Adaptive rood pattern search for fast block-matching motion estimation

IEEE TRANSACTIONS ON IMAGE PROCESSING

Yao Nie, Kai-Kuang Ma, IEEE senior member
Contents

1. Paper Study
2. Experiment Result
3. Conclusion
Block-matching algorithm

1) Divide every image into square blocks
2) Find one displacement vector for each block
3) Within a search range, find a best match that has the minimum matching error
4) Intelligent search algorithm can reduce computation

The most straightforward BMA is the full search which exhaustively searches for the best matching block within the search window. However, FS yields very high computational complexity. Thus, using a fast BMA is indispensable to reduce computational cost.
Disadvantage in FBMA using a fixed set of search patterns:

In this category, the methods are based on the assumption that ME matching error decreases monotonically as the search moves toward the position of the global minimum error, so the error surface is unimodal.
Disadvantage in FBMA using a fixed set of search patterns:

But in fact, the uni-modal error surface assumption is not valid in most of the cases especially when the motion is large and complex.

The picture on the left shows non-unii-modal error surface with multiple local minimum error points.

When the size of the fixed search pattern does not match the magnitude of the actual motion, over search or under search will be incurred which can cause certain search deficiency and inaccuracy.
• If the pattern size we choose is small, the small pattern tends to be trapped into local minimum point along the search path and leads to wrong estimation.
• If the pattern size we choose is big, it well cause unnecessary search for small motion (small mv).

Therefore, it is highly desirable to use different search patterns according to the estimated motion behavior for the current block, this is the so-called adaptive. This boils down to two issues required to be addressed:

1) How to pre-determine the motion behavior of the current block for performing efficient ME?

2) What are the most suitable size and shape of the search pattern?
1) How to pre-determine the motion behavior of the current block for performing efficient ME?

- **Two ways:**

  In the temporal domain:
  - block in the reference frame at the same position as that of the current block is a straightforward choice.

  In the spatial domain:
  - neighboring blocks from the same frame could also provide promising candidates for prediction as well.

  But if we choose temporal prediction, we need recording the entire previous mv. It takes more storage space which we don’t want to see. So, the author choose the spatial prediction.

Raster-scan order:
- left
- above-left
- above
- above-right
1) How to pre-determine the motion behavior of the current block for performing efficient ME?

- 4 types of prediction method
  - type A
  - type B
  - type C
  - type D

- 2 types of computational methods
  - mean value
  - median value

With extensive experiments performed between all the type and computation way, the author found that both PSNR and search point number are similar.

- PSNR within 0.1dB
- search point number within 5%

So, the author choose type D

For saving storage space and less computation complexity
2) What are the most suitable size and shape of the search pattern?

For the initial search:

The shape of our rood pattern is symmetrical with four search points locating at the four vertices. The main structure of ARP has a rood shape and its size refers to the distance between any vertex point and the center point.

Advantage:

1) Most motion are in this 4 directions.
2) At least detect the major trend of the moving object.
3) Benefits hardware implementation.
2) What are the most suitable size and shape of the search pattern?

So, the author chose the simplify formula to determine the size of ARP. From the experimental results, the simplify formula is superior to the first one in terms of higher PSNR.

\[
\text{size} = \text{Round} \left| \text{MV}_{\text{predicted}} \right|
\]

\[
= \text{Round} \left[ \sqrt{\text{MV}_{\text{predicted}}^2(x) + \text{MV}_{\text{predicted}}^2(y)} \right]
\]

Round means taking the nearest integer value.

Involves square and square-root operation which increase difficulty on hardware implementation.

\[
\text{size} = \text{Max}\{\left|\text{MV}_{\text{predicted}}(x)\right|, \left|\text{MV}_{\text{predicted}}(y)\right|\}
\]
2) What are the most suitable size and shape of the search pattern?

For refined local search:

The initial search leads the new search center directly to the most promising area which is around the global minimum.

The assumption of uni-modal error surface formed in this area would be quite valid.

So we can use a fixed and small search pattern to perform local refined search unrestrictedly for identifying the global minimum.

Experiment results show that the 3x3 square pattern yields similar PSNR but requires 40%-80% more checking points among all the fixed pattern.

Iterative procedure: the MME point found in the current step will be re-positioned as the new search center of the next search iteration until the MME point is incurred at the center of the fixed pattern.
Paper Study

- Zero-Motion Prejudgment (ZMP)

<table>
<thead>
<tr>
<th>Video and Coding Bit-rate (Kbps)</th>
<th>Average Static Blocks Percentage per Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo(10)</td>
<td>93.36%</td>
</tr>
<tr>
<td>Container(10)</td>
<td>90.55%</td>
</tr>
<tr>
<td>Hallmonitor(10)</td>
<td>95.18%</td>
</tr>
<tr>
<td>Mother&amp;Daughter(24)</td>
<td>80.36%</td>
</tr>
<tr>
<td>Silence(24)</td>
<td>78.79%</td>
</tr>
<tr>
<td>News(112)</td>
<td>84.61%</td>
</tr>
<tr>
<td>Tennis(1024)</td>
<td>70.44%</td>
</tr>
<tr>
<td>Coastguard(112)</td>
<td>17.29%</td>
</tr>
<tr>
<td>Foreman(1024)</td>
<td>37.41%</td>
</tr>
</tbody>
</table>

From the table, we can see that significant additional reduction in computational cost is possible if we perform a zero-motion prejudgment (ZMP) at the beginning of ME.

So, we first computing the matching errors between the current block and the block at the same location in the reference frame and comparing it with a predetermined threshold $T$. If the SAD value is less than $T$, we consider it as the static block and performing no search.

In this paper, the author choose $T=512$. 
Experiment Result

Experiment environment:

Operating System:
Windows Xp

Software Development Environment:
Visual C++ 6.0

To compare the search time and the PSNR between Full Search, Diamond Search and ARPS-ZMP, I write those three BMA methods and perform those methods with different video sequence.

To simplify the source code, I use the number of search point instead of the search time. Because for a certain computer, the more points to search means more times to cost.
## Experiment Result

<table>
<thead>
<tr>
<th>video</th>
<th>FS</th>
<th>DS</th>
<th>ARPS-ZMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman_cif</td>
<td>984.92</td>
<td>16.33</td>
<td>7.28</td>
</tr>
<tr>
<td>News_qcif</td>
<td>984.92</td>
<td>13.74</td>
<td>2.01</td>
</tr>
<tr>
<td>Mother_cif</td>
<td>984.92</td>
<td>13.59</td>
<td>2.49</td>
</tr>
<tr>
<td>Mobile_cif</td>
<td>984.92</td>
<td>13.27</td>
<td>6.23</td>
</tr>
<tr>
<td>Crew_cif</td>
<td>984.92</td>
<td>17.52</td>
<td>9.16</td>
</tr>
<tr>
<td>Missa_cif</td>
<td>984.92</td>
<td>16.76</td>
<td>5.44</td>
</tr>
</tbody>
</table>

*Average number of search point per mv generation*
## Experiment Result

<table>
<thead>
<tr>
<th>video</th>
<th>FS</th>
<th>DS</th>
<th>ARPS-ZMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman_cif</td>
<td>33.09</td>
<td>32.86</td>
<td>32.71</td>
</tr>
<tr>
<td>News_qcif</td>
<td>35.28</td>
<td>35.02</td>
<td>35.02</td>
</tr>
<tr>
<td>Mother_cif</td>
<td>39.45</td>
<td>39.09</td>
<td>39.11</td>
</tr>
<tr>
<td>Mobile_cif</td>
<td>23.91</td>
<td>23.79</td>
<td>23.80</td>
</tr>
<tr>
<td>Crew_cif</td>
<td>30.31</td>
<td>30.03</td>
<td>30.05</td>
</tr>
<tr>
<td>Missa_cif</td>
<td>37.67</td>
<td>37.30</td>
<td>37.49</td>
</tr>
</tbody>
</table>

Average PSNR performance on FS DS and ARPS-ZMP
Conclusion

- Compared with FS DS, ARPS improves the search speed with a great computational gain. Meanwhile, ARPS maintains similar PSNR performance of FS in most of the sequences.

- Therefore, ARPS is a very efficient and robust ME algorithm for video coding.