Introduction

Many systems in computer vision end up trying to emulate some function that the human perceptual system either for its efficiency or by a natural tendency for humans to solve problems based on how we perceive them. Fundamental in this field is the concept of key points and features. Key points in this context are loosely defined to be interesting patches of pixels that can be used to identify features in an image, this can take the form of local maxima and minima such as in the work of Lowe (2004), or more geometric structures such as corners like are used in the FAST feature detector by Rosten (2010). In this study, key points are used as a technical term indicating image coordinates for which a feature has been detected. As Wikipedia defines it: “a feature is a piece of information which is relevant for solving the computational task related to a certain application.” The word feature can also be loosely related to saliency, as often an engineer may wish to define features in an image based on their own intuition first while formulating their own algorithms. Such as in Lowe’s Scale Invariant Feature Transform (SIFT), features are further grouped into descriptors, a means for describing the interrelation of related features for matching larger objects of interest, whose features are often rigid or geometric properties such as the corners of a hard cover book. In the SIFT algorithm, many nice properties are achieved such as scale and transform invariance, or the property that a descriptor and its key points can be matched given some change in viewpoint of the imaging device.

The importance of such a transform is pivotal in the realm of digital image processing, it forms the basis of the majority of higher-level work, such as object detection in autonomous robots and face matching on smart phones. One such application is known in the industry as stereo correspondence, which can be applied to object detection and 3D scene detection and replication, and novel entertainment applications such as augmented reality. Stereo correspondence may be the problem of determining the spatial variation in objects between two related images, or by reversing that dependency, it can also be used as a basis for detecting objects using inferences about how two images are related. In the former, augmented reality depends heavily on determining the depth of a scene, which humans perceive via the natural stereo correspondence of our binocular vision. To augment what a user sees by creating an image sprite, the developer must know how to translate the sprite in the two images applied to each eye to give it the appropriate depth, or how to occlude the sprite based on objects that may be in front of it. This study will first consider two methods for detecting features and descriptors. First, the merits of the SIFT algorithm for matching descriptors between two images. Then, two algorithms aimed at improving performance of feature detection and descriptor formation on resource constrained platforms are compared to the results obtained from SIFT. The results will be compared using well known metrics provided by Scharstien et al. known as the Middlebury tests and data sets. Finally, the results will be analyzed for the merits of each algorithm considering the tradeoffs made by their authors.

Detection Algorithms

The task of detecting key points, or features can often be separated from the process of defining descriptors. In the canonical SIFT algorithm, key points are detected by computing various gaussian blurred images over varying scales and blur-factors to produce Difference of Gaussian (DoG) images as the base processing space in a an extrema detection scheme. The minima and maxima are found by comparing a cube of neighbor pixels formed by stacking adjacent DoG images within and without the scale space, and an extremum is detected at the center of the cube if the center pixel has the largest value in the cube. SIFT incorporates a hand full of very well composed methods for rejecting false detection, which are outside the scope necessary to understand this study. Finally, a key point is formed by computing an orientation and magnitude for the selected extrema, which for a single pixel is calculated like so:

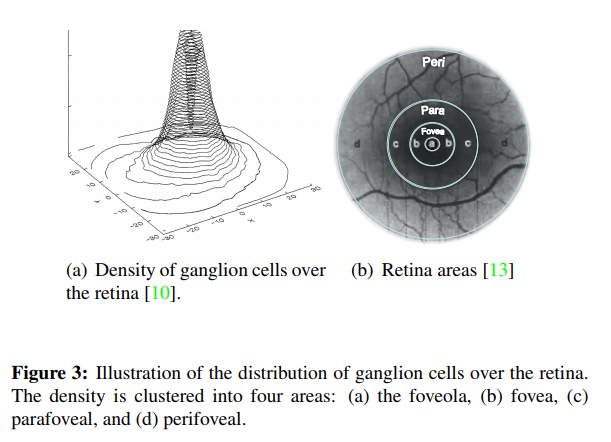
A histogram of orientations is then computed covering a 360-degree window around the point of weighted gradients at each bin orientation. The key point’s orientation is select as the largest peak at the point, while more key points will be generated at the point if more peaks are detected within 80% of the original key point’s magnitude.

The FAST feature detector based on two successive works by Rosten et al. is presented as a faster key point detector solution. As opposed to local point extrema detected with SIFT, the FAST detector instead focuses on corner detection, arguably a simpler problem which leaves room for speed optimization, such as their conclusion that processing PAL video required a fraction of their processing resources. Based primarily on machine learning techniques, Rosten et al. claim that their key point detection benefits from removing human intuition from the equation, relying instead on trained models and learning algorithms. As a comparison, the authors present pixel rates for the Gaussian and the FAST algorithms where the FAST performed a more than 180 megapixels per second faster than a DoG solution, and while only requiring an estimated 5% of their processing power on a 3.0GHz Pentium based Windows PC processing PAL video.

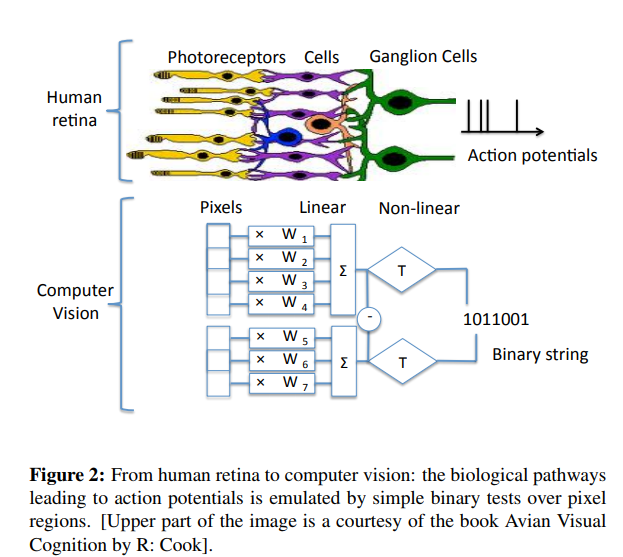
The primary detection test is implemented in FAST as a segmentation test which immediately produces resource efficiency gains of DoG in that only one set of pixels is tested, while sacrificing some generality. Instead of detecting local extrema, the algorithm is looking in a circle around a point of interest for a contiguous line of pixels with either intensity greater or less than the pixel of interest, above or below a certain threshold respectively. The type of edges detected can be adjusted by the parameter which is the minimum number of pixels that must match in a circle of 16 pixels formed around the interest point, which can also significantly affect the performance of the algorithm. The algorithm also saves some time by intelligently checking only as many pixels as it needs to determine if contiguity meeting the criteria is possible. The authors note that weaknesses include inability to reject false features for and adjacent features are often detected in excess when one feature could have been detected representing a few of the ones found using FAST.

The machine learning used in FAST comes from the learned decision tree using the ID3 generator. The learning is based on optimizing the entropy of measuring whether a point is detectable given a pixel in its circular search space. The learned tree can then be pruned for redundancy. The final tree can detect all the same corners as the original FAST algorithm, given the same training set. However, the learned decision tree can never precisely detect all the corners that the FAST algorithm will in practice since all possible corners cannot be contained in a finite training set. Other formulations of the FAST algorithm known as the FAST-*n* family of detectors are theoretically capable of detecting the same set of corners but take a slightly different tact in pixel detection. Later iterations of the FAST algorithm also consider a band around the detection circle, make improved cost functions, and choose different learning methods also based on decision trees, in addition to special case handling for point noise and other imperfections.

For descriptor generation, the Fast Retina Keypoint algorithm is presented for comparison to SIFT’s build in descriptor generation. Once they keypoint has been detected with FAST, FREAK descriptors are computed using a “retinal” sampling patter which mimics the human eye. For a given circular region the algorithm uses sampled intensities from points in concentric rings, where each ring moving inwards is blurred with a Gaussian function of decreasing standard deviation. The effect mimics the retina by its nature of having denser sampling points near the center, with higher sensitivity (due to smaller sigma).



Each point is treated like a photo receptor cell, where multiple cells feed into an integrator (the ganglion cell) and produces information in strings. Consequently, this will have implementation impact in a later section, our matcher will be picked to better match binary strings. More on this later.



# References

|  |  |
| --- | --- |
| [1] | E. Rosten and D. T. , "Machine Learning for Hight-Speed Corner Detection," *Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science,* vol. 3951, 2006. |
| [2] | H. Moravec, "Rover visual ostacle avoidence.," *International Joint Conference on Artificial Intelligence,* pp. 785-790, 1981. |
| [3] | D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *IJCV,* vol. 60, no. 2, p. 91–110, 2004. |
| [4] | E. Rosten, "Faster and Better: A Machine Learning Approach to Corner Detection," *IEEE Transacrtions on Pattern Analysis and Machine Intelligence,* vol. 32, no. 1, pp. 105-119, January 2010. |
| [5] | D. Scharstein, R. Szeliski and H. Hirschmuller, ""Stereo"," 2001. [Online]. Available: http://vision.middlebury.edu/stereo/. |
| [6] | D. Scharstien, R. Szeliski and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, Kauai, HI, 2001. |