N-tree Disjoint-Set Forests for Maximally Stable Extremal Regions

Erik Murphy-Chutorian and Mohan Trivedi
Computer Vision and Robotics Research Laboratory
University of California, San Diego
erikmc@ucsd.edu, mtrivedi@ucsd.edu

Abstract

In this paper we introduce the NDS-Forest data structure, which can be used for the calculation and representation of Maximally Stable Extremal Regions in real-time video. In contrast to the standard MSER algorithm, the NDS-Forest stores information about the extremal regions as they are formed, making it unnecessary to regrow the regions from seed pixels. Using the NDS-Forest structure, we describe a system that uses MSERs in an automobile for face registration, segmentation, and pose estimation of the driver.

1 Introduction

While the visual world is comprised of three-dimensional objects transformed by projective geometry, it is often much easier to represent visual information as a set of local two-dimensional features. As a consequence, there has been much interest in processes that locate and describe salient image regions that are invariant to changes in pose, rotation, and scale.

Although no methods exist to completely reverse the projective deformation without depth information, affine-covariant region detectors have been proposed which can repeatably detect regions that undergo affine distortion [10, 4, 8, 17]. Since a projective transformation is locally-affine, these techniques are often applied to images under the assumption that local image patches are small enough to merit an affine approximation.

The central concept underlying affine-covariant regions is the ability to apply an affine transformation to an image region that transforms it into a canonical shape. Quite frequently this is accomplished by mapping an elliptical image patch onto a uniform circular region. This can be accomplished by iteratively changing the shape of the ellipse such that the normalized circular image patch will have an isotropic texture [10, 6]. This leaves a rotational ambiguity, but this last unknown can be overcome with rotationally-invariant region descriptors [11, 5], or by careful choice of a dominant direction [7] although such direction can be highly unstable given the isotropic constraint.

Another approach is to use *Maximally Stable Extremal Regions (MSERs)*, as they provide an affine-covariant region with an arbitrarily shaped boundary [8]. In this case, affine covariance can be achieved by normalizing the region's covariance matrix about its center of gravity and then using the characteristic shape of the region itself to resolve the rotational ambiguity [18].

There are further advantages of MSERs. In a systematic comparison of affine-covariant region detectors, MSERs were shown to outperform all other detectors in terms of repeatability given viewpoint and lighting changes [12]. In the same comparison, it was shown that MSERs can be computed much more quickly than any of the other affine-covariant regions.

MSERs have been successfully applied to a variety of problems, including wide-baseline stereo [8], object recognition [19, 9, 14], image retrieval [13, 18, 15], and scene classification [1]. In this paper we will also demonstrate how MSERs can be used for face registration and pose estimation, although our primary contribution in this work is the NDS-Forest for efficient calculation of the regions.

For any image processing program, there are some simple rules that can dramatically reduce the computational requirements, which can ensure that the program will continually process frames in real-time.

1. No heap memory allocation at runtime. Allocating and deallocating memory in every frame can dramatically slow down realtime applications. The solution is to preallocate enough memory at initialization, and continuously write and overwrite this space.

2. Efficient algorithms. Although very intuitive, it is imperative to use the algorithm that achieves the least upper-bound on efficiency.

3. No unnecessary recalculations. The results of calculations that will be required later should be maintained in memory. Assuming that the memory requirements are not unreasonable, there is no reason to recompute information that is required in multiple steps in the program.

It is with this third rule especially that the MSER algorithm can be improved. In its published form, the algorithm provides a method to characterize all of the MSERs in a single pass through the image, but it provides no method to retain specific information about the regions. To determine the pixels that comprise the region, one must complete a flood fill algorithm once for each region, although all such information existed and was lost during the first pass through the image. Since there can be hundreds or thousands or these regions in a single image frame, this is a substantial portion of the total computation.

In this paper, we introduce the NDS-Forest data structure that allows for faster calculation of Maximally Stable Extremal Regions. We show that it that satisfies all of the rules mentioned above, while retaining all necessary information during the detection step, so that finding the pixels that make up the region requires a simple tree traversal. We have used this framework to create a system that is able to detect and display maximally stable regions in real-time. In addition, we demonstrate a novel application in which MSERs coupled with a skin-tone prior can be used for real-time face registration in a pose estimation system. In comparison to color-based thresholding approaches, this method can succeed despite affine changes in hue and persons whose skin color significantly deviates from the prior.

In Section 2 we describe MSERs in more detail, and in Section 3, we describe our new data structure. In Section 4 we describe our method of face registration and segmentation, and in Section 5 we provide concluding remarks.

2 Maximally Stable Extremal Regions

Maximally Stable Extremal Regions are best described using a watershed topography analogy [8]. Fundamentally, a grayscale image is a two-dimensional function, mapping an (x,y) coordinate to an intensity value. Similarly, a watershed can be represented as a function assigning a depth to every 2-D position. To understand MSERs, you must first imagine that the the watershed is initially dry and then slowly filled with water. Initially, puddles would begin to form in the deepest crevasses. As the water level increases, the puddles would become ponds and lakes, and occasionally two of these would merge to form a larger body of water. This step can be viewed as the termination of the smaller lake, and the addition of all of its water into the larger lake. When this occurs, the volume of water in the lake is highly unstable as a tiny increase in the water level changed the volume dramatically. With Maximally Stable Extremal Regions, the focus is to discover water levels that are instead local minima in the rate of change of the water volume.

In general, an *extremal region* is a set of pixels connected by their 4-neighbors that satisfy the property that all of their intensities are uniformly greater or less than the intensities of every pixel that surrounds the region. An extremal region is *maximally stable* if given an intensity i and a margin Δ, the change in the number of pixels in the region from $i-\Delta$ to $i+\Delta$ is a local minimum. A large value for Δ will generate only a few highly stable regions, while a smaller value will detect many less-stable regions. The advantage of a small delta value is that more regions will be detected which allows more invariant information to be extracted from an image. The primary trade-off is an increase in complexity to characterize these regions, which suggests the need for an efficient representation. Figure 1 illustrates the effect of different values of Δ.

In the original formulation, MSERs are detected as follows. First all of the pixels in the image are sorted by intensity, which can be quickly accomplished in $O(n)$ with the non-comparison based BinSort or CountingSort [8, 2]. Next, the pixels are added in order to a list of connected components maintained by a *disjoint-set data structure*. The data structure has an associated *union-find* algorithm, which provides methods to unite two components and find the component in which a particular pixel belongs [8, 2]. As more and more pixels are added to the list, the components grow and merge with each other. After all of the pixels have been added, nothing is left but a single component comprising every image pixel. To discover the MSERs, a separate data structure is used to keep track of the area of each connected component as pixels are added to the components. This structure stores the area of each component as a function of intensity, terminating any component if it merges with a larger one. Once all of the pixels have been added, the MSERs can be found as the components that satisfy the maximal stability criterion mentioned above. Since the typical disjoint-set data structure is not *persistent* [3], it is impossibly to recover the connected components that existed after a subset of the pixels had been added. Thus, at the termination of the algorithm, nothing is known about the MSERs other than a single pixel within the region and a boundary threshold, since all of the information characterizing the connected components has been lost. In the following section we describe a solution to this problem.

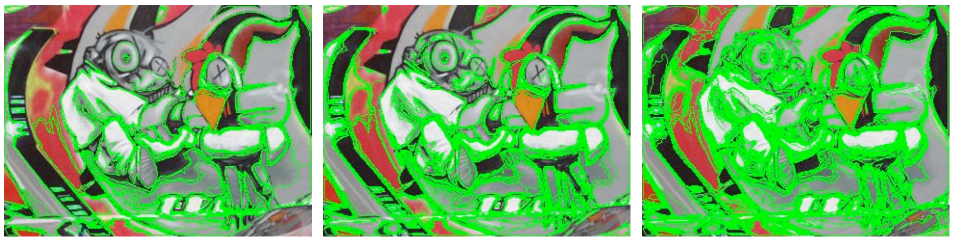


Figure 1: Maximally Stable Regions displayed with bounding boxes. Note the increase in the number of regions for different margins. (left) 254 Regions. (center) 681 Regions. (right) 1533 Regions.

3 NDS-Forest

In graph theory, a *node* is the basic building block of a *graph*, which is comprised of nodes connected to each other by *edges*. A *path* is a sequence of consecutive edges. A *tree* is a graph in which any two nodes that are connected by exactly one path, while an *N-tree*, is a tree in which any node has at most N edges. A *forest* refers to a graph comprised of one or more trees. A *disjoint-set forest* is a disjoint-set data structure that represents each connected component as a separate tree in a forest [2].

We now introduce the *N-tree Disjoint-Set Forest* (NDS-Forest). The NDS-Forest is a disjoint-set data structure that is *partially persistent*, meaning that one can recover the nodes that existed in any tree at any given time. In the context of MSERs, the NDS-Forest can recover the full set of region pixels given a single point and intensity threshold.

The NDS-Forest combines aspects of a doubly-linked N-tree data structure and a disjoint-set forest. It exceeds the functionality of the disjoint-set forest by maintaining an N-tree that keeps track of the order in which nodes are added to each connected component and how components merge together.

3.1 Union Find

As mentioned in Section 2, a *union-find* algorithm provides two methods: Union(X,Y) and Find(X). *Union* merges the two sets that contain nodes X and Y, while *Find* returns the characteristic *root* node of the set containing X. Given a disjoint-set forest, these methods can be efficiently performed with the use of two heuristics [2]:

Union by Rank is the process of always connecting the smaller tree to the root of the larger tree. This can be achieved by maintaining the parameter *rank*, which provides an upper-bound on the height of each node. A tree whose root has a greater rank is considered larger than one with lesser rank, and whenever two trees of the same rank are merged together, one is arbitrarily connected to the other and the rank of the root is incremented by one.

Path Compression is a method to flatten the trees. Whenever Find(X) traverses a node that does not directly connect to the root of the tree, the node is made to point to the root of the tree. This way, subsequent calls to Find(X) require only a single operation.

Combining both Union by Rank and Path Compression, n image pixels can be added to the forest in $O(n\alpha(n))$ where $\alpha(n)$ is the extremely slow-growing inverse Ackermann

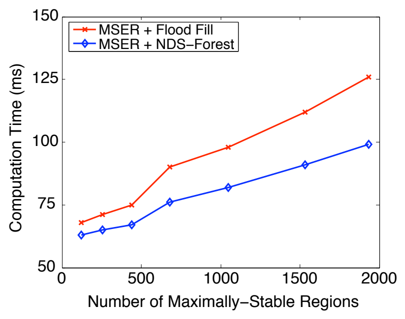


Figure 2: Comparison of the NDS-Forest to MSER + Flood Fill (320x240 pixel image)

function. This has been proven to be the optimal efficiency for any disjoint-set data structure [16].

3.2 N-tree Graph

Each node in an NDS-Forest possesses an array of N initially $NULL$ pointers to neighboring nodes, as well as an index corresponding to the time at which it was added to the forest. Whenever Union(X,Y) is called, one of the free pointers of X is used to connect X to Y, and another of Y to connect Y to X. In the case of MSERs, any pixel can have as many as four possible neighbors, so in this application $N = 4$. The full algorithm is presented in Algorithm 1. At any time index, every node that has been added to the NDS-Forest is a member of a doubly-connected N-tree along with every other node in the connected component. To find all of the nodes that existed in the component at a specific index i, one can traverse the tree beginning at any node, stopping at any leaf or any node with $index > i$. A recursive implementation of this traversal is shown in Algorithm 2, and it should be noted that a queue-based iterative solution is also easily realized.

In summary, each NDS-Forest Node contains the following minimum data:

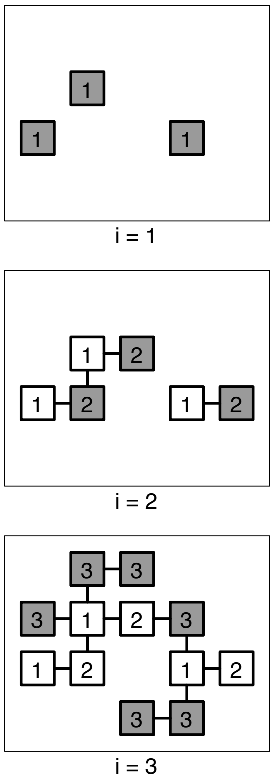
parent - A pointer leading to the characteristic member of each set
index - The time index at which the node was created
rank - An integer providing an upper-bound on the height of each node
neighbor[N] - An array of up to N pointers connecting nodes to each other

3.3 MSER + NDS-Forest

To quickly calculate MSERs using an NDS-Forest, we can preallocate an NDS-Forest node for each pixel in the image. We sort the image pixels by increasing intensity, and then we add them one-by-one into the forest. This involves one call to MakeSet(X,i) for each insertion, followed by a call to Union(X,Y) for each of the 4-neighbors that have already been added to the forest. As in the standard MSER algorithm, we main a separate list of extremal regions and their areas, and once all of the pixels have been added, we find all of the extremal regions that are maximally stable. We now can realize any of these regions using the method described in Algorithm 2 to traverse the region in $O(m)$ where m is the number of pixels in the region. This is the lowest possible bound on a traversal,

Algorithm 1 NDS-Forest Insertion

```
Procedure MakeSet($X, index$)
  $X.parent \leftarrow X$
  $X.index \leftarrow index$
  $X.rank \leftarrow 0$
  $X.neighbor[1:4] \leftarrow NULL$
Procedure Find($X$)
  if $X \neq X.parent$ then
    $X.parent \leftarrow$ Find($X.parent$)
  end if
  return $X.parent$
Procedure Union($X, Y$)
  $A \leftarrow$ Find($X$)
  $B \leftarrow$ Find($Y$)
  if $A \neq$B then
    $X.neighbor$[length($X.neighbor$) + 1] $\leftarrow Y$
    $Y.neighbor$[length($Y.neighbor$) + 1] $\leftarrow X$
    if $A.rank > B.rank$ then
      $B.parent \leftarrow A$
    else
      $A.parent \leftarrow B$
    end if
    if $A.rank = B.rank$ then
      $B.rank \leftarrow B.rank + 1$
    end if
  end if
```



NDS-Insertion Example

since each node is visited exactly once. By including the pixel position as an additional data member in each node, we can perform operations on any region such as estimating the center of gravity, fitting an ellipse, finding a convex hull, or simply recovering the pixel-level segmentation. To find MSERs with decreasing intensity, we repeat the above steps but insert the pixels in decreasing order.

In terms of performance, we have implemented the MSER algorithm using an NDS-Forest on a personal computer with a 3 GHz processor. Given a 320x240 video stream, our implementation can discover all of the MSERs at approximately 15fps. We have compared our implementation to one which requires an explicit flood filling[1]. Figure 2 shows the overall improvement using the NDS-Forest which is more pronounced for larger images with many regions.

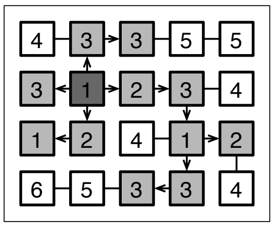
4 Face Registration and Segmentation

While MSERs have the ability to locate invariant regions with high repeatability, the regions themselves can be tailored to a specific task by modifying the input. To use MSERs

[1]We compared our system to a basic recursive flood filling algorithm. Faster methods exist, but they are more complex to implement and still cannot achieve the performance efficiency of a tree traversal.

Algorithm 2 NDS-Forest Traversal

```
**Procedure** StartTraverse($X, threshold$)
  Traverse($X, threshold, NULL$)
**Procedure** Traverse($X, threshold, previousX$)
  **if** $X \neq NULL$ **and** $X.index \leq threshold$ **then**
    **for** $k = 1$ to length($X.neighbor$) **do**
      **if** $X.neighbor[k] \neq previousX$ **then**
        Traverse($X.neighbor[k], threshold, X$)
      **end if**
    **end for**
  **end if**
```



Traversal to $i = 3$

as a detector of face regions, we can pre-filter the image to create extremal regions out of skin-tone. This can be accomplished by considering the chrominance components from a digital color image. In our experiments, we convert an RGB image into the CIE Lab perceptual colorspace. In this space, every pixel has a luminance component L and two chrominance components a and b. By ignoring the luminance components and considering only the color vector $c = [a\ b]^{\mathsf{T}}$ we can create a system that is sensitive to color, but invariant to changes in intensity. We create a skin-tone prior by assuming a jointly-gaussian distribution on c and estimating the mean and covariance μ_c and Σ_c from color samples taken from face images of ten people. With these parameters, we create the Mahalanobis distance image,

$$M(x,y) = \big(I(x,y) - \mu_c\big)^{\mathsf{T}} \Sigma_c \big(I(x,y) - \mu_c\big). \tag{1}$$

We then apply the MSER algorithm to the Mahalanobis distance image since the skin-tone of a face clearly delineates a stable extremal region. This method has multiple advantages over a thresholding approach. Since the algorithm has a preference for extremal regions rather than a range of intensity, persons with skin-tone that greatly varies from the prior will still provide a stable regions so long as their skin is uniformly different than the background. Another advantage is that the algorithm yields multiple hypotheses. Rather than make a critical decision as to which pixels belong to the face and which to the background, our method may classify multiple overlapping regions as stable segmentations. This can be of great benefit in situations involving partial shadows such as in Figure 4.

Although this algorithm successfully finds faces, it also yields many non-faces as well. Thresholds on region size and improbable intensities can greatly reduce the number of facial candidates, but to obtain better results, the region must pass a more sophisticated verification step.

As an example, we created a real-time face pose estimation system for driver awareness. As shown in Figure 3, we fit a square to each candidate region, and then scale it to a 32x32 pixel square. We perform recognition and pose estimation on each region by extracting a SIFT feature [7], and matching it to its nearest neighbor from a set of training views. If the euclidian distance between the new feature and the template is below a threshold, the system identifies the region as a face with the pose of the template. Since there are typically many overlapping regions, there can be multiple hypotheses in a single frame. To provide a single estimate, we use the median consensus over 5 frames to choose the head direction. This system is able to run at 10fps on a 3 Ghz personal computer.

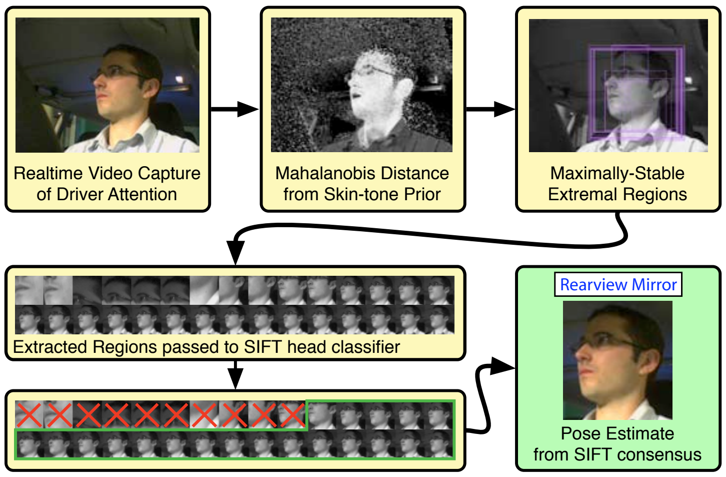

Realtime Video Capture of Driver Attention
Mahalanobis Distance from Skin-tone Prior
Maximally-Stable Extremal Regions
Extracted Regions passed to SIFT head classifier
Rearview Mirror
Pose Estimate from SIFT consensus

Figure 3: Overview of Pose Estimation with MSER + NDS-Forest

5 Conclusions

In this paper we introduced a new data structure which can be used to find and characterize Maximally Stable Extremal Regions. The system requires no heap memory allocation at runtime and attains real-time performance in streaming video. It attains the lowest possible bound on efficiency for the disjoint-set problem as well as the lowest possible bound on region traversal. In doing so, the data structure extends the original MSER formulation since it explicitly provides a method to traverse all of the detected regions.

In addition, we present a system for face registration and pose estimation which applies the MSER algorithm to a Mahalanobis distance image rather than a standard grayscale image. This enables the MSER algorithm to function as a robust skin-tone segmentation algorithm, and we have exploited this property by detecting a driver's head pose in an automobile. In this context, the NDS-Forest is particularly useful since fast driver intent analysis can be the crucial step in preventing an accident.

In future work we would like to exploit the shape information of each region as well as the texture inside it. For head pose estimation, shape can provide complementary information which could boost performance. We also would like to investigate the utility of MSERs for automobile lane detection and sign detection. Figure 6 shows an example segmentation result of the MSER + NDS-Forest algorithm applied to a freeway scene.

6 Acknowledgments

Support for this work was graciously provided by Volkswagen of America, Inc. and the U.C. Discovery program. We would like to thank Joel McCall and the rest of our colleagues in the Computer Vision and Robotics Research laboratory for their helpful suggestions and assistance.

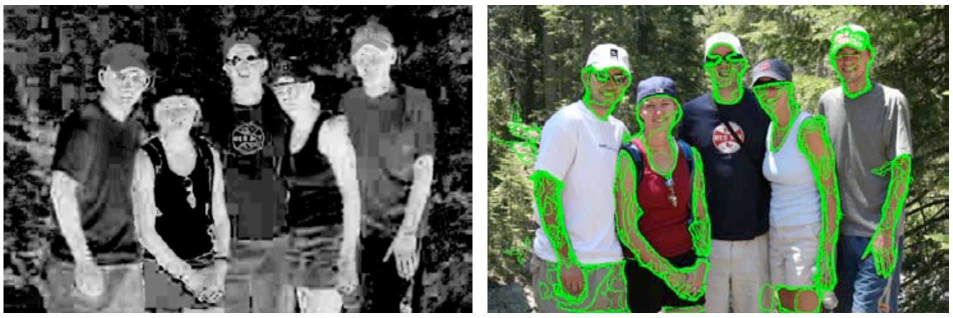
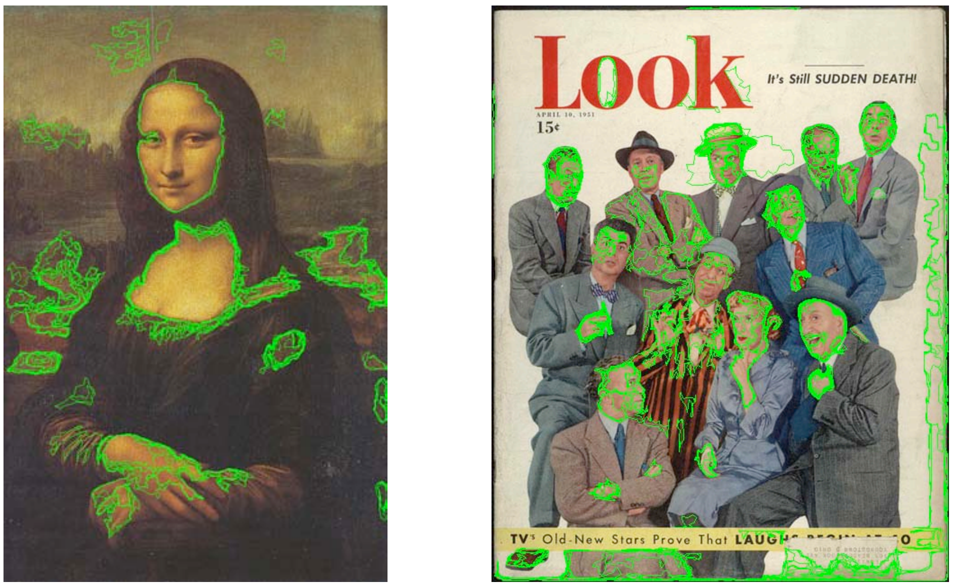

Mahalanobis input MSER boundaries

Figure 4: Face registration with JPEG compression artifacts and partially-shadowed faces.



MSER boundaries MSER boundaries

Figure 5: Registration of synthetic faces.

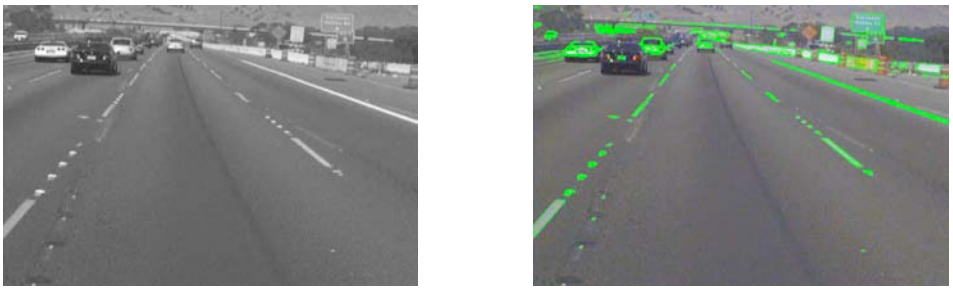

Grayscale input MSER boundaries

Figure 6: MSERs can be used for traffic lane and traffic sign detection.

References

[1] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Scene classification via pLSA. In *Proc. European Conf. Computer Vision*, 2006.

[2] Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill Book Co., second edition, 2001.

[3] Zvi Galil. Data structures and algorithms for disjoint set union problems. *ACM Computing Surveys*, 23(3):320–344, 1991.

[4] Timor Kadir, Andrew Zisserman, and Michael Brady. An affine invariant salient region detector. In *Proc. European Conf. Computer Vision*, May 2004.

[5] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using affine-invariant regions. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 319–326, 2003.

[6] Tony Lindeberg and Jonas Garding. Shape from texture from a multi-scale perspective. In *Proc. Int'l. Conf. Computer Vision*, 1993.

[7] David Lowe. Distinctive image features from scale-invariant keypoints. *Int'l J. Computer Vision*, 60(2):91–110, 2004.

[8] Jiří Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. British Machine Vision Conference*, 2002.

[9] Krystian Mikolajczyk, Bastian Leibe, and Bernt Schiele. Local features for object class recognition. In *Proc. Int'l. Conf. Computer Vision*, pages 1792–1799, 2005.

[10] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *Int'l. J. Computer Vision*, 60(1):63–86, 2004.

[11] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[12] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiří Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *Int'l. J. Computer Vision*, 65(1/2):43–72, 2005.

[13] Frederik Schaffalitzky and Andrew Zisserman. Viewpoint invariant scene retrieval using textured regions. In *Proc. Dagstuhl Seminar Content-based Image and Video Retrieval*, pages 11–24, 2004.

[14] Josef Sivic, Bryan Russell, Alexei Efros, Andrew Zisserman, and William Freeman. Discovering object categories in image collections. In *Proc. Int'l. Conf. Computer Vision*, 2005.

[15] Josef Sivic and Andrew Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. Int'l. Conf. Computer Vision*, volume 2, pages 1470–1477, October 2003.

[16] Robert Tarjan. Worst-cases analysis of set union algorithms. *J. ACM*, 31(2):245–281, 1984.

[17] Tinne Tuytelaars and Luc Van Gool. Matching widely separated views based on affine invariant regions. *Int'l. J. Computer Vision*, 59(1):61–85, 2004.

[18] Štěpán Obdržálek and Jiří Matas. Object recognition using local affine frames on distinguished regions. In *Proc. British Machine Vision Conference*, 2002.

[19] Štěpán Obdržálek and Jiří Matas. Sub-linear indexing for large scale object recognition. In *Proc. British Machine Vision Conference*, 2005.