APPROXIMATING IMAGE FILTERS WITH BOX FILTERS

Bernardo Rodrigues Pires, Karanhaar Singh and José M. F. Moura

Carnegie Mellon University
Department of ECE
Pittsburgh, PA
{bpires, singhk}@cmu.edu, moura@ece.cmu.edu

ABSTRACT

Box filters have been used to speed up many computation-intensive operations in Image Processing and Computer Vision. They have the advantage of being fast to compute, but their adoption has been hampered by the fact that they present serious restrictions to filter construction. This paper relaxes these restrictions by presenting a method for automatically approximating an arbitrary 2-D filter by a box filter. To develop our method, we first formulate the approximation as a minimization problem and show that it is possible to find a closed form solution to a subset of the parameters of the box filter. To solve for the remaining parameters of the approximation, we develop two algorithms: Exhaustive Search for small filters and Directed Search for large filters. Experimental results show the validity of the proposed method.

Index Terms— box filters, filtering theory, integral images

1. INTRODUCTION AND PROBLEM STATEMENT

1.1. Introduction

The use of box filters and integral images traces its roots to the field of computer graphics where they were developed for texture mapping (under the name of summed area tables) by Crow in 1984 [1]. The use of box filters was introduced in the computer vision and image processing communities by Viola and Jones in 2001 with their landmark face detection algorithm [2].

More recently, several works have shown that box filters and integral images can be used to greatly speed up various computing intensive operations in computer vision and image processing. Bay, Tuytelaars, and Van Gool have used box filters to speed up the detection of the SURF image features [3]. Zhu et al. have used box filters to speed up the construction of Histograms of Oriented Gradient (HoG) features for fast human detection [4]. Tuzel, Porikli, and Meer have used box filters to speed up the computation of Region Covariance descriptors for object detection and texture classification [5].

Despite the excellent results produced by the algorithms mentioned above, box filters have not enjoyed widespread application. A possible reason for this is the fact that box filters impose restrictions on filter design. Although the filtering operation can be computed very quickly, the fact that the filter must be composed of rectangular regions limits its applicability.

This work was partially supported by ONR under grant # MURI-N000140710747. Bernardo Rodrigues Pires was partially supported by the Portuguese Foundation for Science and Technology, grant SFRH/BD/37378/2007

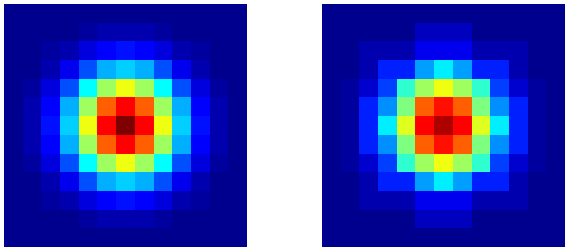

Fig. 1. Sample of the output of our algorithm. Left: Original 2-D Gaussian filter to be approximated. Right: Box filter approximation using 10 boxes.

A very recent effort by Kovesi [6] uses box filters to approximate Gaussian filtering. The result is an algorithm that is able to approximate convolution by a Gaussian filtering using only 25 additions and 5 multiplications per pixel. Although this method produces good results, it cannot be applied to approximate an arbitrary 2-D filter.

Unlike [6], the objective of this paper is to present a new method that automatically creates a box filter approximation to any 2-D filter. An example of our approach is shown in Figure 1. Starting with a general filter (on the left), the objective is to create a box filter approximation (on the right). Note that the approximation shown uses only 10 boxes and achieves an error of about 0.5% of the power of the original filter (See Section 5 for more details). The authors are not aware of any similar effort to create such an algorithm in the literature.

1.2. Problem Statement

As illustrated in Figure 2, the integral image (or summed area table) is defined such that its value at any pixel is equal to the sum of all the values of the previous rows and columns. I.e., if $i(x, y)$ is an image, the integral image $I(x, y)$ will be defined as:

$$I(x,y) = \sum_{m=0}^{x} \sum_{n=0}^{y} i(m,n) \quad (1)$$

The integral image can be computed using four sums per pixel if a recursive (raster-scan) algorithm is used:

$$I(x,y) = I(x-1,y) + I(x,y-1) - I(x-1,y-1) + i(x,y) \quad (2)$$

Or, if separate row sums are maintained, using only two sums per pixel as shown by Viola and Jones [2] (see [7] for more details).

As Figure 2 shows, once the integral image is constructed, the sum of the pixel values in any rectangular area of the image can be computed in constant time, i.e. the sum of all the pixels in the

rectangle $[a,b]\times[c,d]$, is equal to $I(b,d)-I(b,c)-I(a,d)+I(a,c)$. This shows that the integral image can be used to quickly compute the output of a filter comprised of a single rectangular section, or box, defined as:

$$B(x,y) = \begin{cases} 1, & (x,y) \in [a,b]\times[c,d] \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Naturally, a single box cannot be used to approximate complex 2-D filters. To do so, we use a linear combination of N box filters:

$$B_N(x,y) = \sum_{i=1}^{N} \alpha_i \, B_i(x,y) \quad (4)$$

where the various boxes may overlap (this is a key point since it allows us to represent complex filters using a small number of boxes). By linearity, applying the filter $B_N(x,y)$ to an image is equivalent to applying each box filter $B_i(x,y)$ separately, scaling each result with α_i, and then summing all the terms together. A box filter thus constructed is a 2D step filter. It is well known that step functions can approximate with an arbitrarily small error any measurable function, and so finite energy ($\mathcal{L}_2$) or, a posteriori, continuous functions [8].

The set $\theta_c = \{a_i, b_i, c_i, d_i\}_{i=1}^{N}$ corresponds to the corner point locations of the box filter, whereas the set $\theta_s = \{\alpha_i\}_{i=1}^{N}$ corresponds to the scaling factors of the filter. The set of parameters of the filter is denoted by $\theta = \theta_c \cup \theta_s$.

The problem of finding a box filter approximation to any filter, $G(x,y)$, corresponds to determining the set of filter parameters θ that minimizes the error $E(\theta)$ between the original filter, $G(x,y)$, and the approximation, $B_N(x,y)$. The error $E(\theta)$ is defined as the 2-norm between the original filter and the approximation:

$$E(\theta) = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} \left(G(x,y) - B_N(x,y)\right)^2 \quad (5)$$

Organization of the Paper The paper is organized as follows: In the next section we show that, given a set of corner points θ_c, it is possible to construct a closed-form solution that determines the optimal scaling factors θ_s. Armed with a closed form solution for the scaling factors, Section 3 discusses two methods of determining the optimal corner points θ_c: Exhaustive Search, which is better suited for small filters with a small number of boxes, and Directed Search, which is better suited for large filters with a larger number of boxes. Section 4 overviews the proposed algorithms and Section 5 presents experimental results that show the validity of the proposed approach. Finally, Section 6 concludes the paper.

2. FINDING THE OPTIMAL SCALING FACTORS

Assuming that the corner points θ_c in the box filter approximation are constant, we can find the optimum values for the scaling factors θ_s by making:

$$\nabla_\alpha E = \vec{0} \quad (6)$$

where $\nabla_\alpha E = \left[\begin{array}{cccc} \frac{\partial E}{\partial \alpha_1} & \frac{\partial E}{\partial \alpha_2} & \cdots & \frac{\partial E}{\partial \alpha_n} \end{array}\right]^T$ is the gradient with respect to all scaling factors. Using calculus and algebraic manipulations, it is possible to show that Equation (6) is equivalent to:

$$\nabla_\alpha E = \vec{0} \;\Leftrightarrow\; \boldsymbol{A}\,\vec{\alpha} = \vec{b} \;\Rightarrow\; \vec{\alpha} = \boldsymbol{A}^{-1}\,\vec{b} \quad (7)$$

where $\vec{\alpha} = [\;\alpha_1 \;\; \alpha_2 \;\; \cdots \;\; \alpha_N\;]^T$, and the matrix $\boldsymbol{A}$ and the vector $\vec{b}$ are defined to be: (for compactness, we omit the dependences in (x,y); all double summations are over the entire $\mathbb{R}^2$ plane)

$$\boldsymbol{A} = \begin{bmatrix} \sum\sum B_1 B_1 & \sum\sum B_2 B_1 & \cdots & \sum\sum B_N B_1 \\ \sum\sum B_1 B_2 & \sum\sum B_2 B_2 & \cdots & \sum\sum B_N B_2 \\ \vdots & \vdots & \ddots & \vdots \\ \sum\sum B_1 B_N & \sum\sum B_2 B_N & \cdots & \sum\sum B_N B_N \end{bmatrix} \quad (8)$$

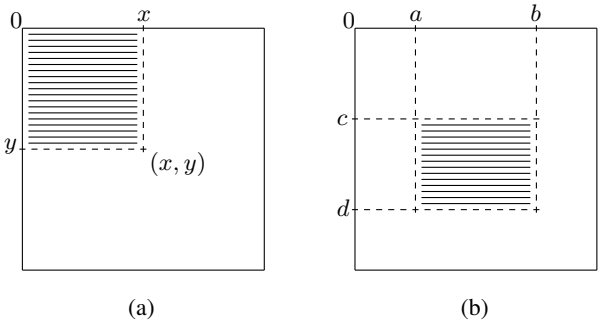

Fig. 2. Construction and usage of an integral image. (a) The value of the integral image at (x,y) is equal to the sum of the image values in the shaded area. (b) Once the integral image is constructed, the sum of all image values inside the shaded rectangle can be computed in constant time.

$$\vec{b} = \left[\;\sum\sum B_1 G \;\; \sum\sum B_2 G \;\cdots\; \sum\sum B_N G\;\right]^T \quad (9)$$

Note that the $\boldsymbol{A}$ matrix can be interpreted as a matrix that measures the "overlap" between the boxes, i.e. each entry A_{ij} of the matrix measures the number of pixels that are simultaneously inside boxes i and j. In particular, the matrix $\boldsymbol{A}$ does not depend on the original filter $G(x,y)$. The $\vec{b}$ vector can be interpreted as the sum of the values of the original filter $G(x,y)$ inside each of the boxes, i.e., each entry b_i of the vector gives the sum of the values of the filter inside box i. Additionally, note that equation (7) assumes that the matrix $\boldsymbol{A}$ is invertible. The cases when this does not happen will be discussed in Section 4.

3. FINDING THE CORNER POINTS

We present two alternative greedy approaches to finding the ideal corner points of the filter. The first method corresponds to Exhaustive Search and is well suited to situations where one wants to approximate a small filter using a small number of boxes. The Directed Search method, on the other hand, is more complex but is able to efficiently handle situations where one wants to approximate large filters.

3.1. Exhaustive Search

The Exhaustive Search method determines the ideal values of the corner points $\theta_c = \{a_i, b_i, c_i, d_i\}_{i=1}^{N}$ iteratively by working on one box at a time. The method starts by taking $N = 1$ and searching all combinations of $\{a_1, b_1, c_1, d_1\}$. For each combination, the optimal scaling factor is determined by using equation (7) and the error E in equation (5) is computed. The values of $\{a_1, b_1, c_1, d_1\}$ that minimize the error are kept.

After the values of the first box are determined, the algorithm considers the case of $N = 2$ but assumes that the first box was already placed at the optimum location and therefore it is only necessary to search for the optimum values for the corner points of the second box, $\{a_2, b_2, c_2, d_2\}$. Again, the algorithm searches for all combinations of $\{a_2, b_2, c_2, d_2\}$ and determines, for each combination, the optimal scaling factors by using equation (7) and the error by using equation (5). The values of $\{a_2, b_2, c_2, d_2\}$ that minimize the error are kept.

To find the optimal box filter approximation, the process described is repeated until the approximation error in (5) is below a threshold, or the maximum number of boxes is achieved.

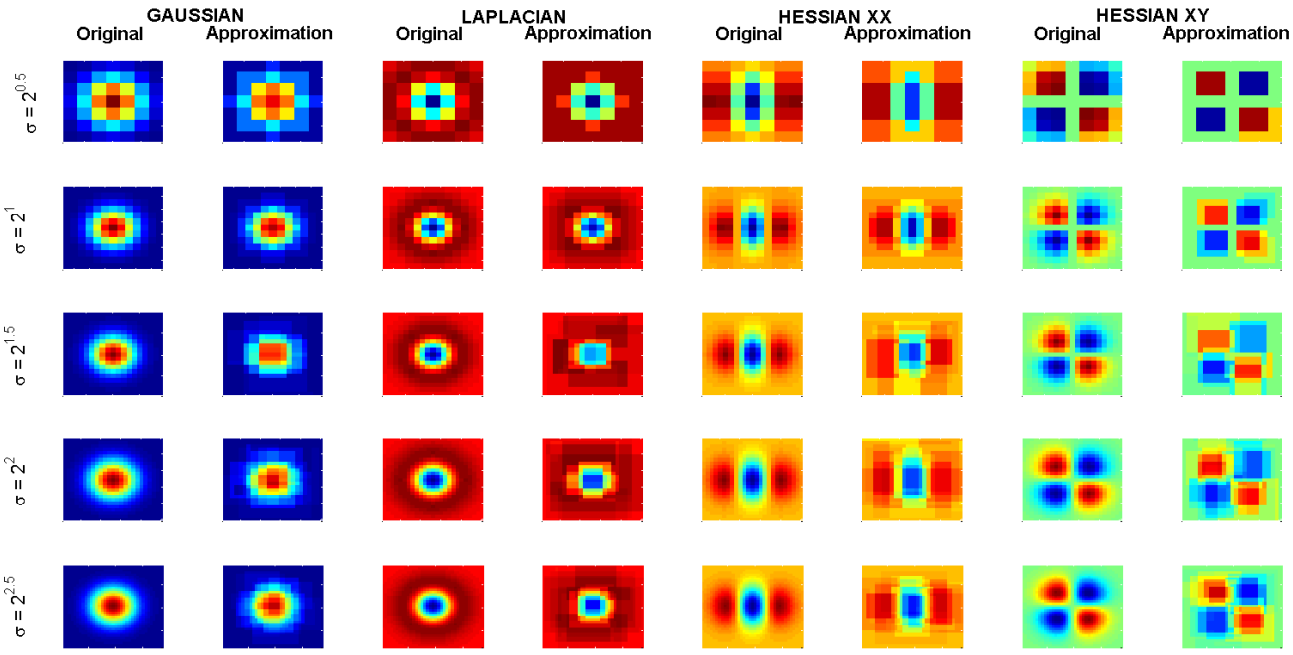

Fig. 3. Experimental Results

3.2. Directed Search

As an alternative to the Exhaustive Search algorithm, we develop an iterative Directed Search algorithm. At each step k of the algorithm, we assume that we have a parameter configuration $\theta = \theta^k$, i.e.:

$$\{\alpha_i, a_i, b_i, c_i, d_i\}_{i=1}^{N} = \{\alpha_i^k, a_i^k, b_i^k, c_i^k, d_i^k\}_{i=1}^{N} \tag{10}$$

To move to step $k+1$, we compute the finite difference in the error function $E(\theta)$ with respect to each of the corner points and move each corner point by a fixed amount in the direction that reduces the error function.

The finite difference in the error function with respect to corner point a_i corresponds to the change in the error function when all parameters are kept constant and a_i is increased by 1. So, if the set of parameters $\theta_{a_i}^k$ is defined to be equal to θ^k except that the parameter a_i is increased from a_i^k to a_i^k+1, then the finite difference in the error function at point θ^k with respect to the corner point a_i is equal to:

$$\Delta_{a_i} E(\theta^k) = E(\theta^k) - E(\theta_{a_i}^k) \tag{11}$$

Using algebraic manipulations, it can be shown that the finite difference with respect to each of the corner points is equal to (where we omit the superscript k on all parameters for compactness):

$$\begin{aligned}
\Delta_{a_i} E(\theta) &= 2\alpha_i \textstyle\sum_{y=c_i}^{d_i} \left(G(a_i, y) - B_N(a_i, y) + \frac{\alpha_i}{2}\right) \\
\Delta_{b_i} E(\theta) &= 2\alpha_i \textstyle\sum_{y=c_i}^{d_i} \left(B_N(b_i, y) - G(b_i, y) - \frac{\alpha_i}{2}\right) \\
\Delta_{c_i} E(\theta) &= 2\alpha_i \textstyle\sum_{x=a_i}^{b_i} \left(G(x, c_i) - B_N(x, c_i) + \frac{\alpha_i}{2}\right) \\
\Delta_{d_i} E(\theta) &= 2\alpha_i \textstyle\sum_{x=a_i}^{b_i} \left(B_N(x, d_i) - G(x, d_i) - \frac{\alpha_i}{2}\right)
\end{aligned} \tag{12}$$

4. PROPOSED ALGORITHMS

4.1. Exhaustive Search

The algorithm for Exhaustive Search method is the following:

1. Start with $N = 0$.
2. Make $N = N + 1$.
   Keep all values of $\{a_i, b_i, c_i, d_i\}_{i=1}^{N-1}$ constant.
   For all combinations of $\{a_N, b_N, c_N, d_N\}$:
   2.1 Use eq. (7) to compute optimal values of $\{\alpha_i\}_{i=1}^{N}$.
   2.2 Use eq. (5) to compute current error $E(\theta^N)$.
3. Stop if error $E(\theta^N)$ below threshold.
   Otherwise return to Step 2.

The value of N in the algorithm corresponds to the number of boxes in the approximation (note that, if necessary, one may additionally define a threshold on the number of boxes used).

As discussed in Section 2, it is not possible to use Equation (7) when the matrix $\boldsymbol{A}$ is not invertible. Since matrix $\boldsymbol{A}$ depends solely on the configuration of the corner points, the configurations that lead to non-invertible matrices (for example, when two boxes perfectly overlap) are discarded.

4.2. Directed Search

In each initialization of the Directed Search, we:

1. Start with $N = 0$.
2. Make $N = N + 1$.
   Initialize $\{a_N, b_N, c_N, d_N\}$ randomly.
   Repeat until convergence:
   2.1 Use eq. (12) to determine finite differences for all corner parameters $\{a_i, b_i, c_i, d_i\}_{i=1}^{N}$.
   2.2 For each corner parameter, if finite difference is above threshold, take step in direction that minimizes error.
   2.3 Use eq. (7) to compute optimal values of $\{\alpha_i\}_{i=1}^{N}$.

3. Use eq. (5) to compute current error $E(\theta^N)$.
4. Stop if error $E(\theta^N)$ below threshold.
   Otherwise return to Step 2.

Again, note that the value of N above corresponds to the number of boxes in the approximation (and, again, one may define a threshold on the number of boxes used). Additionally, in a practical implementation one should limit the number of iterations inside the loop of Step 2. As before, Equation (7) cannot be used when the matrix $\boldsymbol{A}$ is not invertible. A solution for this problem is to re-start the algorithm when the configuration of corner points leads to such a matrix.

Since this is a greedy algorithm (similar to gradient search), there is no guarantee that the optimal minimum will be found. To deal with this problem we initialize the algorithm a large number of times and keep the best solution.

5. EXPERIMENTAL RESULTS

To test the performance of the proposed methods, we created approximations for some common 2-D filters used in image processing: the Gaussian filter $g_\sigma(x, y)$, the Laplacian filter $l_\sigma(x, y)$ and the filters for both the xx and the xy entries of the Hessian matrix $h_\sigma^{xx}(x, y)$ and $h_\sigma^{xy}(x, y)$ (note that the filter for the yy entry of the Hessian matrix is just the transpose of the filter for the xx entry). The formulas for the filters are as follows (were we omitted the dependence in (x, y) and the multiplicative constants for compactness):

$$\begin{aligned} g_\sigma &\sim \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) & l_\sigma &\sim g_\sigma \cdot \left(x^2+y^2-2\sigma^2\right) \\ h_\sigma^{xx} &\sim g_\sigma \cdot \left(x^2-2\sigma^2\right) & h_\sigma^{xy} &\sim g_\sigma \cdot (x \cdot y) \end{aligned} \tag{13}$$

Figure 3 presents the approximations obtained by using our method for the various values of the σ parameter, while Table 5 shows the time to compute each approximation and the error obtained. As the σ parameter increases, the size of the filter also increases. The filters are presented in color for clarity, each (x, y) entry of each filter is a real number (not a RGB value).

All approximations were constrained to generate a specific number of boxes N. For small filters (sizes up to 13×13), both algorithms were used to generate an approximation, and the best approximation was kept. For larger filters, only the Directed Search algorithm was used. In practice, we have observed that the Exhaustive Search almost always outperforms the Directed Search (but is limited by the fact that it can only be applied to small filters).

Visual inspection of Figure 3 shows that, in most cases, our algorithm produces very good approximations. This can be corroborated by looking at the error entries presented in the table. Here, for comparison purposes, the error function $E(\theta)$ was normalized by the 2-norm of the original filter. I.e., the error presented is defined as $E(\theta)/||G||$, where $||G||$ is the 2-norm of $G(x, y)$. Note that, to obtain better approximations, one can always increase the number of boxes used at the expense of making the filtering operation slower.

6. CONCLUSION

We have presented a method for automatically approximating an arbitrary 2-D filter by a box filter. To this end, we started by formulating the approximation as an optimization problem where one is trying to minimize the approximation error with respect to the parameters of the box filter: the corner point locations and the scaling factors.

	σ	$2^{0.5}$	2^{1}	$2^{1.5}$	2^{2}	$2^{2.5}$
	N	5	10	15	20	25
	Size	7×7	13×13	19×19	25×25	37×37
g_σ	Time (s)	0.80	18.67	21.97	40.73	79.64
	Error (%)	1.75	0.54	2.86	1.89	1.14
l_σ	Time (s)	0.78	18.73	25.24	48.54	99.27
	Error (%)	2.55	11.60	9.08	5.57	3.58
h_σ^{xx}	Time (s)	0.80	19.08	25.21	46.81	96.67
	Error (%)	3.77	1.47	10.94	6.36	4.29
h_σ^{xy}	Time (s)	0.76	18.49	25.40	45.71	95.01
	Error (%)	11.93	5.25	18.17	11.04	7.58

Table 1. Experimental results. First three rows show characteristics of the original filter that we are approximating. Subsequent rows show time to generate the box filter approximation and error of the approximation. All times were measured using a 2.26GHz Intel Core 2 Duo laptop with 3GB of RAM.

We have shown that, when one assumes fixed corner points, it is possible to find a closed form solution for the optimal value of the scaling factors. Armed with this knowledge, we develop two algorithms that determine the optimum locations for the corner points: the Exhaustive Search algorithm, that is well suited for approximating small filters; and the Directed Search algorithm, that is well suited to approximating large filters.

To test our algorithms, we generated approximations for commonly used filters including the Gaussian, the Laplacian, and the components of the Hessian matrix. Experimental results confirm the validity for the presented methods.

7. REFERENCES

[1] Franklin C. Crow, "Summed-area tables for texture mapping," in *Proceedings of SIGGRAPH*, 1984, vol. 18, pp. 207–212.

[2] Paul Viola and Michael J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[3] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, "SURF: Speeded up robust features," in *Proceedings of the European Conference on Computer Vision*, 2006, pp. 404–417.

[4] Qiang Zhu, Shai Avidan, Mei-Chen Yeh, and Kwang-Ting Cheng, "Fast human detection using a cascade of histograms of oriented gradients," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'06*, 2009.

[5] Oncel Tuzel, Fatih Porikli, and Peter Meer, "Region covariance: A fast descriptor for detection and classification," in *Proceedings of the European Conference on Computer Vision, ECCV'06*, 2009.

[6] Peter Kovesi, "Arbitrary Gaussian filtering with 25 additions and 5 multiplications per pixel," Tech. Rep. UWA-CSSE-09-002, The University of Western Australia, September 2009.

[7] Richard Szeliski, *Computer Vision: Algorithms and Applications*, Springer, First edition, 2010.

[8] Irving E. Segal and R. A. Kunze, *Integrals and Operators*, McGraw-Hill, 1968.