CoMaL: Good Features to Match on Object Boundaries

Swarna K Ravindran* and Anurag Mittal
Indian Institute of Technology Madras
Chennai INDIA - 600036
swarnakr@cs.duke.edu, amittal@cse.iitm.ac.in

Abstract

Traditional Feature Detectors and Trackers use information aggregation in 2D patches to detect and match discriminative patches. However, this information does not remain the same at object boundaries when there is object motion against a significantly varying background. In this paper, we propose a new approach for feature detection, tracking and re-detection that gives significantly improved results at the object boundaries. We utilize level lines *or* iso-intensity curves *that often remain stable and can be reliably detected even at the object boundaries, which they often trace. Stable portions of long level lines are detected and points of high curvature are detected on such curves for corner detection. Further, this level line is used to separate the portions belonging to the two objects, which is then used for robust matching of such points. While such CoMaL (Corners on Maximally-stable Level Line Segments) points were found to be much more reliable at the object boundary regions, they perform comparably at the interior regions as well. This is illustrated in exhaustive experiments on real-world datasets.*

1. Introduction

Feature points in an image are points that have a distinctive image structure around them and have been used in several applications such as point tracking [25, 15, 31], Visual Odometry (for Automotive Applications, for instance) [32, 11, 43], Optical Flow [22, 2], Stereo [17, 13, 18, 41], Structure from Motion (SfM) from video [41, 49], and Simultaneous Localization and Mapping (SLAM) [21] among others. Most of the popular feature detectors (Harris [16], Shi and Tomasi [42] SURF [4] and Hessian [28]) utilize the whole information in a patch surrounding the point to find feature points. For instance, the Harris detects points that have significant aggregated gradients in orthogonal directions in a surrounding patch. Many recent detectors

Figure 1. A car moving against a varying background. Nearly half of the patch centered on a Harris corner at the object boundary is part of the background.

(AGAST [26], FAST [38] and FAST-ER [39]) use intensity comparisons in different directions and use machine learning techniques to significantly speed up the computation. These features have been matched using a variety of techniques. The simple Sum-of-Squared-Distance(SSD), with some local optimization [10, 23, 2, 25], is still typically the method of choice when there are only small changes in the illumination or viewpoint (for e.g. point tracking, flow and stereo applications) while more complicated descriptors such as SIFT [24] have been utilized where there are more such variations. Many modern variants output a binary descriptor for extremely efficient matching (BRIEF [6], ORB [40], Daisy [45], FREAK [1] and NSD [5]).

While these Feature Detection and Matching approaches perform reasonably in the interior of objects, they perform quite poorly on the object boundaries [49]. This can be attributed to two reasons. First, the detectors rely on fixed (scalable) image patches which may straddle object boundaries and depth discontinuities and a change in these can lead to a change in the detected object. Second, even if a boundary point is detected at the same location w.r.t. one of the objects, matching is very difficult as the part in the patch belonging to the other object changes. (Fig. 1).

In this paper, we try to address these problems by proposing an approach for Feature Detection and Matching that can detect points accurately even in the presence of a changing background. At the same time, the support region is automatically segmented into two parts which often correspond to the regions belonging to the two objects. This enables independent matching of these two parts and by considering only the matching part, the point can be matched

*The author is currently at Duke University accurately even in the presence of a changing background.

We utilize level lines (curves connecting points with the same intensity) for this purpose by noting that the boundaries of objects are typically traced by such level lines, which often move with the object (Fig 2). By detecting turns/corners on level lines that do not change much with intensity variations (stability property), discriminative points can be found. We refer to such points as Corners on Maximally-stable Level Line Segments (CoMaL). Furthermore, this level line itself typically separates the two objects in the case of object boundaries and thus, we match the portions on either side of this curve separately and take the higher score of the two. This makes the matching method robust even in the boundary regions.

Several detectors have used level lines in the past [7], the most popular among them being the Maximally Stable Extremal Regions(MSER) detector [27]. MSERs are stable, *closed* level lines that were shown to return high Repeatability and Matching scores in image matching experiments [29]. They have also been used in hand and object tracking [9], where the object is typically homogeneous and has little interior texture, causing the other detectors to underperform. However, since MSER considers only *small* closed level lines and throws away the information in longer level lines in order to preserve the locality of a feature, it typically returns very few points and is not a popular choice for many other detection and matching applications where one needs to obtain a sufficient number of points (Fig. 2(a)). In this work, we detect corners along long level lines (Fig. 2(b)), which in fact are more stable than small level lines in many cases such as blur [34, 36]. Such long level lines have been used in the past by some detectors such as LAF [34] and SAF [36], that build affine-invariant detectors using some key tangent points on the curve. However, they rely on very few particular key points on the curves to compute the features, which makes them quite noisy. Also, their affine-invariant property makes them less suitable for the basic task of feature detection, where such methods underperform [29]. Edges, which are closely related to level lines, have been used to detect corners [47]. However, edge-based feature detection is prone to a higher error as edges can often be fragmented. In this paper, we restrict ourselves to the problem of basic feature detection (without any scale or affine invariance) that also allows us to use much more robust measures for corner detection on such level lines.

Our detection and matching technique gave superior results compared to other state-of-the-art algorithms on the KITTI Vehicle dataset [14] with real-world sequences, with significantly improved results on the object boundaries. Although our method is applicable in many scenarios, results are illustrated for two applications from this dataset: Point Tracking and Optical Flow.

(a) (b)

Figure 2. (a) A long level line that forms the boundary of an object. The information present along such level lines is discarded by MSER. (b) A few corners (marked in red) detected by us on locally stable portions of the level lines .

1.1. Related Work on Handling Boundary Regions

Several algorithms have been tried to address varying backgrounds in boundary regions. The dominant edge is used to separate the two regions at the object boundary for the problem of Object Recognition in [30]. In object tracking, SegTrack [3], Chen et al. [8] and Oron et al. [35] iteratively build probabilistic appearance models for the foreground and background in order to separate them for superior object tracking.In stereo, Kanade and Okutomi [20] and DAISY (Tola et al.) [45] adaptively determine the window/mask to use while matching each point. Almost all of the above approaches for different problems utilize smoothness constraints in a large region in an iterative manner to disambiguate the possible matches at the object boundaries. Thus, they have limitation when the object boundaries dominate the object appearance (for e.g. thin objects). Furthermore, they need a good initialization. Our algorithm can match points without such smoothness restrictions and on objects having very little internal texture and can also be used to provide some good matches as initializers for these algorithms.

2. Corners on Maximally Stable Level Lines

We define our corners on level lines, which are lines connecting points having the same intensity. If the intensity variation across the image is smooth or has been sufficiently smoothed by a smoothing operation, then such level lines form smooth curves in an image with nearby level lines having close intensities (Fig. 3). Thus, by varying the intensity of the level line, one can move these curves in space. Portions on these level lines that do not move much when the intensity is varied are portions with good perpendicular gradients on the level line and are called *stable* in this work. When additionally, such level lines turn, then such corner points can be discriminated from other points in the neighborhood and detected as feature points. We first consider the stability of a level line segment extending on either side

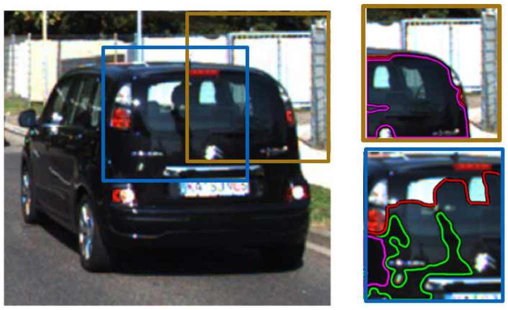

Figure 3. Stable and Unstable level lines in brown (top right) and blue (bottom right) boxes respectively. The Green level line $\mathcal{LL}$ is tested by considering $\mathcal{LL}_{\mathbb{N}}(\mathcal{LL}, \delta)$ and $\mathcal{LL}_{\mathbb{N}}(\mathcal{LL}, -\delta)$ level lines in red and purple respectively. Note that the lines are very close in the brown box due to which they mostly overlap in the illustration.

of a given candidate point p on a given level line, the extent of the segment being determined by the scale at which points are to being detected.

2.1. Stability of a Level Line Segment

The first condition we desire for a corner point is that it should lie in a region of high gradients. A level line that has a high gradient on it is thus desirable. Although we can compute this directly, a more robust approach is to consider neighboring level lines and compute the distance between these. We define a level line that is a neighbor of $\mathcal{LL}(I)$ and is detected at an intensity of $I + \delta$ as $\mathcal{LL}_{\mathbb{N}}(\mathcal{LL}(I), \delta)$. A high value of the gradient on $\mathcal{LL}$ (which is always perpendicular to it) is characterized by close $\mathcal{LL}_{\mathbb{N}}$'s for small δ's. The stability of a level line $\rho(\mathcal{LL}(I))$ can then be defined by considering the distance between $\mathcal{LL}_{\mathbb{N}}(\mathcal{LL}(I), +\delta)$ and $\mathcal{LL}_{\mathbb{N}}(\mathcal{LL}(I), -\delta)$ for some given small value δ. This is illustrated in Figure 3.

The Distance Measure The distance between neighboring level lines may be calculated using a variety of measures. A straightforward measure is to establish explicit correspondences between the two curves by considering the nearest points and then summing the distances between the corresponding points. While this can be speeded up using the Distance Transform, the corresponding points may not be unique and may not cover all the points, especially in the case of concave and convex curves, leading to noisy results.

Stable Affine Frames (SAF) [36] uses the maximum of the distances between three particular pairs of corresponding points instead of all the corresponding points. These are two adjacent bi-tangent points on a level line and a central high-curvature point. However, the detection of these points, especially the bi-tangent, is known to be noisy. Further, relying on just 3 points is not very robust.

In this work, we use the area between the two level lines $\mathcal{LL}_{\mathbb{N}}(\mathcal{LL}(I), +\delta)$ and $\mathcal{LL}_{\mathbb{N}}(\mathcal{LL}(I), -\delta)$, normalized by the length of the level line, as the distance measure. This measure, based on the number of points between the two curves, is more robust to noise in the curves. It is inspired by MSER [27], which has been shown to be a robust detector in many evaluations [29].

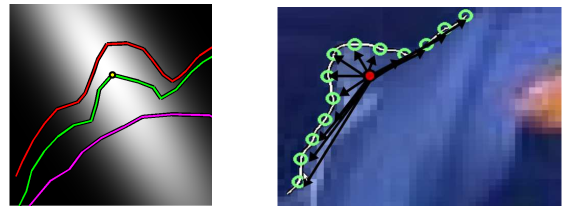

Figure 4. (a) The Gaussian weight centered on point p (yellow) on a level line $\mathcal{LL}$. (b) The vectors connecting the points (in green) on the level line segment to their mean $(\bar{x}, \bar{y})$ (in red). The distribution of these vectors is used to determine the cornerness of this level line segment.

Weighting the Points in the Patch We make a modification to this measure in order to make it more robust. Essentially, the points closer to the *candidate corner* p are more important than points far from p. To achieve this effect, while computing the area between the curves and the segment length of the level line, the points in the image patch centered at the point p are weighed using a 2D Gaussian $G_I(p, \sigma_I^{low}, \sigma_I^{high}, \theta)$ centered at the candidate corner point p. The Gaussian is aligned along the direction θ of the tangent to the level line at the point p such that a high sigma σ_I^{high} is used in the direction perpendicular to θ and a low sigma σ_I^{low} is used in the direction of the tangent (Fig 4(a)). These σ's are multiplied by the scale s at which the point is to be detected. G_I is truncated at 2 σ_I for efficiency purposes.

Given such a weighting for the points in the surrounding patch, the weighted length len_w is computed for the level line segment $\mathcal{LL}(I, p, s)$ at intensity I centered along the level line at the point p at scale s. Further, the weighted area ΔA_w is calculated from the weighted points between $\mathcal{LL}_{\mathbb{N}}(\mathcal{LL}(I, p, s), \delta)$ and $\mathcal{LL}_{\mathbb{N}}(\mathcal{LL}(I, p, s), -\delta)$. Then, the stability ρ of the level line segment $\mathcal{LL}(I, p, s)$ using the variation parameter δ is defined as:

$$\frac{1}{\rho(\mathcal{LL}(I, p, s), \delta)} = \frac{\Delta A_w(\mathcal{LL}(I, p, s))}{len_w(\mathcal{LL}(I, p, s))} \quad (1)$$

Essentially, $1/\rho$ measures the average *weighted* motion of a point on $\mathcal{LL}(I, p, s)$ when the intensity I is varied. This stability measure is computationally simple, symmetric and more stable compared to many other alternatives since it relies on the characteristics of the entire curve and not just a few points on it which can be noisy.

Given the stability of the level line segments, a non-maximal suppression is finally done by picking only those segments $\mathcal{LL}(I, p, s)$ that have a higher ρ than their immediate neighbors: $\mathcal{LL}_{\mathcal{N}}(\mathcal{LL}(I, p, s), 1)$ and $\mathcal{LL}_{\mathcal{N}}(\mathcal{LL}(I, p, s), -1)$. Such maximally stable level line segments are denoted as $\mathcal{MLL}(I, p, s)$ in this work.

Such $\mathcal{MLL}$'s are distinctive in their neighborhoods from neighboring level lines. However, points on such level lines are distinctive from each other only where the curve turns. Such turns or corners on such level lines are detected using the following approach:

2.2. Corners on $\mathcal{MLL}$'s

The turn points or corners on maximally stable level line segments $\mathcal{MLL}$ are distinctive and can be differentiated from other points in the neighborhood. Thus, such points will be detected as corner points in this work.

A popular and straightforward approach to find corners on curves is by using the curvature [36, 7] which measures the rate of change in the curve direction at any given point of the curve (second derivative). Local maxima of such curvature along the curve can be used as corner points. However, this measure can be somewhat noisy due to the use of the second derivative. To make it less sensitive to noise, one must use a fairly high precision which increases the running time of the algorithm. We use a more robust and computationally much more efficient approach as it does not require a high precision while computation.

The distribution of points on the curve centered at the candidate corner point p is determined (Fig. 4(b)). The Covariance matrix Σ_s of such points at scale s is:

$$\Sigma_s = G(p, \sigma_s) \otimes \begin{bmatrix} (x-\bar{x})^2 & (x-\bar{x})(y-\bar{y}) \\ (x-\bar{x})(y-\bar{y}) & (y-\bar{y})^2 \end{bmatrix} \quad (2)$$

where $\bar{x}$ and $\bar{y}$ are the x and y means of points and a 1D Gaussian $G(p, \sigma_s)$ is used to weigh the points on the level line such that σ_s is proportional to the scale s.

The eigenvalues of Σ_s reflect the distribution of the points along two principal orthogonal directions and high values of both indicate a corner. Shi and Tomasi [42] and Tsai et al. [46] use the minimum of the two eigenvalues as a measure for cornerness, arguing that it better represents the corner. However, computing the eigenvalues explicitly is slow, due to which the original Harris Corner detector [16] works on the second moment matrix of the image gradients directly, defining cornerness as: $det(\Sigma_s) - k \cdot trace(\Sigma_s)^2$. Forstner et al [12] and Lowe et al. [24] use:

$$\kappa(s) = det(\Sigma_s)/trace(\Sigma_s)^2 = \frac{(\lambda_1 \cdot \lambda_2)}{(\lambda_1 + \lambda_2)^2} \quad (3)$$

Due to the normalization, it is scale invariant and since eigenvalues themselves are rotation invariant [16], this measure is also rotation invariant. This measure was found to be suitable for our purposes and can also be computed fast and is thus used in this work.

A threshold is applied on the cornerness $\kappa(s)$ in order to find points of high cornerness at scale s. Furthermore, a non-maximal suppression is employed along the $\mathcal{MLL}$'s to yield corners that are well localized along the level line.

Finally, corner points are defined as:

Definition: A point p is a feature point at scale s if $\mathcal{LL}(I(p), p, s)$ is *maximally stable* according to the stability measure ρ and the cornerness $\kappa(s)$ of p is the local maxima along $\mathcal{LL}(I(p))$ at scale s.

The important point to note here is that all the tests above have to be done by centering the curve and the patch at the point p. Calculation of such stability for every point on every level line is prohibitively slow. We next discuss an iterative approach to search for such corner points efficiently.

3. Algorithm: Iterative Feature Detection

In order to perform this search efficiently, we note that the maximally stable segments do not shift much when the scale is varied. This allows us to run an initialization step at a slightly higher scale (we use 2 times the scale of the final detection) in overlapping blocks for an initial estimate of the points. Furthermore, no weighting is used in this step which allows it to be fast. Each of such initial corners is passed through an iterative refinement step where the full constraints of patch centering at the detection point and point weighting are applied for stability and cornerness computations.

3.1. Initialization

The first step in the Initialization is to divide the image into overlapping blocks of size $2Bs \times 2Bs$, where B is a multiplying factor specifying the support region to be used for corner detection and s is the scale at which we want to detect the final corners. Maximally stable level line segments at scale $2s$, $\mathcal{MLL}(2s)$, are detected in each image block using a modified-MSER algorithm described next. No weight scaling as described in the previous section is applied. On such $\mathcal{MLL}'s$, an initial set of corners C_s^{init} is determined using Eq 3. The cornerness threshold is also lowered a bit compared to the final detection threshold in order to not miss any final corners.

Efficient $\mathcal{MLL}$ Detection using a Modified MSER Algorithm: We modify the MSER algorithm to efficiently detect maximally stable level line segments since the MSER detector efficiently maintains the set of level lines and the area of the associated regions by the union-find algorithm. We replace the MSER's stability formulation with our formulation in Eq. 1, which involves a division by the *weighted length* of the level curve $\mathcal{LL}$, which is an open curve, rather than a division by the *area* of the closed level line, which may not be the best thing to do in these blocks which often truncate such level lines and extremal regions. This modified MSER algorithm is run on each image *block* to get an initial set of stable level line segments $\mathcal{MLL}$. Note that no point weighting as proposed in Section 2.1 is applied here.

Approximations in Corner Computation Corner computation is run on such detected $\mathcal{MLL}$'s from each block. The time consuming step is the convolution with a Gaussian weight filter (Eq. 2) for the point contributions for each test point. This step can be made efficient by using the Central Limit Theorem to replace the Gaussian with an average filter that can be applied multiple times to approximate the effect of a Gaussian (The average filter is run 3 times for the results in this work). The averaging operation is extremely fast due to the applicability of Dynamic Programming. The idea is similar in spirit to the approximate 2D Gaussians implemented in SURF [4]. Such an approximation is possible in our approach since our cornerness measure is quite robust to weight errors compared to other measures such as the curvature which require more precise computations. Note that there is no need to run this step at scale $2s$ and we run this corner detection step at scale s itself.

Running Time A maximum init window stride of $2Bs/2$ ensures that each of the points is captured in at least one of the init windows. Since the MSER is a linear-time algorithm [33], the computation of the $\mathcal{MLL}$'s takes around 4 times the amount of time the MSER algorithm would take on the entire image. The computation of the corners on such $\mathcal{MLL}$'s is again linear in the number of pixels on the level lines, which is actually much lower than the number of pixels in the image and is thus extremely fast.

3.2. Iterative Point Refinement

Given an initial set of approximate corner locations obtained from the initialization stage, we run an iterative refinement algorithm for each point so that in the end, the level line is locally maximally stable with the detected point p as its center, and the stability measure is computed with the appropriate point weighting as specified in Section 2.1.

The first step in the refinement is to recompute the maximal level line $\mathcal{MLL}$ when the patch is centered at the current estimate of p. A block of size of $Bs \times Bs$ is used as the support region for point detection. The modified-MSER algorithm as described in the previous section is used. Among the many maximal level lines that may be found in this block, the one that is closest in terms of shape and distance to the current one is taken as the new $\mathcal{MLL}$. Appropriate Gaussian weighting of the points is used, which also ensures that blocking causes minimal errors as the points near the block boundaries will have very low weights. Corners are re-detected on the new $\mathcal{MLL}$ at scale s and the one closest to the previous one is taken as the updated corner point.

This process is repeated till the point stops moving. At this stage, the point p satisfies both the conditions for our feature point and is output as a corner point at scale s.

Typically, the initial level line remains fixed or moves to only a nearby level line during the iterations and the maximum number of iterations was found to be only around 3 or 4 in our experiments. Each iteration is an order $(Bs)^2$ operation where most of the time is taken by the linear-time MSER algorithm running on the block of size $Bs \times Bs$. It is also important to note that the algorithm is trivially parallelized, for example, by the use of GPUs. The whole iterative procedure is illustrated in Fig. 5.

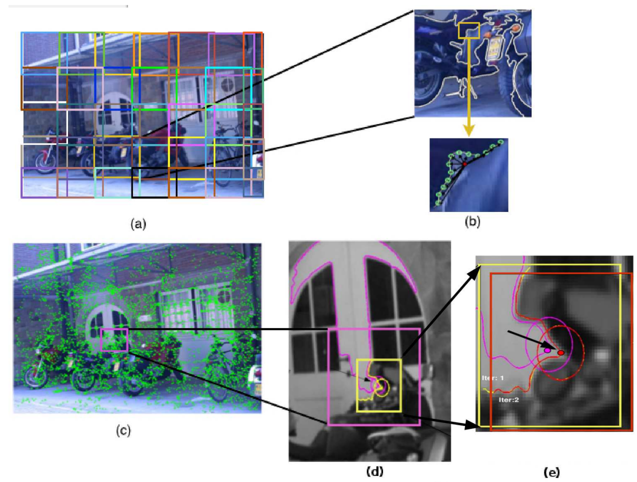

Figure 5. (a) An Image divided into overlapping blocks of size $2Bs$. Different blocks are shown in different colors for clarity purposes. (b) A sample $\mathcal{MLL}$ in one sample block (top) and a corner found on it (bottom). (c) The set of initial corners detected. (d & e) The iterative procedure for point refinement. The $\mathcal{MLL}$ and the initial point (pink) detected in the initial stage with a block window of size $2Bs$ are used to center a block (yellow) of size Bs in the first step of the iteration. This point moves to the red point. When the window is now centered at this (red) point, it remains the same and is thus detected as the final corner.

4. Point Matching

While one can use simple strategies, such as the SSD for point tracking or descriptors such as SIFT to handle more variations, they don't work very well for the points on the boundary of two objects as the surrounding patch may contain regions from two relatively moving objects (an object

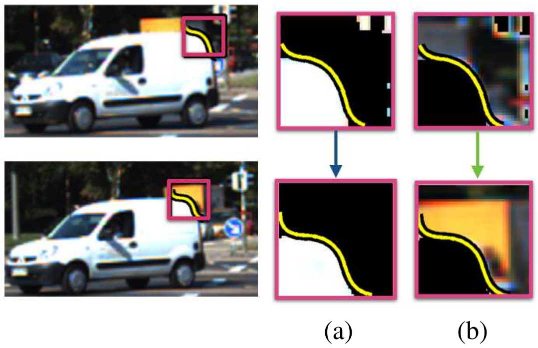

Figure 6. (a) +ve region (side with higher intensities) and (b) -ve region (side with lower intensities) separated by the level line shown in yellow. They are matched separately for better matching.

vs. its background). To handle such cases, we propose to use the maximal level line on which the point was detected as a separation boundary between two regions of the patch and match the two regions separately (Fig. 6(b)). The basic idea is that the boundary between two objects is typically traced by a maximal level line which moves along with the object and thus an $\mathcal{MLL}$ is a good separation boundary between two moving objects. By using such an approach, one can correctly match boundary points on many objects, even if the objects themselves are homogenous and textureless and do not yield any corner points in the interior.

The SSD is used as the distance measure between the two patches in our work, although more complicated descriptors such as the SIFT can also be considered for many applications. A gradient descent is applied to the (half) patch before computing the SSD in order to deal with small shifts and errors in point localization. The SSD values are computed in the common(intersection) region of the masks of the two patches being matched and are normalized by the size of this region.

In case no external information is available, one can utilize the above two-region matching approach for all the points. However, if it is possible to classify the points as interior and boundary points, perhaps in an iterative way, then one can apply the two-region matching only to the boundary points as considering only a part of the patch for matching for the interior points does reduce the discriminability of the matcher due to utilization of lesser information.

5. Experiments and Results

While scale and affine-invariant features have been developed for many applications, in this work, we have developed a basic feature detector that does not handle scale or affine variations. Hence, we compare against basic feature detectors (Harris, Hessian, FAST) which are useful in applications such as point detection and tracking in videos. Thus, we evaluate for these applications only.

Harris, Hessian and FAST have been found to be the best basic detectors in many evaluations [39, 48]. More recent ones such as FAST-ER and AGAST improve the speed of detection but their performance is quite similar to FAST [26, 39] and thus only FAST was compared against. All the scale and affine-invariant detectors [28, 29] including level line based methods such as SAF [36] and LAF [34] performed significantly worse than the basic point detectors for these applications and are not shown, due to lack of space.We include results for MSER since our method is closely related to theirs.

Dataset: The dataset that we choose for evaluation is the publicly available KITTI dataset [14]. The dataset has realistic, challenging outdoor sequences with good ground-truths. We evaluate our method on 11 video sequences for Point Tracking and 194 image pairs for Optical Flow from this dataset. Each vehicle in the tracking sequence moves through roads against different backgrounds and the Optical Flow sequences consist of vehicles and other real-world structures with significant depth discontinuities.

5.1. Vehicle tracking

We first consider a vehicle tracking application which uses interest point tracking [11, 44, 43, 37, 41, 32, 19]. The seminal KLT algorithm [25] is still quite popular for such an application [43, 44, 32] along with its variants [11, 37]. In this application, interest points are detected and matched in subsequent frames. While simple tracking might work for a few frames, the tracks eventually get lost and have to be re-detected and matched to the original ones for longer term tracking.

11 challenging sequences from the KITTI dataset that have significant variations in the background were selected and we compare results for point matching at a gap of 1 and 5 frames to test the efficacy of the detectors and matchers for shorter and longer range point matching respectively. While matching, an appropriate neighborhood was set as the search region in order to restrict the amount of motion that each point can undergo.

Since the dataset contains only car tracking bounding boxes, the ground truth for point matches was generated from the annotated bounding boxes by assuming that the relative location of a point w.r.t. to the bounding box remains the same across frames. A small amount of error is allowed, as the object is not rigid in 2D and there might be some errors in the bounding box annotations. A 10-pixel allowance was found to be sufficient for this dataset.

For a fair comparison, we equalize the average number of detected features detected by a detector as far as possible. For detectors that return very few points (e.g MSER), the threshold is lowered as much as reasonably possible. For CoMaL, the threshold used to vary the number of points is

Seq	CoMaL+SSD	SSD				NSD				SIFT			
		Harris	Hessian	MSER	FAST	Harris	Hessian	MSER	FAST	Harris	Hessian	MSER	FAST
Gap of 1													
CarA	**165.7**/0.7	99.9/0.7	150.3/0.7	25.5/0.6	84.3/0.7	105.8/0.6	150.5/0.7	25.8/0.6	103.7/0.6	95.5/0.6	126.2/0.6	26.0/0.6	105.7/0.7
CarB	**159.4**/0.7	53.9/0.7	126.9/0.7	20.4/0.5	108.1/0.7	57.9/0.6	130.3/0.7	19.8/0.6	140.0/0.7	55.1/0.7	125.5/0.7	20.3/0.6	115.4/0.7
CarC	**111.4**/0.7	46.3/0.7	88.2/0.7	14.1/0.6	82.7/0.7	41.1/0.7	93.3/0.7	12.2/0.6	92.3/0.7	43.6/0.7	99.2/0.7	14.2/0.6	96.2/0.7
CarD	**162.8**/0.7	39.7/0.7	125.0/0.7	16.5/0.6	95.0/0.7	50.9/0.7	141.0/0.7	17.8/0.7	140.5/0.7	52.0/0.7	142.8/0.6	19.2/0.6	137.2/0.7
CarE	**134.4**/0.8	36.4/0.7	115.6/0.8	13.2/0.7	109.5/0.8	39.2/0.8	120.8/0.8	14.6/0.8	127.1/0.8	39.9/0.8	123.4/0.8	14.8/0.8	127.0/0.8
CarF	**95.1**/0.8	45.7/0.7	79.1/0.7	14.0/0.6	51.5/0.7	37.3/0.8	83.1/0.8	11.5/0.8	52.4/0.8	43.0/0.7	74.3/0.7	13.2/0.7	61.8/0.7
CarG	**66.5**/0.8	26.0/0.8	57.3/0.8	6.9/0.7	48.4/0.8	25.3/0.8	48.0/0.8	7.0/0.8	53.9/0.8	27.0/0.8	51.1/0.8	7.5/0.8	55.4/0.8
CarH	**98.4**/0.8	42.3/0.7	58.8/0.8	9.3/0.6	43.9/0.8	31.6/0.8	79.6/0.8	7.5/0.8	56.1/0.9	33.5/0.8	74.6/0.8	8.9/0.8	60.4/0.8
CarI	**370.6**/0.7	171.5/0.6	280.3/0.6	54.0/0.5	262.5/0.6	185.9/0.6	339.0/0.6	57.1/0.6	324.5/0.6	179.6/0.6	342.7/0.6	58.1/0.6	345.0/0.6
CarJ	**360.8**/0.7	117.3/0.6	248.6/0.6	31.7/0.5	231.0/0.6	146.1/0.7	325.7/0.7	35.7/0.6	315.0/0.7	129.5/0.6	316.8/0.6	33.9/0.6	323.9/0.7
CarK	**364.1**/0.7	195.8/0.5	296.8/0.5	54.5/0.5	289.3/0.5	202.9/0.6	329.5/0.6	55.5/0.6	320.3/0.6	189.9/0.5	333.5/0.5	56.5/0.5	342.1/0.6
Gap of 5													
CarA	**84.6**/0.7	55.7/0.7	60.6/0.6	15.9/0.6	30.3/0.6	45.6/0.7	74.7/0.7	11.5/0.7	42.5/0.7	30.7/0.7	53.5/0.7	8.7/0.7	25.1/0.7
CarB	**120.2**/0.7	49.7/0.7	102.8/0.7	21.1/0.6	91.2/0.7	48.1/0.7	71.2/0.7	18.7/0.7	63.4/0.7	31.4/0.7	89.2/0.7	15.0/0.7	80.9/0.7
CarC	**78.8**/0.7	32.5/0.7	49.6/0.7	11.1/0.6	49.4/0.7	18.6/0.7	51.2/0.7	6.8/0.7	47.6/0.7	19.2/0.7	59.8/0.7	7.4/0.7	54.9/0.7
CarD	**57.4**/0.8	10.6/0.7	36.2/0.8	4.6/0.7	33.7/0.8	15.7/0.8	47.4/0.8	6.0/0.7	45.6/0.8	9.7/0.8	38.6/0.8	3.8/0.7	40.3/0.8
CarE	**91.2**/0.8	20.1/0.8	80.1/0.8	7.9/0.8	83.7/0.8	23.9/0.8	83.4/0.8	8.8/0.8	82.8/0.8	17.5/0.8	76.5/0.8	6.7/0.8	85.8/0.8
CarF	**62.7**/0.7	36.1/0.7	44.6/0.7	8.9/0.7	31.1/0.7	16.6/0.7	40.1/0.7	5.1/0.7	26.6/0.7	15.2/0.7	40.1/0.7	5.5/0.7	27.5/0.7
CarG	**57.4**/0.8	21.1/0.8	39.5/0.8	6.2/0.8	36.6/0.8	21.0/0.8	52.8/0.8	5.8/0.8	50.0/0.8	17.8/0.8	53.8/0.8	5.2/0.8	56.5/0.8
CarH	**52.4**/0.8	27.0/0.8	43.9/0.8	6.9/0.8	37.1/0.8	27.2/0.8	43.3/0.8	6.7/0.8	40.6/0.8	20.9/0.8	48.1/0.8	6.0/0.8	40.0/0.8
CarI	**275.7**/0.7	112.9/0.7	191.6/0.7	40.0/0.6	193.6/0.7	110.8/0.7	215.2/0.7	37.8/0.7	202.8/0.7	85.8/0.7	184.5/0.7	31.8/0.7	191.3/0.7
CarJ	**232.5**/0.7	94.2/0.6	212.1/0.6	26.1/0.5	214.5/0.6	106.8/0.7	183.5/0.7	30.6/0.7	194.3/0.7	82.7/0.7	198.5/0.6	22.8/0.7	201.3/0.7
CarK	**237.2**/0.7	95.1/0.7	154.7/0.7	31.2/0.6	162.8/0.7	127.4/0.7	163.2/0.7	35.2/0.7	184.0/0.7	112.4/0.7	200.9/0.7	25.5/0.7	170.8/0.7
Total	**122.7**	50.4	91.9	15.3	87.6	51.0	93.2	15.7	89.1	40.3	94.8	12.5	88.6

Table 1. Number of Correct Matches M_{cor} for 11 video sequences from the KITTI Vehicle Tracking dataset averaged over all the frames in the sequence. The second number is the Matching accuracy M_{acc} for the method. The top rows show the results at a gap of 1 frame (consecutive frames) while the bottom rows show results at a gap of 5 frames. The best result is highlighted in bold while the second best is underlined.

Region	CoMaL+SSD	SSD				NSD				SIFT			
		Haris	Hessian	MSER	FAST	Harris	Hessian	MSER	FAST	Harris	Hessian	MSER	FAST
B	**37.4**/0.9	18.2/0.7	26.3/0.8	4.0/0.4	23.9/0.9	23.3/0.9	22.3/0.9	2.2/0.7	23.2/0.9	24.8/0.9	30.5/0.9	3.1/0.8	23.7/0.9
N-B	90.6/0.9	75.6/0.6	87.4/0.8	14.6/0.2	60.8/0.8	82.5/0.9	90.1/0.8	15.6/0.6	68.8/0.9	88.9/0.9	**91.7/0.9**	11.2/0.7	75.9/0.9

Table 2. Average number of correct matches M_{cor} for 194 pairs from the KITTI Flow dataset with the corresponding Matching accuracy M_{acc} in the B (Boundary), and N-B (Non-Boundary Regions). The best is in bold and the second best is underlined.

Figure 7. Frame numbers 88 and 93 in the sequence Car-B from the KITTI dataset showing CoMaL + SSD matches in the first row followed by next performing combination: Hessian + SIFT and FAST + NSD in the 2nd and 3rd rows respectively. CoMaL points are matched more numerously and accurately at the object boundary regions in spite of a significant change in the background.

the threshold on the stability value ρ. For a fair comparison, we use a typical scale value of 8.4 for all the detectors and all the other parameters for the detectors and descriptors are kept at their default values used in standard implementations. Comparative results for other scale settings were similar. While CoMaL is combined with only the SSD matcher, the other detectors are combined with SSD, NSD [5] and SIFT. CoMaL doesn't work very well with SIFT or NSD as the regions on either side of the level line are often homogeneous and not suitable for these descriptors.

We define the matching accuracy or precision M_{acc} as the ratio of the number of correct matches M_{cor} to the total number of matches M found: $M_{acc} = M_{cor}/M$. Equalizing this for the different algorithms by varying the matching thresholds, one can compare the number of correct matches generated by the algorithm averaged over all frames.

Fig 7 and videos in the supplementary section show some qualitative results of the approach, while Table 1 shows the quantitative results. It is clear from the results that CoMaL yields a much higher number of correctly matched points compared to other approaches at a similar or higher accuracy. Generally, Hessian performs second, closely followed by FAST. The superior performance of our approach can be attributed to a much better performance and resilience in the boundary regions that are quite significant for these vehicle objects, while the interior points are correctly matched by most methods.

Type	Seq	CoMaL+SSD	SSD				NSD				SIFT			
			Harris	Hessian	MSER	FAST	Harris	Hessian	MSER	FAST	Harris	Hessian	MSER	FAST
Textured	Pens	**27.6**/0.9	12.6/0.8	8.5/0.9	0.2/0.2	3.2/0.8	14.8/0.9	17.0/0.9	0.3/0.3	15.1/0.9	12.1/0.8	20.8/0.9	0.5/0.4	12.3/0.9
	Doll	**39.0**/0.9	14.8/0.8	11.7/0.8	0.6/0.3	9.0/0.8	21.4/0.9	29.4/0.9	0.4/0.5	31.1/0.9	16.2/0.8	22.3/0.9	0.7/0.6	24.7/0.9
	Toy	**31.2**/0.8	9.0/0.7	13.6/0.8	0.7/0.4	12.3/0.8	13.1/0.8	11.9/0.8	0.3/0.3	11.7/0.8	10.4/0.7	12.8/0.8	0.4/0.4	13.6/0.8
	Hero	**47.4**/0.9	16.4/0.9	17.8/0.9	1.2/0.5	16.7 /0.9	18.2/0.8	29.6/0.9	1.1/0.6	30.0/0.9	15.9/0.8	24.0/0.9	1.3/0.7	25.1/0.9
	Race-car	**52.5**/0.9	17.0/0.8	18.6/0.8	0.7/0.5	12.4/0.9	20.7/0.9	30.6/0.9	0.4/0.4	27.9/0.9	16.5/0.8	31.0/0.9	0.6/0.5	28.3/0.9
Homogeneous	Box	**37.5**/0.9	14.5/0.9	21.8/0.9	0.5/0.2	19.3/0.9	19.4/0.9	19.2/0.9	0.3/0.4	18.8/0.8	16.3/0.9	25.3/0.9	0.5/0.4	24.5/0.9
	Tape-Box	**39.1**/0.9	15.9/0.9	16.0/0.9	0.8/0.5	16.5/0.9	18.9/0.9	25.2/0.9	0.5/0.3	26.2/0.9	16.3/0.9	21.3/0.9	0.6/0.4	26.3/0.9
	House	**32.9**/0.9	13.2/0.9	21.0/0.9	0.6/0.4	22.4/0.9	17.7/0.8	25.0/0.9	0.5/0.5	27.5/0.9	13.9/0.8	25.7/0.9	0.7/0.6	28.8/0.9
	Average	**38.4**	14.2	16.1	0.7	14.0	18.0	23.5	0.5	23.6	14.7	22.9	0.7	23.0

Table 3. Number of Correct Matches M_{cor} on the boundary regions for sequences in the CoMaL dataset averaged over all the frames in the sequence. The second number is the Matching accuracy M_{acc} for the method. The best is in bold and the second best is underlined.

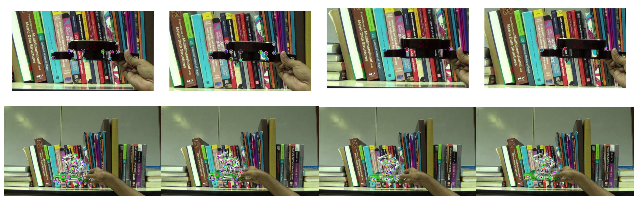

Figure 8. Top Row: Matches on a Homogenous object - Box. Bottom Row: Matches on a Textured Object. CoMaL + SSD matches are shown in the first two images while Hessian + SIFT matches are shown in the last two.

5.2. Optical Flow

Optical flow is a dense point tracking problem and many optical flow techniques use point feature tracking as an input to the computation of flow for the entire scene [22, 2]. We evaluate our features for this application by matching points across pairs of images and verifying them using the ground-truth flow map provided with the dataset.

Since in this dataset, the full flow is available, one can determine the boundary regions by looking at motion discontinuities. This helps us evaluate the detectors separately at the boundary and non-boundary regions. The evaluation criteria is chosen to be same as the vehicle tracking application and Table 2 shows the average number of correctly matched points across the given flow pairs on the boundary and internal/non-boundary points separately.

In this test, it becomes clear that CoMaL + SSD outperforms the other approaches in the boundary regions while performing close to the best detector and matcher combinations in the non-boundary regions. Slightly lower performance for our method can be expected in the non-boundary portions as others use information from the whole patch for matching while we use only around half of it.

5.3. Our Own Dataset

Finally, to evaluate the performance of the detectors at the boundary regions and under a varying background, since no suitable dataset exists in the literature, we have developed our own dataset. The background and the camera are kept static that allows the use of background subtraction to separate out the foreground from the background. This also enables detection of the boundary regions between the foreground and background for evaluation purposes.

Ground-truthing is done by extracting foreground blobs and assuming that the the relative location of a point with respect to the blob center does not change drastically over the frames. Matches obtained with CoMaL+SSD and Hessian+SIFT (which performs second-best) are shown visually for a homogeneous and textured object in Fig. 8.

Table 3 presents quantitative results at the boundary regions. As can be seen, CoMaL beats all the competing methods by a large margin at the boundaries for both homogenous and textured objects, with an overall increase of 14.8 correctly matched points on an average over FAST + NSD which performs next best, closely followed by Hessian + NSD. Results at non-boundary regions are comparable to other detectors (shown in supplementary section).

Discussion For applications with significant boundary portions, our method can be used on its own. For other applications, one could use our method with others such as the Hessian, perhaps in an iterative framework, where the boundary and non-boundary portions are estimated iteratively in order to determine the best algorithm to use for different regions.

6. Conclusion and Future Work

We have presented an algorithm for corner detection and matching that was found to be much more robust in the boundary regions compared to existing approaches. This is accomplished by detecting corners on maximally stable level lines that often trace the object boundaries and by matching the two regions separated by such level lines separately. Results on point tracking on several datasets including the challenging real-world KITTI dataset show that our method is able to extract and correctly match much more points compared to existing approaches in the boundary regions. Future work includes application to other problems where our approach might be useful, such as SfM in video sequences and stereo.

References

[1] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *CVPR 2012*, pages 510–517. Ieee, 2012. 1

[2] S. Ali. Measuring flow complexity in videos. In *ICCV 2013*, pages 1097–1104. IEEE, 2013. 1, 8

[3] R. Almomani and M. Dong. Segtrack: A novel tracking system with improved object segmentation. In *ICIP 2013*, pages 3939–3943. IEEE, 2013. 2

[4] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, pages I: 404–417, 2006. 1, 5

[5] J. Byrne and J. Shi. Nested shape descriptors. In *ICCV 2013*, pages 1201–1208. IEEE, 2013. 1, 7

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *ECCV 2010*, pages 778–792. Springer, 2010. 1

[7] F. Cao, P. Mus, and F. Sur. Extracting meaningful curves from images. *Journal of Mathematical Imaging and Vision*, 22(2-3):159–181, 2005. 2, 4

[8] D. Chen, Z. Yuan, Y. Wu, G. Zhang, and N. Zheng. Constructing adaptive complex cells for robust visual tracking. In *ICCV 2013*, pages 1113–1120. IEEE, 2013. 2

[9] M. Donoser and H. Bischof. Efficient maximally stable extremal region (mser) tracking. In *CVPR 2006*, volume 1, pages 553–560. IEEE, 2006. 2

[10] L. B. Dorini and S. K. Goldenstein. Unscented feature tracking. *CVIU*, 115(1):8–15, 2011. 1

[11] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, 2014. 1, 6

[12] W. Förstner. A framework for low level feature extraction. In *ECCV 1994*, pages 383–394. Springer, 1994. 4

[13] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *PAMI*, 32(8):1362–1376, 2010. 1

[14] A. Geiger, P. Lenz, et al. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2, 6

[15] S. Hare, A. Saffari, and P. H. Torr. Efficient online structured output learning for keypoint-based object tracking. In *CVPR 2012*, pages 1894–1901. IEEE, 2012. 1

[16] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, page 50, 1988. 1, 4

[17] S. S. Intille and A. F. Bobick. *Disparity-space images and large occlusion stereo*. Springer, 1994. 1

[18] R. Jensen, A. Dahl, Vogiatzis, et al. Large scale multi-view stereopsis evaluation. In *CVPR 2014*. 1

[19] J.-P. Jodoin, G.-A. Bilodeau, and N. Saunier. Urban tracker: Multiple object tracking in urban mixed traffic. In *WACV*, pages 885–892. IEEE, 2014. 6

[20] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *PAMI*, 16(9):920–932, 1994. 2

[21] G. Klein and D. Murray. Improving the agility of keyframe-based slam. In *ECCV 2008*, pages 802–815. 1

[22] P. Lenz, J. Ziegler, A. Geiger, and M. Roser. Sparse scene flow segmentation for moving object detection in urban environments. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 926–932. IEEE, 2011. 1, 8

[23] M. Lourenço and J. P. Barreto. Tracking feature points in uncalibrated images with radial distortion. In *ECCV 2012*, pages 1–14. Springer, 2012. 1

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004. 1, 4

[25] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981. 1, 6

[26] E. Mair, G. D. Hager, D. Burschka, Suppa, et al. Adaptive and generic corner detection based on the accelerated segment test. In *ECCV 2010*, pages 183–196. 2010. 1, 6

[27] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, pages 36.1–36.10. BMVA Press, 2002. 2, 3

[28] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 60:63–86, 2004. 1, 6

[29] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, et al. A comparison of affine region detectors. *IJCV*, 65:43–72, 2005. 2, 3, 6

[30] K. Mikolajczyk, A. Zisserman, and C. Schmid. Shape recognition with edge-based features. In *BMVC 2003)*, volume 2, pages 779–788. 2

[31] G. Nebehay and R. Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *WACV 2014*, pages 862–869. IEEE, 2014. 1

[32] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *CVPR 2004*, volume 1, pages I–652. IEEE, 2004. 1, 6

[33] D. Nister and H. Stewenius. Linear time maximally stable extremal regions. In *ECCV 2008*, volume 5303, pages 183–196. Springer, 2008. 5

[34] Š. Obdržálek and J. Matas. Object recognition using local affine frames on maximally stable extremal regions. In *Toward Category-Level Object Recognition*, pages 83–104. Springer, 2006. 2, 6

[35] S. Oron, A. Bar-Hille, and S. Avidan. Extended lucas-kanade tracking. In *ECCV 2014*, pages 142–156. Springer, 2014. 2

[36] M. Perdoch, J. Matas, and S. Obdrzalek. Stable affine frames on isophotes. In *ICCV 2007*, pages 1–8, 2007. 2, 3, 4, 6

[37] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, Mordohai, et al. Detailed real-time urban 3d reconstruction from video. *IJCV*, 78(2-3):143–167, 2008. 6

[38] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *ICCV 2005*. 1

[39] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE PAMI*, 32:105–119, 2010. 1, 6

[40] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *ICCV 2011*. 1

[41] K. Sakurada, T. Okatani, and K. Deguchi. Detecting changes in 3d structure of a scene from multi-view images captured by a vehicle-mounted camera. In *CVPR 2013*, pages 137–144. IEEE, 2013. 1, 6

[42] J. Shi and C. Tomasi. Good features to track. In *CVPR 1994*, pages 593–600, 1994. 1, 4

[43] H. Song, S. Lu, X. Ma, Y. Yang, X. Liu, and P. Zhang. Vehicle behavior analysis using target motion trajectories. In *Vehicular Technology, IEEE Transactions on*, 2013. 1, 6

[44] S. Tanathong and I. Lee. Translation-based klt tracker under severe camera rotation using gps/ins data. *Geoscience and Remote Sensing Letters, IEEE*, 11(1):64–68, 2014. 6

[45] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *PAMI*, 32(5):815–830, 2010. 1, 2

[46] D.-M. Tsai, H.-T. Hou, and H.-J. Su. Boundary-based corner detection using eigenvalues of covariance matrices. *Pattern Recognition Letters*, 20(1):31–40, Jan. 1999. 4

[47] T. Tuytelaars and L. J. V. Gool. Matching widely separated views based on affine invariant regions. *IJCV*, 59(1):61–85, 2004. 2

[48] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *FnT Comp. Graphics and Vision*, pages 177–280, 2008. 6

[49] G. Zhang, Z. Dong, J. Jia, T.-T. Wong, and H. Bao. Efficient non-consecutive feature tracking for structure-from-motion. In *ECCV 2010*, pages 422–435. Springer, 2010. 1