Fast Pattern Recognition using Normalized Grey-Scale Correlation in a Pyramid Image Representation

W. James MacLean and John K. Tsotsos

December 16, 1999

Abstract

The ability to quickly locate one or more instances of a model in a grey scale image is of importance to industry. The recognition/ localization must be fast and accurate. In this paper we present an algorithm which incorporates normalized correlation into a pyramid image representation structure to perform fast recognition and localization. The algorithm employs an estimate of the gradient of the correlation surface to perform a steepest descents search. Also described is an algorithm for fast sub-pixel localization. Test results are shown.

1 Introduction

Fast pattern recognition is an invaluable component of many machine-vision algorithms used in industry today. The ability to quickly identify an object or a marking on an object can aid in finding a desired part amongst a selection of parts, or to register a part so that work may be performed on it. This paper outlines an algorithm for fast (less than 0.5 S) detection and localization of a pattern using simple and inexpensive hardware. Specifically, the algorithm is suitable for implementation on a personal computer equipped with an image acquisition board and a camera.

The algorithm relies on a pyramid representation of both the model image and the search image, as well as an estimate of the gradient of the correlation

surface. The algorithm begins by creating a pyramid representation of the model image. A worst-case analysis is then used to determine the effect on the correlation score of i) the number of pyramid levels combined with ii) the possible different registrations of the pyramid sampling grid with respect to the model to determine the optimum number of pyramid levels to use. This analysis is done once off-line.

During the search phase, a pyramid representation is built from the image being searched for instances of the model. It is built to the same depth as the model pyramid. The top-level of the model pyramid is convolved with the top-level of the search-image pyramid to produce a correlation surface from which likely candidates for model instances can be chosen. For each candidate, the search process descends through the pyramid performing a correlation-gradient search at each level. When the bottom level is reached, either a model instance has been found or the candidate is rejected for failing to meet a minimum threshold for its correlation score. Once a model instance is found, a bi-quadratic sub-pixel localization algorithm can be applied if desired.

This algorithm does not allow for size or rotational orientation variations, nor does it allow for model instances which are partially occluded. Non-uniform lighting variations will also detract from the algorithm's performance, but it is assumed that these may be controlled in industrial settings. This algorithm may be suitable for incorporation into a larger algorithm which addresses the the issues of size and rotation variance.

An outline of this paper is as follows: Results of previous approaches are considered in section 2. Section 3 deals with the idea of performing normalized grey-scale correlation in a pyramid structure. Section 3.1 develops an estimate of the correlation gradient for gradient-descent search. An important feature of this estimate is that it can be estimated quickly. In section 3.3 a method for choosing the depth of the pyramid is presented. Section 4 describes details of the algorithm's implementation. A discussion of the results of applying the algorithm are presented in section 5.

2 Review of Previous Work

- reference book on pyramid re-presentations

- draw link to scale-space representation
- discuss different methods:
  - correlation techniques (slow)
  - moments (can be faster, and also rotation/size invariant, but the latter require *a priori* knowledge of area of object)
  - neural net methods (slow[?], works best for classification, not localization)

3 NGC in a Pyramid Representation

This paper describes the use of a pyramid image representation to reduce the computational complexity associated with correlation search. Let $I(x, y)$ represent an image $I_w \times I_h$ pixels in size. The top-left corner of the image is defined to be the origin. Each pixel is assumed to encode a grey value in the range $0 \ldots 255$, one byte per pixel. If we wish to use correlation to search for a model $M(x, y)$ of size $M_w \times M_h$ in the image, the complexity for performing the correlation is $O(I_w I_h M_w M_h)$. The complexity increases if normalized correlation is used (see Eq. 1).

$$C(x,y) = \frac{\sum_{i=0}^{M_w-1} \sum_{j=0}^{M_h-1} M(i,j) I(x+i, y+j)}{\sqrt{\sum_{i=0}^{M_w-1} \sum_{j=0}^{M_h-1} M(i,j)^2} \sqrt{\sum_{i=0}^{M_w-1} \sum_{j=0}^{M_h-1} I(x+i, y+j)^2}}, x < I_w - M_w + 1 \text{and} y < I_h - M_h \quad (1)$$

An overview of the method is as follows: build pyramid representations of both the model and the image (search space), and perform correlation search at the top level of the pyramid. This can be done very quickly. The estimate of the model's location can be refined using a coarse-to-fine strategy in which the best estimate of location at level k in the pyramid is used as the starting point for the search at level $k-1$. When the base of the pyramid is reached, the model has been found. Multiple models can be found by repeating this procedure by choosing more than one match at the top level of the pyramid.

The pyramid is built by averaging squares of 4 pixels in one level and mapping the result onto one pixel in the next level. Let K be the number of levels in the pyramid, and assign an index $k = 0 \ldots K-1$ to each level. Each

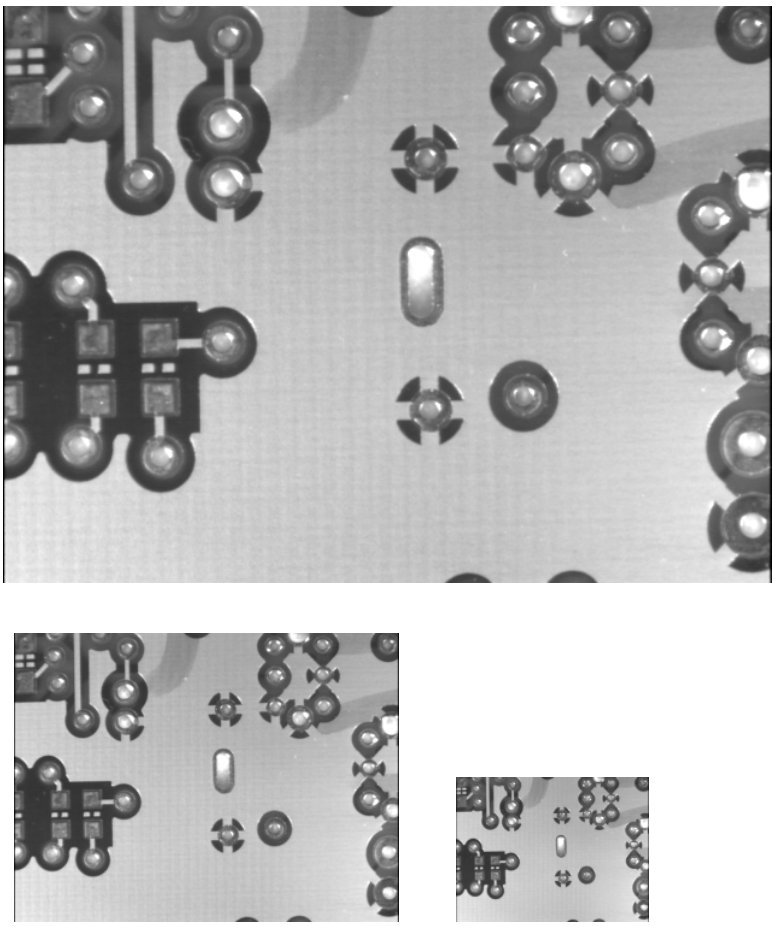

Figure 1: The pyramid representation for a typical image is shown. The pyramid has three levels, with level 0 being the largest image (top), and level 2 being the smallest (bottom right). In the level 0 image, a search model is defined.

layer of the pyramid has no more than one-quarter of the number of pixels of the previous level, and the entire pyramid can be built in less than twice the amount of memory used to store the original image. This type of pyramid can be built quickly, as each pixel in a new level only requires 3 adds and one shift to compute. The algorithm for building the pyramid is $O()$. The number of levels is limited by $K_{max} \leq \log_2 min(M_w, M_h)$.

We now describe briefly the method of building both the image pyramid and the model pyramid. The structure of the pyramid is quite simple. We have used an averaging-with-no-overlap scheme, and used a sampling rate of 2. This means that each level in the pyramid has dimensions equal to one-half of those for the level below, meaning that each level is one-quarter the size of the one immediately below. Each pixel in a given layer is the average of four pixels from the layer below. These four pixels are termed "the receptive field" of the pixel on the higher level, and the fact that the pyramid is non-overlapped means that on each level the receptive fields do not overlap each other. There are of course many ways to build a pyramid, this is the one we have chosen.

By way of example, consider building a 4-level pyramid from a 640×480 image. The first level of the pyramid is just the raw image, and the subsequent layers have dimensions of 320×240, 160×120, and 80×60. A pixel in level 3 having indices (x, y) is the average of the following pixels from level 2: $(2x, 2y)$, $(2x+1, y)$, $(2x+1, 2y+1)$ and $(2x, 2y+1)$. The advantages of buidling a pyramid become quickly obvious: in this case the top level of the pyramid has 64 times fewer pixels than does the original image. Remember that the model pyramid also has 64 times fewer pixels, the total cost of NGC at the top level of the pyramid is $64^2 = 4096$ times smaller than NGC on the original image.

The only real difference between the model and image pyramid is the representation used. The model image is converted to **double**, while the image pyramid is left as **BYTE**. This does of course mean that information is lost while building the image pyramid since only integral values can be represented.

If we now perform correlation search for the model at the top-level of the pyramid, then the complexity is $O((I_w I_h M_w M_h)/2^{4(K-1)})$. For an image built into a 4 level pyramid, this is a factor of 4096 faster. Of course, the localization of the pattern at the top level of the pyramid is not perfact, but this can be used as an initial estimate for the next finer level of the pyramid.

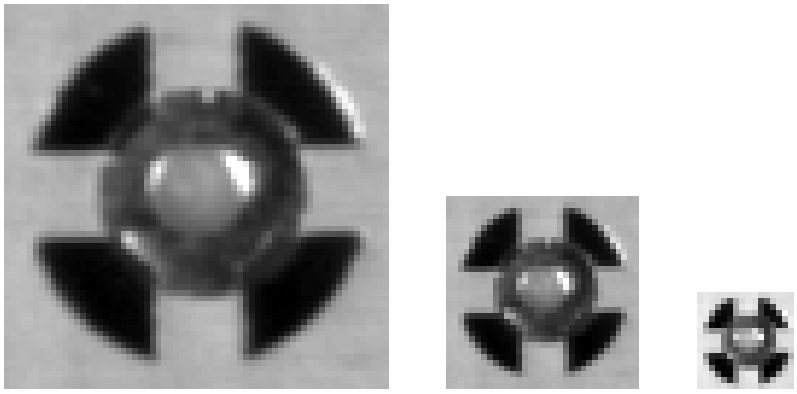

Figure 2: The pyramid representation of the model in Figure 1 is shown. The pyramid has the same number of levels as the image pyramid.

At each level of the pyramid a small number of correlations is used to refine the location estimate.

Instead of performing an exhaustive local search for the new maximum at each level of the pyramid, it is possible to estimate the gradient of the correlation surface and use a steepest descents method to perform the search. The following section describes the derivation of the gradient estimate.

3.1 Derivation of Correlation Gradient

When refining the estimate of model location is performed at each level of the pyramid, it is possible to use an estimate of the gradient of the correlation surface to perform a steepest descent search. While it is possible to estimate the correlation surface in a 3×3 neighbourhood centred on the current location estimate, this is computationally expensive (it requires 9 full correlations). It is possible to estimate the correlation gradient in a much less expensive manner. In the derivation which follows the continuous case is developed: the transition to the discrete case is straighforward.

Given an image $I(x, y)$ and a model $M(x, y)$, with M being differentiable over all x and y, the correlation coefficient surface can be defined as

$$C(u, v) = \frac{\iint I(x, y)(MW)(x - u, y - v)dx\,dy}{[\iint I^2(x, y)W(x - u, y - v)dx\,dy]^{1/2}}\,. \tag{2}$$

The function $W(x,y)$ is a windowing function which is used to force the contribution from M to zero outside a given region:

$$W(x,y) = 0 \;\forall\; x < 0,\; x > M_w,\; y < 0 \text{ and } y > M_h \;. \tag{3}$$

The notation $(MW)(x,y)$ is shorthand for $M(x,y)W(x,y)$. For simple correlation computation, W is normally chosen to be a box function. However, since we will want to differentiate Eq. 2, the windowing function should be chosen such that its derivative goes to zero at the borders of the window. It is assumed that $\iint M^2(x,y)W(x,y)dx\,dy = 1$, *i.e.* that $M(x,y)$ is normalized with respect to the windowing function. Since this need only be done once, it can be done "off-line" before the search is performed. Assuming that our functions M and W are "well-behaved" we can differentiate Eq. 2 to derive the gradient:

$$\nabla C(u,v) = \begin{bmatrix} \frac{\partial C}{\partial u} \\ \frac{\partial C}{\partial v} \end{bmatrix} \;. \tag{4}$$

The gradient of the windowed correlation is

$$\begin{aligned} \nabla C^2(u,v) = & \quad -2\frac{\iint I(x,y)(MW)(x-u,y-v)dx\,dy}{\iint I^2(x,y)W(x-u,y-v)dx\,dy} \\ \times & \quad \iint I(x,y)\nabla(MW)(x-u,y-v)dx\,dy \\ + & \quad \left[\frac{\iint I(x,y)(MW)(x-u,y-v)dx\,dy}{\iint I^2(x,y)W(x-u,y-v)dx\,dy}\right]^2 \\ \times & \quad \iint I^2(x,y)\nabla W(x-u,y-v)dx\,dy \;. \end{aligned}$$

This leaves us with just four terms to calculate:

$$\iint I(x,y)(MW)(x-u,y-v)dx\,dy \tag{5}$$

$$\iint I(x,y)\nabla(MW)(x-u,y-v)dx\,dy \tag{6}$$

$$\iint I^2(x,y)W(x-u,y-v)dx\,dy \tag{7}$$

$$\iint I^2(x,y)\nabla W(x-u,y-v)dx\,dy \tag{8}$$

Since $C^2(u,v) \in [0,1]$ we don't expect $\nabla C^2(u,v)$ to be huge so long as the correlation surface is reasonably smooth, *i.e.* no discontinuities in I or M. As the amount of high-frequency content in M and/or I increases, we expect the correlation surface to become less smooth. When we move to the discrete case, the derivative will always exist, but it is still desirable to avoid large discontinuities in the model and image.

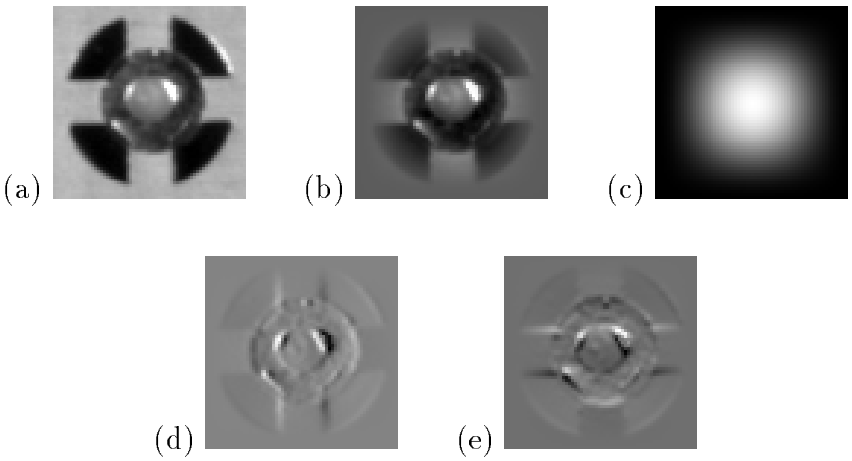

Figure 3: This figure shows the various representations required at each level of the pyramid for gradient correlation search. The model is given in (a). The windowed version of the model is shown in (b), with the windowing function shown in (c). The horizontal and vertical components of the gradient are shown in (d) and (e).

3.2 Choice of Image Vector Space Origin

Modern imaging technology typically represents images as one (or more) 2-D arrays of 1-byte pixels, with individual pixels having values in the range of $0 \ldots 255$. While correlation values are in the range of -1 to 1, images represented solely by positive semi-definite integers will lead to purely positive correlation values.

An important example of this occurs when comparing the model to a section of the image which has uniform intensity, *i.e.* all the pixels have the same value. Let $\vec{I}_{const} = [a \ldots a]^T$ be a column vector representing the uniform background. The correlation of a model with this "uniform background vector" (UBV) yields the following result:

$$C = \frac{\vec{M}^T \vec{I}_{const}}{||\vec{M}||\ ||\vec{I}_{const}||} \ .$$

Substituting the known value of $\vec{I}_{const}$ we get

$$C = \frac{\sum_{i=1}^{M_h M_w} m_i}{\sqrt{M_h M_w}} \tag{9}$$

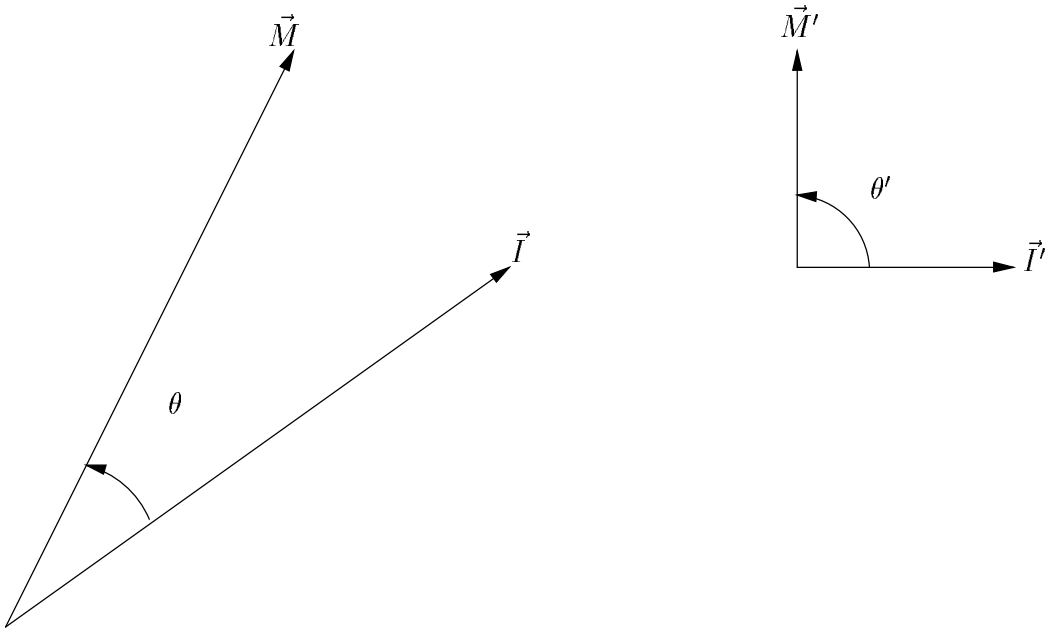

Figure 4: In this figure we see the effect of the choice of origin on the correlation score computed for the two vectors. On the left we see two vectors which could easily result from a positive-only imaging system, and the angle between them. On the right we see a much larger angle achieved by choosing a different origin for the vector coordinates. In general, for any two distinct vectors it is possible to choose an origin which makes the angle between them any value, including 90 and 180 degrees.

where m_i is the ith element of $\vec{M}$. We see that we get a correlation value which depends only on the model. We also see that the choice of origin affects the value of the correlation score. This is easily illustrated by remembering that the correlation score is just the cosine of the angle between the two vectors, i.e. $C = \cos\theta$ where θ is the angle between $\vec{M}$ and $\vec{I}$.

In Figure 4 we see a graphic depiction of the effect of choice of the origin on the correlation score. On the left we see a pair of vectors, $\vec{M}$ and $\vec{I}$, which could be a typical pair of image vectors. The angle between the two vectors is θ, which is directly related to the correlation score (smaller angles correspond to higher correlation scores, and *vice versa*). When all the vector components are positive, then the resulting image vectors are limited to some

subset of the image space. In the 2-D case shown here (think of the image vectors as having two pixels), the vectors are limited to one quadrant of the vector space. In the 3-D case (think of the image vectors as having three pixels), the vectors are limited to one octant of the vector space. In general, the image vector will be confined to $\frac{1}{2^N}$ of the space, where N is the total number of pixels in the image. Since the models we are searching for will routinely have hundreds and even thousands of pixels, we see that the net result of having only positive components in the vectors is that correlation scores will tend to be high in most cases. On the right of Figure 4 the same two vectors are depicted, but a new origin is being used. In this case the vectors are represented by $\vec{M}'$ and $\vec{I}'$, and the angle between them is θ'. In this case the origin has been chosen to make $\theta' = 90°$. In general we can choose the origin to make the correlation score anything we want (we can only make $C = 1$ in the limiting case where the origin tends to infinity).

Given an infinite number of choices for a new origin, which one should we choose? One reasonable choice would be to choose an origin such that correlation of the model with a uniform background leads to a score of 0. Examining Eq. 9, we see that subtracting the mean of the model vector components from the model will cause

$$\sum_{i=1}^{M_w M_h} m_i' = 0 \ .$$

Since a correlation score of $+1$ indicates a perfect match, and a score of -1 indicates a perfect mismatch, then a score of 0 for comparison with a featureless background seems reasonable. This method of subtracting the mean from the model has been very successful in practice. It should also be noted that subtracting the mean from the model is the equivalent of removing any D.C. component from the model image's energy—thus only the energy related to model features is left.

A final issue to consider involves the effect of subtracting the model's mean from the image region being correlated with the model. While it seems reasonable in practice to treat the image region in an identical fashion to the model, there are some practical problems. If the mean is subtracted before the normalization is applied, then raw intensity of the model can affect the matching process. Since one of the reasons we use NGC is for its robustness against multiplicative changes in the image (*i.e.* changes in image intensity),

this is undesirable. In an image captured with pixel values in the range $0 \ldots 255$, pure multiplicative changes about the origin (0) are equivalent to a translation and a multiplication when any other origin is chosen. It was observed in practice that subtracting the same mean from both the model and the image region being compared made the entire process very sensitive to changes in lighting. As a result, it is proposed that the mean of the image region be subtracted instead. In this case we are no longer doing pure NGC (since each vector has a different value subtracted from it), but it works far better in practice. Again, we can think of having removed the D.C. component from the image region as well.

3.3 Tuning the Pyramid

Up until now the only mention made of K is that it is limited by the size of the model. In many cases the details in the model may degrade too quickly for this limit to be realistic. Take, for example, a model which consists of a checkerboard pattern, with alternating pixels black and white. In this case even a 2-level pyramid is too much, because the second level of the model pyramid will be a uniform (featureless) grey region which will be useless for correlation.

Instead, the depth of the pyramid must be chosen according to the characteristics of the model itself. One method of doing this is to build the model pyramid one level at a time, and as each new level is added estimate the worst-case correlation score of the model at that level.

By "tuning the pyramid" we refer to the problem of deciding how many levels to choose for a pyramid representation of a given model. The more levels the pyramid has, the greater the savings in computation, but building the pyramid with too many levels may render the model unrecognizable to the NGC algorithm running at the top of the pyramid. The following discussion on the effects of pyramid sampling gives a possible starting point to tackle the problem of tuning the pyramid.

3.3.1 Issues Surrounding Sampling of the Pyramid

Usually a model is chosen as a sub-image in a larger image. It is possible that we will then look for other instances of that model in the same image, or we may wish to remember the model and try and find it again later.

Let's say (for the sake of argument) that we have a model of size 50×50 located at $(96, 96)$ in the image. The model pyramid will have 4 layers of size 50×50, 25×25, 12×12 and 6×6. If we look at the top-level of the model we will find an exact replica of it in the top-level of the image pyramid (save integer division truncation errors), since the location of the model is on an integral 8-pixel boundary. Remember that our 4 level pyramid has a top-level where each pixel is represented by $8 \times 8 = 64$ pixels from the bottom-level. This guarantees that the same information averaged and combined in (almost) the exact same way while building both the image and model pyramids. But what would happen if the model had been at $(97, 97)$? Then the top-level representation of the target would not be the same in the image pyramid as in the model pyramid. Since at $(96, 96)$ the top-level instance should have a score of 1, the score when the model is at $(97, 97)$ is guaranteed to be worse. For a two-level pyramid we require the model to be on an even pixel boundary in order for the image and model pyramid to match at the top. Therefore, for a randomly placed model there is one chance in four that it will land 'perfectly'. For a three-level pyramid, the model must land on an even boundary of 4 pixels, giving one chance in sixteen of perfect alignment. There is one chance in 64 for perfect alignment in a 4-level pyramid.

It should be noted that perfect alignment is not required in order to be able to find the target at the top level of the pyramid. How well an imperfectly aligned model will match at the top of the pyramid is dependent on the features of the model.

3.3.2 Tuning the Pyramid using Worst-Case Analysis

As discussed earlier, the representation of a target at the top of the pyramid may be better or worse depending on its location in the image. This suggests a possible method for tuning the pyramid.

The first step is to set an upper limit on the number of levels in the pyramid. Obviously correlation is meaningless with only one pixel at the top level of the pyramid, so we can choose a minimum size for the model at the top-level. For the current implementation of the search tool this minimum size has been chosen to be 4×4. This means that the maximum number of levels in the pyramid will be chosen so that the model has a minimum dimension between 4 and 7 (a minimum dimension of 8 could be reduced to

4 by adding another level to the pyramid). Depending on the features of the model it may be possible to successfully search at this maximum pyramid depth.

The second step is to consider the worst-possible-case score of the model at different pyramid depths. The worst case will be be due to the effects of pyramid sampling as discussed previously. In order to determine the worst case score of a model in a K-level pyramid, we build the model pyramid to K levels, and then build an image pyramid for offsets (x, y) where both x and y are in the range $0 \ldots 2^{K-1} - 1$. Therefore we end up considering $2^{2(p-1)}$ different offsets. Of course, the offset $(0, 0)$ is expected to have the best score, since only truncation error is expected to differentiate between the model and "image" pyramid.

It is important to note that this worst-case analysis is only an estimate of the worst score of the model at the top of the pyramid, since in a real search situation that score will depend on image content outside the target for which we have no *a priori* knowledge. As we consider different offsets, the size of the top-level "image" pyramid (really just a shifted version of the model) can be as much as one-pixel smaller than the model in each dimension. If we consider the most extreme case, we compare a 4×4 model to a 3×3 "image". Even in this case our estimate involves comparing 56.25% of the model with a shifted version of itself.

The third step is to perform this worst-case analysis for pyramids from 2 levels up to and including the maximum possible number of levels, and choosing the maximum pyramid depth which still yields an acceptable worst-case score.

A number of "pathological" targets can be imagined in order to test this scheme. Consider black & white checkerboard patterns where each cell of 2 white and 2 black squares takes on the following dimensions: 2×2, 4×4, 8×8 and 16×16. The first pattern can only be searched with a 1-level pyramid, since the 2×2 receptive field used to go to the next level will blur the checks into a featureless pattern of grey. The second pattern could be searched with a 2-level pyramid if we were confident that the target would always align perfectly on a 2-pixel boundary. However, since this is not guaranteed, we will likely end up with a featureless grey again (corresponding to an offset of $(1, 1)$), so we must restrict ourselves to a 1-level pyramid. The next two patterns (for very similar reasons) may be searched reliably with a 2-level pyramid, but no more.

To recap the "worst-case analysis" algorithm, we perform the following steps:

1. Based on the size of the model in the original image, determine the maximum depth (number of levels) of the pyramid which will make the top-level representation of the model larger than a pre-set minimum size (in our case 4×4 pixels). Call this depth K_{max}.

2. Build the model pyramid to this depth.

3. For each pyramid depth p in the range of $2 \ldots K_{max}$ perform a worst case analysis of the image representation of the model at level p. This is done by building an "image" pyramid based on the original model, but offset by (x, y) where each of x and y takes on values in the range $0 \ldots 2^{K-1} - 1$. For each "image", compute the correlation score between the Kth level representation of the "image" and the model.

4. Choose the number of levels in the pyramid to be the largest value of K for which the worst-case correlation score between model and "image" is above some pre-set threshold (currently we are using 0.1 for this threshold).

Intuitively this method is quite appealing, in that it mimics the actual search process when deciding how to tune the pyramid. Unfortunately, it may well be that always catering to the worst possible case is unduly pessimistic. Variations on the method, in which the average or median of returned scores is used instead of just the worst case itself, may prove useful.

**** When searching for a model in an image, we can use the worst-case score for the top level of the pyramid as a cut-off point (threshold) for accepting candidates ****

4 Implementation, Experiments & Results

- implemented using C++
- test results run on Pentium 100 MHz processor
- algorithm given number of targets in advance, plus minimum acceptable threshold for instances

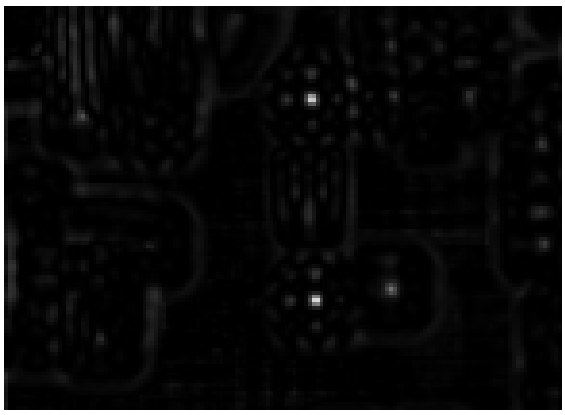

Figure 5: The correlation surface from the top level of the pyramid is shown. Two strong peaks, representing the location of the two instances of the model, are evident. These peaks provide coarse location estimates, which are refined as the algorithm descends through the pyramid.

- search time *vs.* model size
- search time *vs.* number of instances
- search time distribution (peaks depending on number of levels of pyramid)
- percentage false-negatives and false-positives

4.1 Subpixel Localization

In identifying an object in an image using normalized grey-scale correlation (NGC), it may be the case that the object as found will not align with the pixel grid in the same way as the model (or template) did. In this event, our estimate for the location of the object is expected to lie *between* pixels. Since NGC only allows us to compute the correlation at integral pixel locations, some method of determining the sub-pixel location is needed. One such method would be interpolation using a *bi-quadratic* surface.

A computationally simple method is now presented for doing sub-pixel localization. A bi-quadratic surface is employed to locally model the correlation surface. Nine samples from this surface are the data used to estimate

Instance	Location	Score (level 0)	Score (level 2)
1 (model)	(322,304)	1.000	1.000
2	(318, 95)	0.950	0.796

Table 1: Results are shown for searching for the model defined in Figure 1. The search was completed in 1272 mS. The number of levels in the pyramid was determined by the worst-case analysis of model scores in a pyramid structure. The level 2 scores indicate the score of each instance at the top level of the pyramid (see Figure 5). While the model has a perfect score at the top of the pyramid, this is not normally the case. Instance 2 has a lower score at the top level.

the 'best-fit' (least-squares) parameters of the bi-quadratic. No justification is given for the bi-quadratic form: while it is simple and provides some estimate of a maximum it is by no means the only possible model, and it may be that more appropriate models exist.

Assume the location of the peak in the correlation surface lies at $(0,0)$. (this can be done without loss of generality.) Proceed by calculating the correlation at the 8-neighbours of $(0,0)$, namely $C(x,y)$ evaluated at the points:

$C(-1,-1)$	$C(0,-1)$	$C(1,-1)$
$C(-1,0)$	$C(0,0)$	$C(1,0)$
$C(-1,1)$	$C(0,1)$	$C(1,1)$

We wish to fit $C(x,y)$ to a function of the form

$$\hat{C}(x,y) = \alpha x^2 + \beta y^2 + \gamma x + \delta y + \xi xy + \varphi$$

where the parameters of the model are

$$\vec{\pi} = [\ \alpha \quad \beta \quad \gamma \quad \delta \quad \xi \quad \varphi\ ]^T\ .$$

Write $\vec{\hat{C}} = F\vec{\pi}$ where

$$F = \begin{bmatrix} 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 & 1 \\ 1 & 1 & -1 & 1 & -1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

and

$$\vec{\hat{C}} = [\hat{C}(-1,-1) \quad \ldots \quad \hat{C}(1,1)]^T \ .$$

In order to estimate the parameters $\vec{\pi}$ we use the observed values of the correlation function at these points:

$$\vec{\pi} = (F^T F)^{-1} F^T \vec{C}$$

where

$$\vec{C} = [C(-1,-1) \quad \ldots \quad C(1,1)]^T \ .$$

The quantity $(F^T F)^{-1} F^T$ can be computed exactly, and its value is

$$(F^T F)^{-1} F^T = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ -\frac{1}{6} & 0 & \frac{1}{6} & -\frac{1}{6} & 0 & \frac{1}{6} & -\frac{1}{6} & 0 & \frac{1}{6} \\ \frac{1}{4} & 0 & -\frac{1}{4} & 0 & 0 & 0 & -\frac{1}{4} & 0 & \frac{1}{4} \\ -\frac{1}{9} & \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} & \frac{5}{9} & \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} & -\frac{1}{9} \end{bmatrix} \ .$$

Examination of this shows that it has a simple structure. It is advantageous to be able to calculate this quantity off-line (*i.e.* ahead of time). Each parameter can be estimated with a small number of adds, subtracts, and at most two divides. No multiplications are required to estimate any of the parameters except φ, which we do not need as shown below. Table 2 outlines the cost of calculating the individual parameters of the model.

Parameter	Multiplies	Divides	Adds	Subtracts
α, β	0	2	7	1
γ, δ	0	1	4	1
ξ	0	1	2	1
φ	2	1	7	2

Table 2: This table shows the computational cost of computing each parameter for fitting the bi-quadratic surface to the correlation data.

It is necessary to find the location of the maximum. At the maximum

$$\frac{\partial \hat{C}}{\partial x} = 0 \quad \Rightarrow \quad 2\alpha x_c + \gamma + \xi y_c = 0$$
$$\frac{\partial \hat{C}}{\partial y} = 0 \quad \Rightarrow \quad 2\beta y_c + \delta + \xi x_c = 0 \; .$$

The subscripts in x_c and y_c indicate those values of x and y which satisfy the requirements for a maximum. This can be rewritten as

$$\begin{bmatrix} 2\alpha & \xi \\ \xi & 2\beta \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix} = - \begin{bmatrix} \gamma \\ \delta \end{bmatrix}$$

with a solution of

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = - \begin{bmatrix} 2\alpha & \xi \\ \xi & 2\beta \end{bmatrix}^{-1} \begin{bmatrix} \gamma \\ \delta \end{bmatrix} = - \frac{1}{4\alpha\beta - \xi^2} \begin{bmatrix} 2\beta & -\xi \\ -\xi & 2\alpha \end{bmatrix} \begin{bmatrix} \gamma \\ \delta \end{bmatrix} \; .$$

The computation of (x_c, y_c) requires a further 8 multiplications, 2 adds, 2 subtractions and 2 divides.

The use of a bi-quadratic surface to interpolate the correlation surface and locate its peak is efficient. In fact, the cost of computing the correlation data $\vec{C}$ far outweighs the cost of estimating a sub-pixel location of the peak.

It remains to consider the error associated with the subpixel estimation. Two sets of tests were done. The first set involves a series of 10 images produced by synthetic means. Software was devised to simulate a target moving in the horizontal direction at 0.1 pixels/frame. The main advantage to using

synthetic images is that "ground truth" is known for the target location. However, the synthetic data ignore the possibility of effects introduced by imperfections in the imaging system.

The synthetic images were created by creating a target at a higher resolution than the original image, and then averaging over the required subset of pixels in order to determine the target's representation in the low-resolution image.

The second set of images show a printed circuit board, with the board being shifted left by about 1/20 of a pixel in each subsequent image. There are 20 images in the sequence. This image sequence permits meaningful testing of the subpixel localization algorithm on real images.

In both sets of images, the 'model' is defined in the first image, where its location (by definition) lies exactly on an integer pixel boundary. In each image sequence the target is assumed to shift by roughly equal amounts from one image to the next.[1] As a model is tracked, therefore, the subpixel estimate for its location should form a straight line with respect to the frame number.

For analysis of the real images, four image regions were selected as models, and each region was searched in each image in order to obtain an estimate of its location using subpixel localization.

4.1.1 Subpixel Localization—Synthetic Images

Figure 6 shows the results of subpixel estimation in the horizontal direction. The data has been shifted to show the movement in the range of 0 to 1.[2] The error in the estimates ranges from 0.0018 pixels (when the target position is integral) to a maximum of 0.0886 pixels. The mean error for the synthetic data is 0 (when taken over a complete cycle of the error curve). The RMSE is found to be 0.06 pixels.

Figure 7 shows the vertical error. Since the target was not moving vertically, we expect this error to be roughly constant, and ideally 0. The mean error was -0.0025 pixels, and the RMSE was 0.0026 pixels. We see a sharp change in the error around frame 5. To understand this, recall that the subpixel localization algorithm starts from 9 correlation values centred on the

[1] For the synthetic images this is known to be true, for the real images this was the intention, and will be true within limits on physical precision.

[2] In the original image, the horizontal location of the target starts at 318 pixels.

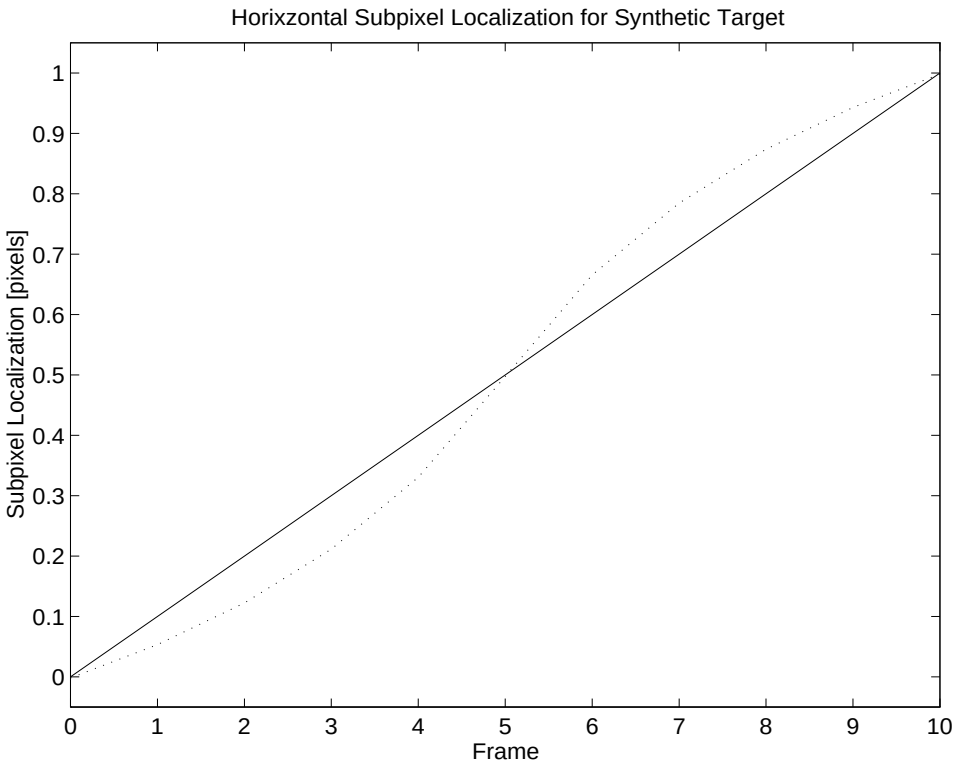

Figure 6: This image shows the localization in the horizontal direction of the moving target. The solid line indicates the ground truth position for the target, and the dotted values the subpixel localization estimates. The error is roughly symmetric about the half-way point between integral pixel values.

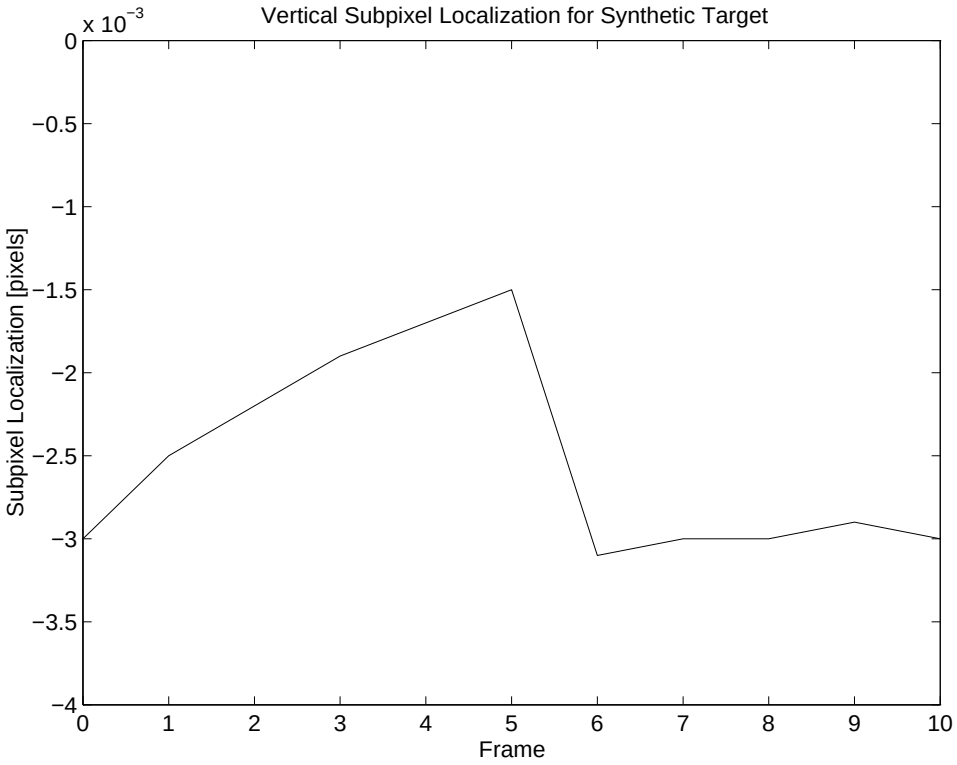

Figure 7: This image shows the localization in the vertical direction of the moving target. There was no target movement in this direction, so ideally the error should be zero. The sudden change in the error at frame 5 is likely due to the search algorithm switching it's (pre-subpixel) "best" estimate for the target from one pixel to the next.

"best integral estimate" of the target position. At frame 5 we are half-way between pixels, and we should expect the best integral estimate to switch from one pixel to the next at this point.

4.1.2 Subpixel Localization—Real Images

The data derived from the four image regions tracked through the real image sequence was analyzed in an attempt to gauge the performance of the sub-pixel localization algorithm. First, since it was assumed that all movement in the image sequence was purely in the horizontal direction, any changes in the vertical ordinate of the estimates should represent noise. In Figure 8 we see a plot of the y-ordinates from the four data sets. Each data set has been adjusted to start at 0 so that they could be plotted together. The mean error, $\bar{y}$ is calculated as

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

where y_i are the adjusted y data. The value obtained is 0.003 pixels. Since each target is defined in the first image in the sequence, we expect the error to have zero average over the image sequence. The root-mean-square error, defined as

$$\sigma_y = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

was measured to be 0.009 pixels. As expected from looking at Figure 8, this error is quite small.

The horizontal (x) ordinate of the estimates is expected to change. Since the image moves approximately one pixel to the left over 20 frames, we would expect a change of about -0.05 pixels/frame. As seen from the plot in Figure 9, the slope of these lines is indeed negative, and a quick estimate shows it to be approximately the right size. In order to get a more accurate measure of the error, a linear least-squares regression was performed on the data.

Figure 10 shows the best-fit line from the regression done assuming the line had zero intercept. The line has a slope of -0.0356 pixels/frame. The mean error in the data with respect to the regression line is -0.0034 pixels, and the RMSE is 0.09 pixels.

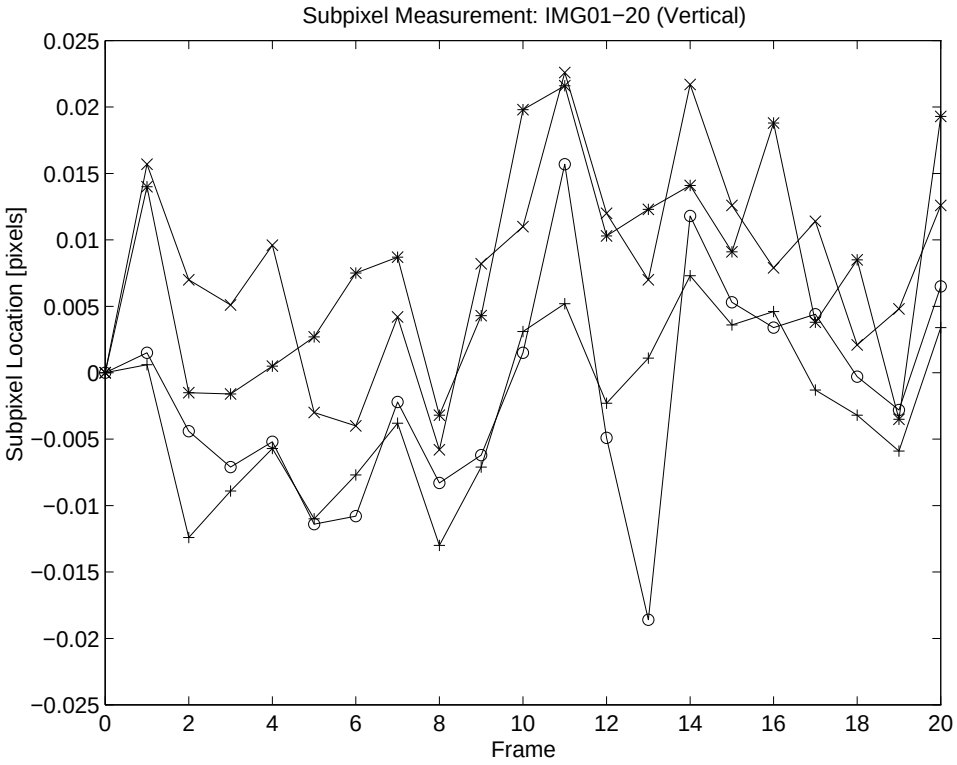

Figure 8: This figure shows the vertical ordinates of the subpixel localization estimates over the entire image sequence. The estimate shown for Frame 0 is that obtained with subpixel localization turned off. Each data set has been adjusted so that the first element is zero. Data sets 1–4 are represented by *, +, X and o, respectively.

Another meaningful test is to pick out multiple regions in an image as targets, and then search for them in the same image. In this case the exact target location is known, and it is known to be an exact integral value. The subpixel location estimates will not return exactly integral values and are expected to be in error—the question is "how much?". In a set of data collected using this approach (15 regions), the horizontal location error was found to have a mean error of 0.0676 pixels and an RMSE of 0.1264 pixels in the horizontal direction. The corresponding values for the vertical direction are 0.0094 and 0.0305 pixels, respectively. It must be mentioned that several of the targets were "structure deficient", *i.e.* they were composed of lines almost exclusively in one direction, in this case horizontal, leading to some large error values in the horizontal estimate. The mean and RMSE values would doubtlessly be lower if these samples were omitted.

4.1.3 Bias in Subpixel Location Estimates

It is worth considering whether a bias exists. Recall that the input data to the subpixel localization algorithm is a 3×3 array of correlation values, where it is assumed that the centre value is larger than the surrounding values.

In the case of a perfect match this centre value will be 1.0. If we were performing an autocorrelation we would expect the remaining values in the array to be symmetric around the centre. However, in calculating the remaining correlation values we consider parts of the search area that are not a part of the original model, so in general we cannot expect the correlation values passed to the subpixel localization algorithm to be symmetric about the centre. As a result, even if the centre correlation is 1.0, indicating a perfect match, the algorithm will always adjust by a non-zero amount because of the lack of symmetry. It may be then that "context" effects will produce a bias in the subpixel location estimates.

Since each line in Figure 9 represents the x-ordinate of a region in an image which is assumed to be moving rigidly, our regression should fit the same slope to each line. However, if a systemmatic bias exists, then it is possible that each line may have a different y-intercept. Another regression was performed that forces each line to have the same slope but different intercepts (see details in next section). The measured values for the parameters are shown in Table 3.

The slope value (α) is seen to be very close to our expectations, especially

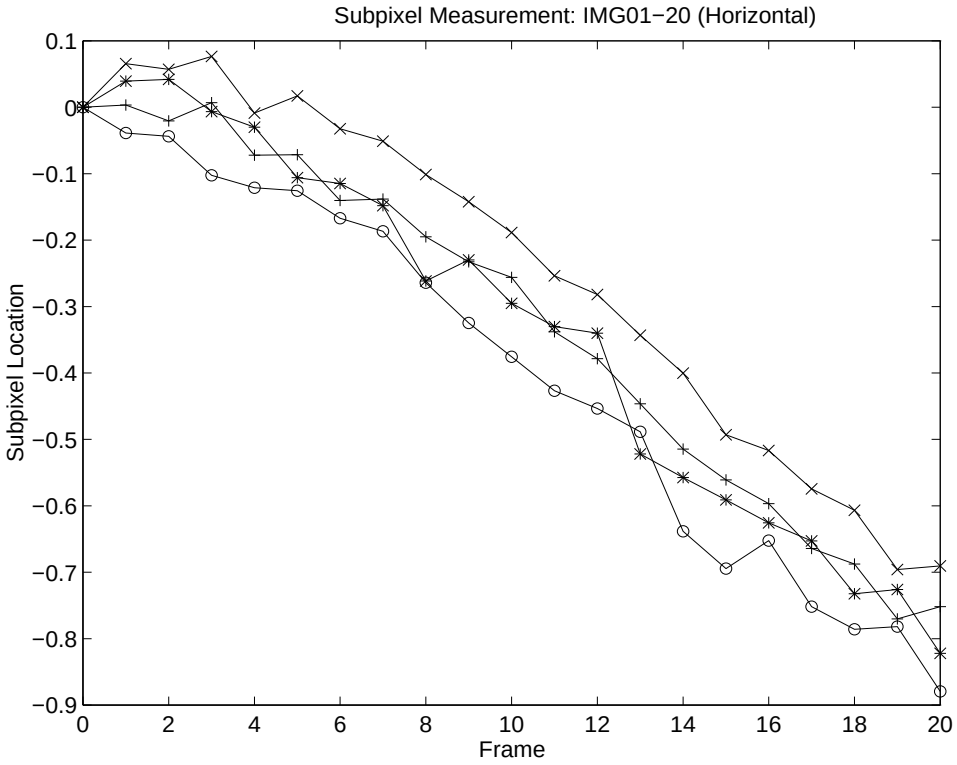

Figure 9: This figure is the same as Figure 8, except that here the horizontal ordinate of the subpixel localization estimate has been plotted. Over the 20 frames in the sequence, we see that the image shifts to the left by about 0.9 pixels.

Parameter	Value	
α	-0.0457	pixels/frame
β_1	0.1297	pixels
β_2	0.1389	pixels
β_3	0.2220	pixels
β_4	0.0651	pixels

Table 3:

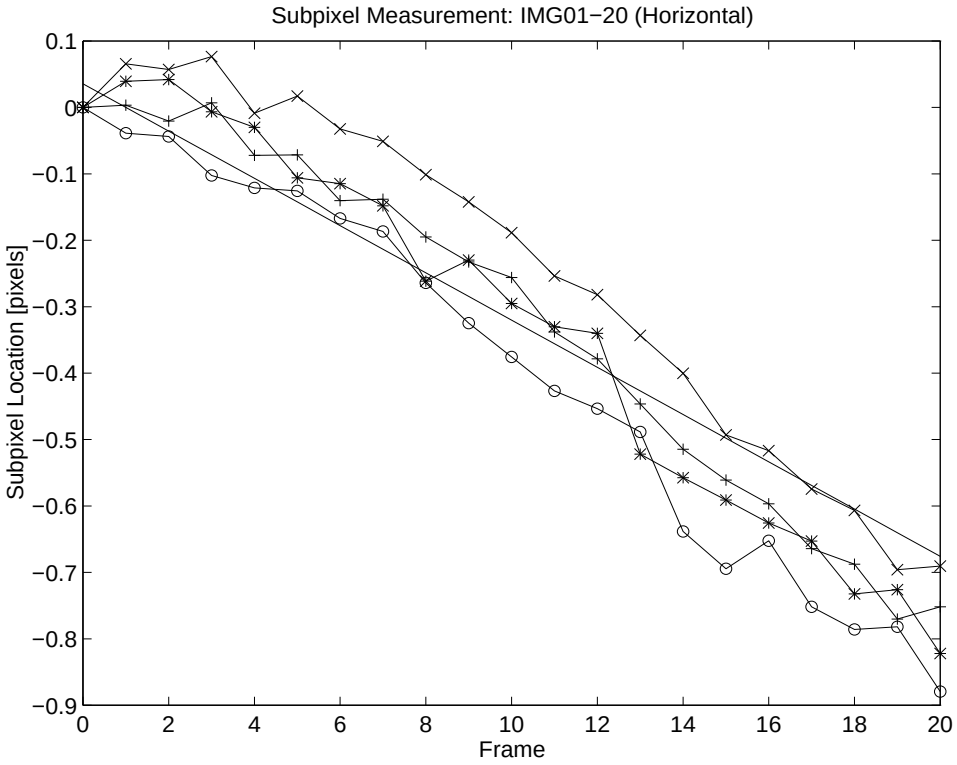

Figure 10: This figure shows the best-fit line (least-squares regression) fitting the subpixel location estimates. The regression was done assuming the regression line would have zero intercept.

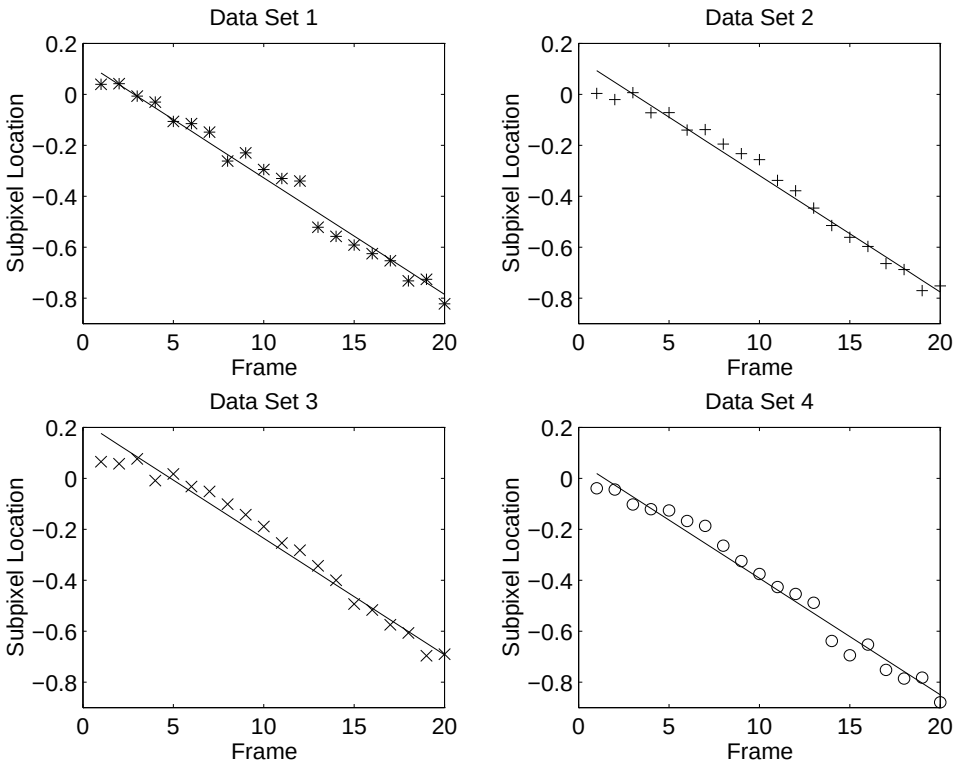
Data Set 1
Data Set 2
Data Set 3
Data Set 4
Subpixel Location
Frame
0.2
0
−0.2
−0.4
−0.6
−0.8
0
5
10
15
20

Figure 11:

when we consider the fact that the image doesn't appear to move by a whole pixel, but rather between about 0.7–0.9 pixels. In Figure 11 we see the fit of the regression for each data set. The data sets have been plotted separately in order to avoid confusion. This regression also appears to have provided a good fit to the data.

The y-intercept values are of particular interest. If a bias exists they could be reflected in the intercept values. In Table 3 we see that the intercepts range from 0.0651 pixels to 0.2220 pixels.

The regression parameters allow us to estimate error quantities. We define the error of an individual estimate as the difference between the measured value and the value predicted by the regression model. The mean error, calculated as

$$\bar{e}_x = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)$$

was found to be -1.7×10^{-16} pixels (this is less than machine epsilon – 2.2×10^{-16} – for the machine on which the regression was done). The root mean-square error, defined as

$$\sigma_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}\ ,$$

was found to be 0.0392 pixels. The mean error and RMSE values are lower than in the case of zero-intercept, but since we have fitted 5 parameters and not 1, this lower error does not necessarily mean the model is any better.

If such a bias exists, can it be removed? This is not sure, but there is one piece of information we haven't used yet, and that is the constraint that the correlation surface must never be greater than 1.0. The case detailed above indicates a bias exists even though a perfect match has been found: this indicates that the bi-quadratic model of the correlation surface has a maximum which is greater than 1.0. It may be possible to consider this new constraint and thereby reduce the bias.

5 Discussion

- talk about structurally deficient images

- talk about speed of recognition, and effect of number of pyramid levels
- talk about choice of threshold at level 0

6 Conclusion

Contributions:

- application of pyramid image representation to pattern recognition
- method for tuning the pyramid
- derivation/application of correlation gradient
- sub-pixel localization algorithm

Limitations:

- does not deal with size variation
- does not deal with rotation
- does not deal with illumination variation such as cast shadows which were not present when the model was analyzed
- does not deal with occlusion

7 Aknowledgements

The authors would like to thank Fernando Nuflo for his contributions in testing the recognition algorithm.

A Symbols Used

$I(x, y)$	the image
I_w	width of the image in pixels
I_h	height of the image in pixels
$\vec{I}$	subregion of image, same dimensions as model, represented as vector
$M(x, y)$	the model
M_w	width of the model in pixels
M_h	height of the model in pixels
$\vec{M}$	the model represented as a column vector
$C(x, y)$	the correlation surface defined by the model and the image
$W(x, y)$	a windowing function, of the same dimensions as the model
K	the number of levels in the pyramid
k	an indice in the range $0 \ldots K-1$ identifying a level in the pyramid
∇	the gradient operator
a	a scalar constant
θ	the angle between the $\vec{M}$ and $\vec{I}$
α, β, γ, δ, ξ and φ	parameters in the bi-quadratic correlation surface model
$\vec{\pi}$	parameter vector for bi-quadratic model
$\hat{C}(x, y)$	bi-quadratic estimate of $C(x, y)$
F	matrix of location coefficients for bi-quadratic model
(x_c, y_c)	location of maximum of bi-quadratic model
(x_i, y_i)	subpixel location estimates
N	number of subpixel location estimates
$\overline{y}$	average of y_i
σ_y	standard deviation of y_i