Robust Multi-scale Extraction of Blob Features

Per-Erik Forssén and Gösta Granlund

Computer Vision Laboratory, Linköping University,
SE-581 83 Linköping, Sweden
{perfo, gosta}@isy.liu.se
http://www.isy.liu.se/cvl/

Abstract. This paper presents a method for detection of homogeneous regions in grey-scale images, representing them as blobs. In order to be fast, and not to favour one scale over others, the method uses a scale pyramid. In contrast to most multi-scale methods this one is non-linear, since it employs robust estimation rather than averaging to move through scale-space. This has the advantage that adjacent and partially overlapping clusters only affect each other's shape, not each other's values. It even allows blobs within blobs, to provide a pyramid blob structure of the image.

1 Introduction

In signal processing it is useful to attach a confidence measure to each measurement [1]. Such algorithms usually work with a measurement–confidence pair, but there are advantages with using the *channel representation* [2, 3]. The channel representation of a measurement–confidence pair (p, c) is a vector $\boldsymbol{h} = c(\, H_1(p)\; H_2(p)\; \ldots\; H_K(p)\,)^T$ of *channel values* h_k, computed using a set of localised *kernel functions* H_k. The kernels should be symmetric, non-negative, and preferably have a compact, localised support. In this paper we will use a family of integer displaced, windowed $\cos^2()$ functions

$$H_k(p) = \begin{cases} \cos^2(\pi/3(p-k)) & \text{if } |p-k| \leq 3/2 \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

This setup gives us at most 3 non-zero channels for any signal value. Other common choices of kernels are B-splines [4], and Gaussians [5]. The values represented in a channel vector may be recovered using a *local decoding* [3]. That is, we cut out an interval of channel values from the vector, and decode the (p, c) pair using only these. A local decoding allows several (p, c) pairs to be retrieved without their interfering, provided that the measurement values are sufficiently different. For a channel representation according to (1), the local decoding has to involve at least $N = 3$ adjacent channel values $\{h_l, h_{l+1}, h_{l+2}\}$, and is calculated as

$$\hat{p}_l(\boldsymbol{h}) = l + \frac{3}{2\pi}\arg\left[\sum_{k=l}^{l+N-1} h_k e^{\mathbf{i}2\pi/3(k-l)}\right] \quad \text{and} \quad \hat{c}_l(\boldsymbol{h}) = \frac{2}{3}\sum_{k=l}^{l+N-1} h_k. \tag{2}$$

To decode a channel vector, we simply go through all such adjacent groups, and sort the decoded $(\hat{p}, \hat{c})$ pairs, according to the confidence measures $\hat{c}$.

1.1 Channel Smoothing

If we combine a set of measurements using an average of their channel vectors $\boldsymbol{h} = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{h}_l$, the decoding with the largest confidence measure can be shown to be a robust weighted average [4].[1] In other words, similar measurements are averaged, but the influence of a measurement on the result drops as a function of its distance to the cluster centre.

If we channel encode each pixel, $p(x, y)$, in a grey-scale image with $c(x, y) = 1$, we obtain a set of *channel images*. If we then perform low-pass filtering on the channel images, and reconstruct an image using the local decoding (2), in each pixel, we obtain the result shown in figure 1. This operation is called *channel smoothing*. A low-pass filtering directly on the image is also shown for comparison. The behaviour of channel smoothing is similar to *selective binomial filtering* [6], and *non-linear Gaussian filtering* [7]. In fact, non-linear Gaussian filtering can be shown to correspond to the first iteration of an *M-estimation*, which is a robust estimation technique [8]. Thus, iterated Gaussian filtering will yield very similar results to channel smoothing. One thing missing in the non-linear Gaussian filtering is the confidence measure, which we will need later in this paper.

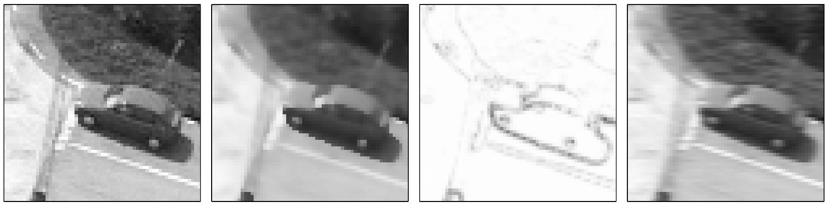

Fig. 1. Illustration of channel smoothing.

For this example we have used $K = 8$ channel images, and averaged each channel image with two one-dimensional Gaussian filters of $\sigma = 1.18$ and 7 coefficients each. A channel representation according to (1) can represent measurements $p \in [3/2, K - 1/2]$, and thus we have scaled the image intensities $i(x, y) \in [r_l, r_h]$ using a linear mapping $p(x, y) = t_1 i(x, y) + t_0$ with

$$t_1 = \frac{K-2}{r_h - r_l} \quad \text{and} \quad t_0 = 3/2 - t_1 r_l. \tag{3}$$

For large amounts of smoothing, the channel smoothing output looks almost like a segmented image, and this is what lead us to develop the blob extraction algorithm presented in this paper.

[1] This result actually only holds when the decoding formula is linear, as is the case for B-spline kernels. For other kernel functions the result is however very similar.

2 A Blob Extraction Algorithm

Homogeneity features are called *blobs* in scale-space theory [9]. In contrast to segmentation, blob detection has a more modest goal—we do not attempt to segment out exact shapes of objects, instead we want to extract robust and repeatable features. A blob representation discards exact shapes, and thin connections between patches are neglected.

Blob features have been used as texture descriptors [10] and as features for image database search [11]. For a discussion of the similarities and differences of other approaches and the one presented here, the reader is directed to [6].

The blob estimation procedure uses a scale pyramid. Each position and scale in the pyramid contains a measurement and confidence pair (p, c). The confidence is a binary variable i.e. $c \in \{0, 1\}$. It signifies the absence or presence of a *dominant* measurement in the local image region. Consequently, when $c = 1$, the dominant measurement is found in p. This representation is obtained by non-linear means, in contrast to most scale-space methods, which are linear [9], and thus obtain the *average* measurement.

The pyramid is used to generate a region image, from which we then extract a list of homogeneous regions. The shape of each region is approximated by its moments of orders 0, 1 and 2. These moments are conveniently visualised as ellipses, see figure 2 (right). The following sections will each in turn describe the different steps of the algorithm, starting with the clustering procedure.

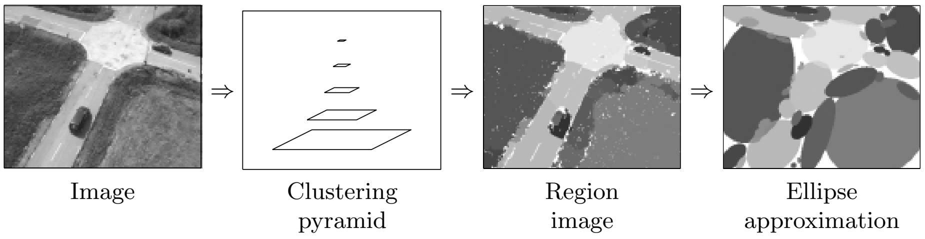

Fig. 2. Steps in blob extraction.

2.1 Building a Clustering Pyramid

To build the clustering pyramid, we view each image pixel as a measurement p with confidence $c = 1$, and expand the image into a set of channel images. For each of these channel images we generate a low-pass pyramid. The channels obtained from the input image constitute scale 1, and successively coarser scales are obtained by low-pass filtering, followed by sub-sampling. The low-pass filter used consists of a horizontal and a vertical 4-tap binomial kernel [1 3 3 1]/8.

When the filter sums to 1, the confidence values of the reconstructed measurements correspond fractions of the *area* covered by the filter. Thus, we construct the low-pass pyramids, reconstruct the dominant (p, c) pair in each position, and finally make the confidence binary by setting values below $c_{\min}$ to zero, and values above or equal to $c_{\min}$ to 1. Typically we use the area threshold $c_{\min} = 0.5$. Row A in figure 3 shows such a pyramid for an aerial image.

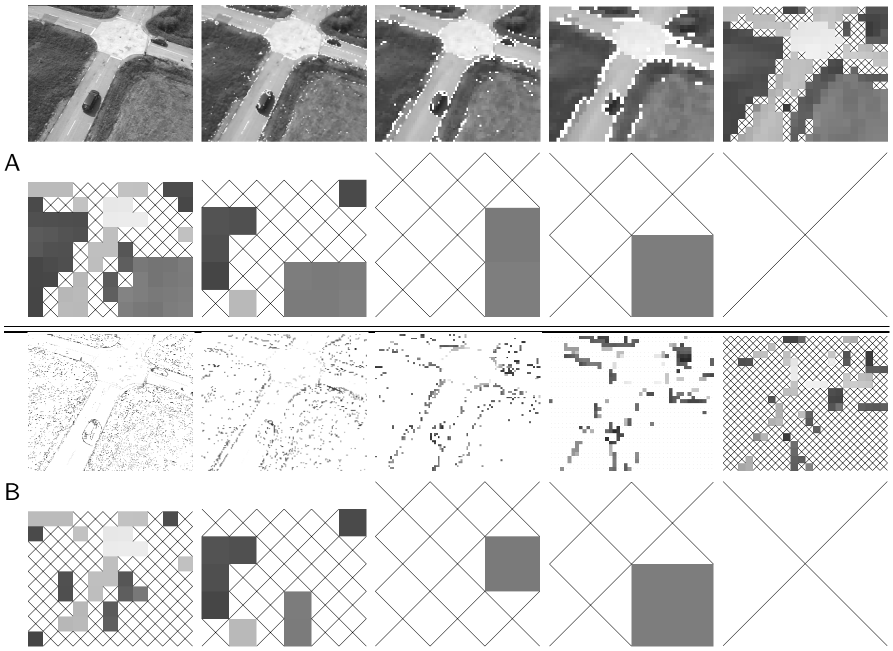

Fig. 3. A: Clustering pyramid. B: Result of the pruning operation on clustering pyramid. Positions with confidence $c = 0$ are indicated with crosses.

2.2 Pruning the Pyramid

If we look at the clustering pyramid in row A of figure 3, we can see that there is a great deal of redundancy, which will now be removed. We do this via a top-down reconstruction of the image. Starting from the top pixel, we expand it to cover its *children* (the four pixels below) by nearest neighbour up-sampling, i.e. duplication. Children which are similar to the parent are set to zero, and the others get to *substitute* the parent description in the reconstruction. This process is repeated until the finest level is reached.

The purpose of this stage is to remove all pixels that have parents (or parent's parents etc. if the parent has zero confidence) with a similar dominant component. By similar, we mean $|p_k - p_l| \leq d_{\max}$, where $d_{\max} = 2t_1$ (see (3)), i.e. the approximate boundary between averaging and rejection (see [5]).

The result of this operation is shown in figure 3, row B. It bears some similarity to the old quad-tree image representation (see e.g. [9], chapter 2). The main differences from quad-trees are that we integrate information over more than a 2×2 region, and that we are representing the dominant component rather than the average.

As we go through the pyramid top down, we also compute two neighbour maps, one signifying that the pixel to the right is similar to this one, and one signifying that the pixel below is similar to this one. These maps will be used in the region image generation.

3 Region Image Generation

If we look at figure 3, row B, we see that the pixels left in the pyramid can be used as seeds for blob extraction. We start at the top scale, and generate a label image, where each seed is given an integer label. We then proceed to expand each region using the neighbour maps extracted at the previous stage. We then do a nearest neighbour up-sampling of the label image, and crop the result to the image size at the finer level. Finally we expand our regions once more, using the neighbour maps. To summarise, the algorithm for region image generation looks like this:

step 1 Generate an empty label image at the top scale.
step 2 Assign new labels to all pixels with $c = 1$, and expand each new region with its neighbours, if the neighbour maps say so.
step 3 Do a nearest neighbour up-sampling of the label image, and crop it to the image size at the finer level.
step 4 Expand all regions with their neighbours, whenever the neighbour maps say so.
step 5 If we are at the finest scale, we are done. Otherwise go back to **step 2**.

During the region expansion (**step 4** above), a region with a lower label number is allowed to expand over one with a higher, if there is a conflict. This will cause large regions to "eat" most of the small regions with similar colour that will otherwise appear near the object boundaries. Note that the algorithm only moves through the pyramid once, and is thus quite fast.

The algorithm above does not quite produce a conventional segmentation. Occasionally it splits homogeneous regions into two or more regions. In most cases this is in line with the shape constraint on the region shapes mentioned in section 2, but sometimes regions which are really better described as one blob are split. This will however be dealt with in a later stage of the blob extraction.

4 Ellipse Approximation

The raw moments of a binary mask $v_n(x,y) : \mathbb{Z}^2 \rightarrow \{0,1\}$ are defined by the weighted sum

$$\mu_{kl} = \sum_x \sum_y y^k x^l v_n(x,y). \tag{4}$$

See e.g. [12] for a more extensive discussion on moments of binary masks. For all the regions $\{v_n(x,y)\}_1^N$ in the image we will now compute the raw moments of order 0 to 2, i.e. μ_{00}, μ_{01}, μ_{10}, μ_{02}, μ_{11}, and μ_{20}. Note that this can be done using only one `for`-loop over the region image.

The raw moments are then converted to measures of the area a_n, the centroid vector $\boldsymbol{m}_n$, and the inertia matrix $\boldsymbol{I}_n$

$$a_n = \mu_{00}, \quad \boldsymbol{m}_n = \frac{1}{\mu_{00}} \begin{pmatrix} \mu_{01} \\ \mu_{10} \end{pmatrix} \quad \text{and} \quad \boldsymbol{I}_n = \frac{1}{\mu_{00}} \begin{pmatrix} \mu_{02} & \mu_{11} \\ \mu_{11} & \mu_{20} \end{pmatrix} - \boldsymbol{m}_n \boldsymbol{m}_n^T. \tag{5}$$

Using the input image $p(x,y)$ we also compute and store the average measurements for all regions,

$$p_n = \frac{1}{\mu_{00}} \sum_x \sum_y p(x,y) v_n(x,y). \tag{6}$$

From the eigenvalue decomposition $\boldsymbol{I} = \lambda_1 \hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T + \lambda_2 \hat{\boldsymbol{e}}_2 \hat{\boldsymbol{e}}_2^T$ with $\lambda_1 \geq \lambda_2$ we can find the axes of the ellipse as $2\sqrt{\lambda_1}\hat{\boldsymbol{e}}_1$ and $2\sqrt{\lambda_2}\hat{\boldsymbol{e}}_2$ respectively. Since $\boldsymbol{I} = \boldsymbol{I}^T$, each blob can be represented by $1+2+3+1=7$ parameters.

4.1 Merge and Cleanup

For different translations and rotations of the image, the detection scale for a blob may change, and this will cause a blob to sometimes be split in two, or several. To reduce this effect, we will make use of the fact that the moments of a combined mask $v(x,y)$ can be computed from the moments of its parts (e.g. v_1 and v_2 with $v_1 + v_2 = v$), and merge blobs whenever appropriate.

Since $\det|\boldsymbol{I}| = \lambda_1 \lambda_2$, the ellipse area is given by $e_n = 4\pi\sqrt{\det|\boldsymbol{I}_n|}$. Unless all pixels in the mask lie on a line, this is an overestimate of the actual mask area, and $r_n = a_n/e_n \leq 1$ is thus a measure of how ellipse-like a blob is. We will merge two blobs if they have similar dominant measurements, and the result becomes more ellipse-like, i.e. if $|p_m - p_n| \leq d_{\max}$ and $e_m + e_n \geq e_{mn}/\alpha$. The parameter α can be set to > 1 to cause more mergers. Here we have used $\alpha = 1.005$. Finally, we remove blobs with areas below a threshold $a_{\min} = 20$.

Figure 4 shows an aerial image and two blob representations of the image. The clustering pyramid has been created using $K = 26$ channels, spaced according to (3). The input image is a 348×287 image, and the blob representation initially contains 146 blobs, which after merging and cleanup drops to 57. This gives a total of 399 parameters for the entire image–a factor 250 of data reduction.

5 Noise Sensitivity

The purpose of the robust estimation scheme is to obtain robustness to noise. To evaluate noise sensitivity we have subjected our test image to three common kinds of noise.

1. **Salt&pepper noise**. This consists of randomly setting pixel intensity to 1 or 0. The density of corrupted pixels chosen is 5%.
2. **White rectangular noise**. This noise is additive white noise with rectangular distribution and amplitude 0.08 i.e. $i(x,y) = i_0(x,y) + \epsilon$ where $\epsilon \in 0.08[-1,1]$.
3. **$1/f$ noise**. This noise is correlated additive noise, i.e. white rectangular noise $\epsilon_0(x,y) \in [-1,1]$, weighted with $1/r$ in the Fourier domain. Formally we have $i(x,y) = i_0(x,y) + \epsilon(x,y)$ with $\epsilon(x,y) = 0.06\text{IDFT}\{d(r)\text{DFT}\{\epsilon_0\}/r\}$. Here $r = \sqrt{u^2+v^2}$ and the frequency coordinates are $u,v \in [-\pi,\pi]$, and $d(r)$ is a function which is zero for $r=0$ and 1 otherwise. I.e. it discards the DC component.

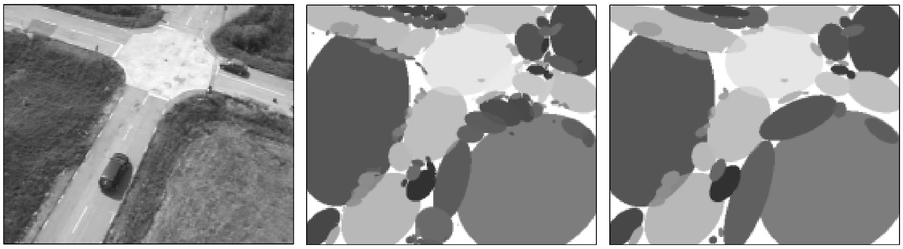

Fig. 4. Aerial image, blobs from scales 3-8, and blobs after merge and cleanup step. Number of blobs are 146 and 57 respectively.

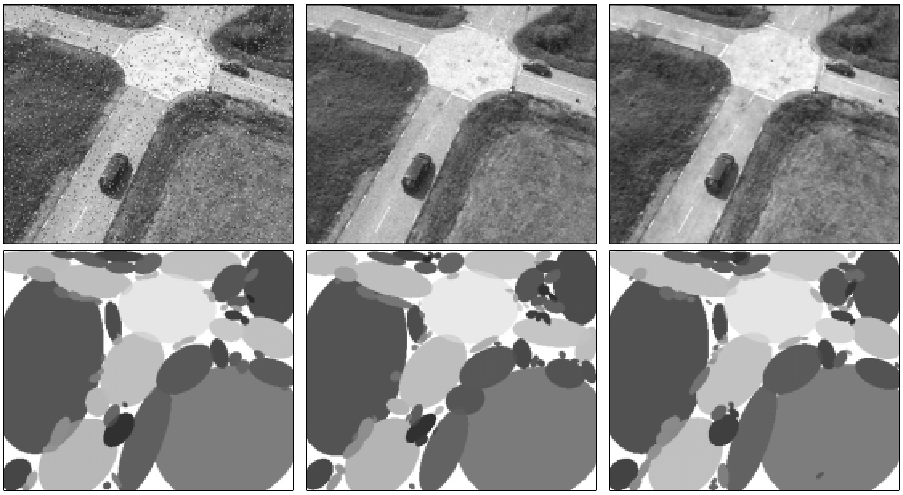

Fig. 5. Top row: noisy input images. Bottom row: corresponding blob images. Columns left to right: salt&pepper noise, white rectangular noise, and $1/f$ noise. Number of blobs for the three cases are 66, 76, and 72 respectively.

As can be seen in figure 5, all three kinds of noise are handled fairly well. There is a small change in number of blobs, we originally had 57 blobs, and afterwards we got 66, 76, and 72 respectively. As can be seen this is mainly due to similar neighbouring blobs being merged, or blobs being split into two.

The robust estimation causes salt&pepper pixels to be rejected as outliers. This will cause small reductions in the blob areas and slight changes of blob shapes, but will not affect the resultant measurements p_k at all. White noise will be averaged if it has an amplitude below $d_{\max}$. For $K = 26$ we have $d_{\max} = 0.083$. $1/f$ noise is a common model of atmospheric distortion, and moderate amounts of this distortion appears to be handled as well.

6 Concluding Remarks

We wish to stress that this is just a first attempt at robust blob detection, and a rigorous evaluation of the performance is still in progress. Similar results could probably be obtained if the pyramid generation using channel smoothing is replaced by another robust estimation technique. The blob features presented in this paper are intended for view based object recognition using learning, something which will hopefully be facilitated by the dramatic data reduction (we typically get 200 times less data).

Acknowledgements

The work presented in this paper was supported by WITAS, the Wallenberg laboratory on Information Technology and Autonomous Systems, which is gratefully acknowledged.

References

1. Granlund, G.H., Knutsson, H.: Signal Processing for Computer Vision. Kluwer Academic Publishers (1995) ISBN 0-7923-9530-1.
2. Granlund, G.H.: An Associative Perception-Action Structure Using a Localized Space Variant Information Representation. In: Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC), Kiel, Germany (2000)
3. Forssén, P.E.: Sparse Representations for Medium Level Vision. Lic. Thesis LiU-Tek-Lic-2001:06, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden (2001) Thesis No. 869, ISBN 91-7219-951-2.
4. Felsberg, M., Scharr, H., Forssén, P.E.: The B-spline channel representation: Channel algebra and channel based diffusion filtering. Technical Report LiTH-ISY-R-2461, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden (2002)
5. Forssén, P.E.: Observations Concerning Reconstructions with Local Support. Technical Report LiTH-ISY-R-2425, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden (2002)
6. Forssén, P.E., Granlund, G.: Blob Detection in Vector Fields using a Clustering Pyramid. Technical Report LiTH-ISY-R-2477, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden (2002)
7. Godtliebsen, F., Spjøtvoll, E., Marron, J.: A nonlinear gaussian filter applied to images with discontinuities. J. Nonpar. Statist. **8** (1997) 21–43
8. Winkler, G., Liebscher, V.: Smoothers for discontinuous signals. J. Nonpar. Statistics **14** (2002) 203–222
9. Lindeberg, T.: Scale-space Theory in Computer Vision. Kluwer Academic Publishers (1994) ISBN 0792394186.
10. Lindeberg, T., Gårding, J.: Shape from texture in a multi-scale perspective. In: Proc. 4th International Conference on Computer Vision, Berlin, Germany (1993) 683–691
11. Belongie, S., Carson, C., Greenspan, H., Malik, J.: Color- and texture-based image segmentation using EM and its application to content based image retrieval. In: Proceedings of the 6:th ICCV. (1998) 675–682
12. Sonka, M., Hlavac, V., Boyle, R.: Image Processing, Analysis, and Machine Vision. International Thomson Publishing Inc. (1999) ISBN 0-534-95393-X.