Optimizing local feature description and matching for realtime video sequence object detection

Katsiaryna Halavataya, Vasili Sadov
Belarussian State University
Minsk, Belarus
katerina-golovataya@yandex.ru, sadov@bsu.by

***Abstract*—The paper proposes an algorithm for local feature extraction, description and comparison on color images for semantic video sequence processing. One of the main problems in implementation of such an algorithm is its ability to work within realtime constraints. Asymptotic computational complexity for proposed algorithm is determined and local performance optimizations are introduced in order to enhance processing time. The optimized algorithm is able to compute local feature vectors and compare them across video frames in realtime, which simplifies further semantic analysis.**

***Keywords*—image processing, image feature extraction, computer vision, algorithm optimization, realtime object detection**

I. Introduction

Local image feature extraction, description and matching algorithms are the main building blocks of a wide range of image processing techniques for computer vision and visual semantic analysis problems, including image classification, object detection, bundle adjustment, object tracking and visual flow, 3D scene reconstruction, shape and texture extraction, and many others [1], [2].

The main problem that feature extraction aims to solve is that image pixel brightness values, per se, are poorly suited as basis for direct semantic analysis – by itself, a single pixel conveys very little information about the actual contents of the image. When used as a feature, the pixel value states that this particular point of the image has a concrete color value, but the implication of a specific visual cue cannot be reversed – for instance, a stop sign appearing in the upper left corner of the image implies that one of the pixels will be red, but the inverse is not true – one single red pixel in the upper left corner doesn't mean that this part of the image contains a stop sign. In essence, this example illustrates how raster image representation is completely different from the actual semantic representation of the objects that can be used as a basis for an automated decision-making processes [3].

The premise of feature extraction is the fact that actual semantic analysis based on visual representation is done based not by individual color values, but rather by meaningful higher-order features and unique characteristics associated with objects represented on the image. For instance, one of the defining semantic features of stop sign is the octagonal shape. For a proper semantic analysis it is required to create a concrete measure that correlates with how much similar to an octagon a particular group of pixels is. It is obvious that this is very hard when using raw pixel values, because the particular set of transforms and combinations required to create an appropriate measure can vary greatly from image to image.

This limitation means that raw image data (i.e. pixel brightness values or CCD/CMOS image sensor jot charges) is very hard to analyze in the context of various classification and object detection problems with classical approaches, like naive Bayesian classifiers or regular feedforward neural networks [3], [4]. For instance, when using a single pixel value as an input feature for an artificial neural network-based classifier, it is usually hard to measure how much this pixel should actually contribute to the output both during training and during classification, no matter the weights assigned to them.

Image feature extraction is a set of methods for transforming raw image data (pixel brightness values) into an alternate representation that is, in turn, better suited as a basis for semantic analysis in the context of a particular domain problem. Feature extraction as a process can be accomplished by the combination of feature detector and feature descriptor algorithms [5]–[7].

II. Image feature detectors and desecriptors

Image feature detector $d(I)$ (also called feature extractor or keypoint detector) is an algorithm that, given an image, produces a set of coordinates of feature points (or keypoints) of this image:

$$d(I) : \forall w, h \in \mathbb{N} | C^{(w \times h)} \to \{(i, j)\}, i \in \overline{1, w}, j \in \overline{1, h}, \quad (1)$$

where I is the input image, w, h – integers representing image width and height, $\mathbb{N}$ – natural numbers set, C – color depth (set of all possible color values), $C^{(w \times h)}$ – set of all images. More formally, feature detector can be represented as a predicate:

$$\forall w, h \in \mathbb{N} | d_p(i, j, I) : (\overline{1, w}) \times (\overline{1, h}) \times C^{(w \times h)} \to \mathbb{B}, \quad (2)$$

where $\mathbb{B} = \{false; true\}$ – a Boolean domain. That is, a feature detector can be implemented as a function that, for any point (i, j) of an image I with color depth C, determines whether this point is, in fact, a keypoint ($d_p(i, j, I) = true$) or not ($d_p(i, j, I) = false$). Based on this, the set of keypoints in (1) can be determined using (2) by applying a predicate over every coordinate set:

$$d(I) = \{(i, j) : d_p(i, j, I) = true | i \in \overline{1, w}, j \in \overline{1, h}\}. \quad (3)$$

A feature descriptor f is a projection of any point p^I_{ij} of any image I to an n-dimensional metric vector space F:

$$f(p^I_{ij}) = \vec{v}^I_{ij} \in F \tag{4}$$

Since vector space F is also a metric space, an appropriate metric is defined on it:

$$m : F \times F \to \mathbb{R} \tag{5}$$

Two arbitary points, p^{I_1} of image I_1 and p^{I_2} of image I_2, are considered similar by a feature descriptor (4) if their feature vectors $\vec{v}_1 = f(p^{I_1})$ and $\vec{v}_2 = f(p^{I_2})$ are similar by measure of metric (5), or $m(\vec{v}_1, \vec{v}_2) < t$. The threshold t is selected based on a specific descriptor implementation.

While there is no concrete definition as to what exactly makes an image point a feature, there is a number of desireable properties for good feature detectors and descriptors. Specifically, feature detectors and descriptors must be invariant (to a certain degree) to displacement, linear scale-space transforms like rotation, shift, skew, etc., and to some of the common non-linear transforms like perspective shifts and distortion [5].

Detector transform invariance means that any point of the image that was classified as a keypoint before the transform must continue being recognized as a keypoint after it. This is mainly achieved by analyzing a certain surrounding area of a keypoint in rotationally-invariant manner (i.e. circular traversal) and on different levels of subsampling.

Descriptor transform invariance means that descriptor vectors of the same point before and after transform must differ, as calculated by metric m, by a small margin. This is achieved by including only spatially-invariant information into the descriptor and making metric calculation process circularly agnostic.

III. Feature detector and descriptor based on midpoint circle traversal

There are many known approaches to both feature detection and description. Some of them treat detection and description as two disjoint operations, while others can build a keypoint descriptor as a by-product of determining whether the point should be treated as keypoint or not [5]–[7].

One of the main disadvantages of most keypoint detection and description algorithms is their rigidity and inability to adapt in terms of input parameters. Known methods usually have a parametric threshold that should be predetermined before feature extraction is carried out, generating spurious point clouds on one image and hardly detecting any on the others if the threshold is off. Moreover, these methods tend to be over-reliant on sharp brightness shifts, while in reality such shifts mostly occur as artifacts (like sharp light reflections or optics chromatic abberations). Finally, most of the known methods are poorly suited for real-time usages. While certain keypoint descriptors, like FAST, are designed to carry out keypoint detection in the most efficient manner, they commonly forego the usage of information used to classify a point to further use it in keypoint description, requiring more complex and computationally expensive keypoint descriptor algorithms like ORB.

The proposed method is designed to be adaptive and able to perform keypoint detection and description in with a single pass.

Input of the algorithm is an image I of width w and height h with 3 color channels (RGB colorspace).

The first step is the selection of image traversal step. A common way to do it is to apply a logarithmic scale on an input image size:

$$\begin{cases} s_g \in \mathbb{N} \\ s_g \propto log(w \cdot h) \end{cases} \tag{6}$$

This allows to reduce the number of exess points in proximity of a larger feature and makes the algorithm less susceptible to scale transforms.

After that, a one-channel transform is applied to the image. While it is possible to use common grayscale transforms like YUV colorspace Y-component formula, doing so usually removes too much of color distribution from the image. A proposed solution is to use Principal Component Analysis (PCA) while treating image color values on 3-component representation (red, green and blue channels, respectively) as input features in 3-dimensional feature space. Applied to image colorspace, PCA will generate a new basis with 3 pseudocolor directions. First direction corresponds to the largest variance of the image, which is, for most of the real-world images, a shift from darker to brighter colors – that is, first component of the new colorspace in PCA is usually very close to Y direction of YUV colorspace. Based on actual variance numbers, however, a second component might be preferable, the one corresponding to the second-largest variance distribution of colorspace. This is usually a shift from one predominant color of the image to the other. In some specific areas, like medical image processing, color hue may provide more destinguishable feature space compared to plain grayscale brightness. As a general rule, if the variance of the second component is above 20% of the total image variance, a second PCA component should be considered for feature extraction. It is important to note, however, that, when comparing feature descriptors across images, descriptors based off first and second components tend to differ significantly, so the same component should be used on both images.

After the step is chosen and image is transformed to grayscale, a set of traversal radiuses $\{r_i\}$ must be chosen. A common choise is $\{3, 5, 7\}$ pixels, with 9 pixels added for hevily distorted images. The maximum radius $r_{max} = \max_i r_i$ defines a padding for image traversai, that is an image is traversed from r_{max} to $w - r_{max}$ horizontally and from r_{max} to $h - r_{max}$ vertically with the step s_g.

For every point (i, j) traversed, a Bresenham (midpoint) algorithm is used to construct a circle around the point at every specified radius $\{r_i\}$. The result for radius r_k is an ordered set of point coordinates $\{p_n^{r_k}\}$. Before the circle traversal, a starting point is determined as a point where brightness values change compared to the next pixel is the greatest across the entire circle. This is done to mitigate the influance starting point has when comparing features before and after rotation transforms.

The goal of Bresenham circle traversal is to determine segment configuration. The first pixel is assumed to belong to the first segment: $p_1^{(r_k)} \in S_0$. After that, every next pixel brightness of the circle in order is compared to the brightness of the first pixel of a current segment. When the brightness exceeds a certain threshold t_{seg}, the pixel is considered a beginning of a new segment. The process iteratively continues until the point circle is confined:

$$\Delta P_n^{(j)} = |I(p_n^{r_k}) - I(S_0^{(j-1)})| \tag{7}$$

$$\begin{cases} P_n \in S^{(j)} & \text{if } \Delta P_n^{(j)} \geq t_{seg} \\ P_n \in S^{(j-1)} & \text{if } \Delta P_n^{(j)} < t_{seg} \end{cases} \tag{8}$$

Brightness difference (7) across two neighbour points in Bresenham circle is used in (8) to determine the beginning of a new segment. Artificial segment edge introduced by the first pixel is mostly mitigated by correct selection of the first pixel.

The segment configuration, i.e. the number of segments and their length, constitute a feature of a certain point. Variable-length natural-numbered segment span vector can be used as a feature vector of a certain point. It is also possible to short-circut descriptor evalutation if a point should not be considered keypoint, in case the number of segments or their respective differential shift is too small on any of the radiuses.

To compare the features, segment configuration similarity measure must be created. This process is complicated by the fact that segment span vectors have variable length, so traditional similarity measures like Euclidean distance cannot be applied to them. One way to solve this problem is to introduce a simple binary representation of a feature vector. For this descriptor, each segment of a specific length constitutes a span of two-bit numbers of the same length, and each segment edge changes the segment two-bit representation with the following circular pattern: 00 $\rightarrow$ 01 $\rightarrow$ 11 $\rightarrow$ 10 $\rightarrow$ 00. For example, for an 12-point circle (radius 3) with segments of length 3, 7, 3, the binary representation will be 24-bit string – 00 00 00 01 01 01 01 01 01 01 11 11 11. The specific pattern is chosen in such a way that Hamming distance (the number of differing bits) increases by 1 for each two-bit marker. To illustrate this, let's consider a 12-point circle with segments of length 3, 6, 1, 3. Its 24-bit descriptor will be the following: 00 00 00 01 01 01 01 01 01 11 10 10 10. The difference beetween the previous 3, 7, 3 segment is the introduction of another small segment, which means that configurations remain relatively similar. If a 1-bit representation was used, an extra segment would invert the rest of the string, resulting in a completely different 12-bit descriptor with large Hamming distance. With 2 bits per segment span, only "half" is inverted. In our example, Hamming distance between feature vectors is 4.

The set of keypoints and their respective descriptors for each radius can be used for further processing in template-based object detection algorithms and 3D reconstruction tasks.

IV. Realtime optimization

Realtime video sequence processing puts a time constraint on feature detector and descriptor performance.

Generally accepted framerate for video sequence image rendering is 24fps (frames per second). That means that it should take around 40ms for an entire frame to render. If the information can be presented as a HUD (heads-up display) overlay over existing video sequence with visual cues (as is the case with object detection and localization, for instance), it is generally acceptable to have HUD re-render at lower framerates, with times of up to 100ms per frame. Still, this constraint is quite severe in terms of computational complexity of algorithms involved in common computer vision tasks. For instance, a well-calibrated feature-based binary classifier needs as much as 5ms to process a single sliding window frame, and needs to run the classification up to 10-15 sliding window positions per frame to correctly localize an object. In practice, this means that, independently of further processing and semantic analysis, feature detector and descriptor should produce extracted feature vectors within 20ms for acceptable performance that can still be percieved as realtime [4], [8], [9].

It's easy to illustrate how the feature extraction procedure (7) - (8) has a linear asymptotic computational complexity. PCA transform requires mean and standard deviation calculation, which require a full image pass and have the complexity of $O(n)$ each. The transform itself is a $O(p^3)$ complexity problem, where p is the number of input features: for color images, $p = 3 = const$, i.e. it doesn't depend on the input image size. Detection requires a full traversal of an entire image, which is a linear $O(n)$ complexity compound problem. For each pixel, a constant number of operations must be performed to construct a descriptor, so the complexity of single point analysis is $O(n_r)$ where $n_r = const$ is the number of radiuses used for descriptor generation. The resulting complexity is, in terms of big-O notation, is $O(2n + p^3 + n_r n) = O(n)$.

The nature of feature extraction process means that it's not possible to further reduce asymptotic complexity, so the optimizations must be performed during key stages of the algorithm.

PCA computation is generally quite costly, since it requires the calculation of eigenvectors and eigenvalues of color feature matrix. However, when working with a video sequence, it can be safely assumed that PCA transform basis will be quite similar as long as the scene doesn't change drastically. In practice, it means that the same PCA transform can be used across the series of similar frames, for times up to several seconds of video sequence. So the first optimization is to skip PCA computation and instead use previous transform matrices across several frames. It is also possible to perform this computation in parallel to the main data stream on a single frame in isolation, and replace the current transform once PCA is calculated, further reducing frame processing time.

Detection itself requires a full image traversal at the minimum [10]. The traversal step grid introduced in (6) means that a portion of image's pixels is skipped already. To further increase processing performance, it's possible to statistically analyze the keypoint detection distribution. With adaptive threshold value the total number of keypoints on the image takes up to 10% of all the image pixels, which means that bail-out optimizations might actually increase processing speed. Bail-out optimizations are a way to short-circut the evaluation in case of a negative answer by performing a short check. In case of Bresenham circle descriptor, one common way to discern a keypoint is to select only those points where number of segments is large enough. It is possible to traverse the smallest radius value only, and mark the pixel as non-keypoint early if it doesn't have enough segments, thus avoiding additional traversals on larger raduises. Since this happens for most of the image pixels (around 90%), this optimization provides a noticeable performance boost.

V. Results and conclusion

The proposed algorithm allows for efficient feature extraction from color images. It is able to both determine a set of keypoints on the image and calculate their respective descriptor values in one pass, while also providing a bit string descriptors values that can be compared very fast using Hamming distance across several frames.

The performance optimizations have been evaluated for a processing task of feature extraction in realtime over 1 minute of FullHD video sequence. The results are presented in table I.

As can be seen from the results, the goal of realtime performance optimization of midpoint feature extraction algorithm was successfully met – the framerate constraint of 20ms per frame for HUD processing tasks was fulfilled using Midpoint circle traversal algorithm with bail-out optimization and staggered PCA computation across 24 frames (basis recalculated every second). For comparison, FAST feature detector and ORB feature descriptor performance was also considered. While both of those algorithms have linear asymptotic complexity, the information used in FAST for keypoint detection is not used for descriptor generation, as ORB is a separate algorithm; thus, the algorithms require twice the amount of image traversals, which impacts performance on larger images, like FullHD-scale image used to evaluate the performance.

Table I
MEAN FRAME PROCESSING TIME t FOR FEATURE EXTRACTION ON FULLHD (1920×1080) 24FPS VIDEO SEQUENCE OVER 1 MINUTE (1440 FRAMES TOTAL)

Algorithm	t, ms
FAST & ORB	27.35
Midpoint without optimizations	35.02
Midpoint with staggered PCA	28.94
Midpoint with bail-out	23.14
Midpoint with bail-out & staggered PCA	19.41

References

[1] R. S. Choraś Image Feature Extraction Techniques and Their Applications for CBIR and Biometrics Systems. *International Journal of Biology and Biomedical Engineering*, 2007, vol. 1, no. 1, pp. 6-16.

[2] L. G. Shapiro, G .C. Stockman *Computer Vision*, New Jersey, Prentice-Hall,2001. 608 p.

[3] Z. Shi, S. Vadera, A. Aamodt Image Semantic Analysis and Understanding. *IFIP Advances in Information and Communication Technology*, 2010, vol. 340, pp. 4-5.

[4] L. Qin, L. Kang Application of Video Scene Semantic Recognition Technology in Smart Video. *Technical Gazette*, 2018, vol. 25, no. 5, pp. 1429-1436.

[5] D. Lowe, Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004, vol. 60, no. 2, pp. 91–110.

[6] K. M. Yi, E. Trulls, V. Lepetit, P. Fua Lift: Learned invariant feature transform. *European Conference on Computer Vision*, 2016, pp. 467-483.

[7] E. Rublee, V. Rabaud, K. Konolige, G. Bradski ORB: an efficient alternative to SIFT or SURF. *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2564-2571.

[8] S. Baier, Y. Ma, V. Tresp Improving Information Extraction from Images with Learned Semantic Models. *27-th International Joint Conference on Artificial Intelligence (IJCAI-18)*, 2018, pp. 5214-5218.

[9] I. Simon, N. Snavely, S. M. Seitz Scene summarization for online image collections. *IEEE International Conference on Computer Vision (ICCV)*, 2007, pp. 1-8.

[10] R. Girshick, J. Donahue, T. Darrell, J. Malik Rich feature hierarchies for accurate object detection and semantic segmentation. *IEEE conference on computer vision and pattern recognition (CVPR)*, 2014, pp. 580-587.

ОПТИМИЗАЦИЯ АЛГОРИТМОВ ОПИСАНИЯ И СРАВНЕНИЯ ЛОКАЛЬНЫХ ПРИЗНАКОВ ИЗОБРАЖЕНИЙ ПРИ ДЕТЕКТИРОВАНИИ ОБЪЕКТОВ НА ВИДЕОПОСЛЕДОВАТЕЛЬНОСТЯХ В РЕАЛЬНОМ ВРЕМЕНИ

Головатая Е.А., Садов В.С.

В работе предложен алгоритм извлечения, описания и сравнения локальных признаков цветных изображений для семантической обработки видеопоследовательностей. Одна из решаемых задач - поддержка работы алгоритма в реальном времени. Для оценки и улучшения производительности проведён анализ асимптотической вычислительной сложности, а также предложены оптимизации. Оптимизированный алгоритм позволяет вычислять вектора локальных признаков и сравнивать их между кадрами, а также может служить основой для дальнейшего семантического анализа.

Received 09.01.19