Global Orientation for Feature Matching in Corresponding Planes

Thomas Scott, Stefan Mellem, Ian Zewiske
Department of Mathematics, Statistics, and Computer Science
St. Olaf College
Northfield, MN 55057
scottt@stolaf.edu, mellems@stolaf.edu, zewiske@stolaf.edu

Abstract

In this work we examine here a new algorithm for the selection, description, and matching of feature points between planar surfaces in multiple images for the purpose of three-dimensional model construction. While feature matching is a rich area in computer vision research, well-known algorithms such as SIFT and SURF are general in nature [1] [3]. Our algorithm is an extension of the OpenSURF algorithm in which we take advantage of the planar nature of the regions of interest via two avenues: global feature orientation and partial feature descriptors. Preliminary evaluation of these modifications indicates improvement over the base OpenSURF algorithm for some otherwise feature-poor tiles, suggesting that these avenues are ripe for future research.

1 Background

The algorithm is motivated by a tile-based approach to 3D model generation in which 3D objects are modeled as polyhedrons, with each planar or approximately planar face of the object a single polygon. We refer to each planar polygon in three-dimensional space as a tile, and the projection of a tile into a two-dimensional image as a contour. Similarly, a corner is a point of interest in three-dimensional space, while a feature point is its corresponding projection into the two-dimensional image.

The pipeline of our algorithm is as follows: First, we identify feature points in the input image that lie within the input contour. Then we generate a global orientation angle for the contour, relying on the assumption that feature points within the tile cannot be transformed independently of each other from one image to the next. The orientation angle can then be used to create an oriented feature descriptor, which includes only that context of the feature point which also lies within the contour. Finally, these oriented partial features are compared and matched to features identified in other contours for the given tile.

2 Integral Images and Non-rectangular Contours

One key aspect of our algorithm is the adaptation of integral images for use on nonrectangular contours. Integral images are a well-known and incredibly powerful optimization for summing the values of all pixels within an arbitrary rectangle [1]. The basic concept is simple: in an integral image, each pixels intensity value is set to the sum of the intensity values of all pixels in the base image which are above and to the left of it, inclusive. That is, the integral image I_Σ can be described in terms of its base image I by:

$$I_\Sigma(x, y) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(x, y).$$

Using the integral image, it is possible to calculate the sum of all the intensity values within any rectangle in the base image with just four operations, as illustrated below.

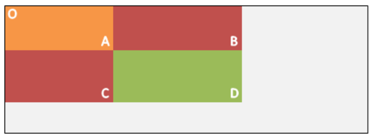


Fig. 1 The box integral of rectangle ABCD can be calculated via: $I_\Sigma(D) - I_\Sigma(B) - I_\Sigma(C) + I_\Sigma(A)$.

When dealing with a specific contour within an image, however, the edges of our contour will seldom be straight lines. In such cases, we use two integral images in parallel rather than a single integral image. The first of our two images is generated from a base image which contains the contour of interest, but in which all pixels outside the contour are set to zero intensity. The second corresponds to a base image with zeroes outside the contour, as before, but only ones inside the contour. We use these to calculate the sum of intensities within the intersection of a rectangle and contour and the total area of the intersection respectively. Thus, we can extract the average intensity per pixel of the intersection, taking advantage of integral images even when our contour does not fit nicely into rectangles.

3 Feature Point Selection

In the feature point selection stage, our algorithm has only slight modifications from the standard OpenSURF algorithm [1]. We begin with an image and a list of the pixels within the image that belong to a single planar contour, which allow us to generate the base images described above. In this stage, we use the contour image with non-contour pixel replaced by zeroes for choosing features.

After the image preprocessing, we identify feature points using the common Hessian matrix approach. The Hessian matrix and its determinant are given by [4]:

$$H(f(x,y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \qquad det(H) = \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2$$

Fig. **2** Hessian Matrix and its determinant. In order to calculate these partial derivatives on the discrete image, we use the pixel intensities in place of the function f and convolve them with the second-order scale-normalized Gaussian, just as performed by OpenSURF[1]. In practice, we use box filter approximations of these convolutions, which can be computed via integral images, to improve performance.

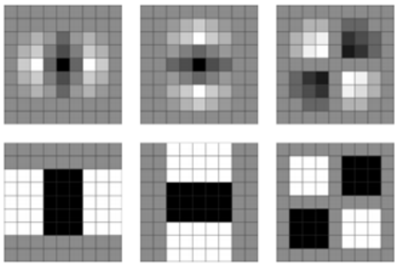


Fig. **3** Laplacian matrix of Gaussian box filters.

Since the box filter approximations $\mathbf{D}_{xx}, \mathbf{D}_{yy}$, and $\mathbf{D}_{xy}$ from left to right are not exactly equivalent to $\frac{\delta^2 f}{\delta x^2}$, $\frac{\delta^2 f}{\delta y^2}$, and $\frac{\delta^2 f}{\delta x \delta y}$ we use OpenSURFs correction to the determinant equation [1]:

$$\det(H) = \mathbf{D}_{xx}\mathbf{D}_{yy} - (0.81\mathbf{D}_{xy}).$$

In order to identify features, we calculate this approximate determinant at each pixel by centering the filters on that pixel and at each scale by simply rescaling the filter. Those points whose Hessian determinant is a local maximum (i.e. is greater in value than the 26 adjacent neighbors in x, y, and scale) are labeled as interest points.

Our modification to OpenSURF in this step is twofold. First, the box filters are calculated via contour-specific integral images, as discussed above. Second, we ignore points that are outside the bounds of the contour they are neither considered as potential maxima, nor as neighboring points in determining whether an in-contour pixel is a maximum. For box filters which are not entirely within the contour; the average intensity value of pixels within the contour is multiplied by the area of the box integral to produce the final value.

4 Global Orientation

The primary advantage yielded by our tile-based approach is that all pixels with a given contour exist on the same plane in three-dimensional space. As a result, we need not determine an orientation for each feature individually based on its local surroundings. Rather, we can be confident that a single global orientation for the contour will be sufficient. Since the entire tile consists of one static, real-life object, every point within the tile must undergo the same transformation between any pair of corresponding contours.

This being the case, we can think of the contour as being much more like a cohesive texture than a collection of independent features. We model our global orientation selection on the orientation routine from Xu et al's LHBP algorithm for texture description [5].

To generate a global orientation, we begin with our contour-only image with black pixels outside the contour. Following the approach of Xu et al, we apply a discrete Haar wavelet transform to this image using box filters. The transform can be computed for different scales by simply changing the number of pixels each Haar wavelet covers in the image.

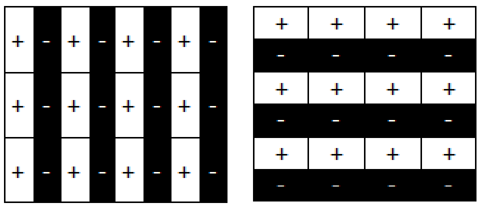
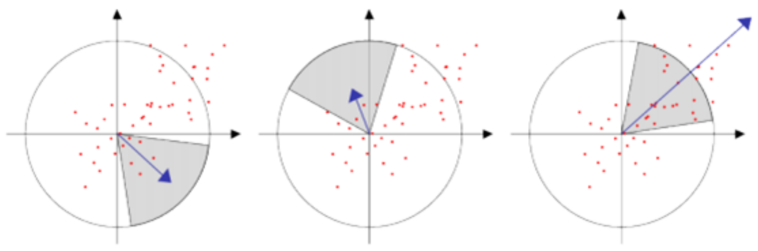


Fig. 4 Sets of 12 Haar wavelets arranged for application to an image to create HL and LH sub bands, corresponding to the x-oriented and y-oriented Haar wavelets respectively.)

The HL and LH high frequency sub-bands provide one ”x” value and one ”y” value for each individual wavelet. These coordinates, in turn, provide a magnitude (distance from the origin) and an angle (counterclockwise from the $+x$-axis). We choose a dominant orientation for each scale in the manner propose by Evans in OpenSURF and illustrated in Figure 5.



A window selects orientation whose covered Haar responses yield the greatest net vector (On right.) [1].

The total vectors for each scale of Haar transform are then summed together, and the angle from the $+x-$axis of this final vector is used as the global orientation for the contour.

5 Partial Feature Descriptors

In feature detection and comparison there are essentially two cases; the first case is texture rich contours or contours with high levels of contrasting detail. The second case is the exact opposite, namely contours with no powerful distinguishing features. "Classical" feature descriptors that rely on contrast, local minima or maxima, and chromatically varietous perform exceptionally well with the texture rich contours commonly examined in stereo vision literature. However, in drab expanses of open wall where colour is uniform and there is little to no physical texture the "classical" detectors waver significantly. It is common when examining case 2 regions with SIFT-like feature detectors to see a great deal of feature points aggregating along the edges of a contour where there is some dramatic change in coloration or texture. Unfortunately edges make flakey feature points because of the likelihood of occlusion, and even more dangerous, the uncertainty associated with translation of a descriptor along the edge.

In the feature descriptor stage, our algorithm once again modifies the base descriptor construction implemented in OpenSURF[1]. We first construct a square window around our feature point, oriented along the global orientation of the contour. This window is split into a 4×4 grid with each of the 16 cells split further into a 5×5 grid. On each of the 25 cells of each cell, we apply an x and a y Haar wavelet to generate the classic SURF descriptor, consisting of 16 cell descriptors with four values each.

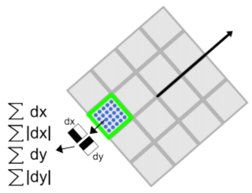


Fig. 5 The SURF feature descriptor. [1].

However, when it comes to contour boundaries, we have several options. We could ignore the boundary entirely by simply using the traditional SURF descriptor unmodified. In order to combat the influence of occluding objects, whose borders with the contour may change radically from one image to the next, we can run feature detection on the contour-only image with black pixels outside it, eliminating the possibility of making a poor feature match based on agreement with the texture of an occluding object. Finally, we can attempt to ignore pixels outside the contour entirely.

We propose a partial feature method as a solution for the final possibility. The difficulty of ignoring pixels outside the contour arises from the fact that some of the components of the SURF feature descriptor will be undefined – there simply won't be data for some of the cells if our feature is sufficiently near the edge of the contour.

Our use of a global orientation ensures that the missing information will not have undue influence on the descriptor's orientation. When we want to compare two partial descriptors, we can simply iterate over the components of the two -vectors and select for comparison only those for which both vectors have defined values. Lastly, we normalize only those components of the vectors which are being compared. By delaying normalization to this stage, we achieve invariance to contrast and produce differences that are comparable to those between full features.

6 Evaluation Methods

The ability to select invariant feature points is useful, but evaluation of feature matches is also critical. Arguably just as important as determining the most likely matches is culling those that are not good matches. The RANSAC algorithm published by Fischler and Bolles in 1981 provides a particularly effective way of discarding outliers and choosing a best fit plane from the consensus of the data points.[2] By randomly sampling subsets of the feature matches and scoring the number of other matches that agree with the plane within some margin of error, the algorithm rapidly and effectively identifies outlier points which might otherwise skew the calculation of a homography. RANSAC can in fact serve a dual purpose: it provides an extra layer of protection against incorrect feature matches while also offering a measure of how consistent a given feature matcher is by identifying how many outliers it produces. We propose here two further methods of comparing the output of feature matchers: a color-distance approach and a distance-between-projections approach. Standards set by the International Commission on Illumination have provided a strong foundation for comparing colors in a quantitative fashion. While color has not been historically popular as a tool in feature matching algorithms proper, it provides a potentially powerful evaluative measure. By implementing the CIEDE color difference metric and using it to measure the chromatic distance between the pixels in a contour and a corresponding contours projection via homography into the same plane, we can quantify the visible color disparities introduced by various homographies. Our second approach also relies on this idea of comparing homographies by comparing the way they project contours into other planes. In the distance-between-projections test, we project a contour onto another plane via two separate homographies generated by different feature-matching algorithms. We then measure quite simply the distances between the two reprojections of each individual point to find a quantitative measure of how closely the two homographies agree with each other.

7 Conclusion

While we have been unable to complete the extensive testing routines outlined above, more rudimentary comparisons between feature matching algorithms are promising. We examined four different variations of the OpenSURF feature detector: 1) "out of the box" OpenSURF with features restricted to the contour only,

2) "out of the box" OpenSURF run on a contour-only image (with black pixels outside the contour), 3) globally-oriented SURF run on the contour-only image, and 4) globally-oriented SURF with partial feature descriptors run on the contour-only image. Over a sample of tiles identified in a single hallway, we saw significant variance in the number of matches found by all of the varieties of SURF. Furthermore, the SURF varieties each seemed to do best on different tiles, which suggests that each modification provides benefits for different types of tiles. While our extensions did not prove to function strictly better than their ancestors, we believe that they introduce promising avenues for future feature matching research.

References

[1] C. Evans. Notes on the OpenSURF Library. The OpenSURF Computer Vision Library, January 2009.

[2] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography. Communications of the ACM, 24: 381-395, June 1981.

[3] D. Lowe. Object Recognition from Local Scale-Invariant Features. The Proceedings of the Seventh IEEE International Conference on Computer Vision, 2:1150-1157, September 1999.

[4] S. Seitz and R. Szeliski. Features and Image Matching. Lecture, University of Washington, Fall 2008.

[5] P. Xu, H. Yao, R. Ji, X. Sun, and X. Liu. A Rotation and Scale Invariant Texture Description Approach. Visual Communications and Image Processing, Proceedings of SPIE, Vol. 7744, 2010.

[6] Gaurav Sharma, Wencheng Wu, and Edul N. Dalal. The CIEDE2000 Color-Difference Formula: Implementation Notes, Supplementary Test Data, and Mathematical Observations. 28 January 2004.