True-Motion Estimation using Feature Correspondences

Ralph Braspenninck and Gerard de Haan
Philips Research Laboratories, Prof. Holstlaan 4, Eindhoven, The Netherlands

ABSTRACT
We present a method for true-motion estimation assisted by feature point correspondences. First the difference between true-motion estimation and motion estimation for coding applications is explained, and an earlier published efficient true-motion estimation algorithm, called 3DRS, is summarized. Then the convergence property of this algorithm is discussed. We present a method for improving the convergence, by using feature point correspondences and show that a significant quality increase can be obtained for sequences containing high velocities.

Keywords: True-Motion Estimation, Feature Correspondence, Picture-Rate Up-Conversion

1. INTRODUCTION
Motion estimation is used in many video processing applications. One of the major application areas is motion compensated prediction for the coding of video signals, like MPEG-4 and H.264. Hence, a lot of research has been performed for this application area. In this paper we focus on picture-rate up-conversion applications. Picture-rate up-conversion increases the image frequency of a video signal by means of temporal interpolation. This application is used to match the image frequency of an incoming video signal to the frequency of the display on which it will be shown. Typically, this display frequency will be higher than the video frequency. Display frequencies often are 60Hz, 75Hz and 100Hz, whereas the video frequencies are 25Hz and 50Hz for respectively PAL film and video material and 24Hz, and 60Hz for NTSC film and video material. Note that, film material is "up-converted" for transmission or storage to interlaced video material by means of so-called 2-2 and 3-2 pull-down schemes, by re-interlacing and picture repetition. However, these video signals are always converted back to their original frequency by means of the inverse process prior to up-conversion.

For picture-rate up-conversion, the best quality is achieved when the objects in the images are interpolated along their motion trajectory [1]. This results in a smooth motion of the objects in the scene, as opposed to the jerky motion or judder created by picture repetition (often called motion judder). Picture-rate up-conversion poses severe constraints on the motion estimation algorithm. Particularly, the estimated vector field should have a very high correlation with the true motion of the objects in the scene. In Section 2, we show that this so-called true-motion vector field generally differs from a vector field that yields the lowest residual signal in predictive video encoding applications, like MPEG4 and H.264. Hence, popular coding motion estimation techniques, such as full-search and logarithmic search block-matching, are unsuitable for picture-rate up-conversion [1]. A more detailed analysis will be given in Section 2 along with some examples.

In the past, different motion estimation algorithms have been designed specifically for picture-rate up-conversion. Some good performing true-motion estimation algorithms are Bayesian motion estimation [2, 3], Phase Plane Correlation (PPC) [4, 5] and 3-D Recursive Search (3DRS) [1, 6]. The implementation cost for the Bayesian motion estimation methods is extremely high compared to the PPC and 3DRS algorithms. The performance of the last two methods is quite similar, but the implementation cost of PPC is still significantly higher than that of 3DRS. Since our focus is on consumer electronics, the implementation cost for an algorithm is very relevant. Hence, we shall start from 3DRS and present an extension to this algorithm. In Section 3, we shall introduce the original 3DRS algorithm.

Since the 3DRS algorithm uses a recursive search strategy, it needs time to converge to the correct solution. This convergence behaviour will be discussed in Section 4. As mentioned above, an extension to the 3DRS

e-mail: {ralph.braspenninck, g.de.haan}@philips.com
algorithm to improve the convergence and correctness of the vector field will be presented in Section 5. Furthermore, results of the comparison between the original and the extended 3DRS algorithm will be shown in Section 6. Finally we will draw conclusions in Section 7.

2. TRUE-MOTION VERSUS MINIMAL RESIDUE

The objective of block-based motion estimation algorithms used for predictive coding is to find for each block of pixels the displacement vector that minimizes the difference between the block of pixels of a reference frame and the block of pixels displaced by this vector of a predicted frame. Thus, minimizing the residual signal when subtracting the two frames using the found displacement vectors.

To find these displacement vectors a full-search strategy is often used as a reference algorithm and the difference metric is usually the Sum of Absolute Differences (SAD). Thus, for each block of pixels $B(x)$ at block position $x$ of frame $n$, the displacement vector $d(x, n)$ is found by testing all vectors from the set $CS$,

$$CS = \{c_i = (e_{i0}, e_{i1})^T | -v_{\text{max}j} \leq e_{ij} \leq v_{\text{max}j}, i = 0, 1\}$$

(1)

where $v_{\text{max}j}$ and $v_{\text{max}1}$ are respectively the maximal horizontal and maximal vertical displacements. In this paper we will use $v_{\text{max}0} = v_{\text{max}1} = 32$. The displacement vector for the block of pixels $B(x)$ then becomes

$$d(x, n) = \arg \min_i (\varepsilon(c_i, x, n)), \ c_i \in CS$$

(2)

and where $\varepsilon(c_i, x, n)$ is the SAD,

$$\varepsilon(c_i, x, n) = \sum_{x' \in B(x)} |f(x', n) - f(x' - c_i, n - 1)|$$

(3)

The block size is denoted by the vector $\beta$. For this study we use a block size of $8 \times 8$, i.e. $\beta = (\beta_0, \beta_1)^T = (8, 8)^T$. The resulting motion vector field using full-search is depicted in Figures 1 and 3 for two well-known sequences, ‘Mobile’ and ‘Foreman’, from the MPEG and II.264 test set. It is immediately clear from these vector plots that the vectors found with full-search do not always represent the true local object motion. For comparison the vector fields produced by the 3DRS algorithm are shown in Figures 2 and 4. These vectors have a much better correspondence with the true object motion. The 3DRS algorithm will be explained in Section 3.

Figure 5 shows the result when using full-search motion vectors for picture-rate up-conversion. This figure shows a detail from an interpolated frame for the Mobile sequence using the motion vectors from Figure 1. Figure 6 shows the same interpolated frame, however, using the 3DRS motion vectors from Figure 2. Clearly, considerable artifacts are visible in the digits of the calendar due to the noisiness of the full-search vector field. To interpolate the images, we used a robust temporal interpolation algorithm called ‘Dynamic Median’ [1].

The noisiness of the full-search vector field can be explained by looking more closely to the so-called SAD profile, which is the 3D surface of the SAD as a function of the motion vector. First consider Figure 9, which shows the SAD profile for position $(104, 56)$ of frame 25 of the Foreman sequence. This block lies near an edge in the building. For this block, there is an uncertainty of the component of the motion vector along the edge. Because of noise in the imaging path, the global minimum does not necessarily correspond to the true motion. The global minimum is shown by the white cross mark. The motion vector found by the full-search algorithm is not the true motion. This situation is a manifestation of the well-known aperture problem [5], i.e the fact that not for all areas of the image a unique motion vector can be found.

Now consider Figure 7. This figure shows the SAD profile for position $(256, 120)$ of frame 100 of the Mobile sequence. This is in the position of the rings of the calendar. As can be seen in both the image and the SAD profile, this is a repeating structure. Hence, there are numerous minima in the SAD profile spaced with the period of the repeating structure. And again the global minimum does not correspond to the true motion. The rings are quite small details. But, Figure 8 shows the same problem at a larger scale. This figure shows the SAD profile at the location of the small flower leaves above the ball. Because there are multiple similar flower leaves, again a wrong (too large) motion vector is found by the full-search algorithm. Finally, Figure 10 shows the SAD profile
of a block inside an area with little detail. For this block the uncertainty in the motion vector is even larger, illustrated by the large black area in the SAD profile. All situations described above are due to the aperture problem.

Because of the aperture problem, the SAD metric alone is not reliable enough to determine the true motion, as is illustrated by the examples mentioned above. Therefore, a better error measure or another search strategy is necessary for true-motion estimation. The efficient search strategies developed for motion compensated coding, like three-step search, four-step search, diamond search [7-9], and others, aim at finding the lowest residue at limited cost. Hence, these algorithms are equally unsuited for true-motion estimation.

3. 3-D RECURSIVE SEARCH

In absence of sufficient texture, there is an ambiguity in the motion vector, because of the aperture problem. Since this usually affects a significant amount of blocks in the image (see Figures 1 and 3), true-motion vectors
can only be estimated for blocks containing enough texture. For all other blocks, we will have to rely on motion vectors already estimated. Therefore, the 3DRS algorithm constructs a small set of candidate vectors based on spatio-temporal predictions. Thus, instead of testing all vectors in a certain search range, it limits the number of vectors to those already estimated, reducing the risk of reaching a wrong minimum.

The 3DRS principle is based on two assumptions. First, objects are assumed to be larger than blocks. Consequently, the vectors estimated for neighbouring blocks are good candidates for the current block. Since, the blocks are processed in a certain scanning order, e.g. from left to right and from top to bottom, there is always a causality problem, i.e. not for every neighbouring block a motion vector has been estimated yet. Motion vectors of neighbouring blocks that have been estimated in the current picture can be used as spatial prediction candidates, $c_i = d(x + p_i, n)$, with $p_i$ indicating the relative position with respect to the current block $x$, e.g. $p_i = (-\beta_0, 0)^T$. Other predictions have been estimated for a previous picture and, therefore, are called temporal candidates, $c_i = d(x + p_i, n-1)$, e.g. $p_i = (2\beta_0, 2\beta_1)^T$. This is depicted in Figure 11.

In addition to spatial and temporal candidates, update candidates are added to the candidate set. Update candidates are generated by adding small random vectors ($u$) to spatial candidates, i.e. $c_i = c_j + u$, $j \neq i$. These (random) update vectors are essential for the convergence of the motion field and to correctly track variable object motion. The update vectors can be relatively small because of the second assumption, which is that objects have inertia. This implies that the movement of objects varies gradually from picture to picture. In principle, the update vector can be a random variable, e.g. with a Gaussian or uniform probability distribution. In practice, it is sufficient to cyclically draw an update vector from a limited update set. An example of an update set, $US$, that in general results in a good performance of the 3DRS algorithm is

$$US = \left\{ \left(\frac{1}{4}, 0\right), \left(\frac{1}{2}, 0\right), \left(0, \frac{1}{4}\right), \left(-\frac{1}{4}, 0\right), \left(-\frac{1}{2}, 0\right), \left(0, \frac{1}{2}\right), \left(-\frac{1}{4}, 0\right), \left(-\frac{1}{2}, 0\right) \right\}$$

(4)

In summary, the candidate vectors $c_i$ of the candidate set $CS$ are constructed as follows

$$c_i = \begin{cases} d(x + p_i, n) & \text{if } c_i \text{ is a spatial candidate} \\ d(x + p_i, n-1) & \text{if } c_i \text{ is a temporal candidate} \\ c_j + u, & \text{if } c_i \text{ is an update candidate} \end{cases}$$

(5)

The candidate set used in this study contains two spatial candidates, one temporal candidate and two update candidates, resulting in five candidate vectors per block. The relative position of the prediction vectors (the $p_i$s) can be seen in Figure 11. Furthermore, the SAD described in Eq. (3) is used as a match error criterion.
Similar to the full-search algorithm, the match error $\varepsilon(c_i, \mathbf{x}, n)$ is minimized varying $i$ in order to obtain the best matching output vector. However, a small penalty is added to the SAD depending on the type of candidate [6]. The reasoning behind this penalty is that candidate vectors that are less likely to be the true-motion vector
for the current block must be less likely to be chosen in case all vectors have a similar SAD value. Typical values for the penalties $p_i$ are

$$p_i = \begin{cases} 
0 & \text{if } c_i \text{ is a spatial candidate} \\
\frac{1}{2} \cdot \beta_0 \cdot \beta_1 & \text{if } c_i \text{ is a temporal candidate} \\
2 \cdot \beta_0 \cdot \beta_1 & \text{if } c_i \text{ is an update candidate}
\end{cases}$$  \hspace{1cm} (6)

The penalty mechanism ensures a preference for the spatial candidates, therefore increasing the smoothness of the vector field. Update candidates are new vectors that have not been found to be the best matching vector in the neighborhood yet, and therefore are less likely than temporal candidates. Note that, after an update candidate has been chosen as output vector it becomes a spatial candidate for neighboring blocks.

In conclusion, the motion vector $d$, assigned to a block of pixels at location $x$ in frame $n$ is found by

$$d(x, n) = \arg\min_{i} (\varepsilon(c_i, x, n) + p_i), \quad c_i \in CS$$  \hspace{1cm} (7)

where the candidate set $CS$ is constructed according to Eq. (5). Note that the number of SAD computations per block is equal to the size of the candidate set, in the presented case five, making the 3DRS algorithm very efficient and suited for the implementation of a real-time system [10]. The vector fields shown in Figures 2 and 4 are estimated by the above described 3DRS motion estimation algorithm with a down-wards and an up-wards scan. In a down-wards scan the blocks are processed from left to right and from top to bottom, whereas in an up-wards scan the blocks are processed from right to left and from bottom to top. Since the location of the prediction vectors is defined relative to the scanning direction, the configuration shown in Figure 11 would be mirrored both horizontally and vertically for an up-wards scan.

4. CONVERGENCE

Since the 3DRS algorithm relies on the update candidates to find new motion vectors, the output needs to converge from an initial vector field to the correct solution. Obviously, for subsequent frames of a video sequence the solutions have a high correlation, and, therefore, small modifications of a previous vector field are sufficient. The 3DRS algorithm implicitly takes care of this by means of its temporal candidate.

However, video material often contains discontinuous changes, e.g. newly (or re-) appearing objects or scene changes. For these situations, new vectors, quite different from current output vectors, need to be found. The process for finding these new vectors works as follows: Starting from an initial motion vector a number of subsequent updates on this vector are necessary to converge to the correct solution. Say, the initial motion
vector is \( v_0 \) and \( N \) update vectors \( u_k \) are necessary to reach the correct solution. Then each intermediate vector \( v_i \) can be found by

\[
v_i = v_0 + \sum_{k=1}^{i} u_k, \quad 1 \leq i \leq N
\]  

(8)

Since update vectors are only added to spatial candidates, new motion vectors need to evolve over a number of blocks. This evolution is guided by the match error criterion, since this criterion is used to select whether to keep an existing vector (spatial and temporal candidate vectors) or to select a new vector (update candidate vectors).

If two neighbouring blocks belong to the same object, then the SAD profiles (as displayed in Figures 7 till 10) for these blocks will in general have a high correlation with each other. Hence, if the SAD profile between the vector and the true motion is a smooth strict monotonically decreasing surface, a lower value of the match error criterion will indicate a vector closer to the true motion and the vector converges with the smallest number of updates to the true motion. We will call this situation the convergence situation. An example of this is depicted in Figure 12.

This requirement on the SAD profile cannot be guaranteed for all distances between the vector and the true motion. Especially for larger distances this requirement cannot be met. If the requirement is not met, a lower value of the match error criterion will not necessarily indicate a vector closer to the true motion (think of local minima) and the convergence of the vector to the true motion is ruled by statistics, i.e. the chance of reaching the true motion is determined by the chance of reaching the convergence situation. Since the distance to the convergence situation (\( d \) in Figure 12) is unknown, the update set should be chosen such that a large search area is covered within the smallest number of updates.

For true-motion vectors with a length larger than the maximal length of the update vectors, it will take either a number of blocks, or a number of iterations or a number of frames before the correct solution is reached, if it is reached at all. In all cases this can lead to annoying artifacts in the interpolated images, especially if the solution is not reached at all, these artifacts are sustained over time. Hence, a fast convergence is important.

In the past, two extensions to the 3DRS algorithm have been proposed that are beneficial for the convergence speed. The first one is the use of an additional candidate vector that results from a parametric model of the global motion, see reference [11]. Good results are obtained when the motion model is an affine transformation:

\[
c_i = Ax + t
\]  

(9)

where the matrix \( A \) is a 2x2 matrix. The diagonal elements model motion produced by zoom, whereas the other two elements are required to model rotational movement. The translational movement is described by the vector \( t \). The estimation of the parameters of this motion model can be done in numerous ways. Since the precise method is not important for this paper, we will not go in detail, and assume a sufficiently accurate estimation of the parameters. The benefit of introducing a candidate from a motion model is mainly present for zoom and rotational motion. Since in these cases the motion vector changes for each block.

Another extension is to include the zero vector \( (c_i = (0, 0)^T) \) in the candidate set for each block. The reason for this is twofold. First of all stationary parts occur frequently in video material and secondly this candidate ensures that exactly the zero vector is found for stationary areas. By including the zero vector, the 3DRS algorithm does not need to converge from a (large) motion vector to zero if necessary.

A different method to improve the convergence speed could be to include larger update vectors in the update set. Since we like to keep the accuracy of the initial update set, the update set would have to be extended with the larger update vectors. Increasing the number of update vectors in the update set, increases the amount of noise inserted into the matching process and it reduces the frequency with which the update vectors are chosen. Although larger update vectors would be beneficial for fast moving sequences, Table 1 shows that it increases the noise of the vector field, by means of the Spatial Inconsistency (SI) metric. The test set used for these experiments, includes sequences with common motion portraits, that have been used before to evaluate motion estimation algorithms for picture-rate up-conversion [12].
As can be seen in Eq. (4) the maximum length of an update vector is 2 for the presented update set. For the experiment of Table 1 two additional update sets were used. For the first one, vectors with length 3 and 4 were added, hence 8 vectors were added to the update set. To create the second additional update set, the first additional set was again extended with vectors of length 8 (adding 4 additional vectors).

Table 1. The $SI$ and $M2SE$ metrics for 5 sequences with maximum length of the update vectors of resp. 2, 4 and 8. The last column indicates the relative $SI$ value with respect to a maximum length of 2.

<table>
<thead>
<tr>
<th>sequence</th>
<th>SI</th>
<th>M2SE</th>
<th>rel. SI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>bicycle</td>
<td>0.875</td>
<td>0.963</td>
<td>1.04</td>
</tr>
<tr>
<td>heli</td>
<td>0.883</td>
<td>0.958</td>
<td>1.02</td>
</tr>
<tr>
<td>hotel</td>
<td>1.29</td>
<td>1.37</td>
<td>1.41</td>
</tr>
<tr>
<td>mummy</td>
<td>0.633</td>
<td>0.660</td>
<td>0.680</td>
</tr>
<tr>
<td>ryan</td>
<td>0.375</td>
<td>0.423</td>
<td>0.428</td>
</tr>
</tbody>
</table>

The two error metrics used here, are the Spatial Inconsistency, which is a measure for the noisiness of the vector field and the Modified Mean Square Error ($M2SE$) [6, 13]. Both metrics have been used to evaluate motion estimation algorithms for picture-rate up-conversion. The $SI$ metric is defined as follows. For vector differences we use the L1 norm, $||v||_1 = |v_x| + |v_y|$. The local inconsistency, $\Delta(x, n)$, is then defined as

$$\Delta(x, n) = \frac{1}{8} \sum_{k=0}^{3} \sum_{l=0}^{3} ||d(x, n) - d(x + \begin{pmatrix} k \\ l \end{pmatrix}, n)||_1$$

The $SI(n)$ is the average local inconsistency over all blocks in the measurement window $W$ (which excludes the borders of the frame), i.e.

$$SI(n) = \frac{1}{|W|} \sum_{x \in W} \Delta(x, n)$$

The other performance indicator, $M2SE$, is a modification of the well-known MSE. The essence of the modification is that the validity of the vectors is extrapolated outside the temporal interval on which they are estimated. Since such extrapolation is only plausible in case the vector describes the true-motion, the $M2SE$ is more relevant than the MSE for judging vectors for scan-rate up-conversion. Frame $n$ can be reconstructed from frames $n-1$ and $n+1$ using the vector field $d(x', n)$ and motion compensated averaging, i.e.

$$f_{m.c}(x', n) = \frac{1}{2}(f(x' - d(x', n), n - 1) + f(x' + d(x', n), n + 1))$$

The $M2SE$ is then the MSE of the original frame $n$ and the reconstructed one, i.e.

$$M2SE(n) = \frac{1}{|W|} \sum_{x' \in W} (f(x', n) - f_{m.c}(x', n))^2$$

Rather than using the $SI$ and $M2SE$ of a single frame, the results presented here are the average value of the $SI$ and $M2SE$ of the first 4 frames of the sequences mentioned. Because we want to capture the convergence behaviour, only 4 frames are used.

From the results of Table 1, it can be concluded that including larger update vectors in the update set often increases the noisiness of the vector field, without an increase in accuracy. Furthermore, even with larger update vectors, it can still take a number of updates to reach the correct vector. And, diagonal vectors are also not covered. Hence, including larger update vectors is a sub-optimal solution for increasing the convergence speed.
5. Feature Correspondences

Therefore, we looked for predictions that are independent from the other predictions in the candidate set, similar to the motion model and the zero vector. Inspired by the work on large-amplitude disparity estimation we turned our attention to methods using feature point correspondences [14]. A method combining block matching and feature correspondences has been published before in [15]. However, both its target application and the block matching algorithm are different. Furthermore, it fixes the output disparity field to the found correspondences and uses a hierarchical full search regularized block matching to fill in the remaining part of the disparity field. Hence, spurious correspondences lead to errors in the disparity field. Our method takes the found correspondences as initial estimates and uses the 3DRS algorithm to construct a locally smooth and accurate vector field. Therefore, our method is more robust to spurious correspondences and can refine the found correspondences to sub-pixel (e.g. 1/4 pixel) accuracy, while maintaining its efficiency. In this Section we will propose a method for combining the sparse feature point correspondences with the 3DRS algorithm in order to improve the convergence.

Since, we are interested in 2-D motion vectors, we require features that can be localized reliably in two dimensions. The most commonly used features for this purpose are corners. To detect the corners in the images we use a well-known corner detection algorithm, called SUSAN [16]. Since, the presented method here is independent of the corner detection algorithm, it can be replaced by another corner detection algorithm. For now, we will assume that given a frame \( f(n) \), a set of feature points \( FP(n) \) is obtained by using a corner detection algorithm.

For calculating feature point correspondences a similar method as presented in [15] is used. First for each feature point \( x_i \) from \( FP(n-1) \) the means, \( \mu(x_i) \), and variances, \( \sigma^2(x_i) \), of the image samples of a block surrounding the feature points are calculated. These means and variances are used to calculate the correlation between two feature points \( x_i \) and \( x_j \), i.e.

\[
C(x_i, x_j) = \frac{1}{|B_{fp}|} \sum_{\delta \in B_{fp}} \frac{(f(x_i + \delta, n - 1) - \mu(x_i)) \cdot (f(x_j + \delta, n) - \mu(x_j))}{\sigma(x_i) \cdot \sigma(x_j)}
\]  

(14)

where \( B_{fp} \) contains the relative pixels position of a block surrounding a feature point, i.e. \( B_{fp} = \{ \delta | -M_i \leq \delta_i \leq M_i, i = 0, 1 \} \). Hence, the block size is \((2M_0 + 1) \cdot (2M_1 + 1)\). Given a feature point \( x_i \in FP(n-1) \), the feature point \( x_{mc}(x_i, n) \in FP(n) \) that maximizes the correlation from Eq. (14) is searched, i.e.

\[
x_{mc}(x_i, n) = \underset{x_j \in FP(n)}{\arg \max} C(x_i, x_j), \quad x_i - x_j \in S
\]  

(15)

where \( S \) is the search range, i.e. \( S = \{ v | -v_{max} \leq v_i \leq v_{max}, i = 0, 1 \} \).*

An additional local candidate vector is generated only if \( x_{mc}(x_{mc}(x_i, n), n-1) = x_i \). Thus, a bi-directional correlation must be found between the feature points. Furthermore, the correlation values must lie above a certain threshold, \( c_{th} \). (We used a value of 0.85 for this study). This local candidate vector is put into an additional vector field, called \( c_{fp} \), i.e.

\[
c_{fp}(x, n) = \begin{cases} 
  x_i - x_{mc}(x_i, n) & \text{if } x_i \in B(x) \land \ x_{mc}(x_{mc}(x_i, n), n-1) = x_i \land \ C(x_i, x_{mc}(x_i, n)) > c_{th} \\
  (0, 0)^T & \text{else}
\end{cases}
\]  

(16)

The zero vector candidate is replaced by the vector from this vector field taken from the appropriate position. Hence, at positions where no correspondence is found, the zero vector is still the candidate, otherwise, the vector found by the feature correspondence becomes a local candidate. The penalty for this candidate is set equal to an update candidate, because it could result from a spurious correspondence.

*For this study we used \( M_0 = M_1 = 4 \) and \( v_{max} = 120, v_{max} = 20 \) pixels.
A second method to incorporate the vectors found by the feature correspondences as global candidates, is to cluster these vectors by means of, for instance, a simple vector quantization scheme. The method used here divides the 2-D plane into equally sized square cells and counts the number of vectors that lie within one cell. A representative vector of the 3 cells containing the most vectors are added as candidate vectors for each block. This representative vector can be, for instance, the mean of all vectors that lie in the cell, or the center of the cell. The latter option was chosen here, in combination with cells centered around the pixel positions. The penalties for the global candidate are set equal to an update candidate, since these vectors are not equally good candidates for all blocks.

6. RESULTS

As was illustrated in Sections 2 and 3, the 3DRS algorithm has little problems with estimating the true motion for the sequences typically used for benchmarking motion estimation algorithms for coding applications. Therefore, to evaluate the effect of incorporating feature point correspondences into the 3DRS algorithm, we selected 6 more challenging sequences that contain fast motion, since for these kinds of sequences in particular convergence is a problem. Thumbnails of these sequences are depicted in Figure 14. All but one sequence have a resolution of 720x576 pixels, the other has a resolution of 720x480 pixels.

The first sequence, called camel, contains a static background with fast horizontal and diagonally moving text. The second sequence, called chophunt, shows a chopper that is tracked by the camera, hence the background moves fast in a slightly diagonal direction. The third sequence, called fnjwar, shows various object moving with different speeds, ranging from slow to fast. The fourth sequence, called subtext, shows a fast scrolling subtext. In the fifth and sixth sequence, called zorro and lord, again the foreground objects are tracked, resulting in a fast moving background.

This test set was used to calculate the M2SE values (see Section 4) for different configurations. The results are shown in Figure 13. It shows a comparison between the original 3DRS algorithm described in Section 3 and three different configurations using the feature point correspondences, i.e., only local injection of candidates (Local), only global injection of candidates (Global) and both (3DRS+FC). The results show that the global

\begin{figure}[ht]
\centering
\includegraphics[width=0.5\textwidth]{figure13.png}
\caption{M2SE numbers of the test set for the normal 3DRS algorithm, the extension using locally the feature correspondences (Local), using globally the feature correspondences (Global) and using both methods (3DRS+FC).}
\end{figure}

\begin{figure}[ht]
\centering
\includegraphics[width=0.5\textwidth]{figure14.png}
\caption{The test set of sequences with from left to right and from top to bottom resp. camel, chophunt, fnjwar, subtext, zorro, lord.}
\end{figure}

The threshold for the feature point detection was set to 10 for the sequences containing motion blur and 50 for the ones without motion blur.

\footnote{The threshold for the feature point detection was set to 10 for the sequences containing motion blur and 50 for the ones without motion blur.}
injection gives a benefit for most sequences, whereas, the local injection gives a benefit for all sequences. Using both at the same time only gives an improvement for the load sequence. As is depicted in the chart, on the average a reduction of around 45% can be obtained for the test set using the methods presented in Section 5. Note that only the \( M2SE \) values are shown since \( SI \) values cannot be compared if the difference in \( M2SE \) is large.

**Table 2.** The \( SI \) and \( M2SE \) metrics for the original 3DRS algorithm, 3DRS with local candidate injection and both local and global candidate injection (3DRS+FC) using the test set used in Section 4.

<table>
<thead>
<tr>
<th></th>
<th>3DRS</th>
<th>Local</th>
<th>3DRS+FC</th>
<th>3DRS</th>
<th>Local</th>
<th>3DRS+FC</th>
<th>3DRS</th>
<th>Local</th>
<th>3DRS+FC</th>
</tr>
</thead>
<tbody>
<tr>
<td>bicycle</td>
<td>0.875</td>
<td>0.930</td>
<td>0.921</td>
<td>66.5</td>
<td>64.2</td>
<td>63.7</td>
<td>1.0</td>
<td>1.06</td>
<td>1.15</td>
</tr>
<tr>
<td>heli</td>
<td>0.883</td>
<td>0.915</td>
<td>0.938</td>
<td>84.0</td>
<td>82.6</td>
<td>81.3</td>
<td>1.0</td>
<td>1.04</td>
<td>1.06</td>
</tr>
<tr>
<td>hotel</td>
<td>1.29</td>
<td>1.26</td>
<td>1.29</td>
<td>267</td>
<td>266</td>
<td>266</td>
<td>1.0</td>
<td>0.98</td>
<td>1.00</td>
</tr>
<tr>
<td>mummy</td>
<td>0.633</td>
<td>0.620</td>
<td>0.680</td>
<td>38.3</td>
<td>38.1</td>
<td>50.1</td>
<td>1.0</td>
<td>0.98</td>
<td>1.07</td>
</tr>
<tr>
<td>ryan</td>
<td>0.375</td>
<td>0.407</td>
<td>0.400</td>
<td>104</td>
<td>103</td>
<td>104</td>
<td>1.0</td>
<td>1.09</td>
<td>1.07</td>
</tr>
</tbody>
</table>

In Section 4 we have shown, using another test set, that including larger update vectors is a sub-optimal solution, because it increases the noisiness of the vector field for regular sequences. Therefore, in Table 2 the results are listed for the method presented in Section 5 using feature point correspondences. For ease of comparison the results for the original 3DRS (max. update length of 2) from Table 1 are restated in Table 2 (column 3DRS). One can see that especially for the local injection of candidates the \( SI \) is not significantly different from 3DRS, as opposed to the results from Table 1. Hence, for regular sequences the noisiness is not increased by using feature correspondences.

**7. CONCLUSIONS**

We have shown that a search strategy with the only objective of finding the lowest residual for each block, is not suited for true-motion estimation. Hence, a different search strategy is required for true-motion estimation. We presented such a search strategy, called 3-D Recursive Search (3DRS), that has been published before. For sequences containing high velocities this algorithm has difficulties with converging to the correct solution. Including larger update vectors in the update set of 3DRS is a sub-optimal solution for improving the convergence, because it increases the noisiness of the vector field for regular sequences.

We have presented an alternative method using feature point correspondences that does not have the same drawback. Two methods of incorporating the correspondences have been discussed and evaluated. The local injection of candidate vectors gave the largest improvement. Using feature point correspondences it is possible to achieve a reduction of around 45% of the Modified Mean Square Error for sequences containing high velocities. Hence, it is beneficial to combine the efficiency and accuracy of 3DRS for finding a true-motion vector field for limited velocities with the ability of feature correspondence methods to find high velocities.

**REFERENCES**