac{j}{w-1}\right), \left(\frac{i}{h-1}, \frac{j-1}{w-1}\right), \left(\frac{i}{h-1}, \frac{j}{w-1}\right)$
7: ▷ corners of ij-th quadrilateral tile
8: $\mathbf{x}_{tl}, \mathbf{x}_{tr}, \mathbf{x}_{bl}, \mathbf{x}_{br} = \mathcal{T}_{\mathrm{LZ}}^{-1}(\mathbf{x}'_{tl}), \mathcal{T}_{\mathrm{LZ}}^{-1}(\mathbf{x}'_{tr}), \mathcal{T}_{\mathrm{LZ}}^{-1}(\mathbf{x}'_{bl}), \mathcal{T}_{\mathrm{LZ}}^{-1}(\mathbf{x}'_{br})$
9: ▷ top-left and bottom-right corners of rectangle enclosing the quadrilateral tile
10: $\mathbf{c}_{tl}, \mathbf{c}_{br} = \min(\mathbf{x}_{tl}, \mathbf{x}_{tr}, \mathbf{x}_{bl}, \mathbf{x}_{br}), \max(\mathbf{x}_{tl}, \mathbf{x}_{tr}, \mathbf{x}_{bl}, \mathbf{x}_{br})$
11: ▷ set of all candidate points in the ij-th tile
12: $B_{ij} = \{\mathbf{x} \in \mathrm{Grid}(H'', W'') : \mathbf{c}_{tl} \leq \mathbf{x} \leq \mathbf{c}_{br}\}$
13: **for** $\mathbf{x} \in B_{ij}$ **do**
14: $\mathbf{x}' = \mathrm{BilinearTransformation}_{ij}^{-1}(\mathbf{x})$
15: **if** $\mathbf{x}'_{tl} \leq \mathbf{x}' \leq \mathbf{x}'_{br}$ **then**
16: $\mathcal{T}_{\mathrm{LZ}}^{-1}(\mathbf{x}) = \mathbf{x}'$
17: **return** $\mathcal{T}_{\mathrm{LZ}}^{-1}$

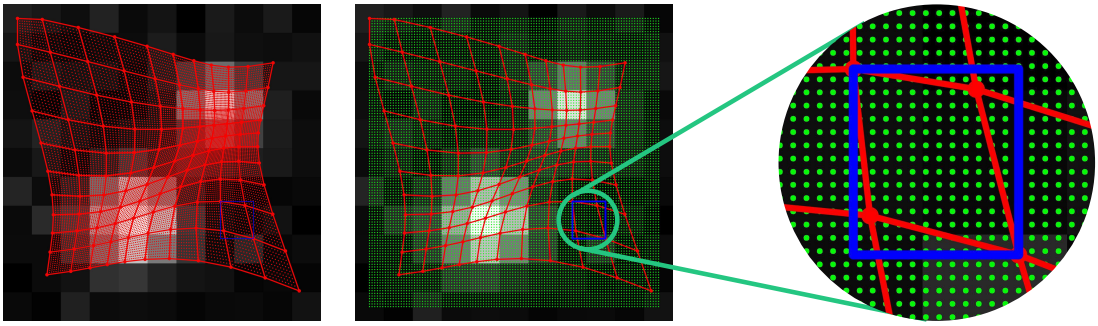


Figure 8.5: Unzooming in the nonseparable case. $\mathcal{T}_{\mathrm{LZ}}$ is approximated as $\widetilde{\mathcal{T}}_{\mathrm{LZ}}$, which is a $(h-1) \times (w-1)$ tiling of bilinear transformations (left). We wish to compute $\widetilde{\mathcal{T}}_{\mathrm{LZ}}^{-1}(\mathbf{x})$ at each green point $\mathbf{x} \in \mathrm{Grid}(H'', W'')$, where $H'' \times W''$ is the desired output size (middle). For the ij-th tile, we consider the set of all candidate green point $\mathbf{x}$ within the enclosing blue box (right) and apply the inverse bilinear transformation. We set $\widetilde{\mathcal{T}}_{\mathrm{LZ}}^{-1}(\mathbf{x}) = \mathrm{BilinearTransformation}_{ij}^{-1}(\mathbf{x})$ only if it falls in the ij-th grid rectangle $R(i, j)$.

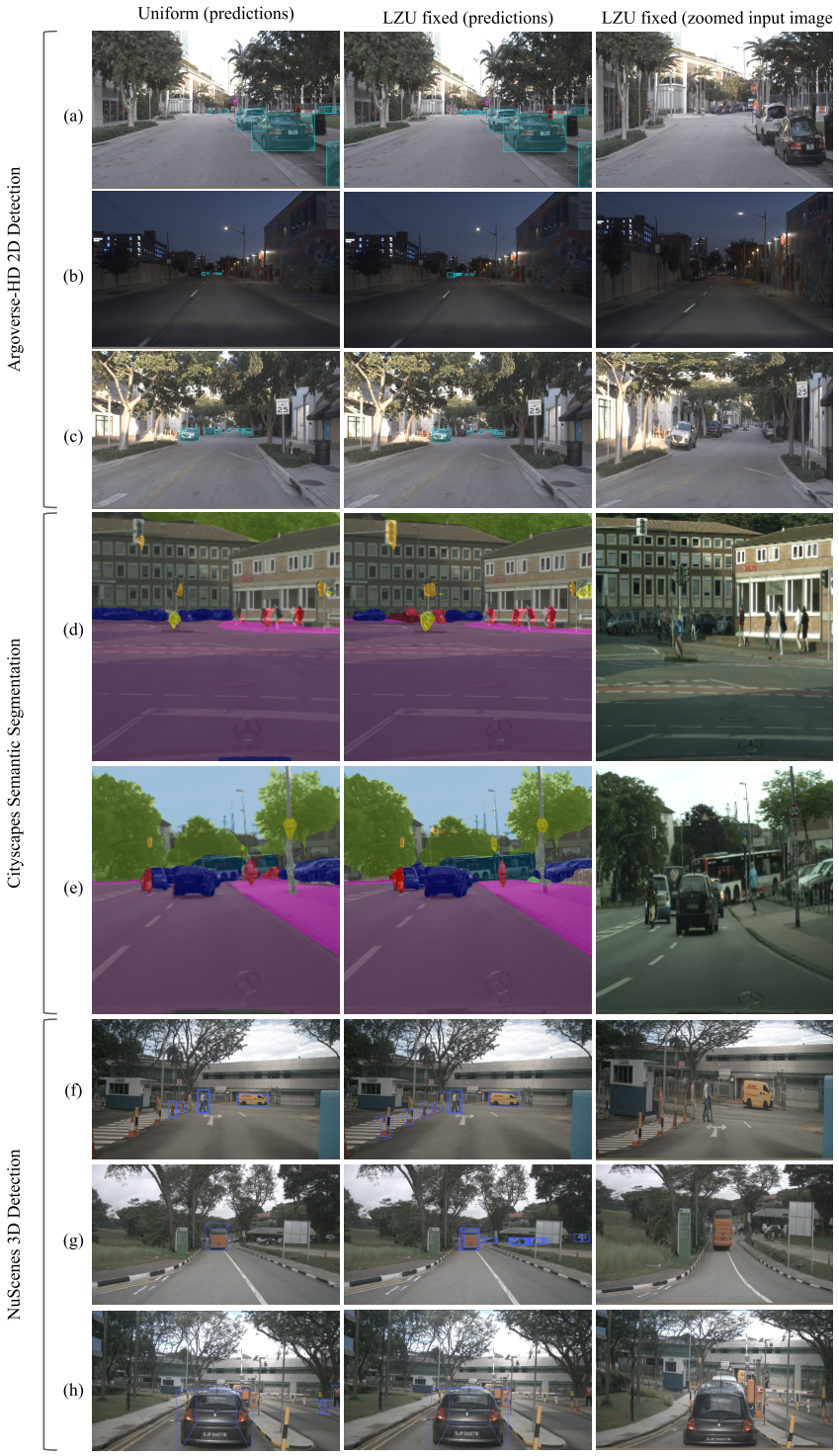


Figure 8.6: Examples of success and failure cases of LZU on 2D detection, semantic segmentation, and 3D detection. Rows a, b, f, and g depict detection examples where zooming in on the horizon helps the detector pick up smaller objects. On the other hand, there are failure cases, where zooming leads to false negatives. For example, in row c, LZU misses the black car, and in row h, LZU misses some objects near the edge of the image. For segmentation, note the consistently improved quality near the center of the image.

Argoverse-HD [29] Detection As done in [51], for our uniform downsampling experiments, we finetune a COCO-pretrained model for 3 epochs with the random left-right image flip augmentation, a batch size of 8, momentum SGD with a learning rate of 0.005, momentum of 0.9, weight decay of 1e-4, learning rate warmup for 1000 iterations, and a per-iteration linear learning rate decay [28].

For both LZU models, we use a learning rate of 0.01 and keep all other hyperparameters identical to the baseline. To "zoom", we use a 31×51 saliency map and the separable anti-cropping formulation $\mathcal{T}_{\text{LZ,sep,ac}}$ (as proposed in [51] and discussed in Section 3.2).

For the fixed saliency LZU model, we use Gaussian distance kernels k_x and k_y of full-width-half-maximum (fwhm) 22. To generate the fixed saliency map, we use kernel density estimation (KDE) on all training bounding boxes with hyperparameters amplitude $a = 1$ and bandwidth $b = 64$. For details on the effects of a and b, refer to [51].

For the adaptive saliency LZU model, we use Gaussian distance kernels k_x and k_y of fwhm 10. To generate adaptive saliency, we use KDE on detections from the previous frame with $a = 1$ and $b = 64$. When training, to simulate motion, we jitter bounding boxes by $\mathcal{N}(0, 7.5)$ pixels horizontally and $\mathcal{N}(0, 3)$ pixels vertically.

For each LZU experiments, we run a grid search over separable/nonseparable, amplitude $a = 1, 5, 10, 50, 100$, and distance kernel's fwhm $= 4, 10, 16, 22$ to determine optimal settings. This is done using an $80/20$ split of the train set, so as to not overfit on the real validation set. We generate this split such that locations between splits are disjoint. All other hyperparameters are chosen one-shot.

Synthetic Video COCO [34] Detection We train the uniform downsampling baseline using default MMDetection [8] hyperparameters. That is, we train from an ImageNet-pretrained backbone for 12 epochs with random left-right flip augmentation, a batch size of 16, momentum SGD with a learning rate of 0.01, momentum of 0.9, weight decay of 1e-4, learning rate warmup for 500 iterations, and step learning schedule of multiplier 0.1 at epochs 8 and 11.

For our adaptive LZU model, we keep the same training hyperparameters as the uniform downsampling baseline. For "zoom" hyperparameters, we use mostly the same configuration that was determined optimal for our Argoverse-HD [29] experiments. That is, we use the separable anti-cropping version $\mathcal{T}_{\text{LZ,sep,ac}}$ with Gaussian distance kernels k_x and k_y of fwhm 10. To generate saliency, we use KDE with amplitude $a = 1$ and bandwidth $b = 64$ (see [51] for details). Finally, we train and test with a jitter of $\mathcal{N}(0, 10)$ pixels in the x and y directions. However, since COCO images have variable aspect ratio, we are unable to maintain a fixed size saliency map. Instead, we scale the shorter side length to 31. More precisely, given an input image of size $H \times W$, we generate a saliency map of size $\lfloor Hs \rfloor \times \lfloor Ws \rfloor$ with a scale factor of $31 \cdot \max(H, W) / \min(H, W)$.

Cityscapes [13] Segmentation We train the uniform downsampling baseline using mostly the default hyperparameters from MMSegmentation [12]. The only changes are to the data augmentation pipeline and the evaluation frequency. Comprehensively, we train for 80K iterations with just the photometric distortion augmentation (random adjustments in brightness, contrast, saturation, and hue), a batch size of 16, momentum SGD with a learning rate of 0.01, momentum of 0.9, weight decay of 5e-4, and a polynomial learning rate schedule with power 0.9. To account

for overfitting, we evaluate every 10K iterations, and report the best performance.

For the fixed LZU model, we use the same training hyperparameters as the baseline. To "zoom", we use the separable anti-cropping formulation $\mathcal{T}_{\text{LZ,sep,ac}}$ with a 45×45 saliency map and Gaussian distance kernels k_x, k_y of fwhm 15. To generate the fixed saliency, we aggregate ground truth semantic boundaries over the train set. Precisely, we define boundaries to be pixels which differ from at least one of its eight neighbors. We compute semantic boundaries for each 256×256 ground truth segmentation, assign boundaries an intensity of 200 and background an intensity of 1, and average pool down to a 45×45 saliency map. The semantic boundary intensity value was chosen qualitatively (for producing a reasonably strong warp) and tested one-shot.

NuScenes [5] 3D Detection We train the uniform downsampling baseline using all default hyperparameters from [11], except the learning rate, which we reduce for stability. Specifically, we train for 12 epochs with the random left-right flip augmentation, a batch size of 16, momentum SGD with a learning rate of 0.001, momentum of 0.9, weight decay of 1e-4, doubled learning rate on bias parameters with no weight decay, L2 gradient clipping, a step learning rate schedule with drops at epochs 8 and 11, and a linear learning rate warmup for the first 500 iterations.

For the fixed LZU model, we use the same training hyperparameters as the baseline. To "zoom", we use the separable anti-cropping formulation $\mathcal{T}_{\text{LZ,sep,ac}}$ with a 27×48 saliency map and Gaussian distance kernels k_x, k_y of fwhm 10. To generate the fixed saliency, we project 3D bounding boxes into the image plane and reuse the same KDE formulation with the same hyperparameters ($a = 1$ and $b = 64$) as used in 2D detection. These are all chosen and evaluated one-shot.

8.2.5 Code

We include code for our NuScenes [5] 3D detection experiments in the supplementary zip submission. To set this up, please install MMDetection3D [11] version 0.18.0 from source. The attached `lzu_mmdet3d.zip` file contains all changes necessary to implement LZU. Please unzip and move the modified files into the MMDetection3D codebase. We do this due to the 100MB supplementary quota. Finally, to run experiments, simply run the scripts under the `experiments` folder.

We plan to release publicly release code for other tasks as well.

Bibliography

[1] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. *ACM SIGGRAPH computer graphics*, 26(2):35–42, 1992. 3.1, 3.2, 5.1.1

[2] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. In *ICCV*, 2019. 2.1.2

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 2.1.1, 8.1.4

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 1

[5] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 1, 6.3, 6.4, 8.2.4, 8.2.5

[6] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3D tracking and forecasting with rich maps. In *CVPR*, 2019. 1

[7] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *CVPR*, pages 4974–4983, 2019. 2.1.1, 4.1.1

[8] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 8.1.5, 8.1.8, 8.2.4, 8.2.4

[9] Qiang Chen, Yingming Wang, Tong Yang, Xiangyu Zhang, Jian Cheng, and Jian Sun. You only look one-level feature. *CVPR*, 2021. 4.1.1, 8.2, 8.1.4

[10] Yu Cheng, D. Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *ArXiv*, abs/1710.09282, 2017. 1

[11] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. `https://github.com/open-mmlab/mmdetection3d`, 2020. 6.3, 8.2.4, 8.2.4, 8.2.5

[12] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 8.2.4, 8.2.4

[13] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 1, 6.2, 6.3, 8.2.4

[14] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 2.1.3, 2.2.1

[15] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 34, 2012. 8.1.5

[16] Mingfei Gao, Ruichi Yu, Ang Li, Vlad I Morariu, and Larry S Davis. Dynamic zoom-in network for fast object detection in large images. In *CVPR*, pages 6926–6935, 2018. 2.1.3, 8.1.5, 8.1.5

[17] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014. 2.1.1

[18] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *TPAMI*, 37:1904–1916, 2015. 2.1.1, 3.2.4

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4.1.1, 4.1.2, 6.1, 6.2, 6.3

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017. 3.2.4

[21] Chen Huang, Simon Lucey, and Deva Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *ICCV*, pages 105–114, 2017. 2.1.3

[22] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015. 1, 2.2.2, 3, 3.1, 3.1, 3.1, 3.2, 3.3, 5.1.1, 8.1.1, 8.1.5

[23] Chen Jin, Ryutaro Tanno, Thomy Mertzanidou, Eleftheria Panagiotaki, and Daniel C Alexander. Learning to downsample for segmentation of ultra-high resolution images. *arXiv preprint arXiv:2109.11071*, 2021. 1, 1, 1.1, 2.2.2, 5.1, 5.2.2, 6.2, 6.3

[24] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, pages 9799–9808, 2020. 2.1.3

[25] Shu Kong and Charless Fowlkes. Pixel-wise attentional gating for scene parsing. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1024–1033. IEEE, 2019. 2.2.1

[26] Adam M. Larson and Lester C. Loschky. The contributions of central versus peripheral vision to scene gist recognition. *Journal of vision*, 9 10:6.1–16, 2009. 1

[27] Jun Li, Yongjun Chen, Lei Cai, Ian Davidson, and Shuiwang Ji. Dense transformer networks. *arXiv preprint arXiv:1705.08881*, 2017. 2.2.2

[28] Mengtian Li, Ersin Yumer, and Deva Ramanan. Budgeted training: Rethinking deep neural network training under resource constraints. *arXiv preprint arXiv:1905.04753*, 2019. 8.2.4

[29] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *European Conference on Computer Vision*, pages 473–488. Springer, 2020. 1, 6.1.1, 6.1, 6.1.2, 8.2.2, 8.2.4, 8.2.4

[30] Mengtian Li, Yuxiong Wang, and Deva Ramanan. Towards streaming perception. In *ECCV*, 2020. 1, 2.1.2, 4.1, **??**, 4.2, **??**, 8.1.8

[31] Mengtian Li, Ersin Yumer, and Deva Ramanan. Budgeted training: Rethinking deep neural network training under resource constraints. In *ICLR*, 2020. 4.1.1

[32] Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *CVPR*, pages 3193–3202, 2017. 2.1.3

[33] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, pages 7083–7093, 2019. 2.1.2

[34] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 1, 6.1.2, 6.2, 8.1.1, 8.2.4

[35] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2.1.1, 4.1.1, 5.2.2, 6.1, 6.1.1, 6.1.2, 6.2, 6.3

[36] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 4.1.1, 6.1, 6.1, 8.1.4, 8.2

[37] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *AAAI*, volume 32, 2018. 2.1.3

[38] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016. 2.1.1

[39] Hao Luo, Wenxuan Xie, Xinggang Wang, and Wenjun Zeng. Detect or track: Towards cost-effective video object detection/tracking. In *AAAI*, volume 33, pages 8803–8810, 2019. 2.1.1

[40] Dmitrii Marin, Zijian He, Peter Vajda, Priyam Chatterjee, Sam Tsai, Fei Yang, and Yuri Boykov. Efficient segmentation: Learning downsampling near semantic boundaries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2131–2141, 2019. 2.1.3, 2.2.2, 6.2, 6.3, 8.1.1

[41] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive

feature pyramid and switchable atrous convolution. *arXiv preprint arXiv:2006.02334*, 2020. 2.1.1

[42] Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, F. Durand, and Saman P. Amarasinghe. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2013. 1

[43] Adria Recasens, Petr Kellnhofer, Simon Stent, Wojciech Matusik, and Antonio Torralba. Learning to zoom: a saliency-based sampling layer for neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 51–66, 2018. 1, 1, 2.1.3, 2.2.2, 3, 3.1, 3.1, 3.1, 3.2, 3.2.2, 3.3, 4.1.2, 5.1, 5.1.2, 5.1, 5.1.2, 5.2.1, 5.2.2, 8.1.1, 8.1.5, **??**, 8.1.8

[44] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2.1.1

[45] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1, 2.1.1, 2.2.2, 3.2.4, 4.1.1

[46] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, pages 658–666, 2019. 3.2.4

[47] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. 2.1.1

[48] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 2.1.1

[49] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2019. 1

[50] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, pages 10781–10790, 2020. 2.1.1

[51] Chittesh Thavamani, Mengtian Li, Nicolas Cebron, and Deva Ramanan. Fovea: Foveated image magnification for autonomous navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15539–15548, 2021. (document), 1, 1, 1.1, 2.2.2, 3, 5.1, 5.1.2, 5.2, 5.2.2, 5.2.2, 6.1.1, 6.1, 6.1.2, 6.3, 8.2.4, 8.2.4

[52] Chittesh Thavamani, Mengtian Li, Francesco Ferroni, and Deva Ramanan. Learning to zoom and unzoom. 2022. (document), 5

[53] Burak Uzkent and Stefano Ermon. Learning when and where to zoom with deep reinforcement learning. In *CVPR*, pages 12345–12354, 2020. 2.1.3

[54] V. Vanhoucke, A. Senior, and M. Mao. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS*, 2011. 1

[55] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 2320–2329, 2020. 2.1.3, 2.2.1

[56] Thomas Verelst and Tinne Tuytelaars. Segblocks: Block-based dynamic resolution networks for real-time segmentation. *arXiv preprint arXiv:2011.12025*, 2020. 2.2.1

[57] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 913–922, 2021. 1, 6.3, 6.4

[58] George Wolberg. *Digital image warping*, volume 10662. IEEE computer society press Los Alamitos, CA, 1990. 5.2.1, 8.2.1, 8.2.1

[59] Zhenda Xie, Zheng Zhang, Xizhou Zhu, Gao Huang, and Stephen Lin. Spatially adaptive inference with stochastic feature sampling and interpolation. In *European conference on computer vision*, pages 531–548. Springer, 2020. 2.2.1

[60] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, June 2020. 1, 4.3

[61] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 1, 6.2, 6.3

[62] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *ICCV*, pages 408–417, 2017. 2.1.1

[63] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. Towards high performance video object detection. In *CVPR*, pages 7210–7218, 2018. 2.1.1

[64] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019. 2.2.1