Learning to detect low-level features

Peter Hall and Martin Owen
Department of Computer Science
University of Bath
Bath, BA2 7AY
pmh | cspmjo@cs.bath.ac.uk

Abstract

We introduce a method to detect low-level features that is prescriptive (as Canny edge-detection is) but trained by a user. The user simply chooses feature classes and points to class instances. Given a input image, we compute a probability map that indicates how likely it is to belong to each of the user-defined classes; so combining different kinds of feature detector into a single, user-trainable system. This paper explains how we characterise features, how we train and detect, and gives an algorithm that automatically determines feature scale. We empirically compare the method to standard edge and corner detectors, showing a measurable advantage in each case.

1 Introduction

Low-level feature detection has long been of major interest to Computer Vision. Attention has traditionally been directed toward developing *prescriptive* methods; that is, methods in which the algorithm's designer provides a prior definition of the feature to be detected. The literature is vast, but typical features/detectors include Canny edges [2], Harris corners [7], and (to a lesser extent) ridges [8]. Prescriptive systems have many advantages: they can be mathematically elegant, with useful analytic properties; they can be computationally efficient; and they are well known.

Yet a characteristic of prescriptive methods is that they tend to disagree with humans. For example, a typical edge-map will both admit and omit edges when compared to an edge-map produced by humans: in so far as line-drawings and edge-maps are analogous, edge-maps are rather poor line-drawings. There is good reason for the computer vision community to care about this. Line-drawings are almost never intended to faithfully reproduce the pattern of light on some surface (such as the retina). Instead, the purpose of line-drawings is to convey information to the viewer: line-drawings are salience-maps in the sense that important elements are highlighted, but unimportant detail is suppressed. It follows that the more closely a detector agrees with a human, the more "information efficient" it likely to be.

Of course, humans disagree with one another — each one of us may produce a different drawing of the same scene — but human/human disagreements are usually smaller than human/machine disagreements. Evidential support for this claim comes from Martin *et al.* [9], who plot a scatter-graph showing "recall" verses "precision" for a variety of edge detectors and many humans. (*Recall*: probability that a ground truth element is

BMVC 2004 doi:10.5244/C.18.36

detected, *Precision*: probability that a detected element is a ground truth element.) These graphs show humans cluster together, exhibiting high recall and high precision; they invariably out-perform all machine-based detections. Furthermore, the data suggest there is sufficient agreement between humans to make the notion of "average human" a meaningful one, indeed the authors plot a "median human" curve.

This paper introduces a method for low-level feature detection in which better agreement with humans is an explicit objective. Our system is a classifier that requires user training, and so is in line with recent work such as that of Martin *et al.* [9] who allow users to train a classifier to detect boundaries in natural images, the boundaries divide homogeneous regions in colour, texture, and brightness. The spirit of our approach is seen also in the work of Konishi *et al* [14], who provide a trainable system for statistical edge detection. We are unique in several ways:

- We provide a multi-classifier in which we allow users to specify the *type* of feature they wish to detect, such as "corner", "edge", or "ridge".
- We provide separate terms for *visibility* and *coherence*.
- We automatically adapt to feature scale.

Our feature detector is based on circular sampling. Smith and Brady have already argued in favour of circular sampling [1]. Krüger and Felsberg [4] use radial-polar coordinates to compute the "intrinsic dimension" of features, which enables them to classify features as flat, edge, or corner; their method is prescriptive, they make no provision for other possible feature types. Martin *et al.* [9] also use a circular windows; they make use of "oriented energy", but they do not classify the boundaries they detect; we base our characterisation on the Fourier transform and classify boundaries.

2 Learning and detecting low-level features

In this section we describe how our classifier is trained, and how it operates. We first define what we mean by *type* in Section 2.1, *visibility* in Section 2.2, *coherence* and *scale* in Section 2.3, and provide an algorithm in Section 2.4 that uses these properties to detect features in an image.

2.1 Feature classification

Low-level features are identifiable patterns of colour in a window, that is, features are classes of pattern. Different applications, and different users, make use of different classes, hence the literature contains "edge detectors", "corner detectors" and so on. We take the view that features should not be defined in advance but learned from the user, who chooses the number of features classes, and provides examples of features from each class.

Our training method is very simple. The user moves a ring over an image, and clicks a mouse button whenever they decide that the window isolated by the ring is an example of a feature for a given class; the keyboard is used to swap between classes. We force the user to provide examples for an additional class we call "other" that contains examples

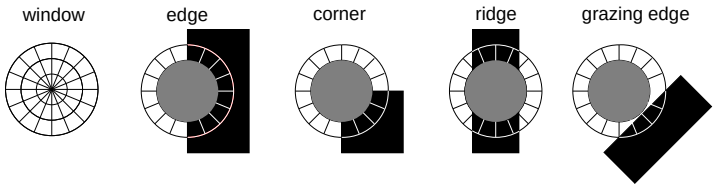

Figure 1: A sampling window and several common-low level features: edge, corner, and ridge. Notice that at the outer ring, a grazing edge gives exactly the same signal as a corner.

of features of no interest to the user. This class is necessary to fully characterise the population distribution in feature space.

We use a ring as a marker because we use a circular sampling window. A pixel with location $\underline{x} = (x,y)^T$ is the centre of the window. The RGB image is a three-valued function $\underline{c}(\rho,\theta)$ that we sample at discrete values of radius ρ and angle θ. We process this into an m-dimensional *feature vector*, $\underline{f}$, where m is the number of angular samples (we use $m = 16$), and estimate the magnitude of the angular differential

$$u(\theta) = \left| \frac{\partial \underline{c}(\rho,\theta)}{\partial \theta} \right| \tag{1}$$

using forward differences and the Euclidean distance between RGB colours. Then characterise this differential signal using

$$f(\omega) = \frac{|\mathscr{F}[u(\theta)]|}{(\sum_\theta u(\theta)^2)^{1/2}} \tag{2}$$

$$f(\omega) \leftarrow f(\omega) \setminus f(0) \tag{3}$$

where $\mathscr{F}$ is the Fourier transform. Using $|\mathscr{F}|$ ensures orientation and mirror invariance. Normalising by unit power enhances discrimination between some feature classes. Removal of the "dc" term, $f(0)$, removes dependence on mean luminance. These invariances mean that features of a given type, edges say, all map to the same point in the parameter space we use, thus giving a strong signal. This is to be preferred over Cartesian based characterisations in which features tend to be more thinly distributed in parameter space; edges, for example, would lie on a ring-like manifold. This approach to feature characterisation may remind the reader of Zernike moments [11]; we have implemented Zernike moments in addition to the above Fourier transform and found no advantage.

Now suppose the user chooses N feature classes, the classifier has $N+1$ classes in it, because of the 'other' class. The user provides n_i examples in the ith class. The posterior probability that a window belongs to class i is estimated using Bayes' rule:

$$p(i|\underline{f}) = \frac{p(\underline{f}|i)p(i)}{\sum_{j=1}^{N} p(\underline{f}|i)p(i)} \tag{4}$$

where $\underline{f}$ is the feature vector of the window. The prior probability, $p(i)$, that a window selected at random belongs to class i is

$$p(i) = \frac{n_i}{\sum_{j=1}^{N+1} n_j} \tag{5}$$

The class conditional probability $p(\underline{f}|i)$ depends on the population density distribution of the features in class i. We have chosen to represent this density with Gaussian Mixture Model (GMM), using Expectation Maximisation [3] to fit the Gaussian components, and automatically select the number of components K_i using a Bayesian approach [10]. In this case we have

$$p(\underline{f}|i) = \sum_{k=1}^{K_i} \alpha_k g(\underline{f}; \underline{\mu}_k, \underline{C}_k) \tag{6}$$

in which α_k is the prior for the kth Gaussian component, which has centre $\underline{\mu}_k$ and covariance matrix $\underline{C}_k$. Thus, to classify a pixel at $(x,y)^T$ we circularly sample around it, process the window into a feature vector using the outer-most ring of samples, and plug that into our classifier to obtain the posterior probability vector $p(i|\underline{f})$, for $i \in [1, N+1]$. The *type* of the feature is thus $\mathscr{T}[\underline{f}] = \text{argmax}_i p(i|\underline{f})$ and

$$\underline{p}_t(x,y) = \begin{bmatrix} p(1|\underline{f}) \\ \vdots \\ p(N+1|\underline{f}) \end{bmatrix} \tag{7}$$

is the $N+1$-dimensional "probability type vector" at pixel $(x,y)^T$.

This characterisation of features is able to discriminate between the common features classes. However, normalising the differential boosts noise and can lead to windows with no visually discernible structure being falsely classified, because it. We therefore introduce the notion of *visibility*, meaning that the colour contrast in a window must be sufficiently large for any feature to be seen.

2.2 Visibility

We measure the visibility of the window $\underline{c}(\rho,\theta)$ using the differential of the colour signal, but measured in *just noticeable difference* (jnd) units. Two colours are separated by one jnd, if they can just be discriminated by the human eye. jnd distance varies with colour, and many "perceptually linear" colour models exist, such as the CIELAB colour space [13]. Unfortunately a unit distance in such spaces does not necessarily correspond to one jnd, so we empirically measured jnd units in RGB space, as explained in [6]. We argue that if a colour space has an associated function that gives the jnd for each colour, $\tau(\underline{c})$), then any colour space will do. We choose RGB because it is used to encode images. The consequence of our experiments are two numbers for each colour: the mean of many trials, $\tau(\underline{c})$), and the variance over the trials $\sigma^2(\underline{c})$. Together (τ, σ^2) specify a normal distribution for the likelihood that some distance t is the jnd distance: $p(t) \propto \exp(-(1/2)(t-\tau)^2/\sigma^2)$

We can now measure the probability that a feature is visible by measuring the magnitude of the total difference in the window. This is proportional to

$$L = \left(\left| \frac{\partial \underline{c}(\rho,\theta)}{\partial \rho} \right|^2 + \left| \frac{1}{\rho} \frac{\partial \underline{c}(\rho,\theta)}{\partial \theta} \right|^2 \right)^{1/2} \tag{8}$$

which measures the total RGB colour change, which is a distance in Euclidean space. We could convert L to jnd units by division by the τ value corresponding to the colour at the

pixel. However, because we want to compute the probability that the window is "visible", we define

$$p_v(x,y) = \frac{1}{2}\left(\text{erf}\left(\frac{(L-\tau)}{2\sigma}\right)+1\right) \tag{9}$$

where σ and τ are taken from the colour are the window centre, $(x,y)^T$. This is justified because the probability of visibility as a colour-distance L in RGB space is the cumulative density $\int_0^L \exp(-(1/2)(t-\tau)^2/\sigma^2)\mathrm{d}t$.

2.3 Coherence

The classifier can give some surprising results. We trained to detect 'corners' and soon discovered that "grazing edges" — edges that intersect the window but which do not pass through its centre — are falsely classified as corners. One might suspect that this is because our classifier considers only the outer-most ring of samples in a window, so that grazing edges and corners look identical, as seen in Figure 1. Yet using all samples in a window did not improved matters. This is because corners have a scale in a sense similar to edges having scale; a "blurred corner" looks flat at its centre, and therefore resembles a grazing edge, which confuses a classifier because training examples are not well separated.

Our solution is to introduce *coherence*, with a role analogous to entropy, but which takes spatial structure into account. A window $\mathscr{W}$ is coherent, if any given ring is similar to any other ring. We define an *incoherence* measure, D, for a window:

$$D \quad = \quad \sum_{(\rho_1,\rho_2)\in\mathscr{W}} \left(\sum_{\theta} |\underline{c}(\rho_1,\theta) - \underline{c}(\rho_2,\theta)|^2)\right)^{1/2} \tag{10}$$

The incoherence at an edge, and at a true corner, is low, incoherence close to but not on an edge is high. It is tempting to use an analogy with the probability of visibility and so compute the probability of coherence with an error function: $p_c(x,y) = (1-\text{erf}((D-\mu)/\phi))/2$. We have found that this reasonable results, but there is advantage in delaying the computation of this probability until after the scale of a feature has been decided.

We define the *scale* of a feature as the smallest radius at which it is probably visible, computed at a pixel using

$$R(x,y) = \min(\{\rho : p_v(x,y;\rho) > 0.5\}) \tag{11}$$

where $p_v(x,y;\rho)$ is updated notation to allow for the fact that the visibility computation depends on the scale parameter ρ, which is the radius of the window; $\min()$ chooses the smallest element in the set which is its argument. We have deliberately omitted coherence from this definition of scale, because a feature could potentially be anything (the user may choose to not use coherence at all).

2.4 The Algorithm

Our algorithm for detecting features first determines scale, then classifies and computes a coherence measure. The coherence measures are later used to compute a coherence probability map:

Input a full colour image
FOR each pixel
    $\rho \leftarrow \rho_{\min}$ // $\rho_{\min}$ is the smallest useful scale
    WHILE $p_v(x,y;\rho) < 0.5$
        $R(x,y) = \rho$ // record the scale at this pixel
        compute $\underline{p}_t(x,y;\rho)$// type probability vector at scale ρ
        compute $D(x,y;\rho)$ // coherence measure at scale ρ
    ENDWHILE
ENDFOR
// Having scaled and classified, compute coherence ...
FOR each pixel
    $\rho = R(x,y)$ // fetch the local scale
    circularly sample the coherence measure map $D(x,y)$ to get a window $m(\rho,\theta)$
    normalise m into the range $[0,1]$
    set $D(x,y) \leftarrow m(0,0)$ // use window centre as new coherence measure
    compute $p_c(x,y) = (1 - \mathrm{erf}((D(x,y) - \mu)/\phi))/2$
ENDFOR
Output probability maps $\underline{q}(x,y) = \underline{p}_t p_c$.

Figure 2: Progress of classification: top-left, a full-colour input image; top-right, output of the classifier without coherence; bottom-left, the coherence map — note the fall-off near to but not on edges; bottom-right, the final classification map. Red shows edges, green ridges, blue corners.

The algorithm outputs $\underline{q}(x,y) = \underline{p}_t p_c$. The feature is probably visible by construction. Each "channel" of $\underline{p}_t(x,y)$, is a probability map that the pixel is of a type (specifically, the ith class). The product $\underline{p}(x,y|i)p_c(x,y)$ gives the joint probability that the pixel belongs to the given type and is coherent. Of course, each probability vector $\underline{p}_t(x,y)$ sums to unity, but there no such guarantee for $\underline{q}(x,y)$. The progress of this algorithm at key-stages in illustrated in Figure 2

3 Results and applications

An important first test of any detector is just to look at some results. We trained a classifier to detect "edge", "ridge", and "corner" features. We took care to train and test on a wide variety of images types (indoors, outdoors, portraits, landscapes, etc) it may be that re-introducing rarity would benefit some image classes. We tested on a variety on images — none of them in the training set. A few results are presented in Figure 3.

The question of scale brings with it the question of how many classifiers are needed: do we need a classifier for every scale, or will just a few classifiers suffice? To answer this question we ran our classifier at set scales over hand-prepared ground-truth images that were not in the training set, but using a single classifier trained at a fixed a scale. We chose a window of radius 3 because that was the smallest window we in which could reliably see interesting features (for us, this meant edges, corners, and ridges). We found that features below that scale were unreliable classified (but recall we could not reliably see them), but features above that scale we reliably classified, See Figure 4. In fact, the the classification of a feature remained stable until the window grew sufficiently large as to cover other objects in the image. On the other hand, windows that contained spurious patterns, those we had not trained on, tended to be far less stable in their classification over scale. All the results in this paper were computed using a single classifier trained at scale 3.

Having established only one classifier is needed, we next validated the features our classifier found against hand prepared line-drawings. While the Berkeley segmentation data set [9] was available, this was considered unsuitable as it tended to exclude small features. We asked humans to sketch a picture, giving no other instruction except "make a reasonably fine drawing". No test image was in the training set. We measured a precision-recall curve for varying thresholds of summed $\underline{q}(x,y)$ vectors. Following Martin *et al.* [9], precision is defined as the fraction of detections that are true rather than false; and recall is defined as the fraction of ground truth data that is detected rather than missed. Precision-recall curves are preferred over ROC curves because the former do not depend on image resolution (see [9]).

For comparison with standard methods we computed the P-R curve for Canny edges. For further reference we plotted data taken from Martin *et al.* [9], showing their method, and the median human performance; but we did not repeat there experiments. Figure 3 shows results for two images that typify results from all those we used. In these, and in fact in all other test images, we consistently out-perform the prescriptive methods. We do about as well as Martin *et al.*, sometimes out-performing them, sometimes not. The goal of reaching human levels of performance eludes all methods, unsurprisingly.

The corner data from the classifications was also considered independently for comparison with Harris corners [7] for the purposes of feature matching between stereo image

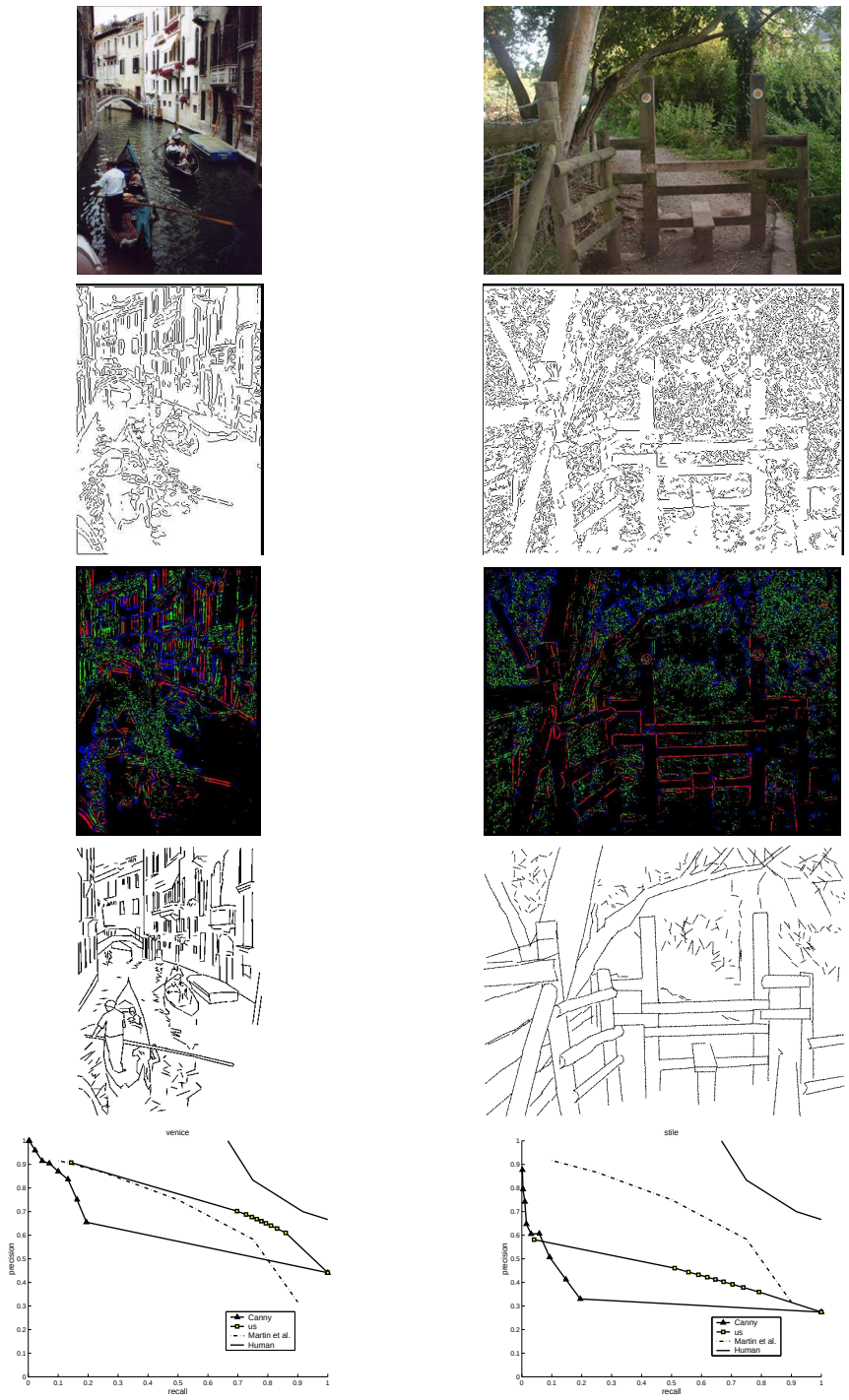

Figure 3: Example test images, a canny edge-map, classification (red indicates edges, green ridges, and blue corners), ground truth, and PR curves.

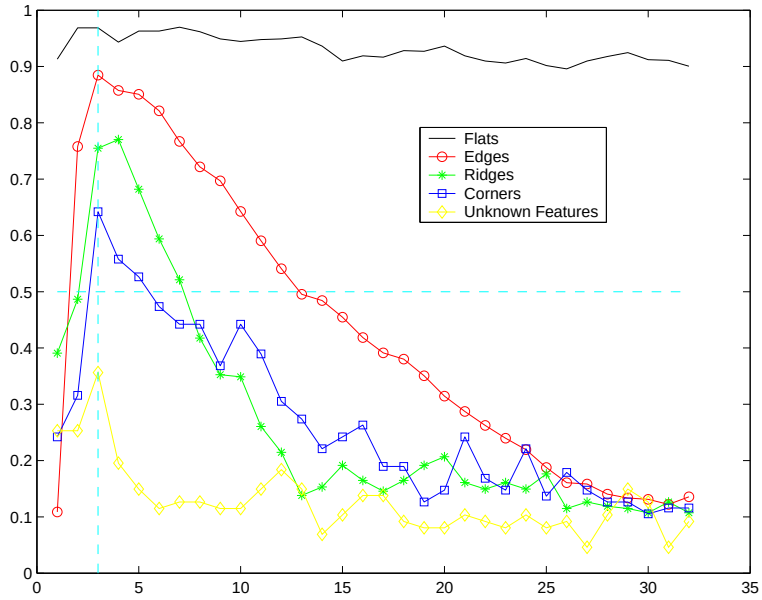

Figure 4: Probability of correct classification over scale. The vertical dotted line shows scale at 3, the trained scale. The horizontal line shows a probability of correct classification of 0.5, above which we consider satisfactory.

pairs. The images ranged from simple block-worlds to noisy outdoor images that included trees and other foliage.

Corner-maps were produced using both techniques, and prospective matches computed using a simple least-square difference between corner neighbourhoods. Homographies were then computing using RANSAC [5]. On average less our our corners were deemed outliers by the RANSAC algorithm (63%) than the Harris corners (75%). If one knows in advance that this much smaller a fraction of the given corner-map are likely to be outliers, then the RANSAC algorithm can be run with fewer iterations, a quarter as many in this case, for full details see [6].

4 Discussion and Conclusions

We have shown that it is possible detect low-level features that are specific to a users interests. Moreover, the scale at which such features are detected can be automatically decided, even though training need only take place at a single scale.

Our system out-performs standard detectors and favourably compares with a state-of-the-art method that fine-tunes its components; it may be our system would benefit by similarly fine-tuning the visibility, coherence, and type sub-components.

A more basic concern is that although “type” is not prescriptive, “visibility” and “coherence” are. The reason for this inconsistency is that including visibility and coherence *measures* as vector elements leads to population distributions that contain sharp discontinuities and so difficult to model with continuous functions (as used by GMMs). Yet

modelling the populations well would allow us to train without prescribing visibility or coherence, or indeed any other terms we might care to use in the future.

Despite these issues, our approach does respond to user needs, and offers improved performance over prescriptive detectors. The ability to deal with types of feature is advantageous; leaves, for example form "ridge noise" that can be filtered away. Moreover, we have observed that patterns of feature class occur commonly. For example, mouths are often characterised by a pattern of two corners joined by a ridge, which is flanked by two edges. If sufficient regularity exists in such patterns, it may be possible to use our feature classifier as the basis of a more complex pattern recognition system. Work on constructing a scale-space description, by analogy with Witken [12] is on-going.

References

[1] S.M. Smith & J.M. Brady. Susan – a new approach to low level image processing. Technical report, FMBIB, Oxford University, 1995.

[2] J.F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Inteligence*, 8(6):34–43, June 1986.

[3] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[4] N. Krüger & M. Felsberg. A continuous formulation of intrinsic dimension. In *Proceedings British Machine Vision Conference*, pages 260–270. BMVA, 2003.

[5] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with apphcatlons to image analysis and automated cartography. *ACM Computing Surveys (CSUR)*, 14:3–71, March 1982.

[6] Reserved for Anonymity. Xxxxxx.

[7] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conference*, pages 189–192, Manchester, UK, 1998.

[8] T. Lindeberg. Edge decection and ridge detection with automatic scale selection. *International Journel of Computer Science*, 25(1):57–74, January 1996.

[9] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, May 2004.

[10] S.J. Roberts, D¿ Husmeier, II Rezek, and W. Penny. Bayesian approaches to gaussian mixture modelling. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 20(11):1133–1142, 1998.

[11] M.R. Teague. Image analysis via general theory of moments. *Journal of Optical society of America*, 70(8):920–930, 1979.

[12] A. Witken. Scale-space filtring. In *Proc. Int. Joint Conf. on Artificial Intelligence*, Karlsruhe, Germany, 1983.

[13] G. Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley and Sons, 2nd edition, 1982.

[14] S. Konishi & A. L. Yuille & J. M. Caughlan & S. C. Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, 25(1):57–74, January 2003.