This WACV 2020 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

SmartOverlays: A Visual Saliency Driven Label Placement for Intelligent Human-Computer Interfaces

Srinidhi Hegde
TCS Research
sri.hegde@tcs.com

Jitender Maurya
TCS Research
jitender.maurya@tcs.com

Ramya Hebbalaguppe
TCS Research
ramya.hebbalaguppe@tcs.com

Aniruddha Kalkar
Walchand College of Engineering
aniruddha.k97@gmail.com

Figure 1: Representative labels generated by our proposed *SmartOverlays*: Row 1: (a) and (c) are the output of the YOLOv2 [28] object detector; (b) and (d) depict the outcome from our *SmartOverlays* algorithm. Note that labels are closer to the objects of interest, do not overlap each other, and do not occlude salient regions in a scene; (e) depicts the non overlapping label placement in a cluttered surveillance scene; Row 2: (f) and (g) show placement of labels for outdoor and indoor sports applications respectively; (h) depicts a comic style label placement feature. **Note**: *SmartOverlays* focuses on determining the label position in a scene while the label content is either provided by the user or derived from AI algorithms. Refer `https://ilab-ar.github.io/SmartOverlays/` for more details.

Abstract

In augmented reality (AR), the computer generated labels assist in understanding a scene by addition of contextual information. However, naive label placement often results in clutter and occlusion impairing the effectiveness of AR visualization. For label placement, the main objectives to be satisfied are, non-occlusion to the scene of interest, the proximity of labels to the object, and, temporally coherent labels in a video/live feed. We present a novel method for the placement of labels corresponding to objects of interest in a video/live feed that satisfies the aforementioned objectives. Our proposed framework, ***SmartOverlays****, first identifies the objects and generates corresponding labels using a YOLOv2 [28] in a video frame; at the same time, Saliency Attention Model (SAM) [7] learns eye fixation points that aid in predicting saliency maps; finally, computes Voronoi partitions of the video frame, choosing the centroids of objects as seed points, to place labels for satisfying the proximity constraints with the object of interest. In addition, our approach incorporates tracking the detected objects in a frame to facilitate temporal coherence between frames that enhances the readability of labels. We measure the effectiveness of SmartOverlays framework using three objective metrics: (a)* Label Occlusion over Saliency (LOS)*, (b) temporal jitter metric to quantify jitter in the label placement, (c) computation time for label placement.*

1. Introduction

Augmented Reality applications fuse contextual synthetic data with the real visual data to enrich perception and efficiency of the user performing a targeted task. Such contextual data superimposed on live video feed is referred as overlays. The overlays can take the forms of, but not limited to, text, audio, 3D objects and GPS coordinates. In this work, we propose *SmartOverlays*, a novel and a generic label placement framework that enhances the effectiveness of situated visualization across most of the computer vision and AR applications. *SmartOverlays* framework can be ported on device (smart-phones, tablets, and head mounts) for effective and unobtrusive label placement eventually. SmartOverlays can be a great addition to low field of view devices like head mounts, where any obstruction in an already constrained environment can greatly hamper the immersive experience.

Overlays could take a variety of geometric shapes/sizes as per application specifications. Thus we propose a solution for text overlay/label with unconstrained geometry. To demonstrate the generic nature of our framework and also to automate label generation as a part of our proposed work, we use an object detector to generate labels for detected objects. The labels generated through object detector is spatially independent, making label placement generic and challenging at the same time. The number of possible label positions grows exponentially with the number of items to be labelled, making the problem NP-Hard [2].

Previous works employed either sensor information such as GPS information, or used cues from the overlay features such as configuration of fixed anchor points [9, 31] as a criteria to place labels. In this work, we use visual saliency for effective label placement as it is an inherent feature of the image aiding in understanding the dynamics of the scene. Our work does not focus on generating meaningful caption which is an open problem in computer vision community, neither do we address label content based on scene understanding of an image. However, these aspects can be combined with the proposed *SmartOverlays* algorithm, to deliver promising artificial intelligence applications. Furthermore, *SmartOverlays* can be used generically as an extension to important problems in computer vision such as object detection and instance segmentation algorithms. The key contributions of our work are:

1. We propose *SmartOverlays*, a multi-label placement framework on video frames/live feed. This method comprises of a Saliency Attention Model for computing visual saliency, a real-time object detector such as YOLOv2, followed by our novel *label placement module*. The label placement module utilizes Voronoi partitioning to avoid label/lead-line overlap and adaptive color scheme to facilitate contrastive label color for dynamic backgrounds. Figure 1 shows outcome of the algorithm on diverse scenes.

2. We introduce object tracking based methods on detected objects to place labels in a temporally coherent fashion.

3. We introduce two metrics, *Label Occlusion over Saliency score (LOS)* and *Temporal Jitter* metric, for measuring the effectiveness of overlay placement spatially and temporally.

2. Related Works

Optimal placement of labels in AR applications enhance users' perception and thereby, reducing cognitive load by placing synthetic data on the real world view. However, when the labels that convey the contextual information are naively placed, they hinder visual perception. To overcome this, Bell et al. [3] propose effective view management techniques that address external labelling at run-time and also use empty spaces in screen to overlay labels. The system does not consider leader lines, it may suffer from crossing leader lines or leader lines occluding annotations. To improve the readability and intelligibility of the annotations in the users' view to ensure that the augmented virtual information show the desired contents intuitively and clearly, view management algorithms are widely used to overlay annotations, which automatically generate layouts of annotations for different applications [20]. Tatzgern et al [33] propose managing the placement of external labels in 3D object space instead of 2D space as 2D view does not encompass the temporal behaviour of a 3D scene and applies changes to the layout based on the 3D geometry of the label.

A broad categorisation of text label placement mainly: *Geometry-based approaches*, *Image-layout based on aesthetic rule*, and *adaptive overlays*. We briefly summarise these approaches below: *Geometric based approaches*: Christensen et al [6] demonstrated that point feature label placement is an NP-hard problem. Thus [6, 15] proposed simulated annealing and gradient descent as solutions. Wu et al.[38] propose a genetic technique combined with image analysis of a vector representation of a map. These complex algorithms require high computational power which can cause latency in execution. *Image aesthetics based approaches*: These approaches consider the visual aesthetics of computer interfaces as a strong determinant of users satisfaction [19]. They utilize a general design principle such as spatial layout rules, symmetry, balance among the element as well color schemes and harmony with the use-case of photobook generation. Owing to occlusions, dim light scenarios, scene variations in the live field of view, overlays have their own challenges. Several approaches discussed below work on comic content unlike our approach intended for real-time augmented reality applications. Kurlander et

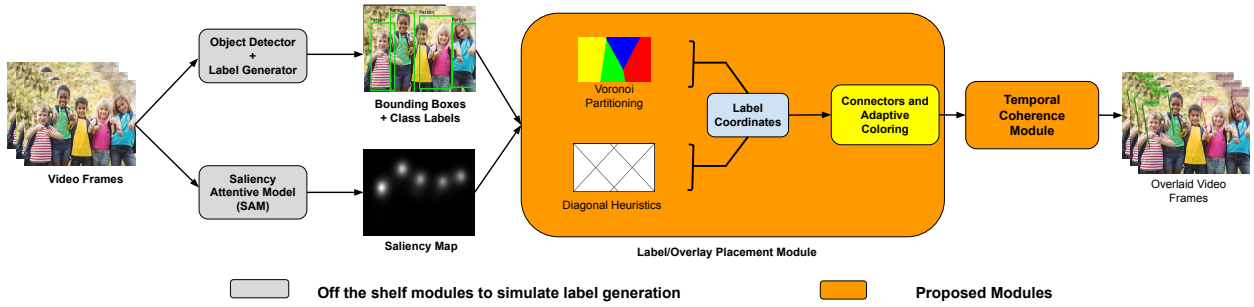

Figure 2: Overview of *SmartOverlays* label placement framework: We take video frames as input to our pipeline. This is passed to object detector with label generation module and Saliency Attention Module for saliency estimation. The object detector and label generator produce bounding box for all the detected objects along with their respective class labels, thus, creating object-label correspondences. SAM [7] computes the saliency maps for each of the video frames. In the Overlay Placement module, we compute the overlay position for each label in a frame based on the object-label correspondences, saliency maps and overlay placement objectives. Finally, the overlay location predictions are passed to Temporal Coherence Module to obtain a jitter free label placement.

al. [18] propose *Comic Chat* comics generation, including balloon construction and layout, the placement and orientation of comic characters while our method is designed for any generic image. User has to enter the text overlay content and then the corresponding baloon is generated based on comic style for a chatroom and then the layout is determined by the set of rules of comic art. However, our labelling is fully automated and is intended for a generic scene that includes comics (refer to Figure 1).

For greedy approach based algorithm, such as ours, it is very essential to have a good metric that decides the priority for choosing a particular location in the search space. Bell et al.[3] presents a priority constraint which they satisfy while placing the labels. We are inspired by [9, 27] which use visual saliency as a primary constraint in deciding the optimal position for labels. These works use method proposed by Achanta et al. [1] which is one of the first baselines in estimating visual saliency in images. But [27] focuses on single label placement unlike our generic multi label formulation. Traditional saliency prediction models were inspired by the biological features that captured low level features in image such as color, edges, texture and semantic abstractions of certain objects of interests such as human faces, people and text [10, 14, 35]. With the rapid advancements in specific deep learning architectures and large annotated datasets [11, 17, 25], data driven saliency prediction approaches have become popular in mainstream computer vision community. These deep learning based approaches outperform the traditional methods for predicting visual saliency in scenes. As per the MIT Saliency Benchmark [5], Liu et al., Cornia et al. and Kruthiventi et al.[7, 16, 22] are the leaders in the visual saliency estimation challenge. All of these approaches use deep neural network models for saliency prediction in images.

Previous methods are not generic for multi-label placement scenario. Our intent is to use visual saliency as metric for the label placement framework that does not use any external sensor information apart from the image/video themselves. Furthermore, to improve readability of overlays, we propose techniques to improve temporal coherence in the label placement. To the best of our knowledge, no techniques have been proposed for solving temporal jitter in label placements in videos capturing dynamic scenes.

3. Proposed Method

We formulate the label placement problem as follows: the input for our pipeline is an RGB video $V < f_1 , f_2 , ... , f_n >$ with frame sequence of length n and each frame of dimension $F_w \times F_h$. Our proposed model outputs an image coordinate $P = (x^i, y^i)$, for the i^{th} label in a frame, where $1 \leq x^i \leq F_w$ and $1 \leq y^i \leq F_h$. P represents the most suitable coordinate in the frame space for placing the i^{th} overlay. This point corresponds to the top left corner of the overlay or overlay bounding box if the overlay is non rectangular. *SmartOverlays* aims to handle overlays that can have unconstrained geometry and not just restricted to rectangular labels. In case of unconstrained overlays, we consider the tightest bounding box surrounding the overlay. Figure 2 shows an overview of our *SmartOverlay* algorithm. Subsections below provide a detailed overview of each of the modules of the pipeline.

3.1. Object Detection and Saliency Map Computation

Object detection algorithms can be used for evaluating the effectiveness of the label placement. For multiple label placement, it is necessary to have correspondences of labels with object of interest; the labels as placed as close to the relevant objects as possible. Hence, we detect and label objects in the scene using a YOLOv2[28].

We then use Saliency Attention Model (SAM) proposed by Cornia et al.[7], for computing saliency maps. Originally, SAM is used for computing saliency maps on static image datasets but, in this work, we use SAM on videos showing its effectiveness on a temporal data. SAM is trained on video frames along with its saliency ground truth in the form of both saliency density map and eye-fixation points. Visual information is given as input to the loss function, for learning to predict saliency maps. Cornia et al.[7] propose a loss function that is a combination of different scoring metrics for saliency maps, given by:

$$L(\hat{y}, y, y^{\texttt{Fix}}) = \alpha NSS(\hat{y}, y^{\texttt{Fix}}) + \beta CC(\hat{y}, y) + \gamma KL(\hat{y}, y) \tag{1}$$

where $\hat{y}$, y and $y^{\texttt{Fix}}$ are the predicted saliency maps, ground truth saliency maps and eye-fixation points respectively and α, β and γ are three scalars which balance the three loss functions. NSS, CC and KL are saliency evaluation metrics(NSS is the Normalized Scanpath Saliency, CC is the Linear Correlation Coefficient and the KL, Kullback-Leibler Divergence as defined in [13]).

These metrics help learn saliency maps for video frames as it is helpful in estimating similarity and dissimilarity between two saliency maps [13]. We quantify the pixel values of y at the eye fixation locations, y_i^{fix}, and normalize it with the variance of y.

3.2. Overlay/Label Placement

At this point, we have multiple objects and their corresponding labels in the scene. We now consider each object sequentially, in the decreasing order of saliency occlusion, for placing labels on the frame. Once we place an overlay we mark the region occupied by the overlay as highly salient. We see that this region is unsuitable for placing other overlays as it will increase occlusion with salient region. We need a metric to decide the saliency occlusion by label and object to decide the order in which we place the labels. We rank the object label pairs as per the label occlusion over saliency (LOS) score of bounding box of object of interest.

$$LOS(N, G) = \frac{\sum_{(x,y)\in N} G(x, y)}{|N|} \tag{2}$$

where N is the set of pixels (x, y) that is occluded by overlay and G is the ground truth saliency map. LOS score ranges from 0 to 1, where LOS score of 0 represents no occlusion with any salient region and 1 means complete overlap with highly salient region. As we place the overlays in the decreasing order of LOS score, the overlays will become a high saliency regions in the corresponding saliency map. Furthermore, we explicitly put a hard constraint for avoiding placement of labels on detected objects and previously placed labels.

Apart from the order of object-label pairs, there are four additional objectives that need to be considered while placing any overlay - (a) *The overlay should be placed to minimize the occlusion with highly salient regions*, (b) *The overlay must be as close to the corresponding object as possible*, (c) *Connector or leader lines, connecting objects and overlays, should not intersect with each other*, (d) *The overlay must satisfy diagonal heuristic and central bias*.

For minimising occlusion, we use LOS score as metric which is used to determine occlusion that might result with label placement. We compute LOS for each label over the saliency maps that are generated from the SAM discussed in the Section 3.1.

3.2.1 Proximity to Objects of Interest

We propose an approach based on a strict Voronoi partioning [36] (boundary excluded) of the image space. We choose the seed points of the Voronoi partitions as the centroids of the bounding boxes that are generated from the YOLOv2 detector. The Voronoi formulation ensures that the label's top left corner, P, is placed close to the corresponding objects due to the following property of the Voronoi partitions.

Theorem 3.1 *Voronoi partitions, generated using centroid of object bounding box, enforces the label to be closest to the corresponding object centroid than any other object centroid.*

We use greedy approach for searching the best location for overlay placement. For such approaches, reducing the search space is a big challenge. These Voronoi partitions, which are much smaller than the image space in size, are the new search space for finding the minimum LOS score. If we have multiple minima in the search space we select the one which has the least Euclidean distance from centroid of object bounding box. The number of seed points are equal to the number of detected objects. Also we can see that the size of Voronoi partitions decrease as the number of seed points increase. Thus with more objects in scene the size of Voronoi partition, that is the search space, decreases.

3.2.2 Avoiding Intersection of Connectors

For clarity of object-label pair correspondences, we use *connectors* or *leader lines* - a path of line segments that connects the center of the object bounding box to one of the corners of the corresponding label. We choose the label corner that (i) has the least Euclidean distance to the object bounding box centroid and, (ii) is strictly within the same Voronoi partition as that of the object bounding box centroid. In our work, we use only a single line segment, $C(s_1, s_2)$ where s_1 and s_2 are the endpoints of the line segment, as connectors. As per the widely accepted aesthetic rule, a connector should not intersect with any other connector [30].

Corollary 3.2 *The proposed connector placement method prevents intersection of a connector, $C(s_1, s_2)$, with other connectors.*

Proof Let p_1 and p_2 be object bounding box centroids, which are also the seed points for the respective Voronoi partitions, V_1 and V_2. Consider two distinct connectors, $C(p_1, p_1')$ between endpoints p_1 and p_1', and $C(p_2, p_2')$ between endpoints p_2 and p_2'. We know that Voronoi partitions are convex polygons [36]. From the definition of convexity, all the points s on the line segment $C(s_1, s_2)$ also lie in the corresponding Voronoi region, i.e., if p lies on the line segment $C(p_1, p_1')$, it also lies within V_1. Let us assume that $C(p_1, p_1')$ and $C(p_2, p_2')$ intersect at x, which implies that $x \in V_1 \cap V_2$. But we know for a strict Voronoi partition, $V_1 \cap V_2 = \emptyset$, and hence, the connectors are the same. However, this leads to a contradiction since $C(p_1, p_1')$ and $C(p_2, p_2')$ are distinct. Thus $C(p_1, p_1')$ and $C(p_2, p_2')$ never intersect.

3.2.3 Diagonal Heuristics and Central Bias

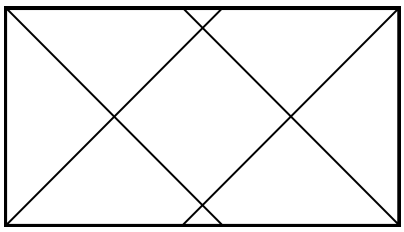

Figure 3: Modeling diagonal heuristic. The figure shows diagonal heuristic mask which consists of angle bisectors of each corner angle of the image.

Malu and Indurkhya [24] show that placing labels on the diagonal angle bisectors tends to increase the user experience in viewing. To improve user experience, we add diagonal heuristics to the saliency map (refer to Figure 3). Furthermore, studies have shown that eye-fixation points tend to cluster towards the centre of the scene [32, 34]. This human tendency is known as central bias. To model this phenomenon, while generating saliency maps, SAM [7] incorporates a combination of multiple learned priors to model the central bias.

Initially we tried black and white color scheme for a consistent coloring patterns used in [27]. For improving the legibility of text in labels, we use an adaptive color scheme where the text color adapts to the texture present in the label's background. Strong color contrast supports efficient text reading and to achieve contrast we use Maximum HSV Complement[8].

3.3. Temporal Coherence in Label Placement

To make a label readable, it should be placed such that the movement of label is coherent with the object throughout the video. However, real-time label placements can be jittery due to dynamic scene changes and drastic object movements in the input videos. Furthermore, since we formulate determining label position as an optimization problem for every frame independently, the placement of labels can vary from frame to frame resulting in jitter in labels placed in videos. To address this, we employ two schemes which are based on object tracking for maintaining temporal coherence in the label locations - *Fixed Label* and *Tracking by Optical Flow* methods.

In *Fixed Label* method, we assume the motion of moving objects to be small. Therefore, for a small time interval within Δk frames, we update the anchor points of the connectors without changing the label location. We track the locations of the corresponding objects by taking bounding box locations with the maximum IoU among all the bounding box pairs (with IoU greater than 0.75). If such bounding box pairs are not found for an object then we stop tracking for that object.

For *Tracking by Optical Flow* method, we extend the *Fixed Label* method. Along with centroid, here, we also update the location of centroid of object bounding box, C_i^n, along with label location, L_i^n, in frame f_i and $n \in \{1, .., N_O\}$ (N_O being the number of tracked objects). We update C_i^n to UC_i^n using exponential weighted average as follows

$$UC_i^n = (1 - \beta)C_i^n + \beta T_{(i-1)}^n, \tag{3}$$

with β being the weight. Exponential weighted average ensures a smooth flow of label resulting in minimum jitter. Figure 4 shows that we choose centroid and 8 points around centroid for robust object centroid tracking. We select multiple points as there might be an error in tracking only a few points. More than 9 points could be used but empirically we found that using 9 was enough to maintain symmetry and reduce computation. We track these 9 points in subsequent frames, using optical flow tracking [4]. In the corresponding object Voronoi partition a location, L_i, is computed with

minimum saliency. Now the updated label location, UL_i^n, is calculated by taking exponential weighted average again, as

$$UL_i^n = (1-\gamma)L_i^n + \gamma UL_{(i-1)}^n, \quad (4)$$

with γ being the weight. We give more weightage to the information obtained from previous frames and, thus, set β and γ to 0.9.

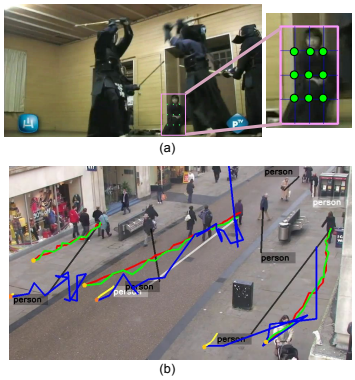

Figure 4: Tracking detected objects with optical flow to reduce temporal jitter. (a) We select 9 points in the object bounding box (centroid and 8 points around the center that are exactly midway between the centroid and bounding box edge midpoints and corners). (b) Visualization of robust tracking of detected objects with optical flow. **Legend**: **Green**-Detected centroid and **Red**-Corrected centroid, **Blue**-Jittery label location, **Yellow**-Corrected Label Location from method proposed in Section 3.3.

4. Experiments and Results

4.1. Experimental Setup

We performed all the experiments on a Linux platform system with an Intel Xeon CPU E5-2697 v3 2.60GHz with an Nvidia Tesla K40c GPU with 12GB RAM. The deep learning models for object detection and SAM were trained in PyTorch. For label generation and object detection, we use an YOLOv2 pretrained on COCO [21] dataset that has 80 classes. We resize input video frames to 608×608 before passing it to YOLOv2. For computing saliency maps, we use a SAM that is pretrained on SALICON dataset [12] containing eye fixation ground truths for images. We use the values of the hyperparameters $\alpha = -1 =$, $\beta = -2$ and $\gamma = 10$, in loss function as specified in [7].

4.2. Performance Evaluation

Figure 5 shows the results of *SmartOverlays* and it also compares with the naive label placement of object detector. We also compared the saliency prediction accuracy with different state-of-the-art and baseline methods. The saliency metrics considered in the study are - NSS, CC, AUC(Judd), sAUC and KL, as defined in [14]. Table 1 shows that the SAM (with Resnet) offer competitive results on all of the saliency evaluation metrics on SALICON [12] dataset.

Metrics	DF [16]	SGAN [26]	SAM-R [7]	SAM-V [7]
NSS(↑)	2.26	2.04	**2.34**	2.30
CC(↑)	**0.78**	0.73	**0.78**	0.77
AUC-J(↑)	0.87	0.86	**0.87**	**0.87**
sAUC(↑)	0.71	**0.72**	0.70	0.71
KL(↓)	**0.63**	1.07	1.27	1.13

Table 1: Comparison of different saliency prediction techniques evaluated on SALICON dataset.

On analysing the runtimes of different sub-modules on DIEM dataset[1], we found object detector, SAM and overlay placement module to be taking 1.17s, 2.53s and 0.51s respectively. On a whole the entire pipeline took 4.34s per frame for execution.

We also investigated a few user defined specifications and their effects in the framework. Firstly, the user can define the shape and size of the labels that we overlay. In cases where the size of a label is larger than that of the corresponding Voronoi partition the search space will be empty. As a result the placement module misses out on such labels and this helps in filtering labels in a cluttered environment (see Figure 6). Secondly, in the cases where the overlay text is long our system will create labels that will wrap around the user provided text.

4.3. Temporal Coherence

We compare the temporal coherence of our label placement algorithms (refer to Section 3.3) with naive label placement technique which involves skipping frames to make the label appear stable. We computed the label locations after skipping 20 frames for a 30 fps video based on the user evaluation. For evaluating the methods we define a metric, *temporal jitter* metric, M_j, as $M_j = \frac{d_l}{d_o+\epsilon}$ where d_o and d_l are the distance traveled by an object and its corresponding label throughout the duration of the video and ϵ is a small constant added to avoid division by zero. M_j measures jitter as more distance travelled by the label corresponds to more temporal jitter. M_j also captures the relative motion of label with respect to the its corresponding object due to the d_o in the denominator. Figure 8 shows the effectiveness of the proposed algorithms in terms of low M_j which evaluates to less jitter in videos with varying number of stationary and moving objects (also shown in Figure 8). Figure 8(a) depicts an important observation that jitter

[1]https://thediemproject.wordpress.com/

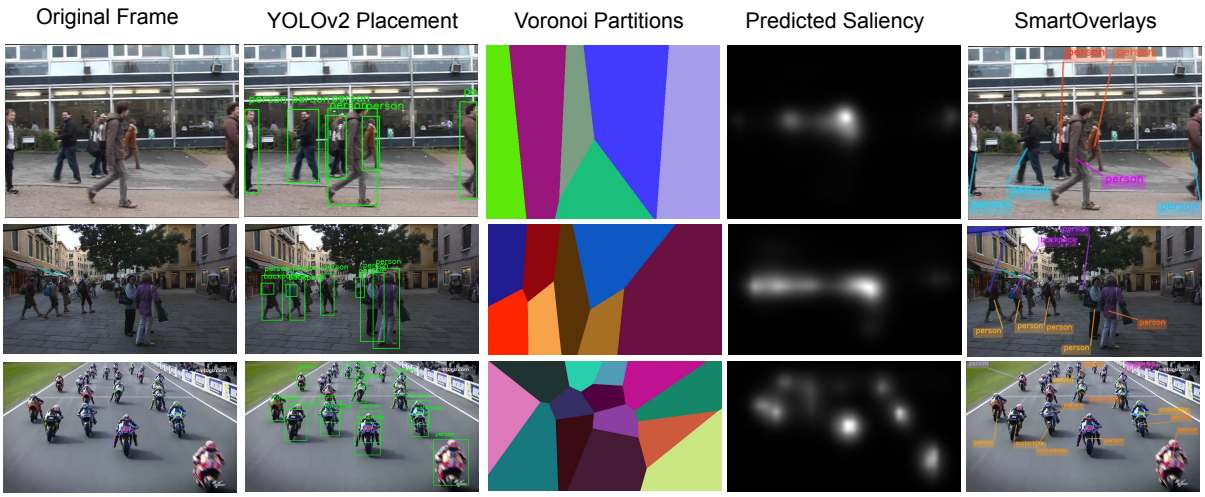

Figure 5: Comparing results of *SmartOverlays* framework. Column 2 corresponds to YOLOV2 output [28] placement of the column 1 images. YOLOv2 uses a fixed template based label placement method that places the label on top left corner of the object detector. (Other current state-of-the-art object detection templates such as the faster RCNN[29], SSD[23] also use top left corner to place labels on a scene.) We address labelling problems, associated with state-of-the-art object detectors, which include rendering labels out of image space (row 1), label occlusions (rows 2 and 3), label overlap (row 2) and so on using *SmartOverlays*. Columns 3 and 4 show the Voronoi partions and the predicted saliency maps respectively of the corresponding original frames in column 1. *SmartOverlays*, whose outputs are shown in Column 5, is applicable to generic object detection techniques as well.

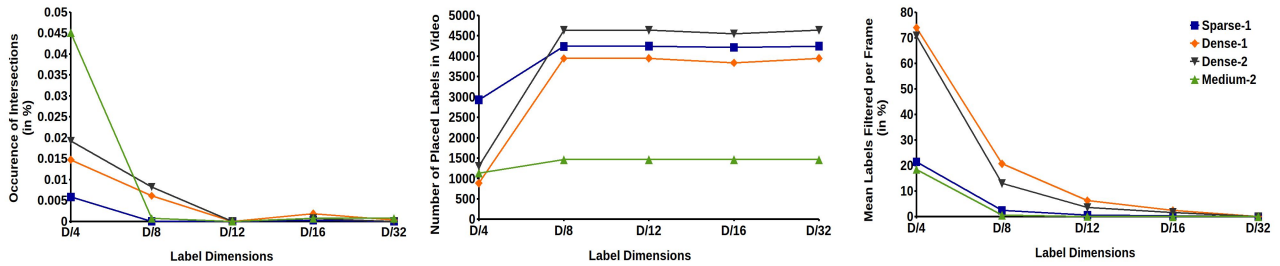

Figure 6: Characterization of label and leader lines intersections. (a) The plot shows the percentage of objects intersecting leader lines in the entire video. As the label dimensions decrease the label-leader line intersections decrease owing to the increase in inter label distances. (b) The plot shows the total number of labels placed in the entire video sequence. From the plot (c) we see the effect of label filtering for larger label dimensions due to unavailability of space for placement. As the label dimensions decrease the filtering effect reduces due to a reduction in the number of empty search spaces for optimal label locations. All the analysis for (a), (b) and (c) are performed on videos with different density of labels per frame and over different label dimensions (which are a function of image dimension, D).

in *Fixed Label* method is lesser than *Optical Flow Tracking* method. This could be because *Optical Flow Tracking* method takes into account the motion of the object which could be jittery due to the inaccuracies in object detection step. Figure 8(b) shows that lower number of stationary objects can also result in jittery label placement highlighting the effectiveness of tracking based methods for label placement in videos.

5. Discussion, Comparison and Future Works

Azuma et al.[2] compare and evaluate different algorithms for searching for optimal overlay positions in the image. They use various evaluation metrics such as counting

Figure 7: Failure cases. (a) Leader line over label (overflowing label problem). (b) Incorrect anchor point due to inaccuracies in object detectors.

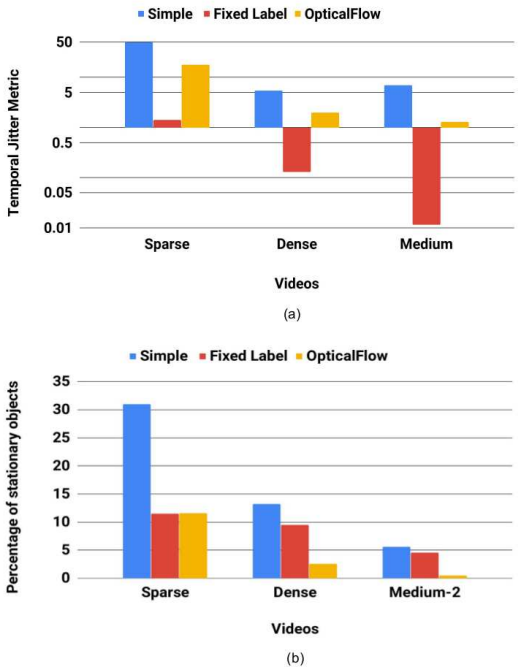

Figure 8: Comparison of temporal coherence in label placement by *SmartOverlays* without temporal coherence, fixed label method and optical flow tracking method. (a) Comparison of changes to temporal jitter metric on different types of videos (y-axis in log-scale), and (b) the number of stationary object vary for different techniques as we track the centroid of object bounding box for motion.

overlaps among labels, computation time, and number of labels moved. However, we do not assume an initial random placement of overlays that are moved to optimal locations by the algorithm. Instead, we predict the optimal label locations for every frame in a video. Thus, metrics analysing motion, like number of overlays moved, are unsuitable in our case. Metrics such as *number of overlaps* among labels is 0 for our algorithm as we mark every placed label as highly salient and our algorithm avoids placing labels on such regions. Hence, we propose a new evaluation metric, Label Occlusion over Saliency (LOS) which objectively quantifies the effectiveness of label placement. Although LOS metric might seem similar to Intersection Over Union (IOU), there are two major differences between the two. Firstly, IOU is used to evaluate fraction of overlap between two bounding boxes (or sets in general) but it ignores the content of the bounding box. We need a metric that considers the content, which is visual saliency in our case, of the region occupied by the label. Secondly, IOU is a function of 2 sets whereas LOS is a function of a set and a heatmap (saliency map in our case).

Regarding the computation time, We observe that the SAM and the object detector are bottlenecks to achieve real-time performance. Thus to make the framework function in real-time seamlessly, in future, we intend to explore computationally efficient methods for detecting objects and predicting saliency maps. We also intend to look at the usage of saliency computation models such as [37] which take temporal element of the data into consideration. We also look forward to create a compressed version of *smartOverlays* that could run on portable devices such as smartphones and HMDs. In our proposed system there are some constraints which is determined by the user. We discuss some of these constraints and their effects in the system. One of the constraints is the shape and size of the label that we overlay. In cases where the size of a label is larger than that of the corresponding Voronoi partition the search space will be empty. This will cause the placement module to miss out on certain labels. Another constraint in consideration is content of the overlay. If the overlay texts are long, then our system will create labels that will wrap around the entire text which is provided by the user. In such cases the text regions themselves may block the saliency objects (see Figure 7). But this would not hamper the placement of the other labels as the previously placed labels will be marked highly salient before we start to place the new labels.

6. Conclusion

We present *SmartOverlays*, a visual saliency driven multi-label placement algorithm that works on videos and real-scenes unlike the previous attempts that work solely on images. We have addressed the label placement objectives that render non-overlapping labels, avoids label occlusion over salient regions and prevents intersection of leader lines. Further, we facilitate temporal coherence in label placement and overlay labels contrastively on a dynamic background. In addition, we have introduced a new metric to evaluate label placement and provided valid proofs for the overlay placement objectives using Voronoi space partitioning. To the best of our knowledge, *smartOverlays* is the first deep learning based approach for label placement. Here, saliency is learnt to simulate the human visual attention behaviour while viewing a particular scene.

References

[1] R. Achanta, S. Hemami, F. Estrada, and S. Ssstrunk. Frequency-tuned Salient Region Detection. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, pages 1597 – 1604, 2009.

[2] R. Azuma and C. Furmanski. Evaluating label placement for augmented reality view management. In *Proceedings of the 2nd IEEE/ACM international Symposium on Mixed and Augmented Reality*, page 66. IEEE Computer Society, 2003.

[3] B. Bell, S. Feiner, and T. Höllerer. View management for virtual and augmented reality. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 101–110. ACM, 2001.

[4] J.-Y. Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001.

[5] Z. Bylinskii, T. Judd, A. Borji, L. Itti, F. Durand, A. Oliva, and A. Torralba. MIT saliency benchmark.

[6] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics (TOG)*, 14(3):203–232, 1995.

[7] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara. Predicting human eye fixations via an lstm-based saliency attentive model. *IEEE Transactions on Image Processing*, 27(10):5142–5154, Oct 2018.

[8] J. L. Gabbard, J. E. Swan, D. Hix, R. S. Schulman, J. Lucas, and D. Gupta. An empirical user-based study of text drawing styles and outdoor background textures for augmented reality. In *Virtual Reality, 2005. Proceedings. VR 2005. IEEE*, pages 11–18. IEEE, 2005.

[9] R. Grasset, T. Langlotz, D. Kalkofen, M. Tatzgern, and D. Schmalstieg. Image-driven view management for augmented reality browsers. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 177–186. IEEE, 2012.

[10] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *Advances in neural information processing systems*, pages 545–552, 2007.

[11] X. Huang, C. Shen, X. Boix, and Q. Zhao. Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 262–270, 2015.

[12] M. Jiang, S. Huang, J. Duan, and Q. Zhao. Salicon: Saliency in context. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1072–1080. IEEE, 2015.

[13] T. Judd, F. Durand, and A. Torralba. A benchmark of computational models of saliency to predict human fixations. 2012.

[14] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *Computer Vision, 2009 IEEE 12th international conference on*, pages 2106–2113. IEEE, 2009.

[15] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[16] S. S. Kruthiventi, K. Ayush, and R. V. Babu. Deepfix: A fully convolutional neural network for predicting human eye fixations. *IEEE Transactions on Image Processing*, 2017.

[17] M. Kümmerer, L. Theis, and M. Bethge. Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet. In *International Conference on Learning Representations (ICLR 2015)*, 2015.

[18] D. Kurlander, T. Skelly, and D. Salesin. Comic chat. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 225–236. ACM, 1996.

[19] T. Lavie and N. Tractinsky. Assessing dimensions of perceived visual aesthetics of web sites. *International journal of human-computer studies*, 60(3):269–298, 2004.

[20] G. Li, Y. Liu, Y. Wang, and Z. Xu. Evaluation of labelling layout method for image-driven view management in augmented reality. In *Proceedings of the 29th Australian Conference on Computer-Human Interaction*, pages 266–274. ACM, 2017.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[22] N. Liu and J. Han. A deep spatial contextual long-term recurrent convolutional network for saliency detection. *arXiv preprint arXiv:1610.01708*, 2016.

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[24] G. Malu and B. Indurkhya. An approach to optimal text placement on images. In D. Harris, editor, *Engineering Psychology and Cognitive Ergonomics. Understanding Human Cognition*, pages 68–74, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[25] P. K. Mital, T. J. Smith, R. L. Hill, and J. M. Henderson. Clustering of gaze during dynamic scene viewing is predicted by motion. *Cognitive Computation*, 3(1):5–24, 2011.

[26] J. Pan, E. Sayrol, X. Giro-i Nieto, C. C. Ferrer, J. Torres, K. McGuinness, and N. E. OConnor. Salgan: Visual saliency prediction with adversarial networks. In *CVPR Scene Understanding Workshop (SUNw)*, 2017.

[27] N. Rakholia, S. Hegde, and R. Hebbalaguppe. Where to place: A real-time visual saliency based label placement for augmented reality applications. In *Image Processing (ICIP), 2018 IEEE International Conference on*. IEEE, 2018.

[28] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525. IEEE, 2017.

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[30] D. Schmalstieg and T. Hollerer. *Augmented reality: principles and practice*. Addison-Wesley Professional, 2016.

[31] T. Stein and X. Décoret. Dynamic label placement for improved interactive exploration. In *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, pages 15–21. ACM, 2008.

[32] B. W. Tatler. The central fixation bias in scene viewing: Selecting an optimal viewing position independently of motor biases and image feature distributions. *Journal of vision*, 7(14):4–4, 2007.

[33] M. Tatzgern, D. Kalkofen, R. Grasset, and D. Schmalstieg. Hedgehog labeling: View management techniques for external labels in 3d space. In *Virtual Reality (VR), 2014 iEEE*, pages 27–32. IEEE, 2014.

[34] P.-H. Tseng, R. Carmi, I. G. Cameron, D. P. Munoz, and L. Itti. Quantifying center bias of observers in free viewing of dynamic natural scenes. *Journal of vision*, 9(7):4–4, 2009.

[35] W.-C. Tu, S. He, Q. Yang, and S.-Y. Chien. Real-time salient object detection with a minimum spanning tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[36] G. Voronoï. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs. *Journal für die reine und angewandte Mathematik*, 134:198–287, 1908.

[37] W. Wang, J. Shen, F. Guo, M.-M. Cheng, and A. Borji. Revisiting video saliency: A large-scale benchmark and a new model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4894–4903, 2018.

[38] H.-Y. Wu, S. Takahashi, C.-C. Lin, and H.-C. Yen. A zone-based approach for placing annotation labels on metro maps. In *International Symposium on Smart Graphics*, pages 91–102. Springer, 2011.