Beyond the Camera: Neural Networks in World Coordinates

Gunnar A Sigurdsson, Abhinav Gupta, Cordelia Schmid, Karteek Alahari

Beyond the Camera: Neural Networks in World Coordinates

Gunnar A. Sigurdsson [1⋆]
Abhinav Gupta [1] Cordelia Schmid [2] Karteek Alahari [2]
[1]Carnegie Mellon University [2]Inria⋆⋆
github.com/gsig/world-features

Abstract. Eye movement and strategic placement of the visual field onto the retina, gives animals increased resolution of the scene and suppresses distracting information. This fundamental system has been missing from video understanding with deep networks, typically limited to 224 by 224 pixel content locked to the camera frame. We propose a simple idea, WorldFeatures, where each feature at every layer has a spatial transformation, and the feature map is only transformed as needed. We show that a network built with these WorldFeatures, can be used to model eye movements, such as saccades, fixation, and smooth pursuit, even in a batch setting on pre-recorded video. That is, the network can for example use all 224 by 224 pixels to look at a small detail one moment, and the whole scene the next. We show that typical building blocks, such as convolutions and pooling, can be adapted to support WorldFeatures using available tools. Experiments are presented on the Charades, Olympic Sports, and Caltech-UCSD Birds-200-2011 datasets, exploring action recognition, fine-grained recognition, and video stabilization.

1 Introduction

The success of recent vision systems is in a way surprising, since the input is typically a 224×224 pixel image, or in the case of videos, a temporal stack of such images [1,6]. This is both low resolution, and does not give much flexibility to investigate important signals in the image. The system is constrained to what the camera recorded, and has typically no active role in collecting the data. In particular, video recognition architectures have been shown to be vulnerable to camera motion, subject size, and temporal scale [28]. In comparison, the human visual system is not a passive receiver—our eyes are constantly moving to increase the effective resolution of the scene and suppress irrelevant signals [8].

These eye movements exist in humans and animals with foveal vision, and can be categorized as: *Stabilization*, the vestibulo-ocular reflex describes a control system the human visual system uses to stabilize images on the retina given proprioceptive information, like head rotation [3]. *Smooth Pursuit*, where the

⋆ Work was done while Gunnar was at Inria.
⋆⋆ Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France.

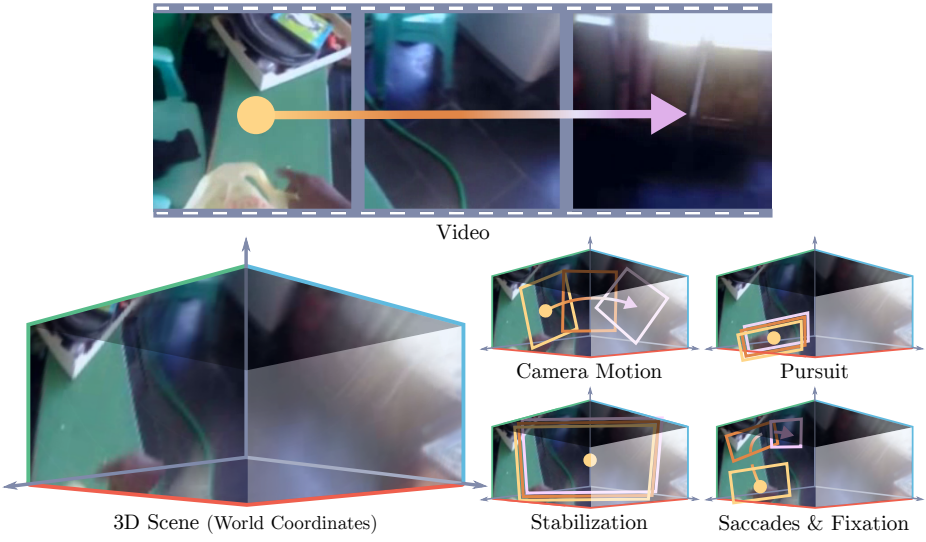


Fig. 1. In this paper we explore how to model the variety of eye motions, even on pre-recorded data. We generalize the concept of feature maps, to WorldFeatures, that include a relative location in space and time, which can encode any type of "eye motion" we want apply to the recorded data, or undo. Orange to pink is used to denote features at time $t-1$, t, and $t+1$. We show a video from a first-person view, along with the *Camera Motion* in the reconstructed world frame. The original video follows a particular sequence of viewpoints, however, crucial information about the activities in the video requires different views—*Stabilization* transforms the views to the world frame, *Pursuit* transforms the views to follow the subject of the video, and *Saccades & Fixation* scans the scene to extract fine-grained information from the video.

gaze is voluntarily shifted to track a moving object, effectively stabilizing the object of interest on the retina [15]. *Fixation*, where gaze is fixed towards a single location, to enhance resolution of that area [24]. *Saccades*, where the eyes quickly jerk between phases of fixation to explore the scene [14]. In Fig. 1 we illustrate how a video is constrained to what the camera recorded, but that also the video may be refocused. How can we build a vision system that has this flexibility to freely explore the data, and move beyond a fixed 224×224 window?

We propose a simple and effective idea—each feature, has a location in real-world coordinates. This combination of `(feature, transformation)` pairs is referred to as a WorldFeature. An example transformation might be image coordinates to real-world coordinates (camera matrix). It turns out preprocessing the video data to, for example, stabilize it, introduces new problems, as demonstrated in Fig. 2. Instead, we use these WorldFeatures in all layers in the network and index the data according to the transformation or transform it as needed. That is, *implicit* transformation instead of *explicit*. With such network, we can utilize any type of "eye motion" we want apply to the recorded data, or undo.

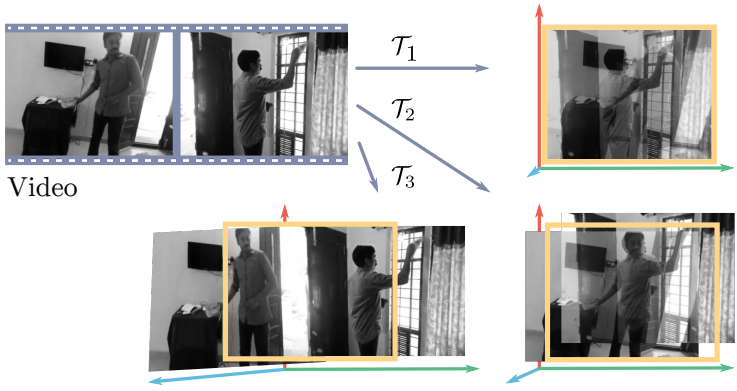


Fig. 2. Illustration of WorldFeatures, attaching transformations to features. Top left is the original video shown as two frame featuremap. We show the temporal average of an aligned feature map, with three different transformations $\mathcal{T}_1$ (identity), $\mathcal{T}_2$ (pursuit), $\mathcal{T}_3$ (stabilization). Observe how different transforms highlight different aspects of the data. We show a potential coordinate frame, that could be used to fit the data into a network, highlighting a problem with naively stabilizing the data, for example, when 64 frames in a row need to be aligned. WorldFeatures get around this by keeping the transformation and only using it as needed.

To illustrate the idea, in Fig. 2 we present a simple two frame featuremap with three different transformations $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ to showcase how different transformations can highlight different signals in the data. For example, in the third it is easy to understand the global layout of the scene, and in the second it is easier to understand the difference in the pose of the person. In Sec. 3 we show more details on WorldFeatures, and how to keep track of the transformations after every layer to build networks with these operations.

Contributions. In this paper we propose the idea of WorldFeatures, where each feature has a transformation, on multiple vision tasks: Fine-grained Image Recognition (Sec. 4.1), Video Stabilization (Sec. 4.2), and Video Activity Recognition (Sec. 4.3). We formalize WorldFeatures, and show how different eye movements can be modeled in this framework. Further, we propose a simple implementation that can adapt optimized neural networks tools to operate on WorldFeatures. This implementation can extend many state-of-the-art building blocks that operate over space and time, such as 3D Convolutions, and MaxPooling. The output of each building block, is also WorldFeatures, such that it can be processed hierarchically (e.g. 3D-ResNet [36]).

2 Related Work

Many image and video understanding architectures have been proposed in the last few decades and we refer the reader to [20,37] for a detailed survey, and focus here on recent video architectures related to our approach.

Video Architectures. Recently, networks have become more evolved for separating content and motion [1,25,30,6]. This direction is analogous to the different specialization of the ventral and dorsal streams in the mammalian brain. Similarly, separating camera motion and subject motion has shown great promise in video segmentation [31]. As the field moves towards utilizing longer temporal context [38], equipping these networks with the capability to separate camera motion, object permanence, scene dynamics, and subject motion can greatly improve those efforts. This paper aims to provide the building blocks necessary to construct these systems.

Video Stabilization. Related to our work on understanding eye motion, is the task of video stabilization that has been pursued directly [9,35,18], or with additional supervision [17]. Of particular relevance is recent work that has incorporated stabilization and tracking of points over time to improve accuracy of video recognition [33,34]. Inspired by these successes, we go a step further, and build a framework that can operate given an arbitrary transformation signal, and use this to improve the vision system.

Improving Visual Input. On the other hand, recent work has also explored how to use the input data more effectively. This has been done through spatially or temporally modifiable connections [4], non-local connections where feature similarity guides the connections [36], or transformations of the input [13]. Furthermore, [22] investigated the idea that not all input features are equally useful, and explored how to learn how to highlight and "zoom in" on the important content in each image. As we will see, eye movement is primarily useful to allow for efficient allocation of resources, and since our system can handle arbitrary transformations, we can explicitly utilize various "zoom in" of the data. Furthermore, our WorldFeatures is a system that allows a network to generalize beyond the camera, and is complementary to frameworks that learn attention to emphasize parts of the data.

3 WorldFeatures: Feature Maps with Camera Movements

We start by introducing the ideas behind WorldFeatures, that is, `(feature, transformation)` pairs. Then we demonstrate how different eye movements can be modeled in this framework. A video is a specific window into the world at the time it was recorded, and a system analyzing this video might have a completely different objective for watching the scene than the camera operator. To undo the camera bias or highlight signals in the video, we allow a transformations that are analogous to eye movements, after the data has already been recorded.

To undo camera transformations, could we pre-process the data? For example, applying affine transformations to each frame to undo camera movement, for stabilization. In fact, even this simple case has problems: How to fit this transformed data into the model? At what scale should we process the data? How to use imperfect transforms? Consider eyes scanning a scene, fixating on a variety of points in the scene at multiple resolutions. Creating data that follows

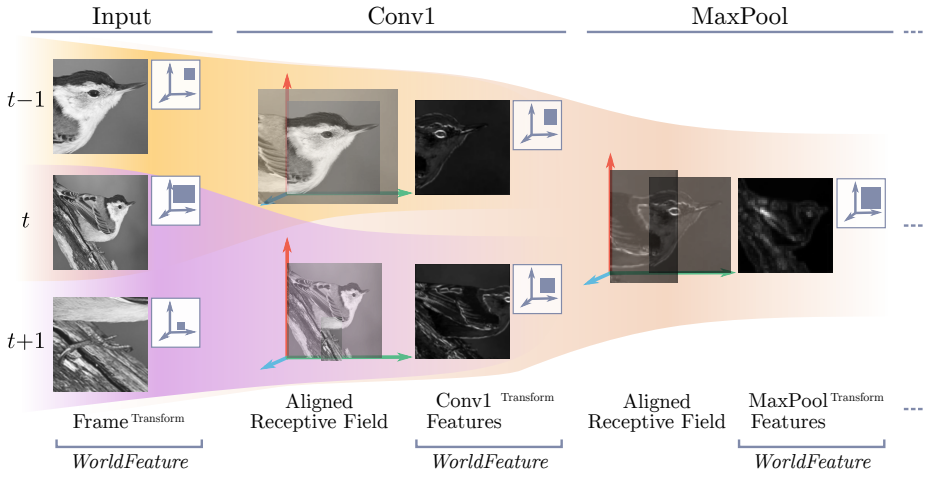


Fig. 3. Demonstration of a model using WorldFeatures at each layer. Here we look at an illustration of the outputs of conv1 and maxpool layers, when applied to an "eye movement" task. The input and output of each layer are WorldFeatures. The receptive field of each output featuremap is aligned, explicitly or implicitly depending on implementation, before computing the output. Since each output feature map has a different alignment, each output has a different transformation, and we repeat the process at every layer. This shows that WorldFeatures allow the network to combine information across multiple locations and scales in the image.

a similar pattern without an understanding how they relate to each other creates a noisy and disjointed view of the scene. Instead, by keeping track of the relative transformations between features, we can crop, scale, and zoom in on the data as required, while still maintaining correct relationships between features.

3.1 WorldFeatures Definition

The key to our idea is the concept of WorldFeatures $\mathbf{x}^{\mathcal{T}}$. That is, a feature map $\mathbf{x}$ of size $T{\times}H{\times}W$ (time, height, width) and an arbitrary transform $\mathcal{T}$ which can encode any type of "eye motion" we want to apply to the recorded data—or undo.* We refer to a single $H{\times}W$ timepoint as a frame, and note that $\mathbf{x}{=}\mathbf{x}^{\mathcal{I}}$, where $\mathcal{I}$ is the identity transform. The feature of $\mathbf{x}^{\mathcal{T}}$ at (t,h,w) is written $\mathbf{x}\,(\mathcal{T}(t,h,w))$. Therefore, extending a convolution operation to work on $\mathbf{x}^{\mathcal{T}}$:

$$\mathbf{y}^{\mathcal{T}'}(t,h,w) = \sum_i \sum_j \sum_k \mathbf{x}(\mathcal{T}(i,j,k)) \cdot \mathbf{f}(t{-}i, h{-}j, w{-}k)\,, \tag{1}$$

where $\mathbf{y}^{\mathcal{T}'}$ is also a WorldFeature, that can then be passed to the next convolution operation and so on. Here $\mathbf{f}$ is a filter, and we omit batch and channel dimensions

* For example, the affine transform, $\mathcal{T}(t,h,w) = (t,\ a_t^0 h{+}a_t^1 w{+}a_t^2,\ a_t^3 h{+}a_t^4 w{+}a_t^5)$, where a_t^n are the parameters at frame t.

for clarity. Compared to regular convolutions, the transformation $\mathcal{T}$ is necessary to transform the features as needed. For example, the valid values of each feature map are only defined for $t \in [0, T)$, $h \in [0, H)$, and $w \in [0, W)$, and transforming the video before processing would move content out of the frame, etc. This convolution layer could use a Spatial Transformer Network [13] independently transforming the receptive field for each output frame, at every layer, exponentially increasing computation. This implementation of WorldFeatures is illustrated in Fig. 3. Fortunately, in Sec. 4 we outline an efficient implementation of this for CNN building blocks, which allows us to explore this type of networks.

3.2 Transformation Types

This framework can be used to model eye movements: Stabilization, Smooth Pursuit, Fixation and Saccades. Fig. 1 contains illustrative examples for each that we refer to for each of these transformations.

Stabilization. The first and most intuitive transformation $\mathcal{T}$ we encounter is the stabilizing transform. We obtain the camera matrix for every frame, and invert it to obtain the stabilizing transform. That is, the transformed feature map $\mathbf{x}(\mathcal{T}(t,h,w))$, will contain the same point in the world at location (h,w) for any t. The *Stabilization* example in Fig. 1 visualizes how the frames of the given video are arranged in the world frame.

Smooth Pursuit. Since the transform $\mathcal{T}$ is arbitrary we can also choose it to stabilize with respect to a different point of reference. That is, instead of (h,w) for any t pointing to the same point in the world, we can make it point to the same point on a moving object of interest, such as a person. The *Pursuit* example in Fig. 1 visualizes how the frames of the given video can be arranged, and combined with attention to the hands of the person.

Fixation. To model fixation, where the high-resolution part of the retina (the fovea) is used, we use a transform that enhances part of the frame. That is, if $\mathbf{x}^{\mathcal{T}}(t,h,w)$ is the original WorldFeature, then we add a "fixation transform" $\mathcal{F}$ to get the WorldFeature $\mathbf{f}^{\mathcal{T}\circ\mathcal{F}}$ where $\mathbf{f} = \mathbf{x}(\mathcal{F}(t,h,w))$. This fixation transform "zooms in" on a part of the input, increasing the resolution.

Saccades / Visual Search. To model saccades, we proceed similarly to fixation, and define a transform $\mathcal{S}$ that gazes at a particular part of the frame, allowing the model to attend to different parts of the image. The similarity with the implementation of fixation is because saccades are in fact defined as the jump between fixations. In the *Saccades & Fixation* example in Fig. 1 we demonstrate how the transformations can be chosen to pay close attention to important aspects of the scene, such as the hands, objects and context.

In conclusion, there are multiple eye movements possible, allowing a variety of augmentations without sacrificing relative location between features. In the experiments we use these definitions to provide the model with a transformation for each video, but learned transformations could be used as well.

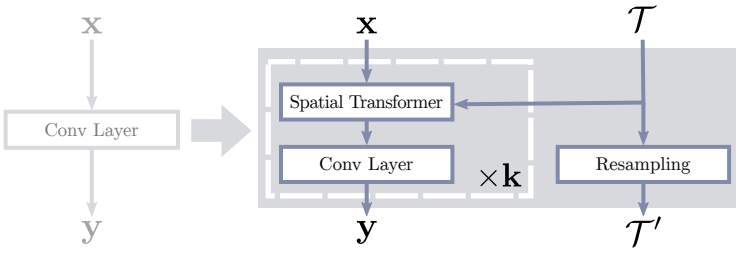


Fig. 4. Our simple implementation of WorldFeatures "wraps" a layer (e.g. conv layer) and applies a spatial transformer to the data to align the receptive field of each output, k times for a layer with temporal extent of k (e.g. 3 in ResNet), and combines the outputs.† More details are provided in the Appendix.

4 Experiments.

Here we explore WorldFeatures for diverse applications. We use a 3DResNet-style architecture for videos built from WorldFeatures. This can be used to incorporate arbitrary eye movements in videos and images, where images are just treated as an uneventful video (replicating images in time). The network can then utilize the transformations to "zoom in" on parts of the image. First, we demonstrate that a 3D network, can outperform 2D networks on fine-grained recognition of nature categories (image task). Second, we look at video stabilization, and show how implicit transformations can bypass the problems with explicitly stabilizing the input data with preprocessing. Third, we utilize enhancing transformations, smooth pursuit and fixation, to improve video recognition accuracy.

Implementation. Our implementation extends a 3DResNet50 model, following [36] (similar to I3D [1]). All convolution, max pool, and average pool layers are converted to WorldFeature layers, which implicitly operate on a transformed video. All models start with parameters from a regular 3DResNet50 pretrained on the Kinetics dataset [1]. In Fig. 4 we illustrate a simple method to convert a layer, such as a conv layer, to use WorldFeatures, without having to engineer special low-level GPU programs for each. More details are provided in the Appendix. We use the grid sampler from Spatial Transformer Networks [13], with arbitrary transformation grids. After fitting the transformer, we use the grid sampler in the nearest (pixel) setting. This avoids blurring of the feature maps as they are repeatedly transformed. To address missing values after transforming, we add a channel denoting missing data at a point, similar to [7]. For comparison, all methods use the same batch-size of 2 unless otherwise noted, and follow the same learning rate schedule tuned for the baseline. The models are implemented using the PyVideoResearch [26] framework in PyTorch, and will be available at github.com/gsig/world-features.

† For k=3 the first transformer aligns frames {0,1,2} (the receptive field for output frame 1), and frames {3,4,5} (receptive field for output frame 4), etc. The second transformer aligns {1,2,3} (receptive field for output frame 2), etc.

	Top-1 (%)	Resolution (px.)
3DResNet50	79.8	224
DT-RAM [16]	82.8	227
ResNet50	83.4	224
Learning to Zoom [22]	84.5	227
3DResNet50 w/Saccades	**85.3**	224
DT-RAM [16]	86.0	448

Table 1. Fine-grained Recognition on the Caltech-UCSD Birds-200-2011 dataset.

4.1 Saccades - Fine-Grained Recognition

Motivated by human gaze studies in image recognition, we explore using 3D video networks for fine-grained recognition on the Caltech-UCSD Birds-200-2011 dataset [32]. To provide a gaze trajectory for the network, we use saliency [12] and objectness [2] to generate 64 bounding boxes. We replicate each 2D image 64 times in time to form a 3D input. The 64 bounding boxes form the basis for the fixation in each frame, and are ordered to maximize the overlap between consecutive boxes, allowing the network to learn filters that combine information between various fixations and scales. See Fig. 5 for an example. We compare with regular *3DResNet50*, and *ResNet50*. We start with a 3DResNet50 and fine-tune with the saccades/fixation setup, and combine with ResNet50.

Results. The results are presented in Table 1. With a input size of 224 pixels, our method outperforms architectures specialized for fine-grained recognition, even with recent methods such as [22] that learn the function of what to zoom to, whereas we use off-the-shelf saliency. Combining our framework and learning saliency is likely to yield additional gains. The method from [16] only outperforms other methods when provided with 448×448 image requiring a special network beyond the scope of this work. These results demonstrate an exciting new avenue for video architectures applied to the conventionally 2D task of fine-grained recognition. In Fig. 5 we visualize the points of interests and scales suggested by the saliency algorithm, and how that is translated into pixel data that the network sees (along with the transformations between those). We also show example of featuremaps from 3 layers in the network, where we can see how the network combines information across the different saccades, and builds detailed fine-grained understanding of the image.

4.2 Stabilization - Video Activity Recognition

Stabilization is a fundamental vision task, and has been explored in various contexts. Here we specifically explore stabilization for improving video recognition.

To evaluate the stabilization performance, we start with videos with synthetic camera motion. We use 50% of the Charades [29] videos with the least motion**,

** We used variance of optical flow, yielding 4396 videos of average 30 sec.

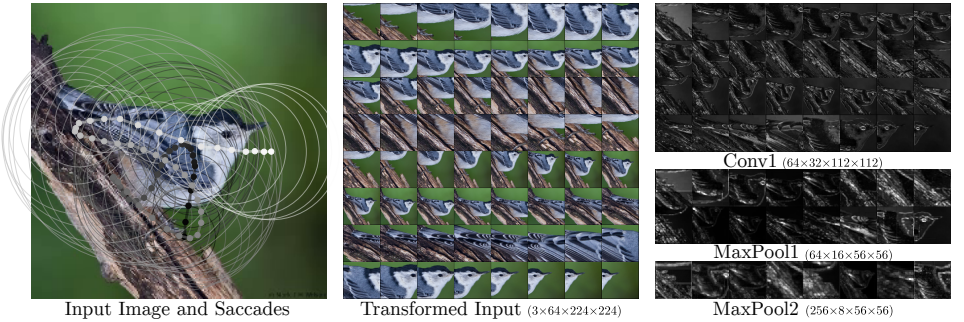


Fig. 5. In the left column we visualize the path (the saccades) that the model chooses, given static saliency and objectness, where dark to light denotes frames 1 through 64, and the large circles denote the scale of the fixaton in that frame. In the the middle column, we show the pixel data that the model uses (the model also receives the transformations). On the right, we show featuremaps from Conv1, MaxPool1, and MaxPool2, high values denote high variance over time at that pixel.

and add synthetic camera movement by linearly moving between 3 locations at randomly chosen scale levels (30%–300%, *Synthetic*). Next we consider Olympic-Sports [19], that was used to showcase the advantage of stabilization in the seminal IDT work [33]. Charades is evaluated with a video-level mean average precision over 157 classes, and Olympic-Sports uses a top1 accuracy for each video over 20 classes. Our baselines are *3DResNet50* [36], and 3DResNet50 with all frames are stabilized to the center frame (*3DResNet50++*). All models use a 16 temporal frame stack as input to make stabilization feasible for the baseline, and are trained with the same learning rate schedule and hyperparameters tuned to the baseline. For each video we construct the camera motion from the video using direct optimization of affine transformations parameters between consecutive frames. The details are provided in the Appendix.

Results. In Table 2 we see that using the stabilization transformation allows our method to improve over the baselines. That is, modelling features and transformations together instead of preprocessing allows our method to avoid compounding errors do to stabilization over many frames, and loss of resolution.

Table 2. Results on stabilization on two video activity recognition datasets. Stabilizing the video before inputting it to the network (*3DResNet50++*) helps in some cases, but has drawbacks, whereas *implicit* stabilization can better utilize the input.

	3DResNet50	3DResNet50++	Pursuit
Synthetic (Video mAP)	13.1	13.5	**14.1**
Olympic-Sports (% Top-1)	96.4	97.2	**97.2**

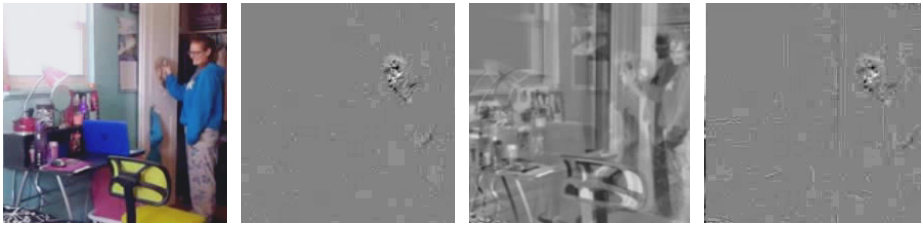


Fig. 6. A single frame from 4 feature maps. First feature map is original video, second is the output of a temporal difference filter applied to an ideal (stable) video, third is output of the filter applied to the actual unstable video, and finally the output of our method given the estimated stabilizing transform.

Analysis. We also looked at an egocentric video dataset, Charades-Ego [27]. With 64 frame inputs, preprocessing (*3DResNet50++*) actually reduces performance in the baseline (23.6 → 22.8 mAP), and with 16 frame inputs (only 0.67 sec video clip), preprocessing only slightly improves performance (17.5 → 18.5 mAP). This demonstrates the challenge of stabilizing a video before passing it to the network, and interestingly, stabilization does not seem to help activity recognition much. Perhaps, since what the camera operator is looking at is highly informative in first-person video. In any case, WorldFeatures have performance at par with the better method in each case (23.4 and 18.4 mAP).

In Fig. 6 we demonstrate how camera motion makes learning from video challenging. We show point in time from 4 feature maps. We observe that WorldFeatures can utilize a transformation to suppress irrelevant signals. Although a deep network might, with sufficient data, be able to learn all possible variations, this adds complexity the model. We will see in the next section how we can go even further and use the transformations to focus on particular parts of the data.

4.3 Smooth Pursuit & Fixation - Video Activity Recognition

To demonstrate how WorldFeatures can utilize any transformations, we explored *Smooth Pursuit*, where we use a transformation to stabilize the content with respect to a human bounding box (*Pursuit*). We use an RCNN person detector [23], and construct a the transform such that the person is always centered and fills the input to the network. Going further, we explored how to utilize the transformations to increase the effective resolution of particular parts of the scene or the subject in a video. Concretely, we used simple video saliency (temporal difference), and fit a bounding box around 80% of the variance in the saliency, and the fixation transforms the input such that the box fills the input (*Fixation*).

Our experiments are on the Charades dataset [29], evaluated with a video-level mean average precision over 157 classes. We start with a standard pre-trained

baseline network from the literature (*3DResNet50*), fine-tune with the new framework, and combine with the old network.

Results. Our results are presented in Table 3. Interestingly, adding smooth pursuit on its own does not significantly improve over the baseline, which stabilizes the video with respect to the person. However, this is expected since many of the videos in Charades have camera motion that already follows the video subject. Fortunately, we do see that using smooth pursuit followed by fixation (*FixationPursuit*) we can improve performance. This suggests that whereas stabilization on its own is not particularly helpful to current neural networks, when it is used to locate an area to increase the resolution it can offer significant benefits, since 224×224 pixels is not provide much detail, and learn filters that combine information across multiple locations and resolutions.

Table 3. Activity Recognition on the Charades dataset using smooth pursuit and fixation transformations.

	3DResNet50	Pursuit	Fixation	FixationPursuit
Charades [29] (Video mAP)	31.3	31.5	32.3	**32.6**

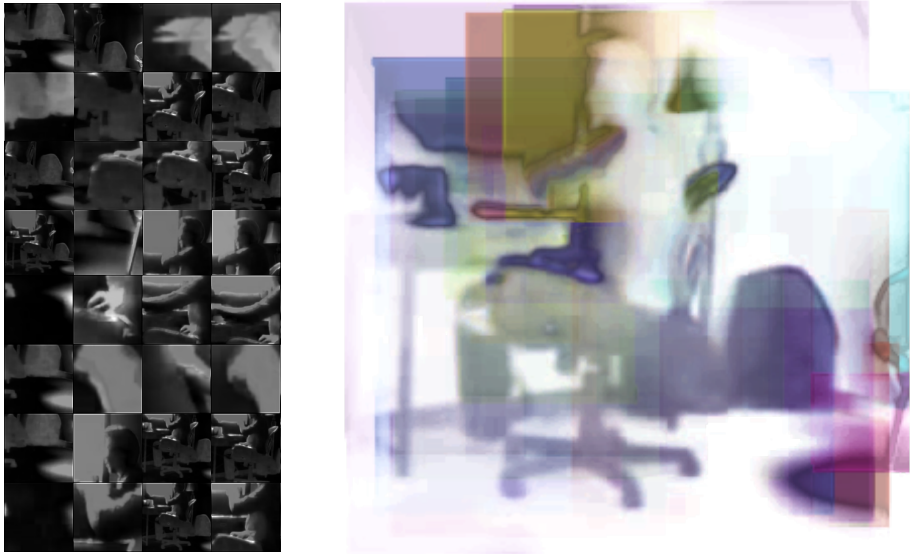
In Fig. 7 we visualize frames from a Conv1 feature map applied to a video on a person sitting at their desk, since each feature has a transformation associated with it, even though some of the feature maps are at very small scales, the network can move them into a common coordinate frame as needed.

5 Discussion & Conclusion

We presented a framework that can use transformations to better use available data, moving beyond the implicit biases of the camera that recorded a video.

Video Stabilization. One of the hypotheses that we explore is stabilization in terms of a world frame, and in terms of a target (smooth pursuit). We discovered that stabilization on its own is not very helpful for current vision systems. However, when combined with fixation and saccades that highlight useful aspects of the input, it has significant advantages.

Fixation in Animals. Similarly, eye movement and fixation in animals is hypothesized to be primarily in order to fully utilize the center of the retina, the fovea, that has high-resolution and only covers about 1-2 degrees of vision [21]. This is, it is about efficient usage of resources. Furthermore, average fixation duration in humans is only 330 ms (8 frames at 24 fps) [11], suggesting long-term stabilization might be unnecessary for some visual systems. Thus eye movement primarily plays a role in allowing efficient visual search of the scene.



Conv1 feature maps

Conv1 transformed (composite)

Fig. 7. Composite of 32 Conv1 feature maps from the fixation model. Different color indicates different frames.

Stabilization in First-Person Videos. Stabilization in the first-person vision case is complicated by the fact that the camera is commonly doing smooth pursuit of the objects/hands of interest already, and zooming in is unnecessary because the field of view is already narrow. In the Charades third-person videos the camera often follows the person, that is, smooth pursuit is already present.

Conclusion. We addressed several challenges that come with moving beyond the camera, and highlighted problems that have to be considered. Such as, variability due to scale and aspect ratio changes, how to include data augmentation, and working with pretrained networks that only consider particular types of input data. We hope this work opens the door for systems that learn policies for visual search and efficient allocation of visual resources.

Acknowledgements. This work was supported by Intel via the Intel Science and Technology Center for Visual Cloud Systems, Sloan Fellowship to AG, the Inria associate team GAYA, the ERC advanced grant ALLEGRO, gifts from Amazon and Intel, and the Indo-French project EVEREST (no. 5302-1) funded by CEFIPRA. The authors would like to thank Kenneth Marino for feedback on the manuscript.

6 Appendix

Implementation Details. Layers using WorldFeatures could be implemented with efficiency similar to their regular counterparts, requiring dedicated engineering. To iterate on the idea, we implemented WorldFeatures in high-level PyTorch code. We observe that current architectures have filter extent of 3 frames [10,36,1]. Thus, we simply transform each chunk of consecutive frames into common coordinates as needed for each filter computation. This is illustrated in Fig. 8, where 3 frames are temporarily transformed into a common coordinate frame (the coordinates of frame t), and used to compute the convolution output.

Concretely, mirroring Eq. (1) above, suppose we need to calculate $\mathbf{y}^{\mathcal{T}}(t,h,w)$ for some particular frame: $t{=}t_0$, $h \in \{0,1,\ldots,H{-}1\}$, and $w \in \{0,1,\ldots,W{-}1\}$. Then we can rewrite as follows:

$$\mathbf{x}'_t(t',h',w') = \mathbf{x}(\mathcal{T}(t',h',w')) \tag{2}$$

$$\mathbf{y}^{\mathcal{T}}(t,h,w) = \sum_i \sum_j \sum_k \mathbf{x}'_t(i,j,k) \cdot \mathbf{f}(t{-}i, h{-}j, w{-}k) \tag{3}$$

which describes convolution on $\mathbf{x}'$. Here $\mathbf{x}'$ is an explicitly transformed version of $\mathbf{x}$. Since $\mathbf{f}$ only has non-zero values at $t = \{-1, 0, +1\}$ (if filter extent is 3) $\mathbf{x}'$ only needs to be computed for $t{-}1$, t, and $t{+}1$. Thus if we need to compute this for all values T of t, we need to create T such versions of $\mathbf{x}'$, which corresponds to 3 versions of the size of the input ($T{\times}H{\times}W$). Finally, we apply the regular convolution to these 3 copies and merge the results to create $\mathbf{y}^{\mathcal{T}}(t,h,w)$.

This has numerous advantages: 1) Since 3 or less frames are being transformed at a time, both cumulative errors in transformation and missing value problems are minimal. 2) Transforming across long timescales is only done in higher layers,

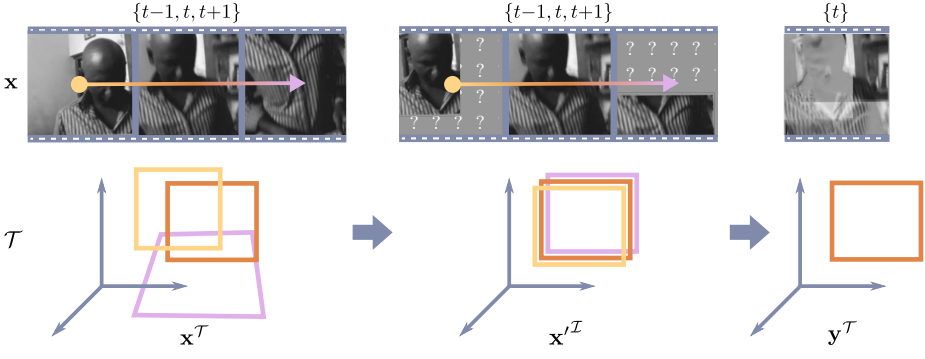


Fig. 8. Illustration of our implementation of WorldFeatures. We start with a world feature map $\mathbf{x}^{\mathcal{T}}$ and and in order to calculate the output of a convolution applied to the feature map at time t, that is $\mathbf{y}^{\mathcal{T}}(t,h,w)$, we first transform the needed features into the coordinate system at time t, and then apply the convolution as normal. Black, gray and white, are used to denote $t{-}1$, t, and $t{+}1$, respectively.

Table 4. Our 3DResNet50 model for video. The dimensions of 3D output maps and filter kernels are in T×H×W, with the number of channels following. The input is 64×224×224, and the pre-computed transformer corresponding to the input is of size 64. Residual blocks are shown in brackets. The table is adapted from [36].

layer		output size	transformer output size
conv_1	7×7, 64, stride 2, 2, 2	32×112×112	32
pool_1	3×3×3 max, stride 2, 2, 2	16×56×56	16
res_2	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$	16×56×56	16
pool_2	3×1×1 max, stride 2, 1, 1	8×56×56	8
res_3	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$	8×28×28	8
res_4	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$	8×14×14	8
res_5	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$	8×7×7	8
global average pool, fc		1×1×1	1

where spatial resolution is lower, meaning less accuracy is needed. 3) Any layer that operates on the time dimension, such as maxpool, average pool, and 3d convolution, can be adapted in the same way using this method.

In Table 4 we illustrate 3DResNet50 extended with WorldFeatures, and how the size of the transformer changes to correspond to each layer. In our setup, all consecutive layers receive a (potentially resampled) transformer that contains the transformations to align consecutive timepoints of the feature map.

Computing Stabilization. Before stabilization takes place, we need to compute the transformations between consecutive frames. Traditionally, optical flow or feature matching have been utilized for this purpose, but with increase in computation, recent work has moved towards direct methods for estimating the transformation. This optimizes the transformation directly on the image intensities, which enables utilizing all of the information in the image, making the estimation higher accuracy, and particularly robust to frames with few keypoints [5]. In particular, we directly optimize the parameters of a transformation (affine or homography), but in a batch fashion on each pair of consecutive frames in each mini batch. The objective from LSD-SLAM [5] operates directly on images intensities (Huber norm) and downweighs large values. We simplify it to:

$$\min_{\theta} \sum_{h,w} \min \left(\left\| \mathbf{x}(t,h,w) - \mathbf{x}(\mathcal{T}_\theta(t{+}1,h,w)) \right\|^2, \delta \right) , \tag{4}$$

where we align the frame at t and $t{+}1$. We scale the source $(t+1)$ and target (t) image equally such that target image has unit variance, and $\delta = 0.01$. We then use gradient-based optimization with adaptive learning rate to minimize this objective with respect to the transformation parameters θ.

References

1. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: CVPR (2017) 1, 4, 7, 13
2. Cheng, M.M., Zhang, Z., Lin, W.Y., Torr, P.: Bing: Binarized normed gradients for objectness estimation at 300fps. In: CVPR (2014) 8
3. Crawford, J., Vilis, T.: Axes of eye rotation and listing's law during rotations of the head. Journal of neurophysiology (1991) 1
4. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: ICCV (2017) 4
5. Engel, J., Schöps, T., Cremers, D.: Lsd-slam: Large-scale direct monocular slam. In: ECCV (2014) 14
6. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. arXiv (2018) 1, 4
7. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. PAMI (2009) 7
8. Findlay, J.M.: Saccadic eye movement programming: Sensory and attentional factors. Psychological Research (2009) 1
9. Grundmann, M., Kwatra, V., Essa, I.: Auto-directed video stabilization with robust l1 optimal camera paths. In: CVPR (2011) 4
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 13
11. Henderson, J.M.: Human gaze control during real-world scene perception. Trends in cognitive sciences (2003) 11
12. Hou, X., Zhang, L.: Saliency detection: A spectral residual approach. In: CVPR (2007) 8
13. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: NeurIPS (2015) 4, 6, 7
14. Javal, É.: Essai sur la physiologie de la lecture. Annales d'Oculistique (1878) 2
15. Krauzlis, R.J., Lisberger, S.G.: Temporal properties of visual motion signals for the initiation of smooth pursuit eye movements in monkeys. Journal of Neurophysiology (1994) 2
16. Li, Z., Yang, Y., Liu, X., Zhou, F., Wen, S., Xu, W.: Dynamic computational time for visual attention. In: ICCV (2017) 8
17. Liu, S., Wang, Y., Yuan, L., Bu, J., Tan, P., Sun, J.: Video stabilization with a depth camera. In: CVPR (2012) 4
18. Liu, S., Yuan, L., Tan, P., Sun, J.: Steadyflow: Spatially smooth optical flow for video stabilization. In: CVPR (2014) 4
19. Niebles, J.C., Chen, C.W., Fei-Fei, L.: Modeling temporal structure of decomposable motion segments for activity classification. In: ECCV (2010) 9
20. Poppe, R.: A survey on vision-based human action recognition. IVC (2010) 3
21. Provis, J.M., Dubis, A.M., Maddess, T., Carroll, J.: Adaptation of the central retina for high acuity vision: cones, the fovea and the avascular zone. Progress in retinal and eye research (2013) 11
22. Recasens, A., Kellnhofer, P., Stent, S., Matusik, W., Torralba, A.: Learning to zoom: a saliency-based sampling layer for neural networks. In: ECCV (2018) 4, 8
23. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NeurIPS (2015) 10
24. Rucci, M., Poletti, M.: Control and functions of fixational eye movements. Annual Review of Vision Science (2015) 2

25. Ryoo, M.S., Matthies, L.: First-person activity recognition: What are they doing to me? In: CVPR (2013) 4
26. Sigurdsson, G.A., Gupta, A.: Pyvideoresearch (2018) 7
27. Sigurdsson, G.A., Gupta, A., Schmid, C., Farhadi, A., Alahari, K.: Actor and observer: Joint modeling of first and third-person videos. In: CVPR (2018) 10
28. Sigurdsson, G.A., Russakovsky, O., Gupta, A.: What actions are needed for understanding human actions in videos? In: ICCV (2017) 1
29. Sigurdsson, G.A., Varol, G., Wang, X., Farhadi, A., Laptev, I., Gupta, A.: Hollywood in homes: Crowdsourcing data collection for activity understanding. In: ECCV (2016) 8, 10, 11
30. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: NeurIPS (2014) 4
31. Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R., Fragkiadaki, K.: Sfm-net: Learning of structure and motion from video. arXiv (2017) 4
32. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. rep., California Institute of Technology (2011) 8
33. Wang, H., Kläser, A., Schmid, C., Cheng-Lin, L.: Action recognition by dense trajectories. In: CVPR (2011) 4, 9
34. Wang, L., Qiao, Y., Tang, X.: Action recognition with trajectory-pooled deep-convolutional descriptors. In: CVPR (2015) 4
35. Wang, M., Yang, G.Y., Lin, J.K., Shamir, A., Zhang, S.H., Lu, S.P., Hu, S.M.: Deep online video stabilization. arXiv (2018) 4
36. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: CVPR (2018) 3, 4, 7, 9, 13, 14
37. Weinland, D., Ronfard, R., Boyer, E.: A survey of vision-based methods for action representation, segmentation and recognition. CVIU (2011) 3
38. Wu, C., Feichtenhofer, C., Fan, H., He, K., Krähenbühl, P., Girshick, R.B.: Long-term feature banks for detailed video understanding. arXiv (2018) 4