FULL QUADRANT APPROXIMATIONS FOR THE ARCTANGENT FUNCTION

Xavier Gironés, Carme Julià and Domenec Puig

This article presents two novel full quadrant approximations for the arctangent function that are specially suitable for real time applications. The key point of the proposed approximations is that they are valid in a full quadrant. As a result, they can be easily extended to two and four quadrants. The approximations we define are rational functions of second and third order, respectively. This article provides a comparison of the precision and performance of the proposed functions with the best state of the art approximations. Results show that the third order proposed function outperforms the existing ones in terms of both precision and performance. The second order proposed function, on the other hand, is the most suitable one for real time applications, since it has the highest performance. Furthermore, it attains an adequate precision for most applications in the computer vision field.

INTRODUCTION

The approximations proposed in this article are introduced to be used in a well-known application in the computer vision field: object recognition. In particular, the arctangent function is required to compute the gradient orientations used in some widespread feature descriptors such as Scale-independent Feature Transform (SIFT) [1] or the Histograms of Oriented Gradient (HOG) [2]. Other popular techniques can benefit from an improved inverse tangent approximation: the performance of the Hough transform, for instance, is significantly enhanced when gradient orientations are considered during the voting process [3].

Since gradient orientations are usually calculated for all pixels in the image, the key point for an arctangent approximation suitable for computer vision applications is featuring a low computational complexity. Furthermore, a low precision approximation should be adequate for most applications in this field, as images are usually quantized in the $[0, 255]$ interval (8 bit pixel depth). Hence, if we consider the maximum pixel quantization error in the image to be $\varepsilon_{\text{image}} \approx 0.5$, the centered gradient calculated employing a $[-1\ 0\ 1]$ mask will lie in the $[-255, 255]$ interval, with a maximum absolute error $\varepsilon_{\text{gradient}} \approx 1$. Therefore, the maximum absolute error in the gradient orientation for a given pair of horizontal and vertical gradient values (u, v) can be defined as

$$\delta(u,v) = \max_{|\varepsilon_x|,|\varepsilon_y|<1} \left\{ \left| \tan^{-1}\left(\frac{v+\varepsilon_y}{u+\varepsilon_x}\right) - \tan^{-1}\left(\frac{v}{u}\right) \right| \right\}. \tag{1}$$

In this case, the maximum resolution that can be achieved in the calculation of the gradient angle is the solution of the minimax problem

$$\varepsilon_{\measuredangle} = \min_{|\nabla x|,|\nabla y|\leqslant 255} \{\delta(\nabla x, \nabla y)\} = \delta(\pm 255, \pm 255) \approx 0.22^{\circ}. \tag{2}$$

Moreover, orientations are usually accumulated into histograms at a much coarser resolution. For instance, [2] uses 9 bins evenly spaced over $[0^{\circ}, 180^{\circ}]$. In view of these results, employing the 24 bit precision arctangent available in many generic numerical libraries to calculate gradient orientations in 8 bit pixel depth images is certainly overkill.

Different approximations for the arctangent function have been proposed in the literature, some of them suitable for standard numerical packages: Kogbetliantz [4] suggests a set of rational and polynomial approximations for single and double precision. Koren and Zinaty [5] provide double precision rational approximations for some transcendental functions. In the context of approximations oriented towards the signal processing field, Lyons [6] proposes a very efficient rational approximation with a maximum error of 0.26° in the angular range of $[-45^{\circ}, 45^{\circ}]$. Rajan *et al.* [7] provide additional polynomial and rational approximations using Lagrange interpolation and minimax optimization techniques. Concretely, the following rational approximation is of especial interest:

$$\tan^{-1}(x) \approx \frac{x}{1 + 0.28086\, x^2} \qquad -1 \leqslant x \leqslant 1. \tag{3}$$

This approximation is appropriate for image processing in several aspects: it achieves a precision of 0.2683°, which is in the order of (2), through a second order rational function with only a single constant to store. However, the fact that its range is only optimized for the interval $[-\pi/4, \pi/4]$ radians means that the identity

$$\tan^{-1}(x) = \begin{cases} -\frac{\pi}{2} - \tan^{-1}\left(\frac{1}{x}\right) & x < 0, \\ \\ \frac{\pi}{2} - \tan^{-1}\left(\frac{1}{x}\right) & x > 0, \end{cases} \tag{4}$$

must be employed to extend the angular range to the interval $[-\pi/2, \pi/2]$. Unfortunately, this step induces an implicit branching that makes it less convenient for an efficient *Single Instruction Multiple Data* (SIMD) implementation.

The main objective of this work is to propose an approximation for the arctangent function more suitable for real time image processing than the existing ones. In particular, the aim

is to use it to calculate HOG descriptors. The key idea under the proposed function is to obtain a *full quadrant* approximation (i.e., defined in the angular range $[0°, 90°]$) with at least the same order of precision as (3) and a similar computational complexity. This function can be easily extended to two or four quadrants on account of the *oddness* of the arctangent function (i.e., $\tan^{-1}(x) = -\tan^{-1}(-x)$).

PROPOSED APPROXIMATIONS

The goal is to compute the angle $\theta = \tan^{-1}(x)$, where $x, \theta \in \mathbb{R}$. This function takes values in $[-\pi/2, \pi/2]$ and, in general, two different quadrants (or four octants) are studied. This article proposes to approximate the *normalized arctangent*. That is, in the case of the first quadrant, instead of approximating the range of the function in $[0, \pi/2]$, the interval $[0, 1]$ is considered. The product by $\pi/2$ can be done later, if necessary. Therefore, if $\phi(x)$ is the normalized arctangent approximation defined in the domain $[0, \infty)$, the identity $\tan^{-1}(x) = -\tan^{-1}(-x)$ can be used to extend it to the full domain $(-\infty, \infty)$:

$$\tan^{-1}(x) \approx \frac{\pi}{2} \operatorname{sgn}(x) \phi(|x|). \tag{5}$$

The approximations studied in this work are rational functions with the following structure:

$$\phi_{m,n}(x) = \frac{P_m(x)}{Q_n(x)} = \frac{\sum_{i=0}^{m} a_i x^i}{\sum_{j=0}^{n} b_j x^j}, \tag{6}$$

where $m, n \in \mathbb{N}$, $a_i, b_j \in \mathbb{R}$, $a_m, b_n \neq 0$. Next, we define the error function as

$$\psi_{m,n}(x) = \frac{2\tan^{-1}(x)}{\pi} - \phi_{m,n}(x), \quad 0 \leqslant x < \infty. \tag{7}$$

The optimum parameters a_i, b_j are the ones that satisfy the minimax criterion

$$J_{m,n} = \min_{a_i, b_j} \left\{ \max_{0 \leqslant x < \infty} \{|\psi_{m,n}(x)|\} \right\}. \tag{8}$$

This problem could be solved using standard methods such as the Remez algorithm. However, the arctangent function in the domain $[0, \infty)$ has some properties that enable the calculation of the a_i, b_j coefficients in a much simpler way:

1. $\tan^{-1}(0) = 0$, which implies that $a_0 = 0$.
2. $\lim_{x\to\infty} 2\tan^{-1}(x)/\pi = 1$, which is equal to 1 if and only if $a_m = b_m$. Recall that the arctangent is being approximated by a rational function (see equation (6)). It is well known that the limit at infinity of a rational function depends on the degree of the polynomials in the numerator and denominator. Concretely, that limit is not null and smaller than infinity if and only if the degree of both polynomials are equal, which implies that $m = n$, in this case. Moreover, this limit is equal to the ratio of the coefficients of the largest degrees of the numerator and denominator. That is:
$$\lim_{x\to\infty} \phi_{m,n}(x) = \frac{a_m}{b_m}, \tag{9}$$
from which it can be assumed that $m = n$, and $a_m = b_m$. Then, the expression (6) can be divided by a_m, giving $a_m = b_m = 1$. Since $m = n$, we can rename $\phi_{m,n}$ as ϕ_n.
3. From the identity in (4): $\phi_n(x) + \phi_n\left(\frac{1}{x}\right) = 1$.
4. $\phi_n(x)$ can have neither real poles nor zeros in the interval $(0, \infty)$.

Henceforth, the first, second and third order approximations for the normalized arctangent $2\tan^{-1}(x)/\pi$ will be calculated by enforcing the aforementioned constraints.

The first order approximation ϕ_1, which is described in [8], arises immediately from the first three conditions and it has the following form:

$$\phi_1(x) = \frac{x}{1+x}. \tag{10}$$

The obtained error is shown in Figure 1. It can be seen that the maximum angle approximation error is about 4^o, which is too coarse for the intended use of calculating HOG descriptors.

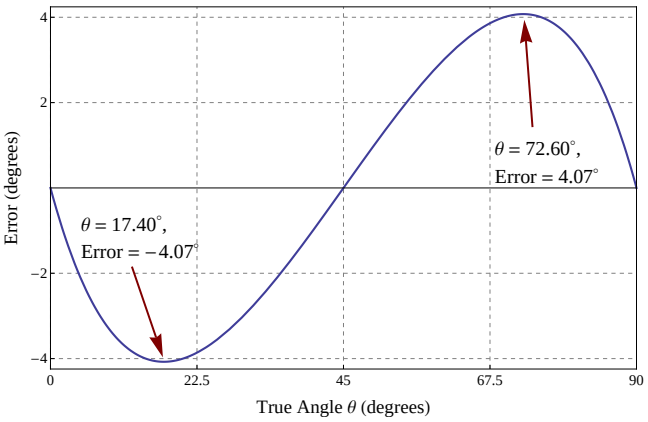

Fig. 1. Approximation error (degrees) obtained by using the function (10), which is proposed in [8].

For the second order approximation, the rational function obtained when enforcing the first and second constraints has the form

$$\phi_2(x) = \frac{P_2(x)}{Q_2(x)} = \frac{a_1 x + x^2}{b_0 + b_1 x + x^2}. \tag{11}$$

Now, applying the third constraint on (11), the following equation results:

$$Q_2(x)\,Q_2\left(\frac{1}{x}\right) - P_2(x)\,Q_2\left(\frac{1}{x}\right) - P_2\left(\frac{1}{x}\right)Q_2(x) = 0, \tag{12}$$

which leads to a system of equations in the polynomial coefficients. The only solution that holds constraint 4 has only one free parameter, and has the form

$$\phi_2(x) = \frac{Bx + x^2}{1 + 2Bx + x^2}. \tag{13}$$

Solving (8) yields a value for $B \approx 0.596227$, which corresponds to a maximum angle approximation error of $0.1620°$. Figure 2 shows the error obtained in the angular domain $[0, \pi/2]$ for both the proposed function (solid line) and the function introduced in [7].

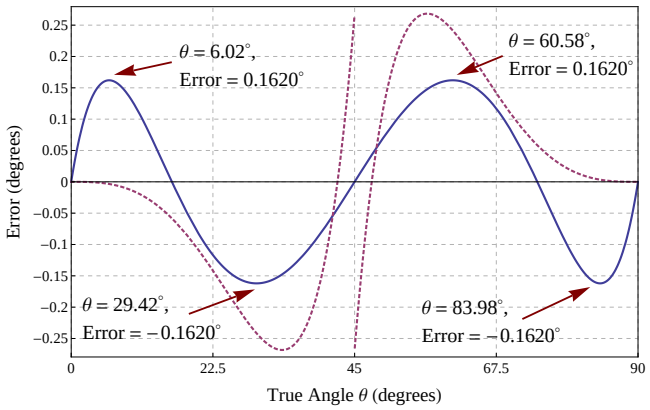

Fig. 2. Approximation error (degrees) obtained by using the proposed function (13). The dotted line corresponds to (3) (introduced in [7]), which has been extended to the $[0, \pi/2]$ range through the identity (4).

An extension of (13) to the $[-\pi/2, \pi/2]$ based on (5) can be implemented very efficiently for numbers expressed in the IEEE 754 representation. As shown in Code 1, the sign of the input parameter x is first determined using a bitwise *AND* operation. Next, the corresponding arctangent in the first quadrant is computed through the absolute value of x, and lastly the sign bit is restored through a bitwise *OR*. For performance reasons the final product by $\pi/2$ is left up to the caller because in many cases it can be aggregated to other constant factors thus saving an operation.

Caution should be taken when employing the function in Code 1 as in some systems the *float* type might not be in the IEEE representation, or not feature the same endianness as *uint32_t*. In addition, the *cast to reference* trick employed for reinterpreting the type of the variables at convenience breaks the strict aliasing rule: the behavior of dereferencing a pointer that aliases another of an incompatible type is undefined, and such assignments could be optimized away by some compilers.

Depending on the final application, a more precise approximation should be required. In this case, we consider third order polynomials to define the rational function, which has the form presented below, once the first and second constraints have been enforced:

$$\phi_3(x) = \frac{P_3(x)}{Q_3(x)} = \frac{a_1 x + a_2 x^2 + x^3}{b_0 + b_1 x + b_2 x^2 + x^3}. \tag{14}$$

As in the previous case, after enforcing the remaining constraints, the resulting function only depends on a single free parameter

$$\phi_3(x) = \frac{Cx + x^2 + x^3}{1 + 4C^2 x + 4C^2 x^2 + x^3}. \tag{15}$$

Solving (8) yields $C = 4C^2 - 1$, from which a closed algebraic representation for C is obtained: $C = \frac{1+\sqrt{17}}{8}$. Using this value for the C parameter, (15) can be reformulated as

$$\phi_3(x) = \frac{Cx + x^2 + x^3}{1 + (C+1)\,x + (C+1)\,x^2 + x^3}, \tag{16}$$

which in turn can be factorized in

$$\phi_3(x) = \left(\frac{x}{1+x}\right) \cdot \left(\frac{C + x + x^2}{1 + Cx + x^2}\right). \tag{17}$$

In this third order rational function, the computational cost is higher, but the maximum angle approximation error is far smaller than in the previous approximation: $0.00811°$. The ensuing approximation error is shown in Figure 3.

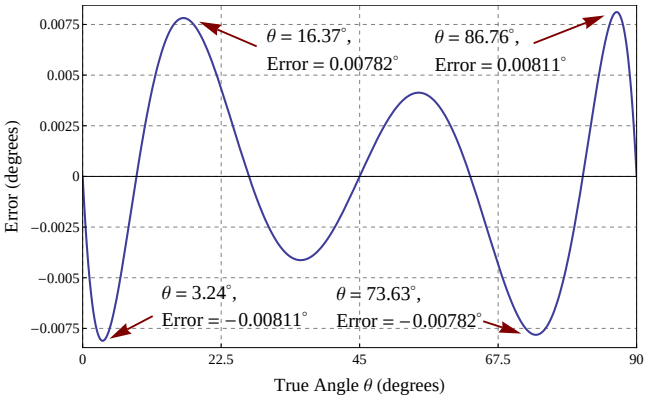

Fig. 3. Approximation error (degrees) obtained by using the proposed function (16).

FOUR QUADRANT APPROXIMATION

This section presents the extension of the second order proposed function to four quadrants. The third order approximation can be extended as well following a similar reasoning, if required.

The regular arctangent function does not distinguish between two vectors in opposite directions. However, many applications in computer vision involving 2D vectors in Euclidean space require taking into account the signs of both vector components and determining the correct quadrant for the angle. The SIFT descriptor [1], for instance, accumulates signed gradients in an orientation histogram comprising 36 bins covering the 360 degree range of orientations. In addition, including sign information in the HOG descriptor does help substantially in some object recognition tasks [2].

It is possible to extend (13) to take a 2D vector by substituting y/x for the independent term, which results in the following function:

$$\phi_2(x,y) = \frac{Bxy + y^2}{x^2 + 2Bxy + y^2}. \tag{18}$$

One of the main advantages of this function is that it approximates the two-argument normalized arctangent on the complete first quadrant (origin is not considered). Previous approaches (e.g., [6], [7]) determine one octant first by checking the condition $y > x$ and require a different function for each half of the quadrant. Another advantage of the proposed approximation is that the term x^2+y^2 in the denominator can be reused for calculating the magnitude, which in the case of HOG is needed for thresholding and determining the voting weights in the histogram.

Insofar as the idea is to compute the Histograms of Oriented Gradients, it is more practical to deal with positive numbers. Hence, rather than extending the function in the angular range $(-\pi, \pi]$, the interval $[0, 2\pi)$ is considered. Furthermore, since it is more efficient to compute the normalized arctangent, the range $[0, 4)$ is used instead.

As depicted in Table 1, the definition of the four quadrant arctangent $tan^{-1}_{[0,4)}$ depends on the quadrant where the pair (x, y) is located.

Table 1. Definition of the $tan^{-1}_{[0,4)}$ function for each quadrant.

Quadrant	$\mathbf{y < 0}$	$\mathbf{x < 0}$	$\mathbf{tan^{-1}_{[0,4)}}$
1	False	False	$\phi_2(\lvert x\rvert, \lvert y\rvert)$
2	False	True	$2 - \phi_2(\lvert x\rvert, \lvert y\rvert)$
3	True	False	$2 + \phi_2(\lvert x\rvert, \lvert y\rvert)$
4	True	True	$4 - \phi_2(\lvert x\rvert, \lvert y\rvert)$

Upon careful inspection, a unique definition can be generalized to the four quadrants in terms of $\phi_2(|x|, |y|)$ and the signs of x and y:

$$tan^{-1}_{[0,4)}(x,y) \approx \mathrm{D}(x,y) + (1 - 2\,\mathrm{sb}(xy))\,\phi_2(|x|, |y|), \tag{19}$$

where $\mathrm{D}(x, y)$ represents the offset to be added to the two quadrant arctangent (0, 2 or 4), which in turn depends on the signs of x and y:

$$\mathrm{D}(x,y) = 2\,\mathrm{sb}(x) + 4\,\mathrm{sb}(y)\,(1 - \mathrm{sb}(x)), \tag{20}$$

and $\mathrm{sb}(t)$ is a helper function used to determine whether a real number is non-negative or not, and defined as

$$\mathrm{sb}(t) = \begin{cases} 0 & t \geqslant 0, \\ 1 & t < 0. \end{cases} \tag{21}$$

Similarly to the case of the one-argument arctangent, for numbers in the IEEE 754 representation functions (20) and (21) can be reformulated more compactly by employing bitwise boolean operations and logical shifts on the sign bits of x and y (see Code 2). This makes them suitable for a SIMD implementation, as no conditional jumps are required.

COMPARISON WITH OTHER APPROXIMATIONS

This section compares the approximations for the arctangent function proposed in this article with the best ones existing in the literature, as far as we are concerned. Concretely, the state of the art functions that are more appropriate for computer vision applications have been considered. The comparison study is divided into two parts: on the one hand, the theoretical error of the studied methods is computed. Results are summarized in Table 2. It can be seen that our proposed approximations outperform the proposed ones in [7] and [8].

Table 2. Theoretical error

reference	**equation**	**range**	**error**
[7]	(3)	$[-\frac{\pi}{4}, \frac{\pi}{4}]$	0.2683°
[8]	(10)	$[0, \frac{\pi}{2}]$	4°
ϕ_2	(13)	$[0, \frac{\pi}{2}]$	0.1620°
ϕ_3	(17)	$[0, \frac{\pi}{2}]$	0.00811°

On the other hand, the empirical error and the throughput obtained with different approximations of the arctangent function are studied. For this comparison two single precision floating point vectors $\mathbf{x}$ and $\mathbf{y}$, each one comprising 256×1024 elements, are initialized by sampling from a uniform distribution $\sim U(-255, 255)$. In the event that an x_i is exactly zero the element is resampled. In addition, a vector $\mathbf{t}$ is calculated by performing the division $\mathbf{y}/\mathbf{x}$ elementwise. The throughput of a particular arctangent approximation is determined from the time spent in performing 20 evaluations on $\mathbf{t}$ and $\mathbf{x}, \mathbf{y}$ for the two and four quadrant cases, respectively. This process is repeated 10 times on one of the CPU cores and the best obtained throughput value is kept. The empirical error is calculated as the maximum of the absolute errors from the double precision arctangent in the C standard library, which is used as the ground truth. Table 3 summarizes the obtained results. The two and four quadrants cases (denoted as 2Q and 4Q from now on) are studied separately. The error is measured in degrees, whereas the throughput is measured in the number of computed arctangents per microsecond (μs). The code has been compiled using Visual Studio 2010, and executed on an Intel Core Quad CPU Q8300

```
// Approximates atan(x) normalized to the [−1,1] range
// with a maximum error of 0.1620 degrees.

float norm_atan( float x )
{
    static const uint32_t sign_mask = 0x80000000;
    static const float b = 0.596227f;

    // Extract the sign bit
    uint32_t ux_s = sign_mask & (uint32_t &)x;

    // Calculate the arctangent in the first quadrant
    float bx_a = ::fabs( b * x );
    float num = bx_a + x * x;
    float atan_1q = num / ( 1.f + bx_a + num );

    // Restore the sign bit
    uint32_t atan_2q = ux_s | (uint32_t &)atan_1q;
    return (float &)atan_2q;
}
```

Code 1. C++ implementation of the normalized arctangent based on (13) and extended to the [-1,1] range using (5).

```
// Approximates atan2(y, x) normalized to the [0,4) range
// with a maximum error of 0.1620 degrees

float norm_atan2( float y, float x )
{
    static const uint32_t sign_mask = 0x80000000;
    static const float b = 0.596227f;

    // Extract the sign bits
    uint32_t ux_s = sign_mask & (uint32_t &)x;
    uint32_t uy_s = sign_mask & (uint32_t &)y;

    // Determine the quadrant offset
    float q = (float)( ( ~ux_s & uy_s ) >> 29 | ux_s >> 30 );

    // Calculate the arctangent in the first quadrant
    float bxy_a = ::fabs( b * x * y );
    float num = bxy_a + y * y;
    float atan_1q = num / ( x * x + bxy_a + num );

    // Translate it to the proper quadrant
    uint32_t uatan_2q = (ux_s ^ uy_s) | (uint32_t &)atan_1q;
    return q + (float &)uatan_2q;
}
```

Code 2. C++ implementation of the two-argument variant of the arctangent based on (18) and extended to the [0,4) range.

2.50GHz desktop computer with Windows 7 Professional 64-bit.

First of all, the arctangent computed using the *atan* and *atan2* functions in the C standard library are considered. Although they feature full 24 bit precision for the *float* type, the throughput is far too small for real-time applications (22.07 in 2Q, 16.24 in 4Q). Secondly, a C++ implementation of the approximation proposed in [7] is evaluated. In this case, the number of arctangents/μs is larger (87.06 in 2Q, 56.30 in 4Q) and the approximation error grows to 0.27°. Then, we compare the previous approaches with a C++ implementation of our proposed approximations employing ϕ_2 (see Codes 1 and 2). It can be seen in Table 3 that it is considerably faster (200.77 in 2Q, 140.59 in 4Q) and that the accuracy is also improved. The ensuing error is smaller (about 0.16°) and matches the theoretical error due to the fact that an exact division is used.

To assess the efficiency of the proposed approximations in a realistic scenario, the Intel *Integrated Performance Primitives* (Intel IPP) library has been incorporated to the comparison. Intel IPP is a library of highly optimized numerical functions tailored for Intel processors. It includes limited accuracy but extremely fast versions of the two and four quadrant arctangent: *ippsAtan_32f_A11* and *ippsAtan2_32f_A11*, respectively. Indeed, it attains a very small error (0.00965°) at a much lower computational cost than the previous examined approximations (488.98 in 2Q, 218.19 in 4Q).

One of the keys to the Intel IPP impressive performance is its extensive use of SIMD instructions to achieve a high parallelism. Therefore, in order to obtain a fair comparison, a SSE2 version of the functions shown in Codes 1 and 2 has been implemented and benchmarked against Intel IPP, with satisfactory results. It can be noticed that, with the SSE2 implementation, the approximation is nearly three times faster than the Intel IPP proposal (1278.97 in 2Q, 692.48 in 4Q) at the cost of an error one order of magnitude larger (0.18°), which is nonetheless acceptable for image processing tasks, as it is lower than (2). It is worth mentioning that with the SSE2 implementation the approximation error is slightly larger than the theoretical error due to a fast reciprocal instruction being used to perform the division.

Finally, the results obtained with the SSE2 implementation of our third order rational function approximation ϕ_3 are also presented. It should be highlighted that it achieves the smallest error of all the studied arctangent approximations (about 0.008124°). It is even more precise than the one provide by Intel IPP (about 15% more accurate) and it is still about 1.5 times faster than it (741.72 in 2Q).

CONCLUSION

This article proposes two novel full quadrant approximations for the arctangent function focused on real time applications. The key point of the proposed approximations is that they can

Table 3. Empirical error and throughput

method	# quadrants	error	# arctan/μs
C Standard Library	2	3.4149e-06°	22.071
	4	6.8301e-06°	16.241
[7], C++	2	0.2683°	87.061
	4	0.2683°	56.308
ϕ_2, C++	2	0.1620°	200.768
	4	0.1620°	140.588
Intel IPP 7.0	2	0.009693°	488.977
	4	0.009650°	218.195
ϕ_2, SSE2	2	0.1843°	**1278.970**
	4	0.1853°	**692.488**
ϕ_3, SSE2	2	**0.008124**°	741.725

be easily extended to two and four quadrants. A comparison of the precision and performance of the proposed functions with the best arctangent approximations existing in the literature is provided. Results show that the third order proposed function outperforms the existing ones in terms of both precision and performance. Specifically, it is about 15% more accurate and about 1.5 times faster than the best existing one. On the other hand, the second order proposed function is the most suitable for real time applications: it is 2.6 and 3 times faster than the best existing one in the two and four quadrants approximations, respectively. Furthermore, it has a precision of about 0.18°, which is adequate for most applications in the computer vision field.

AUTHORS

Xavier Gironés (xavier.girones@urv.cat) received the BSc degree in Computer Science in 1994 from the Polytechnic University of Catalonia, Barcelona, Spain. He worked as an R&D engineer in Hewlett-Packard, Sant Cugat, Spain until 2009. Since October 2009 he is a member of the Intelligent Robotics and Computer Vision Group, in the Universitat Rovira i Virgili, Tarragona, Spain, where he is currently a PhD student after obtaining the MSc degree in Computer Science in 2010. His current topic of research is text detection in natural images.

Carme Julià (carme.julia@urv.cat) received the BSc degree in Mathematics in 2002 from the Polytechnic University of Catalonia, Barcelona, Spain. In September 2004 she got the MSc degree in Computer Science and in 2008 the PhD degree in the Computer Vision Center, at the Universitat Autònoma de Barcelona. Since September 2008 she is a member of the Intelligent Robotics and Computer Vision Group, in the Universitat Rovira i Virgili, Tarragona, Spain, where she is currently a Lecturer. Her research interest is focused on Structure from Motion (SfM) through factorization and on photometric stereo.

Domenec Puig (domenec.puig@urv.cat) received the M.S. and Ph.D. degrees in computer science from Polytechnic University of Catalonia, Barcelona, Spain, in 1992 and 2004 respectively. In 1992, he joined the Department of Computer Science and Mathematics at Rovira i Virgili University, Tarragona, Spain, where he is currently Associate Professor. Since July 2006, he is the Head of the Intelligent Robotics and Computer Vision group at the same university. His research interests include image processing, texture analysis, perceptual models for image analysis, scene analysis, and mobile robotics.

REFERENCES

[1] D. Lowe, "Distinctive image features from Scale-Invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.* June 2005, vol. 1, pp. 886–893 vol. 1, IEEE.

[3] C. Galambos, J. Kittler, and J. Matas, "Gradient based progressive probabilistic Hough transform," *IEE Proceedings - Vision, Image, and Signal Processing*, vol. 148, no. 3, pp. 158, 2001.

[4] E. G. Kogbetliantz, "Computation of arctan n for $-\infty <$ n $< +\infty$," *IBM Journal of Research and Development*, vol. 2, no. 1, pp. 43–53, Jan. 1958.

[5] I. Koren and O. Zinaty, "Evaluating elementary functions in a numerical coprocessor based on rational approximations," *IEEE Transactions on Computers*, vol. 39, no. 8, pp. 1030–1037, Aug. 1990.

[6] R. Lyons, "Another contender in the arctangent race," *Signal Processing Magazine, IEEE*, vol. 21, no. 1, pp. 109–110, 2004.

[7] S. Rajan, S. Wang, R. Inkol, and A. Joyal, "Efficient approximations for the arctangent function," *Signal Processing Magazine, IEEE*, vol. 23, no. 3, pp. 108–111, 2006.

[8] S. Winitzki, "Uniform approximations for transcendental functions," *Computational Science and Its Applications–ICCSA 2003*, pp. 962–962, 2003.