Patch to the Future: Unsupervised Visual Prediction

Jacob Walker, Abhinav Gupta, and Martial Hebert
Robotics Institute, Carnegie Mellon University
{jcwalker, abhinavg, hebert}@cs.cmu.edu

Abstract

In this paper we present a conceptually simple but surprisingly powerful method for visual prediction which combines the effectiveness of mid-level visual elements with temporal modeling. Our framework can be learned in a completely unsupervised manner from a large collection of videos. However, more importantly, because our approach models the prediction framework on these mid-level elements, we can not only predict the possible motion in the scene but also predict visual appearances — how are appearances going to change with time. This yields a visual "hallucination" of probable events on top of the scene. We show that our method is able to accurately predict and visualize simple future events; we also show that our approach is comparable to supervised methods for event prediction.

1. Introduction

Consider the image shown in Figure 1. A reliable modern computer vision approach might at best recognize the objects and regions in the image and list the corresponding nouns — road, car, tree and grass. However, when we humans look at the same image, we can not only infer what is happening at that instant but also predict what can happen next. For example, in the same image, we can predict that the car on the bottom right is either going to go straight or turn left at the intersection. Humans' amazing ability to visualize the future is primarily driven by the rich prior knowledge about the visual world.

We believe the task of visual prediction is important for two main reasons: (a) For intelligent agents and systems, prediction is vital for decision making. For example, in order to perform assistive activities, robots must be able to predict the intentions of other agents in the scene. Even a task as simple as walking through a crowded hallway requires the prediction of human trajectories. (b) More importantly, prediction requires deep understanding of the visual world and complex interplay between different elements of the scene. Therefore, prediction can act as a way to define "what does it mean to understand an image," and the task of visual prediction can act as the litmus test for scene understanding.

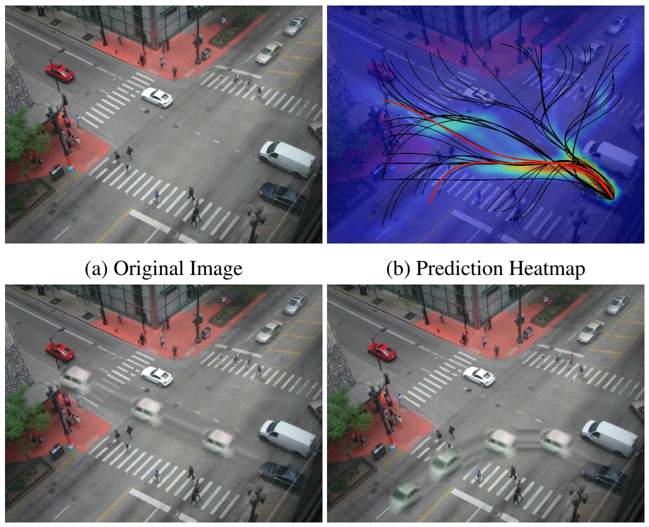

(a) Original Image (b) Prediction Heatmap (c) Predicted Path -1 (d) Predicted Path -2

Figure 1. Consider the scene shown in image (a). Our data-driven approach uses a large collection of videos to predict the likely future of an agent in the scene. The heatmap (b) shows the likely locations the car can visit in the future (along with a few possible trajectories.) (c) shows the hallucination of car moving straight and (d) shows the hallucination of the car turning left.

In this work, we take a step toward this goal of generalized visual prediction — determining *what* is active in the scene as well as *how* the activity should unfold. However, this leaves us with major questions. What do we predict? What does the output space of visual prediction look like? Recent approaches have only focused on predicting the movements and transitions of agents treated as a point object [18] or optical flow of pixels [33]. In contrast, we humans can not only predict the motion but also how the appearances would change with that movement or transition. This allows us to create mental images of prediction. In a similar manner, we argue that the space of visual prediction should be richer and even include prediction of visual appearances. For example, we can guess how a car will look after it turns and how a book unfolds when opened. However, having a richer output space requires richer representation (elements of reasoning) and lots of data to learn the priors. Building upon the recent success of mid-level elements [28], we propose a new framework for visual prediction which uses these mid-level elements as building blocks of prediction. In our framework, we model not only the movement and transitions of these elements in the scene but also how the appearances of these elements can change. Our new framework has the following advantages over previous approaches: (a) Our approach makes no assumption about what can act as an agent in the scene. It uses a data-driven approach to identify the possible agents and their activities; (b) Using a patch-based representation allows us to learn the models of visual prediction in a completely unsupervised manner. We also demonstrate how a rich representation allows us to use a simple non-parametric approach to learn a state-of-the-art visual prediction model; (c) Finally, because our approach exploits mid-level elements instead of full scenes for creating associations, it allows for generalization and sharing across different instances.

1.1. Background

Prediction is a major component of intelligence [13], found even in animals such as rats and pigeons [34]. Researchers in neuroscience have found extensive support for sensory prediction in the human brain [2]. While prediction has been extensively studied in the realm of biological vision [1], in the field of computational vision, most of the research focus has been on understanding and inferring semantic [7, 27, 32] and geometric [10, 14] knowledge of what can be seen in an image or a video. Recently, some researchers have started focusing on modeling static functional interactions of humans with the scene [11, 15] and objects [9, 31] which can be then be used to predict how will humans interact with the scene. However, in this work, we focus on the temporal aspect of forecasting, and our goal is to predict what is likely to happen next.

There have been two classes of approaches for the temporal aspect of prediction. The first is non-parametric. In this case, instead of making any assumption about the agents and the environment, these approaches rely on large databases [33]. For instance, [33] retrieves videos similar to the input scene and then builds a model of expected motion given the retrievals. However, since the matching is done based on the scene, this requires extraordinarily large amounts of training data because one needs to represent all possible spatial and temporal configurations of objects in the world explicitly. Therefore, recent approaches have focused on warping based approaches [20] to generate predictions in case the retrievals are close but not identical.

On the other extreme is a parametric and modeling based approach. Here, humans assume what are the active elements in the scene whether they may be cars or people. Once the assumption is made, then a model is developed to predict agent behavior [35]. Tracking based approaches focus on modeling agent behavior at short time scales and hence use linear models [17]. For longer term predictions, models such as Markov Decision Process (MDP) [18, 35], Markov Logic Networks [29], ATCRF [19], CRF [8], and AND-OR graphs [12, 25] are used. Prediction inference in these models involve approaches from planning and decision making [18, 35]. However, these methods have the following drawbacks: (a) they make strong assumptions about the domain; (b) they are still dependent on semantic classification which still remains a difficult problem to solve; and finally, (c) these approaches explicitly choose active agents such as cars or humans. In most cases, one either needs manual supervision or object detectors (which are robust) to train these models. Instead, we use a data-driven approach and prediction framework based on mid-level elements which are easy to detect and track. This gives us an ability to train our prediction framework in a completely unsupervised manner. But, more importantly, having a rich visual representation gives us the capability to predict visual appearances as well.

In this work, we present a conceptually simple but surprisingly powerful method which combines the two approaches of modeling dynamic scenes. Our work builds upon the recent success of mid-level discriminative patch discovery [4, 5, 6, 16, 28] and proposes a prediction framework based on these mid-level elements. Because the basic elements in our framework are based on these mid-level patches, our approach scales and provides better generalization as compared to scene-based matching approaches. However, instead of just matching patches to the training data, we learn a context-based Markov model over the patches. Our approach not only models how these mid-level elements move and change appearances but also learns a context model (or reward function similar to [18]) which captures the relationship between scene location and the patch. For example, a car patch is less likely to move on the sidewalk and receives a high cost for such a transition, but a person patch will be likely to move on the sidewalk. We build this scene-patch context model directly on image features instead of an intermediate semantic layer. We show our learning approach is robust to errors in tracking and therefore can be learned in a completely unsupervised manner.

2. Our Approach

Given an input scene, our goal is to predict what is going to happen next — what parts of the image are going to remain the same, what parts of the image are likely to move, and how they move. The central idea is that scenes are represented as a collection of mid-level elements (detected using a sliding window) where agents can either move in space or change visual appearances. Each agent is predicted independently assuming a static scene. We model the distribution over the space of possible actions using a transition matrix which represents how mid-level elements can move and transition into one another and with what probability. For example, an element that represents a frontal car can transition to a patch facing right if the car turns. Given the mid-level elements and their possible actions, we first determine which is the most likely agent and the most likely action given the scene. However, this notion of most likely action depends upon goals and the context/scene around the elements. For example, in Figure 1, the visual prediction of a car not only depends upon the goal but also on the other cars, pedestrians, and the sidewalk in the image. Therefore, as a next step, we need to model the interaction between the active element (agent) and its surrounding. We model this interaction using a reward function $\psi_i(x, y)$ which models how likely is it that an element of type i can move to location (x, y) in the image. For example, a car element will have high reward for road-like areas and low reward for grass-like areas — without modeling semantics explicitly. Given a goal, our approach then infers the most likely path using the transition matrix and computed reward (Section 2.4). Finally, if the goal is unknown —which is the case here, we propose to sample several goals and select the most likely goal based on high expected reward.

During training, we need to learn: (a) the mid-level representation; (b) the space and likelihood of transitions for each element; and the (c) reward function $\psi_i(x, y)$ for every possible element. We propose to learn these from large quantities of spatio-temporal visual data in an unsupervised manner. We first create a state space of mid-level patches that distinguish the domain from the rest of the visual world. Within a domain such as videos of cars driving on roads or pedestrians walking outdoors, we first apply the work of [28] to extract mid-level elements. These elements are visually meaningful and discriminative HOG clusters trained against a large set of general visual data. Each element can act as an agent which can move. Instead of using domain based assumptions such as agents being cars or humans, our approach exploits data to decide which features are significant and extract the agents. For example, in the case of the VIRAT dataset [23], one of the elements groups two people since the two people are likely to move together and hence can be modeled as a single agent. Once we have extracted the dictionary of mid-level elements for a given domain, we use temporal information to find patch-to-patch transitions as well as their spatial behavior on the image plane (Section 2.1). We can use the statistics of the transition matrix to determine which elements are the active agents in the scene. Finally, we learn a reward function over the state space which is combined with transition matrix to infer the predictions (Section 2.2).

2.1. Learning The Transitions

Given the dictionary (Figure 3) of mid-level elements, the first step is to learn a temporal model over these elements. The temporal model is represented using a transition matrix where element i can either move in one of the eight directions (top, left, bottom, right, top-left, top-right, bottom-left, bottom-right) or transition into another element in the dictionary. How do we learn these transitions? Given the training data, we extract pairs of frames at least a second apart and detect mid-level patches in both the frames. To learn the transition we need to obtain the correspondence between the detections in the two frames. We obtain this correspondence by counting the number of common features tracked using the KLT Tracker [22] inside the two bounding boxes.

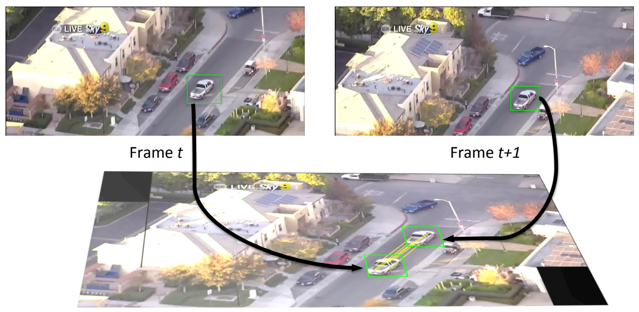


Figure 2. Illustration of patch mapping. Two frames are matched using an estimated homography, and KLT features inside the bounding boxes of detections in each frame direct patch movements.

We interpret the mapping as either an appearance or spatial transition. If the patches are of two cluster types, then the mapping is counted as a transition from one cluster type to another regardless of spatial movement. For a patch to be counted as a spatial movement on the image plane, the mapped patches must be of the same type, and they must not overlap (See the example in Figure 2). In order to compensate for camera motion these movements are computed on a stitched panorama obtained via SIFT matching [21]. For each transition, we normalize for total number of observed patches as well. This gives us the probability of transition for each mid-level patch. Figure 3 shows some of the top transitions for four mid-level elements.

2.2. Learning Contextual Information

A transition matrix captures the most likely action in absence of the contextual information. For example, a car facing right is most likely to move right. However, the actions of agents are not only dependent on the likely transitions but also on the scene and the surroundings in which they appear. For example, if there is a wall in front of the car, it is unlikely to move in that direction. Therefore, apart from capturing the statistics of patch transitions, we need information about how a patch may interact with its environment. We model these interactions using a reward function $\psi_i(x, y)$ which models how likely is it that an element of type i can move to location (x, y) in the image. Because each element is supposed to represent a different underlying

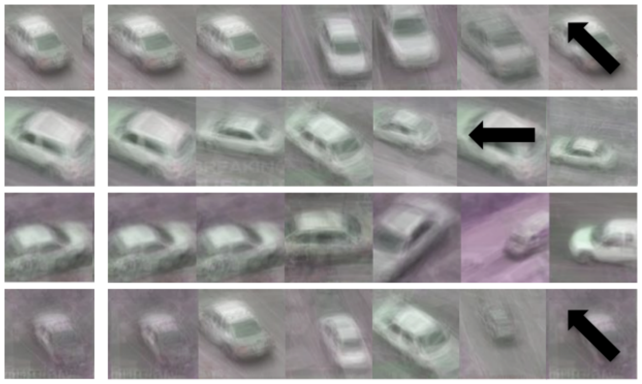


Figure 3. Top possible transitions learned from training data. On the left are the original elements, and on the right are possible transitions. Note each element can either change appearance and morph into another element, or it can just move in space (arrowed squares). The elements are shown as average images of top detections on the training data.

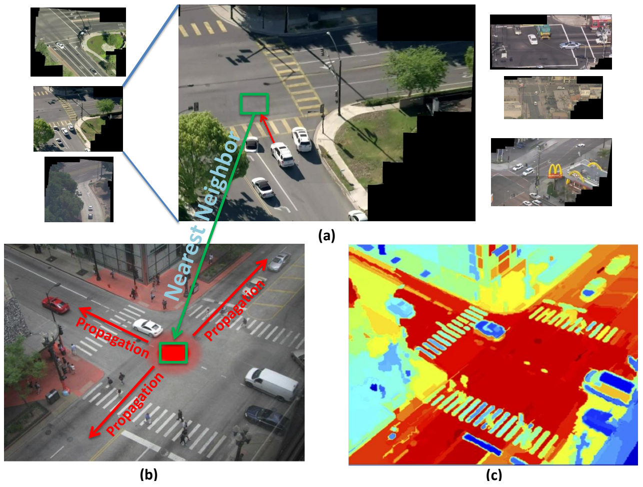


Figure 4. A reward function (c) is propagated by taking texture information from the destinations of observed moving patches in training data (a). During test time (b), the area of the image with the closest texture to the training textures is set as the highest reward in the scene. Other areas of the scene (via graphcut segments) are scored according to the similarity to the chosen window. Warm colors indicate high reward; cooler colors indicate low reward.

concept, we learn a separate reward function for the interaction of each element within the scene.

We use a non-parametric approach over segments to model the reward function. To obtain the training data for reward function of element type i, we detect the element in the training videos and observe which segments are likely to overlap with that element in time. For example, the car elements are likely to overlap with road segments, and hence those road segments act as instances of positive reward areas for car element. Using such instances, we build a training set for every element in the dictionary. Once we have the training sets for every patch type i, we can use this to compute the reward function at test time. Each segment in the test image retrieves the nearest neighbor using Euclidean distance over image features. We choose the top-N nearest neighbors to label high reward areas in the image and then propagate the reward function within the image based on visual similarity — graphcut segments which look similar to high reward regions in the query image also get high reward. Figure 4 shows an example of reward propagation.

2.3. Inferring Active Entities

Once we have learned the transition function and reward function $\psi_i(x, y)$ for every mid-level element, we can predict what is going to happen next. The first step of prediction inference requires estimating the elements in the scene that are likely to be active. Kitani et al. [18] choose the active agents manually. In this work, we propose an automatic approach to infer the likely active agent based on the learned transition matrix. Our basic idea is to rank cluster types by their likelihood to be spatially active. We assume the active agents are the elements that are: (a) likely to move themselves; (b) likely to transition to patches that can move; (c) in a scene that allows the element to move to high reward areas in its neighborhood. In order to detect the top possible elements in a scene that satisfy these properties, we first detect the instances of each element using sliding-window detection. We then rank these instances based on contextual information. The context-score for a patch i at location (x, y) is given by

$$\sum_d p_i^d e^{\psi_i(x+d_x,y+d_y)} \tag{1}$$

where $d = (d_x, d_y)$ is the direction of the movement, p_i^d is the transition probability in direction d and $\psi_i(x+d_x, y+d_y)$ computes the reward for moving the patch from (x, y) to $(x + d_x, y + d_y)$. In this paper, we discretize d into eight directions.

Instead of predicting all the elements discovered, we only predict the activities of elements that are likely to change location either directly or by transition. We compute the likelihood of changing location based on the transition matrix. Therefore, the elements which have high movement transition likelihood or transition to a element that has high movement likelihood are selected.

2.4. Planning Actions and Choosing Goals

Once we have selected the most likely active agents, we use the transition matrix combined with the reward function to search for optimal actions/transitions given a spatial goal in the scene. We first re-parameterize the reward function $\psi_i(x, y)$ such that if the state is $s = (x, y, i)$ (patch i being at location (x, y)), then the reward is $\phi(s)$. Each decision a is quantified by the expected reward: *i.e.,* the product of the probability of the decision p_a and the reward function $\phi(s)$ in the new state. Note that the decision can either be a movement or be a transition. In the first case, the location (x, y) changes in the state while in the second case, we have the same location by a new cluster type.

Our goal is to find the optimal set of actions/decisions $\sigma = (a_1, ..a_n)$, such that these actions maximize expected reward (minimize cost), and these actions reach the goal state g. We formulate this as maximization of the reward function

$$\max_{\sigma} \sum_{a_t \in \sigma} p_{a_t} \phi(s_{t+1}) \quad \text{s.t. } \sigma \odot s_0 = g \tag{2}$$

where s_0 is the initial state and $\odot$ operator applies a set of actions to a state to estimate goal state. We then use Dijsktra's algorithm to plan a sequence of optimal decisions σ from an initial state to all given goal states by *converting rewards to costs*. Specifically, we create a graph where each state is represented as a node in the graph. For example, for a 100x100 image and dictionary size of 750 elements, there will be 100x100x750 nodes in the graph. The edges between the nodes represent the cost of transitioning from state s_i to s_j. This cost depends on the transition probabilities and rewards. Given this graph, the initial state is represented as the source node in the graph, and the goal nodes are considered to be along the edge of image. We then run Dijsktra's algorithm to get the optimal path. We select the best path among different goals based on average expected reward — normalized with respect to the total number of decisions.

2.5. Implementation Details

KLT Tracker: We use the Kanade-Lucas tracking algorithm on extracted SURF features [3] to track how detected patches move in each scene. Given detected patches in two frames, we track the SURF features which initially lie inside the bounding box of a given patch.

Reward Function: The distance metric for the reward function is computed using 69-dimensional feature vector based on RGB and a bag of words.

Other Details: The selected frames during transition matrix learning were 4 seconds apart in the VIRAT dataset and only one second apart in the car chase dataset due to faster motion.

3. Experimental Results

Because there has been little work in the field of visual prediction, there are no established datasets, baselines, or evaluation methodologies. We perform extensive qualitative and quantitative evaluation for path prediction, and we provide detailed qualitative analysis for prediction of visual appearances.

Baselines: There are no algorithms for unsupervised visual prediction; therefore we compare against Nearest Neighbor followed by sift-flow warping [20, 33] and the max-entropy based Inverse Optimal Control (IOC) based algorithm of Kitani et al. [18]. For the NN baseline, we use a Gist-matching [24] approach similar to that of Yuen et. al. [33]. We then use the labeled path from the nearest-neighbor as the predicted trajectory and warp it into the scene using Sift Flow [20]. For Kitani et al. [18], we first learn a reward function using IOC, and then given the initial agent we predict the most likely paths using a Markov Decision Process (MDP).

Datasets: We perform experiments on two different datasets: a Car Chase Dataset (collected from YouTube) and the VIRAT dataset [23].

Evaluation Metric: We use the modified Hausdorff distance (MHD) from [18] as a measure of the distance between two trajectories. The MHD allows for local time warping by finding the best local point correspondence over a small temporal window (3 steps in our experiments). Each algorithm will generate a collection of likely predicted paths and therefore we will compute the distance between the top-N generated predictions and the ground-truth path.

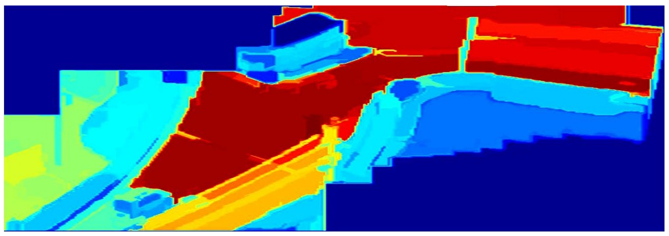
3.1. Car Chase Dataset

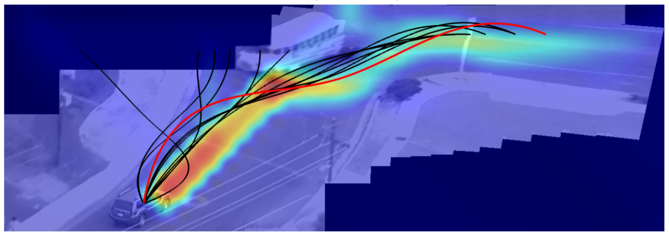
For our main experiments, we created a new dataset by downloading videos from Youtube of aerial car chase videos. In total we used 183 videos (from 48 different scenes) that lasted from 5 to 30 seconds. We used 139 videos from 37 scenes for training and 44 videos from 11 scenes for testing. For extracting discriminative mid-level elements, we used 1871 random frames from the training set in addition to 309 outdoor scenes from Flickr as the discovery dataset, and we use the MIT Indoor 67 [26] dataset for the negative dataset. We manually annotated the trajectories of the car in 44 test videos which were used as the ground-truth to evaluate the algorithm.

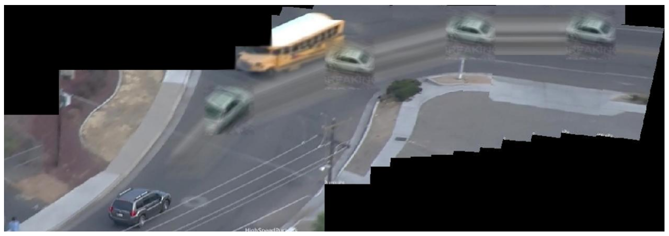
Qualitative: Figure 5 shows some qualitative results. Notice how the reward function captures that the road is a high reward region for the car patch. The bus in the top image and the cars in the bottom image are the areas of low reward. Similarly, sidewalk is also considered a low reward area. Given this reward function, our inference algorithm generated the possible paths, and the marginalization of these paths and some sampled paths are shown in the figure. Finally, notice the visual predictions in (d) and (h). Notice how the car turns in the top image avoiding the bus and how the car maneuvers between the two cars in the bottom image. Figure 6 shows some more qualitative examples of visual prediction generated by our algorithm.

Quantitative: In the first experiment, we compare the performance of our approach with the NN-baseline when no agents are given. In this way, we are measuring the ability of our method not only to find the correct active entity in the scene but also effectively predict its spatial activity. We use our algorithm to identify top three active agents in the scene and predict two paths per agent. We also allow the NN algorithm to generate six paths using top-6 matches. In order to compare path distance across test instances, all query panoramas are resized to a canonical size where the smallest dimension is 50 pixels. Table 1(top) shows the per-

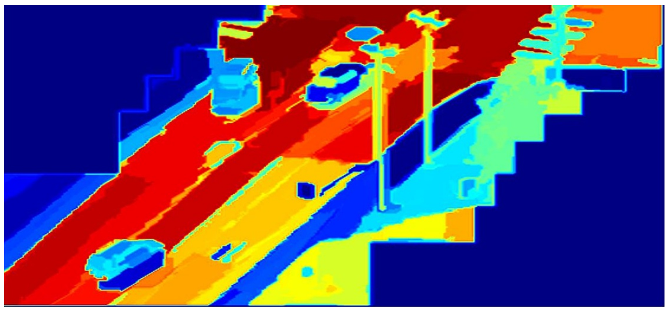
(a) Original Image

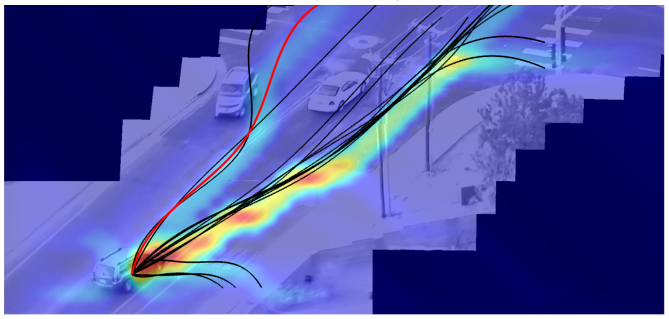

(b) Reward Function


(c) Distribution of Predicted Paths


(d) Predicted Path

(e) Original Image


(f) Reward Function


(g) Distribution of Predicted Paths

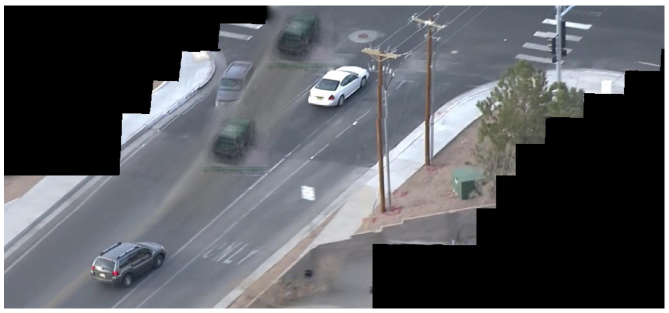

(h) Predicted Path

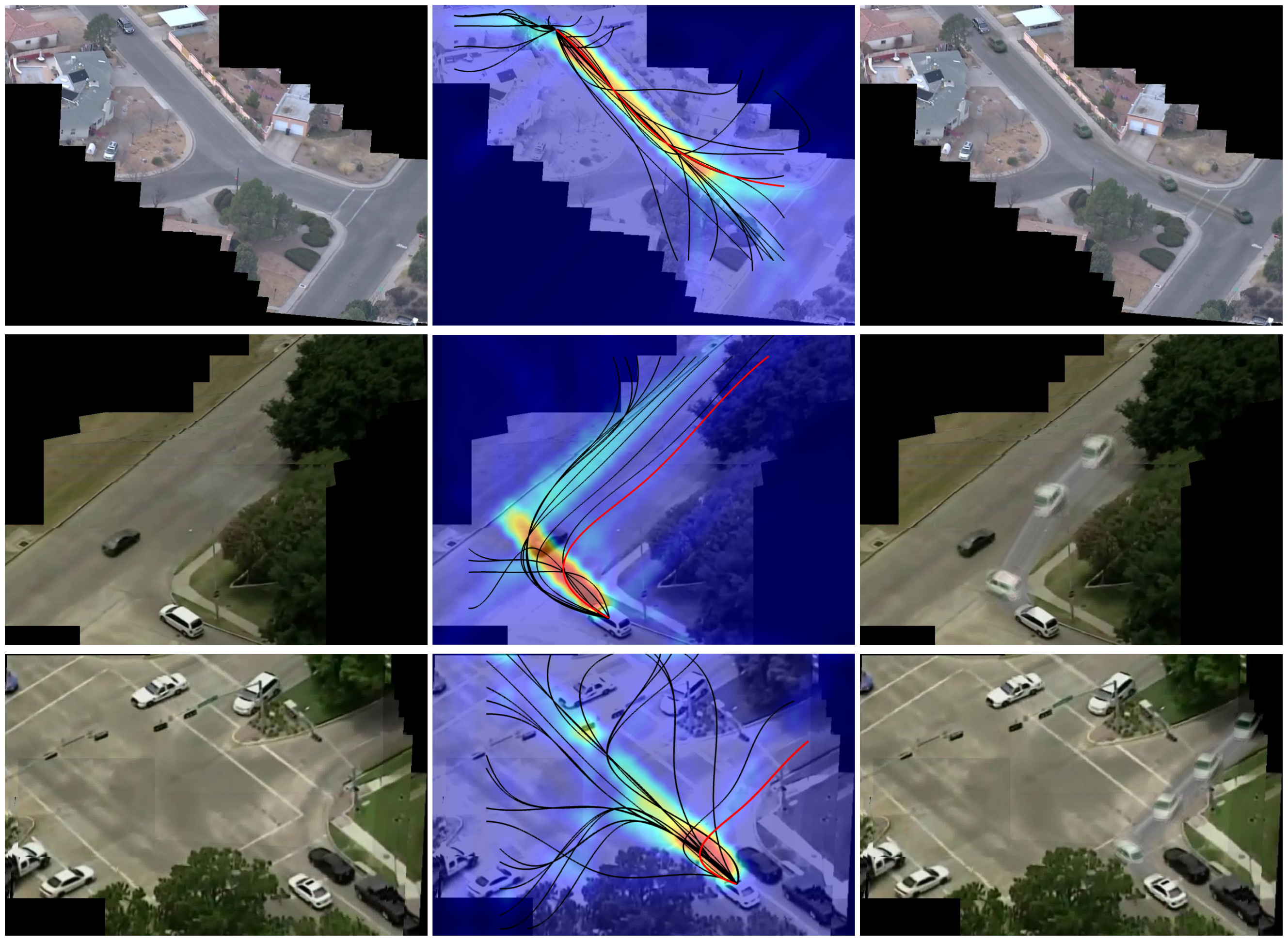
Figure 5. Some qualitative predictions of our approach. The upper right images (b,f) represents a reward function for a given car patch over the image. The lower left (c,g) shows a heatmap of possible locations where the agent can be and some of predicted trajectories, and the lower right (d,h) demonstrates the visualization of one such trajectory.

formance of the approach. Considering the median error, we show 35% improvement over the baseline. The ground truth car was in the method's top three active entities in 73% of cases.

In the second experiment, we ask a different question: given a manually selected agent, how well can we estimate the overall distribution of possible actions? In this case we can add [18] as a second baseline. The paths are ranked according to the expected reward. Since the agent is given, we compare the top paths in this case, not the entire distribution of paths as in [18]. Table 1(bottom) shows the that our approach again is superior to the NN-baseline and even shows improvement over [18]. [18] in this case appears to do poorly because of the underlying semantic features.

Experiment	NN + Sift-Flow	[18]	Ours
No Agent Given			
Mean Distance	22.34	-	**14.38**
Median Distance	16.68	-	**10.91**
Agent Given			
Mean	27.55	37.94	**21.55**
Median	23.77	30.23	**14.98**

Table 1. Mean and Median of the error of closest path over 44 videos in the car chase dataset for no agent given, and Mean and Median error of the top-ranked path for a given agent.



(a) Original Image (b) Prediction Heatmap (c) Predicted Path

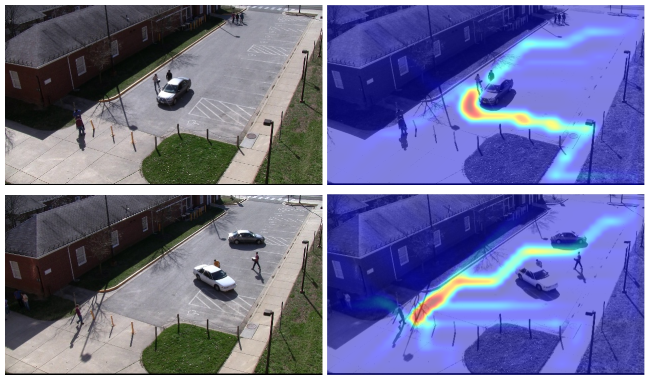
Figure 6. Qualitative predictions for our approach. The far left shows the original image, the center shows a heatmap of possible paths, and the right shows a visualization of one of those paths.

VIRAT	Ours	MDP (Our Reward)	[18]
Mean	**108.81**	128.48	147.32
Median	**77.79**	99.05	150.24

Table 2. Mean and median of the error of top predicted path.

3.2. VIRAT Dataset

For our second dataset, we chose a subset of the VIRAT dataset corresponding to a single scene A used in [18]. Since VIRAT data consists of only one scene, we used frames from the TUD-Brussels outdoor pedestrian dataset [30] to extract mid-level elements. We trained our model following the same experimental design in [18]. We use 3-fold cross validation and use a 15-step window for the MHD as in [18]. We use the full pixel grid (359x634). We find that our method is able to infer goals and paths on a better than [18]. To demonstrate that our reward function is meaningful, we also compare against MDP [18] but using our reward function instead of IOC. Table 2 shows the performance of the approach. For this experiment, since the agent is given, we compare the top trajectory predicted by each approach.



(a) Original Image (b) Prediction Heatmap

Figure 7. Qualitative predictions for our approach on the VIRAT dataset. The left shows the original image; the right shows a heatmap of possible paths.

4. Conclusion

In this paper we have presented a simple and effective framework for visual prediction on a static scene. Our prediction framework builds upon representative and discriminative mid-level elements and combines this visual representation with a decision theoretic framework. This representation allows us to train our framework in a completely unsupervised manner from a large collection of videos. However, more importantly, we can also predict how visual appearances will change in time and create a hallucination of the possible future. Empirically, we have shown that our unsupervised approach outperforms even supervised approaches on multiple datasets. It is important to note that this paper represents an initial step in the direction of general unsupervised prediction; we only predict events assuming the rest of the scene is static. Possible future work includes modeling the simultaneous behavior of multiple elements, predicting their possible coordinated action.

Acknowledgements: This work was supported in part by NSF grant IIS1227495.

References

[1] M. Bar. The proactive brain: memory for predictions. *Philosophical Transactions of the Royal Society*, 364(1521):1235–1243, 2009.

[2] M. Bar, editor. *Predictions in the Brain: Using Our Past to Generate a Future*. Oxford University Press, 2011.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[4] C. Doersch, A. Gupta, and A. A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013.

[5] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes Paris look like Paris? *ACM Transactions on Graphics (TOG)*, 2012.

[6] I. Endres, K. J. Shih, J. Jiaa, and D. Hoiem. Learning collections of part models for object recognition. In *CVPR*, 2013.

[7] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[8] D. Fouhey and C. L. Zitnick. Predicting object dynamics in scenes. In *CVPR*, 2014.

[9] H. Grabner, J. Gall, and L. Van Gool. What makes a chair a chair? In *CVPR*, 2011.

[10] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: image understanding using qualitative geometry and mechanics. In *ECCV*, 2010.

[11] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3D scene geometry to human workspace. In *CVPR*, 2011.

[12] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *CVPR*, 2009.

[13] J. Hawkins and S. Blakeslee. *On Intelligence*. Times Books, 2004.

[14] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

[15] Y. Jiang, M. Lim, and A. Saxena. Learning object arrangements in 3D scenes using human context. In *ICML*, 2012.

[16] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013.

[17] V. Karavasilis, C. Nikou, and A. Likas. Visual tracking by adaptive kalman filtering and mean shift. In *Artificial Intelligence: Theories, Models and Applications*. 2010.

[18] K. Kitani, B. Ziebart, D. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, 2012.

[19] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. In *RSS*, 2013.

[20] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *PAMI*, 2011.

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[22] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

[23] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR*, 2011.

[24] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001.

[25] M. Pei, Y. Jia, and S.-C. Zhu. Parsing video events with goal inference and intent prediction. In *ICCV*, 2011.

[26] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *CVPR*, 2009.

[27] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 81(1):2–23, 2009.

[28] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 2012.

[29] S. D. Tran and L. S. Davis. Event modeling and recognition using markov logic networks. In *ECCV*, 2008.

[30] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. In *CVPR*, 2009.

[31] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010.

[32] B. Yao and L. Fei-Fei. Action recognition with exemplar based 2.5D graph matching. In *ECCV*, 2012.

[33] J. Yuen and A. Torralba. A data-driven approach for event prediction. In *ECCV*, 2010.

[34] T. R. Zentall. Animals may not be stuck in time. *Learning and Motivation*, 36(2):208–225, 2005.

[35] B. D. Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, Carnegie Mellon University, 2010.