Design of a vanishing point algorithm for custom ASIC

M. Villemur*†‡, M. Di Federico*¶, P. Julián*†‡, A. G. Andreou‡, F. Masson*†, E. Nebot§
* Departamento de Ingeniería Eléctrica y de Computadoras, Universidad Nacional del Sur
Av. Alem 1253, Bahía Blanca, Argentina,
Email: martin.villemur@uns.edu
† IIIE-CONICET, Argentina
‡ Electrical and Computer Engineering Department, Johns Hopkins University, USA
§ University of Sydney, Australia
¶ INTI, Argentina

***Abstract*—In this paper we present a vanishing point algorithm variation oriented to a VLSI ASIC. We proposed a simplified voting process and analyze the minimum resolution that can be used for the input image and the filter kernels in order to obtain a good performance.**

***Index Terms*—Vanishing point detection, VLSI, ASIC**

I. Introduction

This paper presents a variation of a vanishing point detection algorithm specifically targeted to achieve a custom Application Specific Integrated Circuit (ASIC). The methods in the literature usually implement a directional filtering using Gabor filters to associate a preferred orientation to every pixel, and then perform a voting process to choose the most voted pixel, i.e., the vanishing point of the image.

For example, [1] proposes a method based on the texture of the terrain using Gabor filters in order to obtain the "dominant orientation" of every pixel. The size of the kernel is determined empirically as a function of the wavelength of the Gabor filters, which in turn are obtained based on the size of the image. This work uses 72 possible orientations on 320×240 resolution images. All pixels vote, and the voting process is binary (not weighted). As a consequence of the high number of orientations and the participation of all the pixels in the voting process, the method is computationally expensive and somehow affected by far away noise.

[2] uses 36 orientations with five different filter scales in order to consider different spatial resolutions. The dominant orientation is obtained by getting the maximum response value among the five resolution filters at every pixel. In addition, a normalized confidence is assigned to each pixel based on the response of all filters at that location. This confidence is used to determine whether the pixel is allowed o vote or not. The voting is done using a Local Adaptive Soft-Voting (LASV) and the voting regions is a half disc below the candidate. The size of the disk is empirically selected to be $0.35 \times \Upsilon$, where Υ is the length of the image diagonal. The voting is not binary. Every voter is weighted according to the alignment with the dominant orientation and the distance to the candidate. Even though the voting scheme is improved with respect to [1], the computational complexity is still high.

[3] uses four filtering orientations and computes the dominant orientation based on the joint activity of them over every pixel. The orientation is then calculated using a tan^{-1} operation between the imaginary and real parts of the joint activity. This method is called Optimal Local Dominant Orientation Method (OLDOM). The voting is done based on the Euclidian distance between the candidate and the voted, and also considering the difference between the dominant orientation and the line that joins the candidate and the voter. The voting is not binary and all pixels in the image vote. The resolution used for the images is 320×240. This method achieves more accuracy in the voting but at the expense of an increased computational load. The number of orientations is low, but on the other hand an inverse tangent calculation is required, which requires a specialized VLSI block with high precision.

In order to implement a vanishing point algorithm on an ASIC, the size and number of kernels, and the resolution in bits of the kernels and the input image must be minimized in order to achieve a reasonable Silicon area. In order to do this, we perform an analysis of the impact of these parameters on the location of the vanishing points. In addition, we propose several simplifications that help achieving a more compact realization.

II. Proposed vanishing point algorithm

A. Texture flow estimation

The first stage of the algorithm is texture flow estimation. The input image $I(x,y)$ is convolved by $2 \times n$ Gabor filters: n of them are odd Gabor filters, namely, (g_{odd}), and the other n are even Gabor filters, namely g_{even}.

For every pixel $p = (x,y)$, the orientation that has the strongest Gabor energy

$$I_e(x,y) = (g_{even} * I)(x,y)^2 + (g_{odd} * I)(x,y)^2 \qquad (1)$$

among all filtered images, is referred to as the *dominant orientation* $\theta(p)$ of pixel p. Based on this concept, two different images are generated. The first image $I_\theta(x,y)$ is

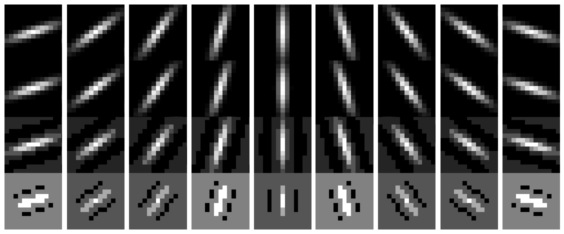


Figure 1: Real Gabor filters, with $n = 9$ orientations; for 4 different levels of quantization: 8, 6, 4, and 3 bits from the top to the bottom.

composed of the dominant orientation and the second image $I_e(x, y)$ is composed of the corresponding modules $I_e(x, y)$.

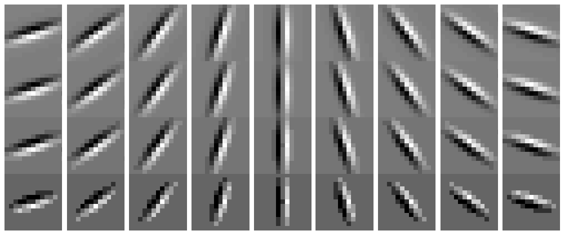


Figure 2: Imaginary Gabor filters, with $n = 9$ orientations; for 4 different levels of quantization: 8, 6, 4, and 3 bits from the top to the bottom.

At this stage, a technique usually found in the literature is the definition of a "confidence rating" for the orientations. For example, in [2] all pixels are assigned a confidence level based on a statistical function over all Gabor responses. However, this is very expensive for an ASIC implementation since all $n \times q_p$ values must be saved on memory.

The Gabor filters are defined by the following equations [4]:

$$g_{even}(x, y, \theta, \lambda, \gamma) = \frac{e^{\frac{-1}{8\sigma^2}\left(\gamma\, {x_\theta}^2 + {y_\theta}^2\right)}}{\sqrt{2\pi}\sigma} \cos\left(2\pi x_\theta/\lambda + \psi\right) \quad (2)$$

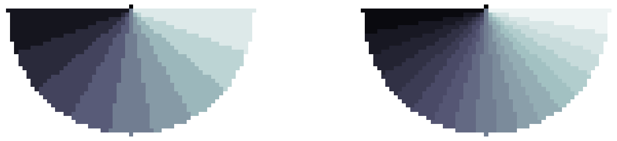
$$g_{odd}(x, y, \theta, \lambda, \gamma) = \frac{e^{\frac{-1}{8\sigma^2}\left(\gamma\, {x_\theta}^2 + {y_\theta}^2\right)}}{\sqrt{2\pi}\sigma} \sin\left(2\pi x_\theta/\lambda + \psi\right) \quad (3)$$

where θ is the filter orientation, $a = x\cos\theta + y\sin\theta$, $b = -x\sin\theta + y\cos\theta$, λ is the wavelength, σ is the gaussian deviation, γ is the x-y aspect ratio of the gaussian surface.

Coefficients λ, σ and γ do not influence the hardware requirements, so they are not analyzed in this paper.

In order to specify the number of filters, which is equivalent to the number of angles or orientations n, several aspects must be considered. Lower resolution images have fewer possible orientations and fewer bits per pixels, resulting in lower computational load. Similarly, lower resolution kernels also reduce the computational load. However, as the resolution of the image/filter kernel reduces, the algorithm performance degrades up to a point where the detection is no longer possible. In addition, the computation of the convolution operation required to filter the image grow linearly with the number of orientations. This result in longer execution time. The necessary memory also grows with the number of orientations. In fact, if there are n possible orientations then $\log_2(n)$ bits must be stored per pixel. This produces more silicon area.

Regarding the kernel size k, the number of computations grows quadratically with the number of elements, and the execution time increases accordingly. On the other side, if the kernels are too small, small texture orientations might contribute at the voting stage, adding undesired noise. Figures 1 and 2 show the real and imaginary filter outputs, considering nine orientations, with a kernel size $k = 13$, and different levels of quantization. The parameter k is determined according to [1] as $k = \frac{10}{\pi} \times 2^{(log_2(w)-5)}$, where w is the input image width. Parameters $\lambda = \frac{k \times \phi}{5}$, $\sigma = 0.15 \times k$ and $\gamma = 20$, are chosen based on the type of images that are being analyzed.


(a) $n = 9$.

(b) $n = 17$.

Figure 3: Example of pattern matrix with radius of 30 pixels for different number of orientations. Equal color represents equal orientation, starting from the first orientation, in dark, until the last orientation (9 or 17), in light-gray

As it can been seen from Figs. 1 and 2, real filters degenerate when 3 bits are used. At first glance, it does not seem recommendable to implement the filters with less than 4 bits.

B. *Voting process*

Once the dominant orientation image $\theta(p)$ has been computed, the voting process can be started. As was mentioned on previous work ([2], [1], [5]) a "hard-voting" scheme will produce a large error in the estimation of the vanishing points, and also will require a high computational cost. Therefore,

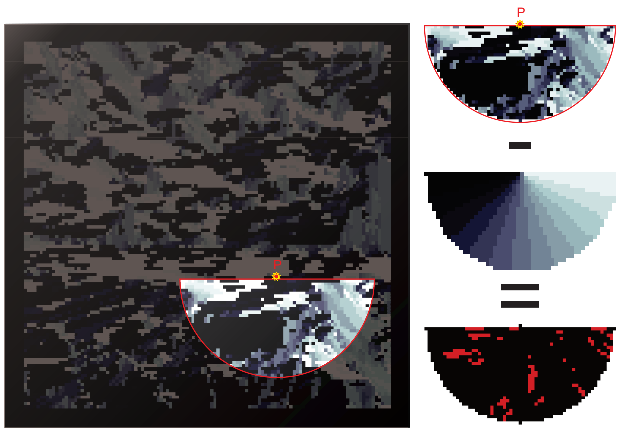


Figure 4: Voting implementation example for one candidate P.

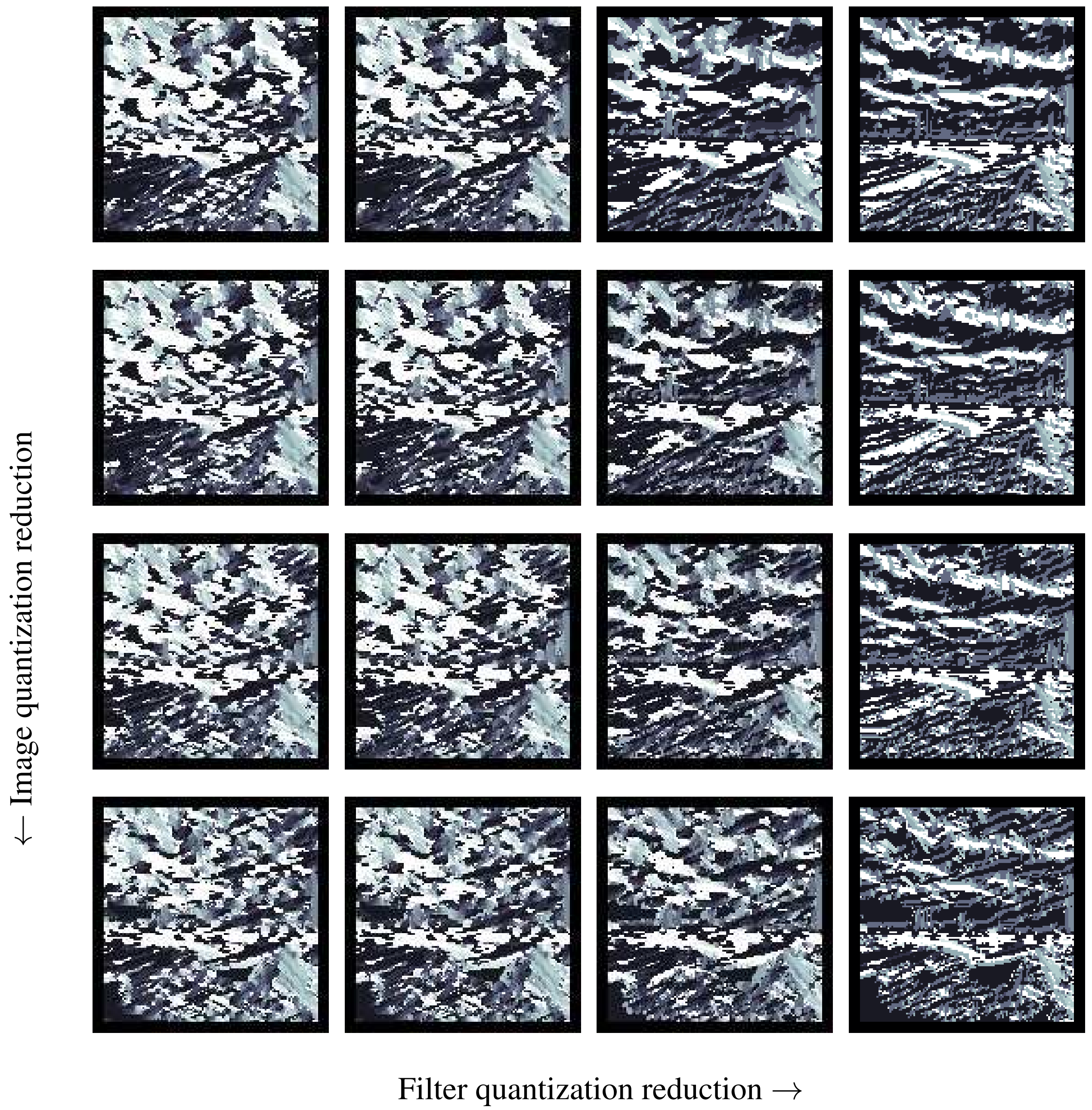

← Image quantization reduction
Filter quantization reduction →

Figure 5: Orientation images for different input image quantization (8, 5, 4 and 3 bits) and different filter quantization (8, 6, 4 and 3 bits).

we implement a locally adaptive soft-voting (LASV) scheme based on the method proposed in [2]. Basically, all pixels $p(x, y)$ from a certain area are considered potentially vanishing points (PF) of the image. Then, for each candidate $P(x, y)$, we count the number of pixels $p(x, y)$ belonging to a voting region V_r whose straight line contains $P(x, y)$. In this particular case, the voting region was selected as a half-disk below $P(x, y)$ centered at it. As a result of this procedure, a bi-dimensional map is generated that contains the voting results.

Unlike other algorithms proposed before, this voting stage does not take into account distance between candidates and their voters, nor any function using angle, distance or confidence levels weighting. Therefore, each voting score is based only on the pixel predominant texture.

A more convenient implementation of LASV for the case of a custom IC is proposed that uses an auxiliary pattern matrix M_p. Each value of M_p is the encoded dominant orientation a voter must have to be added on the voting process.

Figure 3 shows two different pattern matrices, with two different number of orientations (9 and 17 respectively). In order to calculate the voting score of a candidate P, the

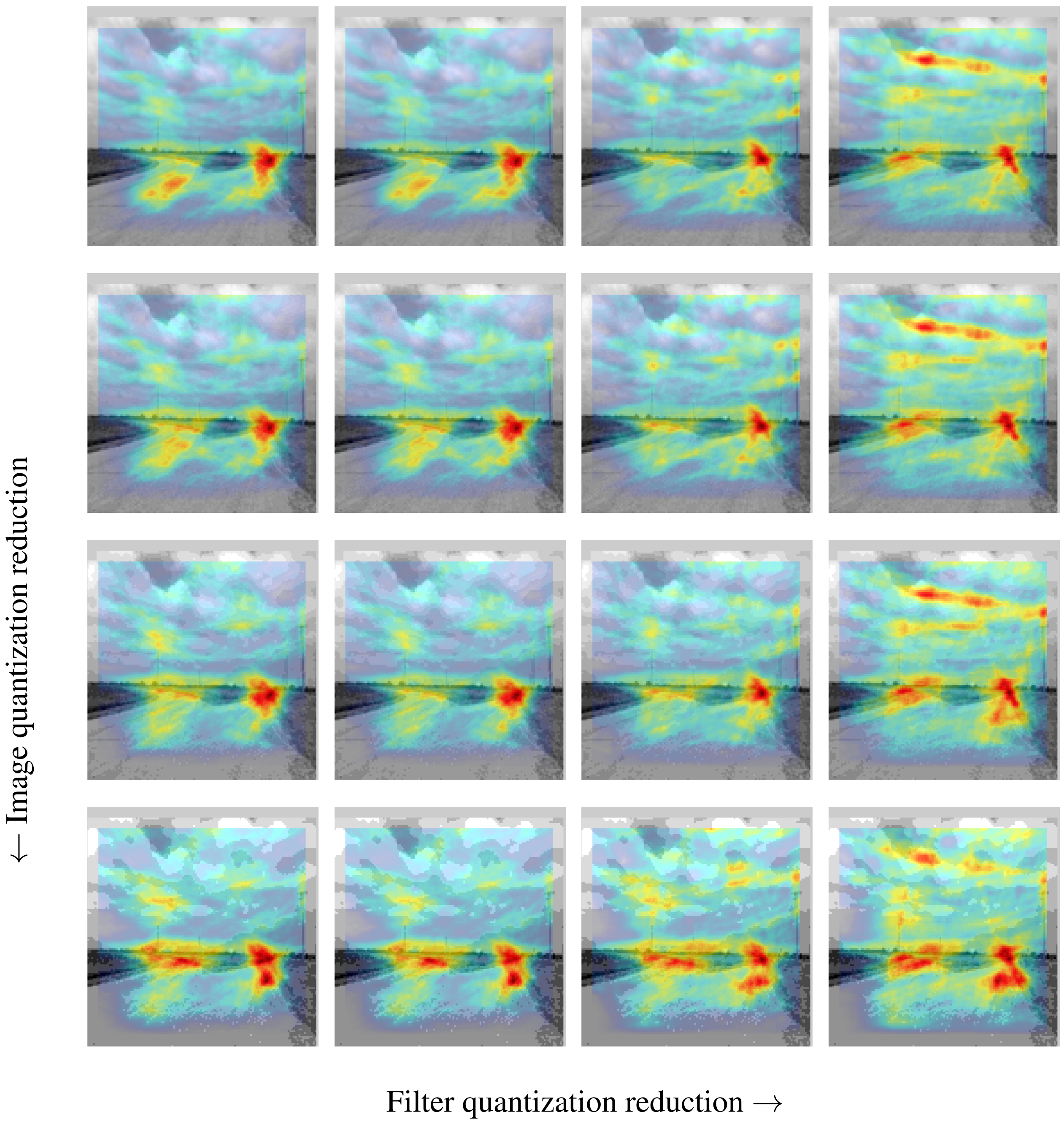


Figure 6: Voting maps for different input image quantization (8, 5, 4 and 3 bits) and different filter quantization (8, 6, 4 and 3 bits).

corresponding part of the orientation image is subtracted from the pattern matrix M_p (see Fig. 4). Then, the number of zeros is counted. This process is repeated over all the possible candidates, so the voting map is generated by performing several times (depending on the number of candidates) a subtraction between two $2r \times r$ size matrix followed by an $2r^2$ search for zeros in the resulting matrix.

C. *Vanishing point selection*

The last stage of the algorithm uses the voting map generated on the previous stage to choose the vanishing point among candidates. This is done by finding the largest values in the map. However, modifications can be introduced if, for example, several local maxima must be found in cases with more than one vanishing points (e.g., more than one road).

III. Resolution analysis

In order to find the simplest possible realization, the results of the algorithm must be evaluated for different image and filter resolutions. At first, the resolution per pixel is the maximum that the software can handle, typically, floating point resolution. Afterwards, both, image and filter kernel resolution are represented with a reduced number of bits and the results are compared.

Figures 5 and 6 show the orientation map and the vote map for a given image containing a road with a bifurcation. These two figures show the results of using 8, 5, 4 and 3 bits for the input image representation and 8, 6, 4 and 3 bits for the filter kernel coefficients. As can be observed from Fig. 6, the performance deteriorates when the filter kernels and the image are represented with fewer bits. However, by using just 4 bits for the image and the filter kernels, the results are comparable to those obtained with full resolution. In this particular case, the filter kernel resolution has a greater influence than the image resolution on the voting map. The change from 6 to 4 bits produces a region of potential vanishing points appearing on the left side of the image, corresponding to the left road in the bifurcation.

The complexity in terms of the number of multipliers, adders and registers required are described in The Appendix.

IV. Conclusion

We have presented an algorithm for vanishing point detection that can be realistically implemented on an ASIC. We have proposed a simplified voting procedure using a pattern matrix, that only requires a substraction followed by the count of the number of resulting zeros. We have also performed a numerical analysis that shows that using 4 bits to represent the input image and the filter kernels is enough to produce similar results to the full resolution case. The algorithm has been tested with a variety of different images, and the results are the same. Only one image is show in the paper due to the lack of space. The Appendix shows the number of multipliers, adders, registers which are needed to be implemented depending on the architecture. There is a clear trade-off between space and the number of cycles to complete the operation. The immediate future steps are the design of an ASIC and its fabrication.

Acknowledgment

This project was partially funded by the International Project PICT 2010 No. 2657 "3D Gigascale Integrated Circuits for Nonlinear Computation, Filter and Fusion with Applications in Industrial Field Robotics", ANPCyT, Ministry of Science and Technology of Argentina.

V. The Appendix

A. Computation complexity

This section presents a brief analysis of the computational cost of filtering the image with the filter kernels using different architectures. The image is processed using a convolution of the form:

$$f[x,y] = i * h = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} h[i,j]\, i[x+i,y+j], \qquad (4)$$

where $[x,y]$ is the location of pixels in the image, $i[x,y]$ is the input image, $h[x,y]$ is the filter kernel, k is the kernel size and h is the image size. The convolution requires $k^2 \times h^2$ multiplications and additions. This operation can be performed using one multiplier for the whole array, k multipliers, k^2 multipliers or one multiplier per row. Assuming a 128×128 image and a kernel of size $k = 13$, Table I summarizes the number of multipliers, adders, registers and cycles required to execute the filter.

Architecture	Multipliers	Adders	Registers	Read cycles
1 multiplier	1	1	2	2768869
k multipliers	13	26	52	212992
k^2 multipliers	169	171	342	16384
1 mult. per row	128	128	256	21632

Table I: Summary of transducers and their characteristics considered in the phantom tests.

References

[1] C. Rasmussen, "Grouping dominant orientations for ill-structured road following," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1. IEEE, 2004, pp. I–470.

[2] H. Kong, J.-Y. Audibert, and J. Ponce, "General road detection from a single image," *Image Processing, IEEE Transactions on*, vol. 19, no. 8, pp. 2211–2220, 2010.

[3] P. Moghadam, J. A. Starzyk, and W. S. Wijesoma, "Fast vanishing-point detection in unstructured environments," *Image Processing, IEEE Transactions on*, vol. 21, no. 1, pp. 425–430, 2012.

[4] T. S. Lee, "Image representation using 2d gabor wavelets," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, no. 10, pp. 959–971, 1996.

[5] Y. Wang, X. Wang, and C. Wen, "Fast vanishing point detection for unstructured road using haar texture," in *Signal Processing, Communication and Computing (ICSPCC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 167–170.