KUNGL TEKNISKA HÖGSKOLAN

Department of Numerical Analysis and Computer Science
TRITA-NA-P02/11 • CVAP-268 • ISSN 1101-2250 • ISRN KTH/NA/P--02/11--SE

Fast Block Matching with Normalized Cross-Correlation using Walsh Transforms

Peter Nillius & Jan-Olof Eklundh

Computational Vision and Active Perception Laboratory (CVAP)

Peter Nillius & Jan-Olof Eklundh
Fast Block Matching with Normalized Cross-Correlation using Walsh Transforms

Report number: TRITA-NA-P02/11, ISRN KTH/NA/P--02/11--SE, CVAP-268
Publication date: September 2002
E-mail of author: nillius@nada.kth.se

Department of Numerical Analysis and Computer Science
Royal Institute of Technology
SE-100 44 Stockholm
SWEDEN

Fast Block Matching with Normalized Cross-Correlation using Walsh Transforms

Peter Nillius* and Jan-Olof Eklundh

Abstract—**Local image matching (block-matching) is a frequent operation in many image processing tasks, such as MPEG compression and the estimation of optical flow and stereo disparities. Normalized cross-correlation (NCC) is particularly useful since it is insensitive to both signal strength and level. However, NCC is computationally expensive. In this article we attempt to speed up NCC first by transforming each sub-block of the image into the Walsh basis. The Walsh transform expansion can be done very efficiently through a binary tree of filters. Calculating the NCC using the Walsh components requires $2N-1$ operations instead of $4N+1$ in a straightforward implementation.**

Further, the Walsh transform expansion is shown to have several scales encoded in it. Using only a part of the Walsh components is the same as doing the correlation at a coarser scale. A coarse-to-fine algorithm for doing block-matching using this is presented and tested. The performance of the algorithm is a trade-off between how well the algorithm can find the correct match and how many calculations that are saved. When matching 99% of the blocks correctly the calculations were reduced to $9-23\%$ of what a full search would require, depending on the images and the size of the search region.

Keywords— **Matching, Block matching, Fast normalized cross-correlation, Walsh functions, Coarse-to-fine**

I. Introduction

Matching of blocks between images is needed for a number of tasks such as MPEG compression, optical flow and stereo disparity calculation. Typically block-matching is done by comparing a block with a number of blocks within a region in another image. The block in the search region with the highest correspondence value is selected as the matching block. There are several ways to measure the correspondence between two blocks. Cross-correlation gives a robust and dense measure of the correspondence between two blocks. In particular if you normalize the cross-correlation in terms of mean and variance you will get a correspondence measure that is insensitive to luminance scale and level. Burt et al [1] showed in a comparison that the normalized cross-correlation consistently gave the lowest error rates compared to non- or partially normalized, Laplacian filtered and binary correlation.

Block-matching requires extensive computations and there exists several algorithms to speed it up, e.g. by reducing the number of blocks to compare with by using different search strategies, [2], [3], [4], [5] and by using pyramid representations and do the search in a coarse-to-fine way, [6], [7]. Some work has also been done in optimizing the calculation procedure itself, [8], [9].

Even though it is relatively robust compared to other measures, normalized cross-correlation (NCC) is rarely used. This is probably because it is computationally expensive due to the normalization. Also the mean-normalization gives the problem a structure that makes it difficult to speed up using clever algorithms such as the running sum algorithm.

In this article we propose a method to speed up matching with NCC using Walsh functions, both by speeding up the calculations themselves and by using a coarse-to-fine search strategy.

The authors are with the Computational Vision & Active Perception Laboratory (CVAP), Department of Numerical Analysis and Computing Science, Royal Institute of Technology (KTH), S-100 44 Stockholm, Sweden. Phone: +46-8-790 6905, Fax: +46-8-723 0302. E-mail: nillius@nada.kth.se, joe@nada.kth.se

* Corresponding Author

II. Normalized Cross-Correlation

Representing the image blocks as vectors, the normalized cross-correlation (NCC) between two blocks $P = (p_0, p_1, \ldots, p_{N-1})^T$ and $Q = (q_0, q_1, \ldots, q_{N-1})^T$, where N is the number of pixels in the block, is given by

$$\frac{1}{\sigma_p \sigma_q}(P - \bar{P}) \bullet (Q - \bar{Q}) \tag{1}$$

where σ_p and σ_q are the standard deviations over the blocks and $\bar{P}$ and $\bar{Q}$ are vectors containing the means:

$$\sigma_p = \sqrt{(P - \bar{P}) \bullet (P - \bar{P})} \tag{2}$$

$$\bar{p} = \frac{1}{N}\sum_{i=0}^{N-1} p_i \tag{3}$$

$$\bar{P} = (\bar{p}, \bar{p}, \ldots, \bar{p})^T \tag{4}$$

If we rewrite the blocks in an orthonormal basis, $\{W_j; j = 0, \ldots, N-1\}$, we get

$$P = \sum_{j=0}^{N-1} w_{p,j} W_j, \tag{5}$$

where the $w_{p,j}$ are P's coefficients in the new basis.

Rewriting (1) and (2) in this basis gives us.

$$\frac{1}{\sigma_p \sigma_q}\left(\sum_{j=0}^{N-1} w_{p,j}(W_j - \bar{W_j})\right) \bullet \left(\sum_{j=0}^{N-1} w_{q,j}(W_j - \bar{W_j})\right)$$
$$= \frac{1}{\sigma_p \sigma_q}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} w_{p,j} w_{q,j} (W_i - \bar{W_i}) \bullet (W_j - \bar{W_j}) \tag{6}$$

$$\sigma_p = (\sum_{j=0}^{N-1} w_{p,j}^2 {\sigma_{w_j}}^2)^{\frac{1}{2}} \tag{7}$$

The expression for the NCC has now expanded to a double sum. However, if the basis vectors W_j were constructed such that

$$(W_i - \bar{W_i}) \bullet (W_j - \bar{W_j}) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

then all terms outside the "diagonal" of the double sum would be zero.

Such a basis can be constructed as follows. Set $W_0 = \frac{1}{\sqrt{N}}(1, 1, \ldots, 1)^T$ and complete the orthonormal basis to get the other basis vectors. Then the mean of W_0 will be $\frac{1}{\sqrt{N}}$, i.e. $\bar{W_0} = W_0$. The means of the other basis vectors can be expressed as the dot product between W_0 and the vectors W_i, times a scalar. Due to orthogonality towards W_0 those means will be zero, i.e. $\bar{W_i} = \vec{0}$ for $i = 1, \ldots, N-1$.

We get

$$(W_i - \bar{W_i}) \bullet (W_j - \bar{W_j}) = \begin{cases} 1 & \text{if } i = j \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

which is what we wanted. All the products outside the diagonal and the first product in the diagonal of the double sum are zero. Now the expression for the NCC will come down to

$$\frac{1}{\sigma_p \sigma_q} \sum_{i=1}^{N-1} w_{p,i} w_{q,i} \tag{9}$$

$$\sigma_p = (\sum_{j=1}^{N-1} w_{p,j}^2)^{\frac{1}{2}} \tag{10}$$

An orthonormal basis that meets these requirements is the Walsh basis, [10], [11].

III. Normalized Cross-Correlation in the Walsh Basis

The Walsh functions form an orthogonal basis. Their discrete version consists only of the values $+1$ and -1, which makes them computationally efficient to calculate.

To calculate NCC using the Walsh functions, each sub-block (including overlaps) of the image needs to be transformed into the Walsh basis. This will be referred to as the Walsh transform expansion (WTE). Calculating the WTE is the same as convolving the image with each of the Walsh functions. This section will show how this can be done efficiently.

First look at the Walsh functions. The discrete Walsh functions can be defined recursively as follows. Let $W_i^{(N)}, i = 0, \ldots, N-1$ be the set of discrete Walsh functions of length N. Set

$$W_0^{(1)} = (1) \tag{11}$$

With $i = 0, \ldots, N/2-1$, the higher order sets are defined as

$$W_{2i}^{(N)} = \begin{pmatrix} W_i^{(N/2)} & W_i^{(N/2)} \end{pmatrix} \tag{12}$$

$$W_{2i+1}^{(N)} = \begin{pmatrix} W_i^{(N/2)} & -W_i^{(N/2)} \end{pmatrix} \tag{13}$$

Equations (12) and (13) can also be expressed with convolutions:

$$\begin{aligned} W_{2i}^{(N)} &= W_i^{(N/2)} * (\overbrace{1 \ 0 \ldots 0}^{N/2} \ 1) \\ W_{2i+1}^{(N)} &= W_i^{(N/2)} * (\overbrace{1 \ 0 \ldots 0}^{N/2} \ -1) \end{aligned}$$

This means that the Walsh functions can be described as a series of convolutions of simple filters, i.e. they are separable. For example

$$\begin{aligned} W_2^{(8)} = W_1^{(4)} * (1 \ 0 \ 0 \ 0 \ 1) = \cdots = \\ (1) * (1 \ 1) * (1 \ 0 \ -1) * (1 \ 0 \ 0 \ 0 \ 1) = \\ (1 \ \ 1 \ -1 \ -1 \ \ 1 \ \ 1 \ -1 \ -1). \end{aligned} \tag{14}$$

For our purposes we should rather use the two dimensional version of the Walsh basis. The 2D Walsh functions can be defined similarly with convolutions alternately in the horizontal and vertical direction. With $W_0^{(1)} = (1)$ and $i = 0, \ldots, N/4-1$:

$$\begin{aligned} W_{4i}^{(N)} &= W_i^{(N/4)} * (\overbrace{1 \ 0 \ldots 0}^{\sqrt{N}/2} \ 1) * \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \\ W_{4i+1}^{(N)} &= W_i^{(N/4)} * (1 \ 0 \ldots 0 \ 1) * (\overbrace{1 \ 0 \ldots 0}^{\sqrt{N}/2} \ -1)^T \\ W_{4i+2}^{(N)} &= W_i^{(N/4)} * (1 \ 0 \ldots 0 \ -1) * (1 \ 0 \ldots 0 \ 1)^T \\ W_{4i+3}^{(N)} &= W_i^{(N/4)} * (1 \ 0 \ldots 0 \ -1) * (1 \ 0 \ldots 0 \ -1)^T \end{aligned}$$

Since the Walsh functions are separable down to simple filters which only require one addition or subtraction, convolving the image with them can be done very efficiently. Further calculations can be saved by arranging these filters into a binary tree of convolutions so that the result of each convolution is reused maximally, see Figure 1.

By using this separable scheme we can reduce the number of calculations required to transform each sub-block of an image, down to $2(N-1)$ additions/subtractions per block, where N is the number of pixels in the block. Without separating the functions the same operation would require N^2 additions/subtractions per block. Also, by calculating the NCC through the Walsh transform we will save $2N$ subtractions per block so for every time we reuse the Walsh transform we will in total save $2N$ subtractions per block. If we e.g. compare a block with all other blocks in a 20x20 block region in the same image we will reuse the Walsh transform for each block 800 times.

IV. Scale Properties of the Walsh Functions

The Walsh functions appear in different order depending on how they are defined, [12]. Equations (11)-(13) define them in dyadic ordering. When defining the 2D Walsh functions the same way, we get what can be called a 2D dyadic ordering. This ordering has some scale properties that can be used to calculate the NCC in a coarse-to-fine manner.

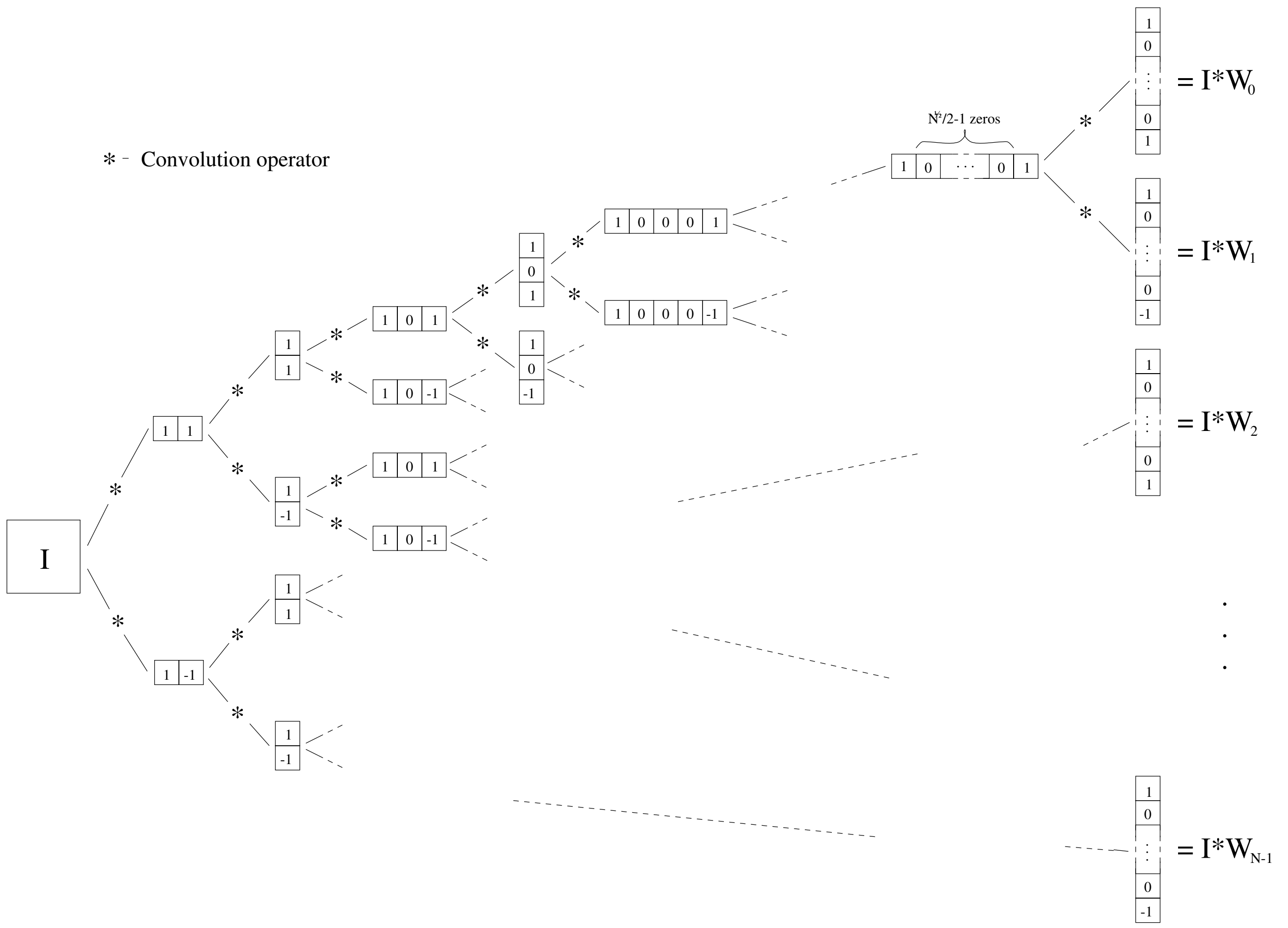


Fig. 1. Calculating the Walsh transform expansion of Image I. The Walsh functions, W_i, can be separated into a chain of simple filters each using only one addition or subtraction. More so calculating the Walsh transform expansion can be done with a binary tree of these filterings. By using this scheme the Walsh transform expansion can be calculated in approximately $2(N-1)$ additions/subtractions per block, where N is the number of pixels in the block.

Looking at the Walsh bases for 8x8 and 4x4 blocks, Figure 2, we see that the vectors in the 4x4 basis match the first 16 vectors in the 8x8 basis. The same holds for all other Walsh bases of size 2^nx2^n. The vectors of a 2^nx2^n basis are the same as the 2^{2n} first vectors in the 2^{n+1}x2^{n+1} basis but at a doubled scale. Why this is so can best be seen in Figure 1. The top sub-branch after the second stage in the filter tree has the image convolved with the 2x2 average operator, as input. All the filters following there have a zero in every second position. That is the same as sub-sampling the image at every second pixel. Removing those zeros we get the same filter tree as the whole tree but with two steps less depth, i.e. a tree that gives the Walsh basis with half the width and height as the whole tree.

This means that using only the first 16 components in the 8x8 Walsh transforms will be the same as doing the same thing with the 4x4 transform on an halved image, where the halved image comes from convolving the image with a 2x2 average operator and then sub-sampling it at every second pixel.

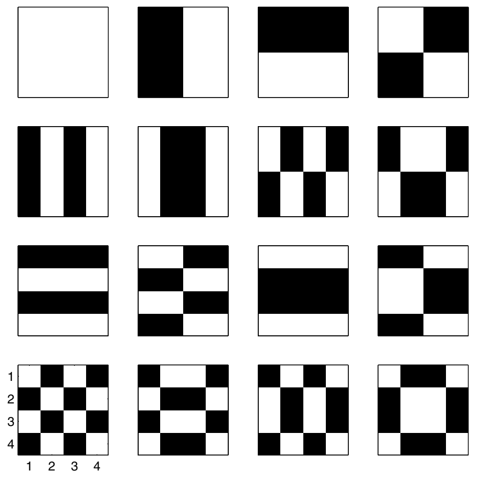
Because this holds for all 2^nx2^n Walsh bases, using the first four components of the 8x8 Walsh transform is the same as using the same number of components of the 4x4 transform on a halved image and that is the same as using all (four) components of the 2x2 transform now on an image scaled down by a factor of four.

In short, we have all octaves of scales downwards, encoded in this Walsh transform expansion.

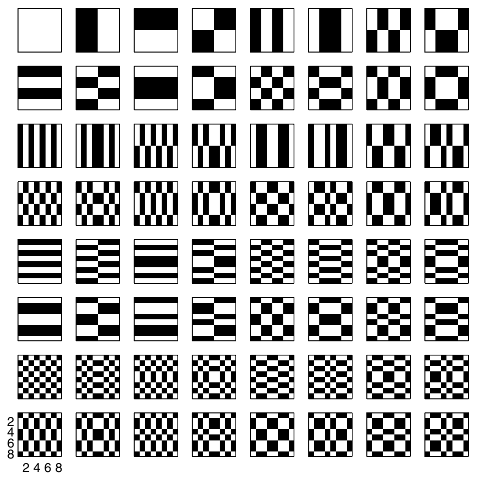
V. A Coarse-to-fine Algorithm

The scale properties of the Walsh functions can be used to do a coarse to fine algorithm for block matching. The following algorithm is proposed.

The idea is to start calculating the NCC on the coarsest scale. That is done by using only four components (actually only three since the first component containing the average of the block is never used) of the Walsh transform. From there you select which points will go on the next finer scale. This is done by thresholding using a threshold that is relative to the current maximum NCC value. Now the NCC for selected points are calculated for the next scale. The new maximum is calculated and the same thresholding process as before takes place. This goes on until you are at the finest scale, where all Walsh components are used. The maximum



(a) 4x4 Walsh Basis



(b) 8x8 Walsh Basis

Fig. 2. The Walsh functions consist only of the values +1 and −1, which makes them computationally efficient to use. Here are the Walsh functions in 2D dyadic ordering. The 2D dyadic ordering has the property that the basis vectors in an 2^nx2^n basis occur in the 2^{n+1}x2^{n+1} basis, first, in the same order, but at a doubled scale. E.g. in the figure the 16 vectors of the (a) 4x4 basis appear in the beginning of the (b) 8x8 basis.

point is the matched block found by the algorithm.

The algorithm step by step:

1. Set $s = 1$
2. Calculate the NCC using the 4^s first components of the Walsh transforms.
3. Find the maximum correlation value c_{max}.
4. Select the points with correlation value $c \geq c_{max} - \alpha_s$.
5. Set $s = s + 1$
6. For the selected points, calculate the NCC using 4^s Walsh components.
7. If $4^s \neq N$ goto 3.
8. Find the point with maximal correlation value.

The calculations at each iteration can be stored so that you just add the extra Walsh components needed for the next scale.

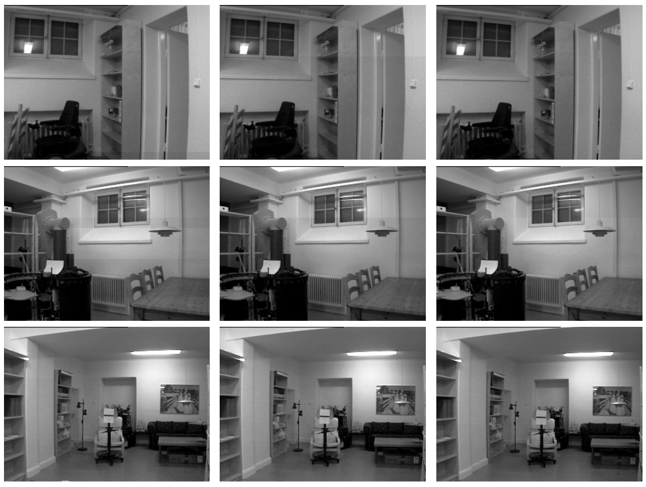


Fig. 3. Some images from the sequences used. The images were recorded with a camera on a pan-tilt unit, with the pan-tilt unit moving stepwise in a constant direction and with constant speed.

A. Using Other Block Similarity Measures

The presented algorithm can be used for block matching using other block similarity measures than NCC, as long as the measure is calculated through the Walsh components. For instance, cross-correlation without luminance-level normalization (without subtracting the mean) can be achieved simply by including the first Walsh component in the calculations.

For other applications it might be more useful to use the sum of squared errors as block similarity measure. Using the same notation as previously, the sum of squared error can be written as

$$(P - Q) \bullet (P - Q). \tag{15}$$

Rewriting the expression in the Walsh basis we get

$$\left(\sum_{i=0}^{N-1} (w_{p,i} - w_{q,i})W_i\right) \bullet \left(\sum_{i=0}^{N-1} (w_{p,i} - w_{q,i})W_i\right) = \\ = \cdots = \sum_{i=0}^{N-1} (w_{p,i} - w_{q,i})^2 \tag{16}$$

Using this formula the algorithm can be used for block-matching using the sum of squared error.

VI. Experiments

Doing block matching in a coarse-to-fine way is an approximation of doing full search. There is no guarantee that the block with the highest correlation value is found. How well the coarse-to-fine method works depends on the images and how the scales are calculated. A series of experiments has been done in order to test if the scales encoded in the Walsh transform expansion are useful for doing block matching in a coarse to fine way.

The images used in the experiments were recorded from a camera on a pan-tilt unit, with the pan-tilt unit moving stepwise in one direction and with constant speed. Figure 3 shows a few images from the sequences used.

The Walsh transforms were calculated for 8x8 sized blocks which means that every block will have 64 Walsh components. The coarse-to-fine algorithm used three scales for its search, using 3, 15 and 63 components respectively (the first Walsh component, which is the average of the block is not used).

The experiments were done by randomly selecting a point in an image and then trying to find it in another image in the same sequence. The distance between the images were selected such that the algorithm was tested for different sub-pixel positions. For each image pair 1000 randomly selected points were used.

As ground truth we used the block who had the maximum NCC value at the finest scale. So when we refer to correct rate we mean when the coarse-to-fine algorithm finds the block with the maximum normalized cross-correlation value at the finest scale.

A. How useful are the encoded scales?

The first set of experiments tests how useful the scales encoded in the Walsh transform are.

For different α, we calculate the percentage of correct points being above the threshold. If it is above the threshold it will be found when the full NCC for those points is calculated. At the same time we record how many points we were able to eliminate and thereby save calculations.

Figure 4a shows the rate of the correct block being above the threshold, as well as the relative number of blocks being above threshold, for different α. This is at the coarsest scale, i.e. using only three Walsh components.

Figure 4b shows the same graph, but at the intermediate scale, i.e. using the 15 first Walsh components.

The figures clearly show that the encoded scales can be used for coarse-to-fine matching. At e.g. $\alpha = 0.1$ when the images are blurred with a 3x3 binomial filter, we eliminate 90% of all points while only missing 3% of the correct points and this when using only three out of 64 Walsh components (Fig. 4a).

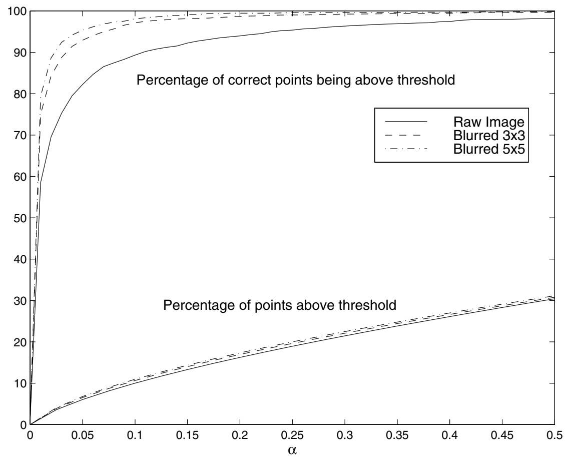
B. Testing the algorithm

Figure 4 shows that the scale properties of the Walsh basis can be used for doing a coarse-to-fine search for the maximum normalized cross-correlation point. However, Figure 4b show the graph for all blocks in the images, while the coarse-to-fine algorithm, at this scale (15 Walsh components), would only calculate the correlation for those blocks that are above the threshold in the previous scale (3 Walsh components). To see the full effect of that we need to look at the different combinations of the parameters α_1 and α_2.

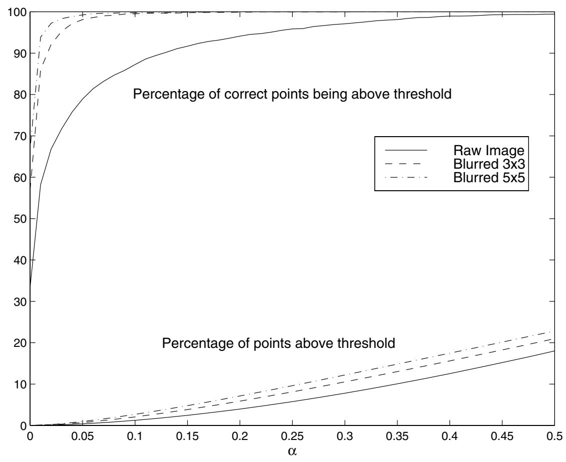
Tables I and II show the correct rate and the rate of calculations needed for different combinations of α_1 and α_2. The calculation rate is in relation to calculating the full NCC map and still using the Walsh transforms. It is calculated as $r_{calc} = (3 + \lambda_1 * 12 + \lambda_2 * 48)/63$, where λ_1 and λ_2 are the rates of pixels above threshold at stages 1 and 2 respectively.

Many of the α_1,α_2 combinations are sub-optimal in the sense that there exists several α_1,α_2 pairs that will give you the same correct rate, but have different calculation rates. The pair with the lowest calculation rate is the optimal. Figure 5 shows the calculation rate as a function of the correct rate when the sub-optimal threshold pairs have been removed. This gives a nice view of the trade-off between the correct rate and the calculations saved. The best results are obtained when the image is slightly blurred. Only minimal blurring is required to get a substantial improvement. When the images are blurred with a 3x3 binomial filter we get 99% correct matches with only 11% of the calculations.

Many of the blocks in the images have little structure, i.e. have almost constant grey-value and are therefore difficult to match correctly. For some applications it might be more



(a)



(b)

Fig. 4. Percentage of correct points and the percentage of all points being above threshold in the correlation image calculated using (a) three Walsh components and (b) 15 Walsh components, for different α. The threshold used was $c_{max} - \alpha$, where c_{max} is the maximum value in each correlation image. In each of the graphs, the three top curves show the percentage of the correct points being above threshold and the three bottom curves show the percentage of all points being above threshold.

TABLE I

Percentage Correct Matches

	α_2				
α_1	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
0.1	83.4	87.4	88.4	88.8	88.9
0.2	85.9	91.4	93.3	93.8	94.0
0.3	86.6	92.6	95.1	96.0	96.3
0.4	86.9	93.4	96.0	97.0	97.3
0.5	87.0	93.6	96.3	97.4	97.9

TABLE II

Percentage of Calculations Required Compared to Full Search

	α_2				
α_1	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
0.1	7.54	9.06	10.4	11.5	12.3
0.2	8.73	10.7	13.0	15.0	16.5
0.3	9.71	11.7	14.5	17.3	19.6
0.4	10.6	12.6	15.5	18.8	21.8
0.5	11.4	13.4	16.3	19.8	23.4

interesting to see how well blocks with some structure are matched. Figure 6 shows the performance when considering only blocks with a high signal energy, in comparison with the performance when considering all blocks. The images were blurred with a 3x3 binomial filter. Only points with $\sigma > 500$ were used in the high energy case. This corresponds to a step edge of height 7.8 (the images range from zero to 255). Overall the performance is slightly better for the high energy blocks. We get 99% correct matches using only 9% of the calculations.

For an illustration of the algorithm see Figure 7.

C. *Performance with different search regions*

The previous experiments were done by searching for a block over the whole image. In many applications such as optical flow and stereo correspondence it is more common to search in a limited search region centered around an expected position. It is likely that the size of the search region affects the performance of the algorithm.

Experiments were done with 11x11, 21x21 and 41x41 block search region. The images were blurred with a 3x3 binomial filter. This time only blocks with high signal energy, $\sigma > 500$, were taken into account. Figure 8 shows the results. As can be seen in the figure for a fixed correct rate, the rate of calculations needed increases as the search region gets smaller.

VII. Implementation

The calculation of the Walsh transform expansion has been implemented in C++ using templates for maximum speed.

The potentially high memory requirement of the intermediate images in the filter tree has been avoided by using buffering. Only the rows needed by the subsequent filter are stored.

The algorithm reads the source image only once and writes full Walsh transform expansion interleaved in one stream for best performance on systems with cached memory architecture.

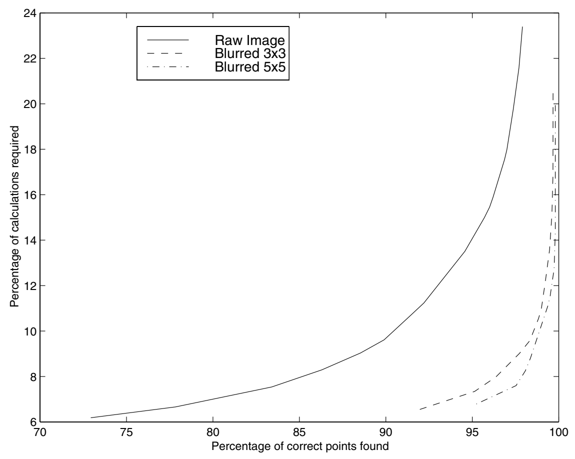


Fig. 5. Calculations required versus correct rate for the best α_1, α_2 pairs.

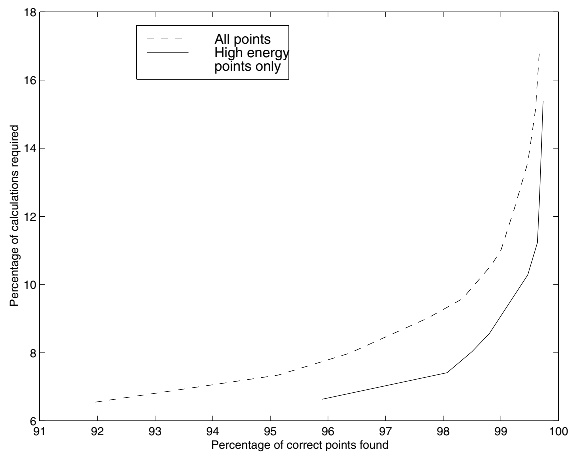


Fig. 6. Comparison of performance when considering all blocks versus looking at blocks with high signal energy. Only blocks with $\sigma > 500$ were taken into account for the high energy case. The images were blurred with a 3x3 binomial filter. The dashed curve is the same as the dashed curve in Fig. 5

Table III shows the computation time of the Walsh transform expansion for different systems and image sizes.

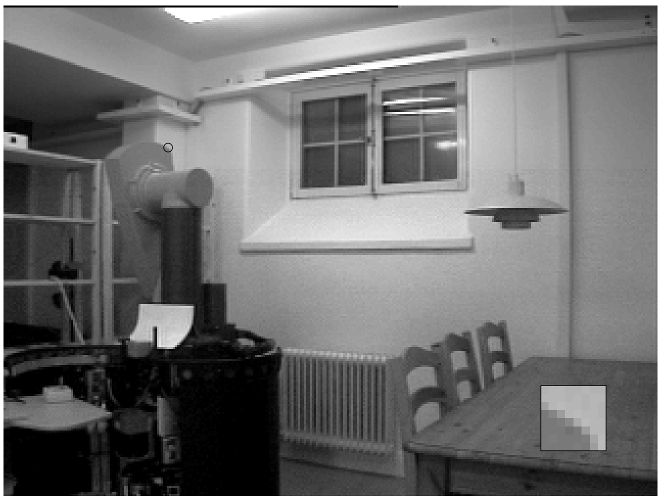
TABLE III

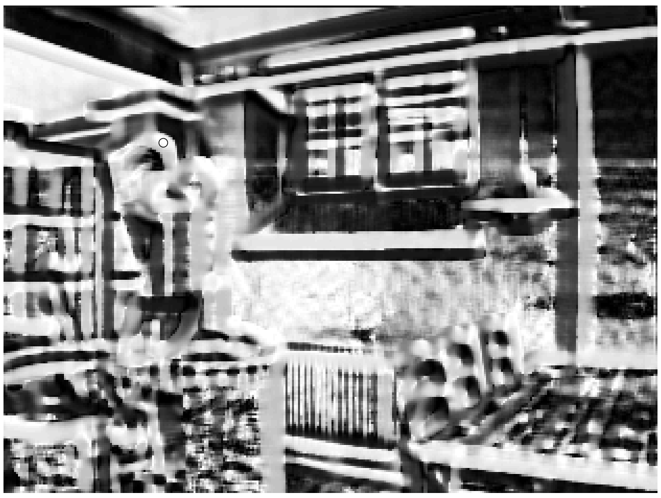
Computation Time of the Walsh Transform Expansion

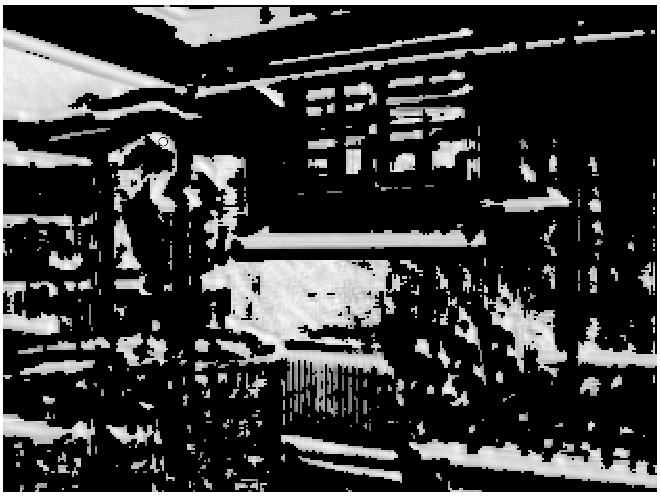
System	256x256 image	128x128 image
400 MHz Pentium II	210 ms	50 ms
Sun Ultrasparc 5	280 ms	66 ms
Sun Ultrasparc 1	400 ms	95 ms

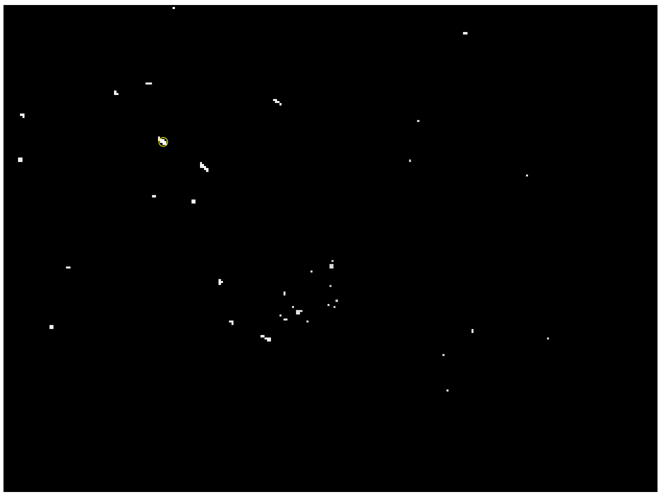
VIII. Calculations

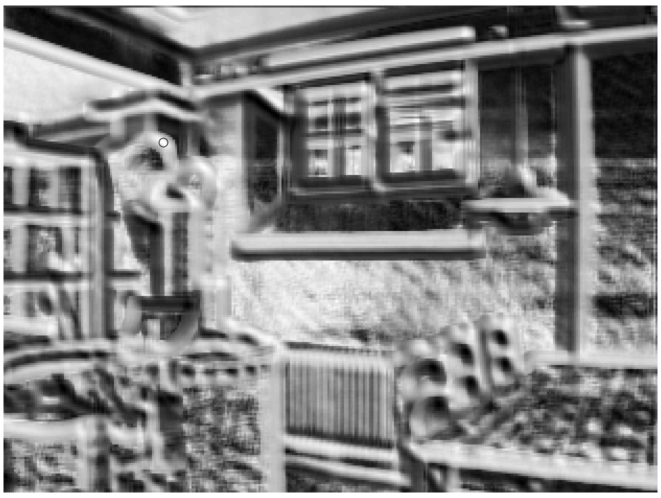
Table IV shows the number of calculations per block needed to compute the NCC normally or using Walsh functions. The mean is only computed when not using the Walsh functions and can be done in approximately 4 adds/subtracts per block using the running sum algorithm. The WTE needed for our algorithm can be calculated in approximately $2(N-1)$ add/subtracts per block. This is the same as the number of nodes in the filter tree in Figure 1. Each filter (node) requires one addition or subtraction.


(a) Search Image and Comparison Block


(b) Coarsest Scale (3 Walsh Components)


(c) Intermediate Scale (15 Walsh Components)


(d) Finest Scale (63 Walsh Components)


(e) Complete Correlation Map at Intermediate Scale

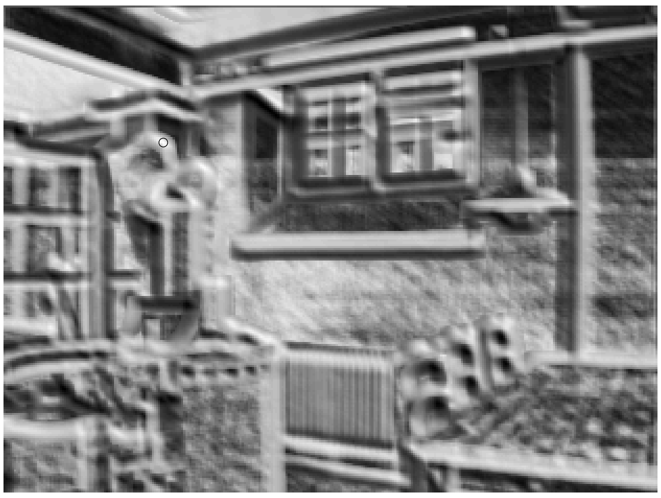

(f) Complete Correlation Map at Finest Scale

Fig. 7. The algorithm at work: Finding the block with the highest normalized cross-correlation value. The circle in all the images marks the correct point. (a) Search image with the enlarged 8x8 comparison block inlined in the bottom right corner. Images (b), (c) and (d) show the different steps of the algorithm. (b) is the correlation map at the coarsest scale calculated using 3 Walsh components. (c) shows the correlation map at the next scale (15 Walsh components) for the points above threshold at the previous scale. (d) shows the correlation map at the finest scale (63 components) for the remaining points at the last step. The point with the highest value in this image is the result of the algorithm and in this case it is the correct one. (e) is the complete correlation map calculated using 15 Walsh components. (f) is the complete correlation map from 63 Walsh components. The parameters in this example: $\alpha_1 = 0.3$, $\alpha_2 = 0.1$

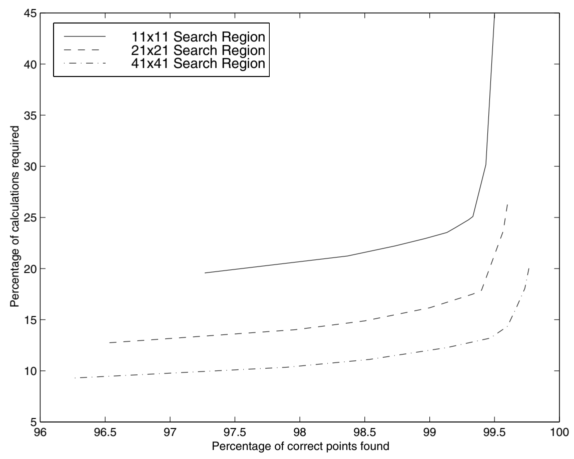


Fig. 8. Performance for different sized search regions.

TABLE IV

Number of Calculations per Block for Normalized Cross-Correlation

Operation	*Without Walsh* +-	*/	√	*With Walsh* +-	*/	√
Mean	4	0	0	n/a		
WTE	n/a			2(N-1)	0	0
Variance	2N	N	1	N	N	1
NCC	3N	N+1	0	N-1	N	0

Assuming that the mean, the WTE and the variance is precalculated and stored for each image, a comparison can be made. Using Walsh functions will require about N more adds/subtracts per block for the precalculations. However, every time the NCC is calculated about $2N$ adds/subtracts per block are saved when using Walsh functions.

IX. Conclusion

We have showed how you can reduce the calculations for NCC by transforming each sub-block of an image to the Walsh basis . Because of the simplicity of the Walsh functions this operation can be done very efficiently through a binary tree of filterings. Calculating the NCC using the Walsh components requires $2N - 1$ operations instead of $4N + 1$ in a straightforward implementation. The Walsh transform expansion takes about $2(N - 1)$ additions/subtractions per block. That means that for every time the Walsh transform is reused $2N + 2$ operations are saved.

Furthermore, it turns out that the Walsh transform expansion has several scales encoded in it. All octaves down are represented in the same transform. Using only some of the Walsh components when calculating the NCC is equivalent to calculating the NCC at a coarser scale. These scale properties are exploited in an algorithm that does block-matching in a coarse-to-fine manner.

The experiments show that the algorithm is an efficient way of speeding up block-matching with normalized cross-correlation. While significantly reducing the number of calculations the properties of the NCC are preserved.

The method presented here should fit well to speed up block matching in various applications. It is a straightforward and fast algorithm. Also, the coarse-to-fine search allows you to easily integrate other constraints used in e.g. optical flow and stereo correspondence.

References

[1] Burt, P. J., Yen, C., Xu, X. Local Correlation Measures for Motion Analysis A Comparative Study, *Image Processing Laboratory Technical Report, IPL-TR-024*, 1982

[2] Jain, J.R. and Jain, A.K., Displacement Measurement and Its Application in Interframe Image Coding, *IEEE Trans. on Communications*, vol. COM-29, no. 12, Dec. 1981

[3] Srinivasan, R. and Rao, K.R., Predictive Coding Based on Efficient Motion Estimation, *IEEE Trans. on Communications*, vol. COMM-33, no. 8, Aug. 1985

[4] Ghanbari, M., The Cross-Search Algorithm for Motion Estimation, *IEEE Trans. on Communications*, vol. 38, no. 7, July 1990

[5] Zeng, B., Li, R. and Liou, M.L., Optimization of Fast Block Motion Estimation Algorithms, *IEEE Trans. on Circ. and Syst. for Vid. Tech.*, vol. 7, no. 6, Dec. 1997

[6] O'Neill, M., Denos, M., Automated system for coarse-to-fine pyramidal area correlation stereo matching, *Image and Vision Computing*, 14(1996), pp 225-236

[7] Anandan, P., A Computational Framework and an Algorithm for the Measurement of Visual Motion., *Int. Journal of Computer Vision*, 2(1989), pp 283-310

[8] Linzer, E., Tiwari, P., Zubair, M., High Performance Algorithms for MPEG Motion Estimation, *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1996, pp 1934-7

[9] Sun, C., A Fast Stereo Matching Method, *Digital Image Computing: Techniques and Applications*, Massey Univ., Auckland, New Zealand, 10-12 Dec 1997, pp 95-100

[10] Beauchamp, K. G., Transforms for Engineers: A guide to Signal Processing, *Oxford University Press*, 1987

[11] Schipp, F., Wade, W.R., Simon, P., Walsh Series: an introduction to dyadic harmonic analysis, *Adam Hilger*, 1990

[12] Ahmed, N., Schreiber, H. H., Lopresti, P. V., On Notation and Definition of Terms Related to a Class of Complete Orthogonal Functions, *IEEE Trans. Electrom. Compat.* EMC-15:75-80, 1973