Illinois Institute of Technology

ECE 565 Multidimensional Digital Signal Processing
(Fall '98)

Project 2: Motion Estimation Using Block Matching

12/08/1998

Jovan Brankov

Step by step procedure with results

1. I wrote tree routines for block-matching of NxN block in ±M surrounding of the block, and one that performs a global search for the block, on the whole image, by using correlation (like the match filter). These routines are called from the *mkdv* program for each block of target image. The *Mkdv* program makes displacement vectors (DV's) for the whole image depending on the desired method, block size, searching area size and constants C and Ro. C and Ro will be defined later.

The routines are *opti*, *sopti*, *fopti*, and *copti*. All routines try to minimize the cost function defined as:

$$SAD_{B_i} = \sum_{\underline{n}\in B_i} \left| I_t(\underline{n}) - I_r(\underline{n}-\underline{v}) \right|,$$

with respect to v, over some searching area S for the block B_i, so we are looking for:

$$v_{-0} = \arg\left\{ \min_{\underline{v}\in S-\{0\}} \sum_{\underline{n}\in B_i} \left| I_t(\underline{n}) - I_r(\underline{n}-\underline{v}) \right| \right\}.$$

In order to measure the quality of the estimated image we can define:

$$SAD = \sum_{B_i} SAD_{B_i} \,,$$

where the summation is over all blocks of the frame.

Routine descriptions:

- *opti* performs the searching for the best match around the current point, at relative positions:

$$l * \begin{bmatrix} (-1,-1) & (-1,0) & (-1,1) \\ (0,-1) & \# & (0,1) \\ (1,-1) & (1,0) & (-1,1) \end{bmatrix},$$

where l is an arbitrary jump from the current point. This routine is called from *mkdv* several times with different l. Although we do not use current point it can happen that v becomes zero.

More about this can be found in the handout from the class on p.502.

- *sopt* performs searching just on two neighboring points of the current point, in the same direction and based on the result, determines the new current point. This is repeated for certain number of times. The possible size of searching area depends on the number of steps.

More about this can be found in the handout from the class on p.502.

These two algorithms can be "trapped"!

---------- ---- ---------

For example, take the image like this:

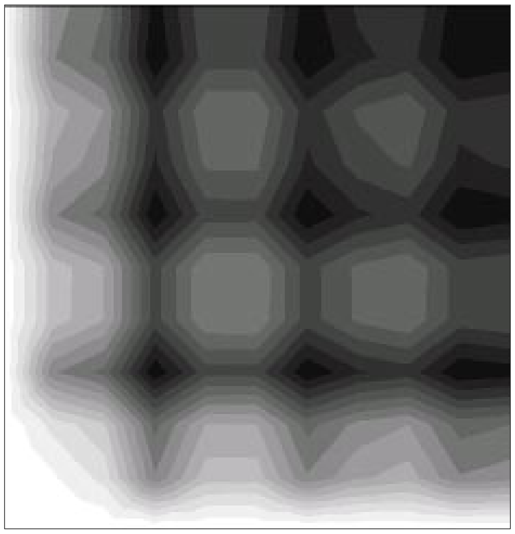


Figure 1. Example of tough reference image.

and we try to find the template:

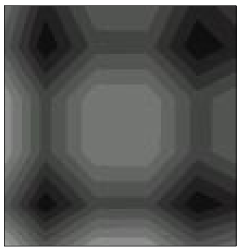


Figure 2. Example of tough target block.

There are several points where the pervious algorithms will stop, because the cost function has several local minimums and the results will be dependent on the starting point.

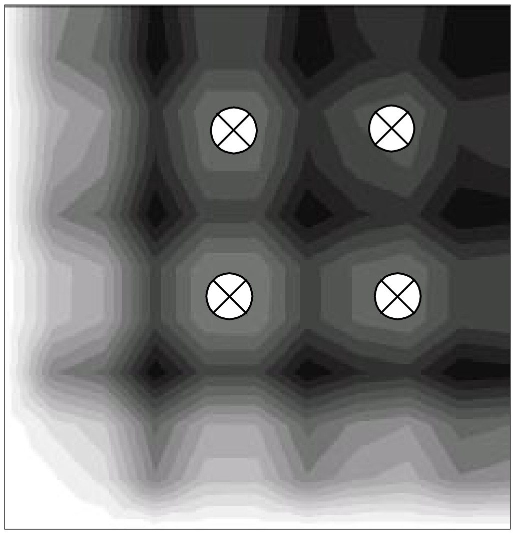


Figure 3. The several solutions.

- *fopti* uses "brutal force" algorithm which goes through every point in the searching area and finds the global minimum.
- *copti* computes normalized cross-correlation between the reference image and target block. Normalization is necessary because the frame has unequal brightness at its points. The maximum of the cross-correlation gives us the center of the region that matches target the best. Searching area is the whole reference image.

2. Done.

In the following experiment, we use two frames, one serves as a reference and another as a target (desired) frame.

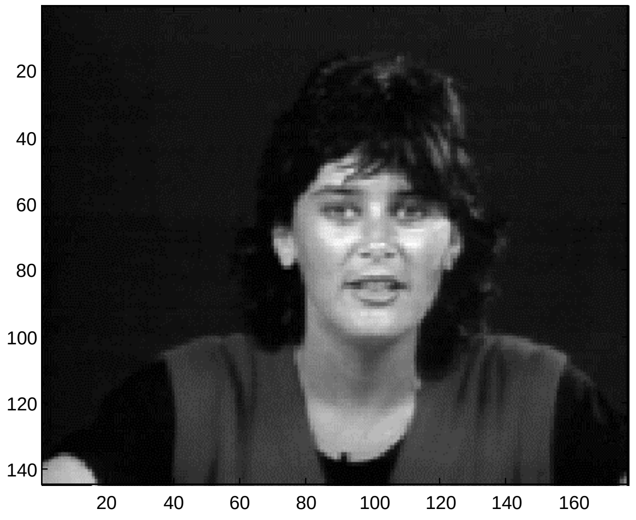


Figure 4. Reference frame.

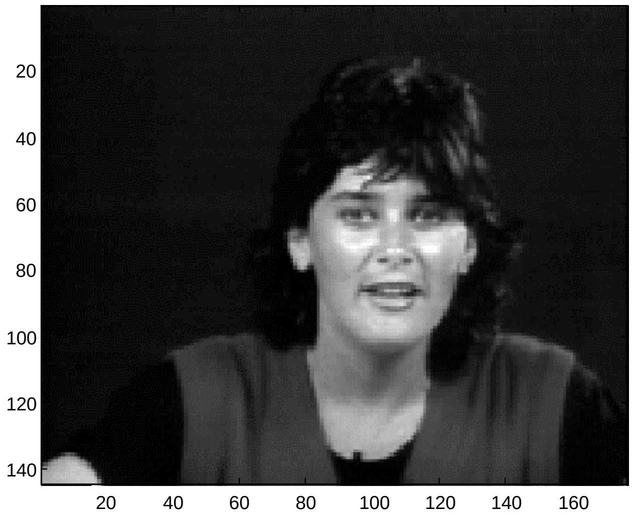


Figure 5. Target frame.

3. Parameter that was obtained by experiment, C, represents threshold parameter, which reduces unneeded motion vectors (v). These motion vectors are present because the solution for v, is not unique, especially for smooth surfaces on the frame. With choosing the suitable C we can improve the quality of the MF.

If $d_{-0} < d_0 - c|B_i|$

then $v^* \leftarrow v_{-0}$,

else $v^* \leftarrow 0$.

In later discussion I will use "brutal force" (point by point search) algorithm, but first let us consider the costs of using such dummy algorithm by looking at the computation time and accuracy of the result. I used C=1, the 8x8 blocks and searching in ±8 surroundings of the block.

	execution time	method	SAD	search area
opti	22.8600	"jump by 3,2,1,3,2,1"	150961	possible 12x12 area
sopti	14.4380	single step decision	66490	possible 16x16 area
fopti	119.4220	point by point search	45657	16x16
copt	143.4380	normalized correlation	49887	whole image

Which algorithm is better, can be determined by using the real motion pictures with real data compression where we can measure the quality of obtained images vs. compression rate or computation time (delay). These measurements must be performed on the bunch of different frame types and then take the average. The goodness of each of these algorithms depends on the type of the motion. For example for "big" motions and noisy frames the last one should be the best, but for relatively restricted motions between frames, the second one, single step decision, should be good enough, yet very simple and fast.

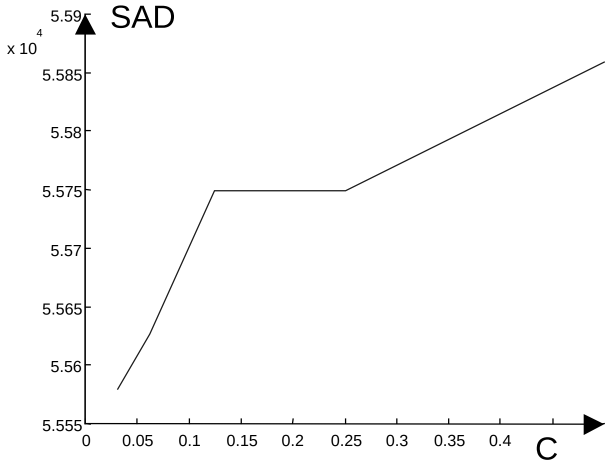
In the MF computations, I assumed that all pixels from the target frame are on the reference frame. That means that there are no new objects in the target frames. That is a rough assumption, but if we remember that, we do not have the way to know what is outside of the frame or behind the object, it seems good enough. In addition, it seems logical that we can deal with boundaries by not searching in the areas that are out of the frame. Because of that, we will have all DV's addressing the blocks inside the reference frame.

In order to determinate the value for C, I used two methods simultaneously:

- measuring the SAD of the estimate frame
- visual inspection

The first one is more precise, but it is hard to define mathematically that you do not care about the motion of the background surface. Moreover, sometimes we can not distinct two frames with different SAD. Therefore, I have chosen C for which the MF has less DV's in the areas where there is "no motion".

On the next figure, SAD vs. C value is shown.



.

Figure 6. SAD vs. C parameter for 16x16 blocks.

It can be seen that for smaller values of C, the SAD is better, as it was expected.

By visual inspection, in order to minimize unnecessary MF vectors I decided to pick 0.25 value for C. For this decision I used a knowledge that, in these cases, there is no motion on the background and that the SAD on Figure 6. starts to rise after that point.

So, to summarize, for the following results, I used point by point search, in ±8 surrounding of the 16x16 block on the reference frame, for the best matching in the target frame, and C= 0.25. The ending of the DV is at the center of the block, like as in given example on the Web, for all MF plots.

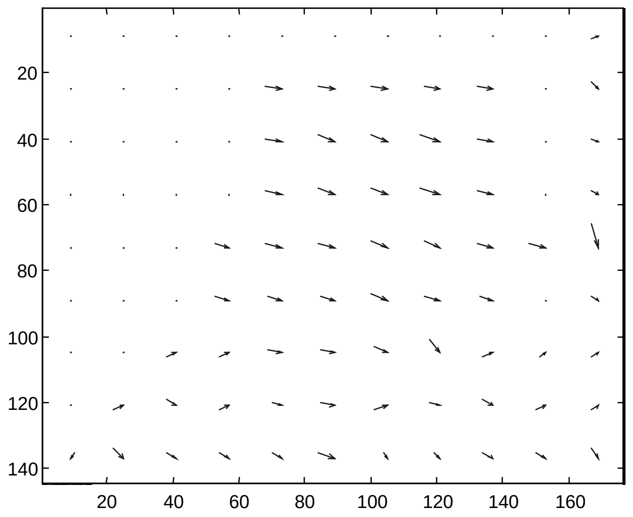


Figure 7. Motion field resulted from using 16x16 blocks and C=0.25 DV Number = 64.

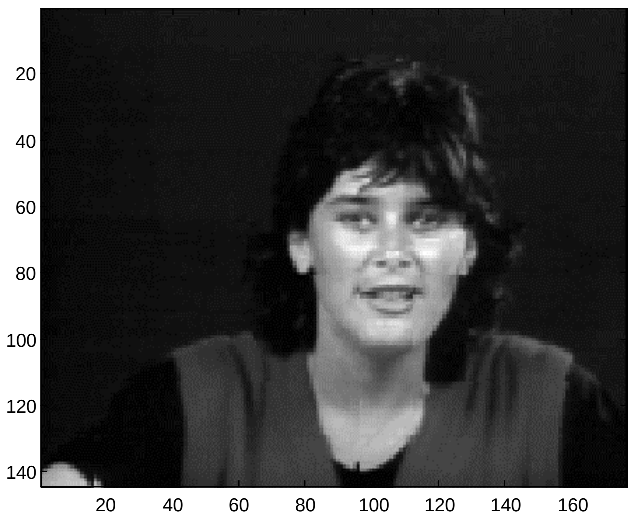


Figure 8. Motion compensated prediction frame resulted from using 16x16 blocks and C=0.25.

The frame difference can be defined as:

$$Error = \sum_{\underline{n} \in frame} \left| I_t(\underline{n}) - I_r(\underline{n}) \right|, \quad Error_{mc} = \sum_{\underline{n} \in frame} \left| I_t(\underline{n}) - \tilde{I}(\underline{n}) \right|,$$

where the first error is the difference between the reference and target frame, and the second is the difference between the reference and motion compensated frame.

For the first case, that can be seen on Figure 9., we have $Error = 177859$, and for the motion compensated frame, Figure 10., $Error_{mc\_16x16} = 55749$. From this, it can be seen that with motion compensation we reduced "the power" of the error.

To display the displaced frame difference (DFD) and the frame difference (FD) I used normalization:

$$x_{norm}(n_1, n_2) = round\left(\frac{255}{\max\limits_{n_1, n_2 \in A}\left(\left|x(n_1, n_2)\right|\right) - \min\limits_{n_1, n_2 \in A}\left(\left|x(n_1, n_2)\right|\right)} * \left(\left|x(n_1, n_2)\right| - \min\limits_{n_1, n_2 \in A}\left(\left|x(n_1, n_2)\right|\right) \right) \right)$$

where A is some area in a frame. We do not take the whole image, since there might be an extremely bright point that would make all the other points look equally dark. For every DFD, and FD the scaling area is in the range of 1 to 130 in n_1 direction, and in the range of 1 to 176 in n_2 direction.

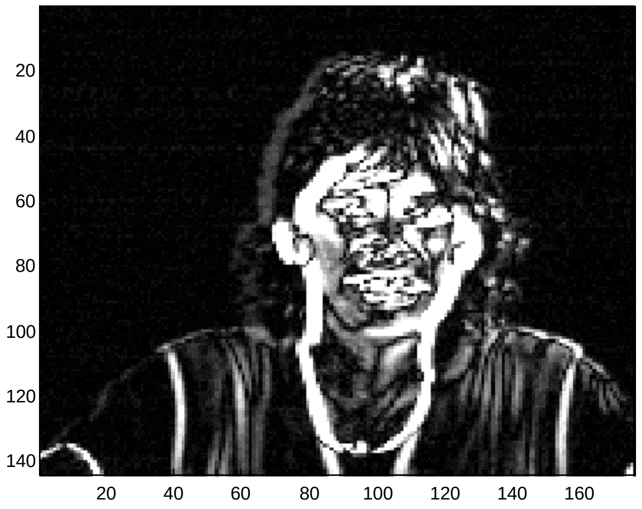


Figure 9. Frame difference without the motion compensation.
$Error = 177859$

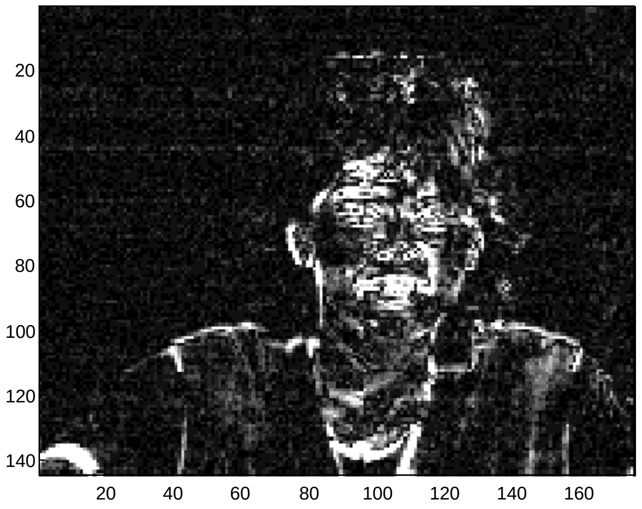


Figure 10. Frame difference with the motion compensation resulted from using 16x16 blocks and C=0.25.
$Error_{mc\_16x16} = 55749$

----

4. Using the same C parameter as above, for 8x8 block size we have:

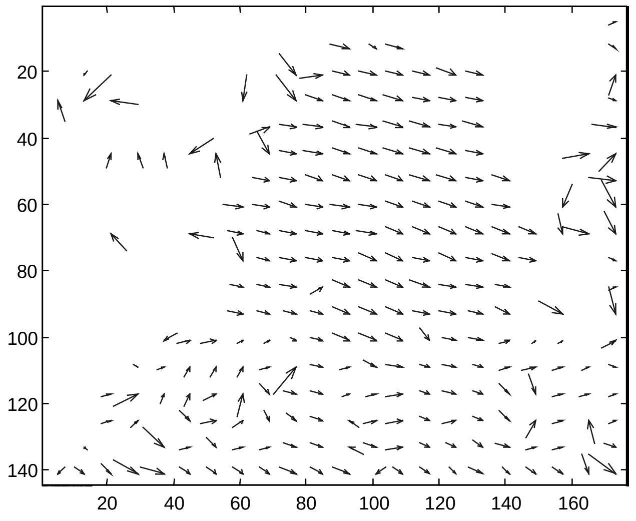


Figure 11. Motion field resulted from using 8x8 blocks and C=0.25
DV Number = 248.

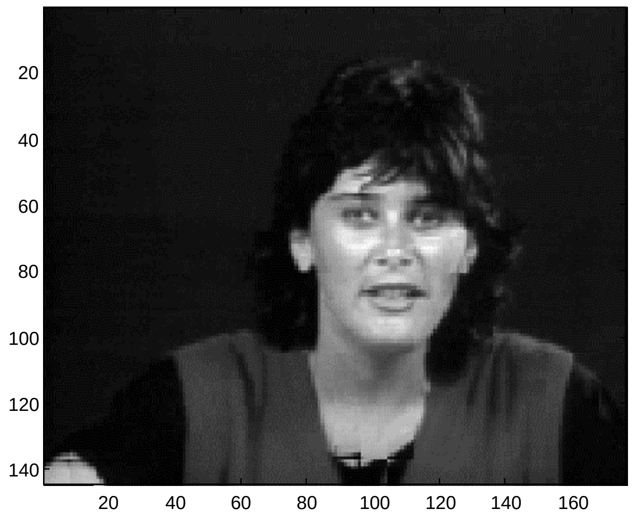
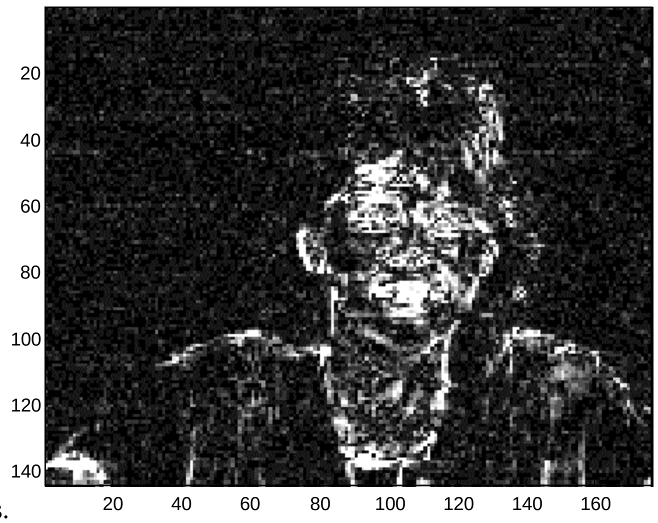


Figure 12. Motion compensated prediction frame resulted from using 8x8 blocks and C=0.25.



\ plots.

Figure 13. Frame difference with motion compensation resulted from using 8x8 blocks and C=0.25.
$Error_{mc\_8x8} = 44571$

In this case, we have $Error_{mc\_8x8} = 44571$.

With the same method as above for obtaining C, I picked C=0.6 for "the best" MF for 8x8 block size.

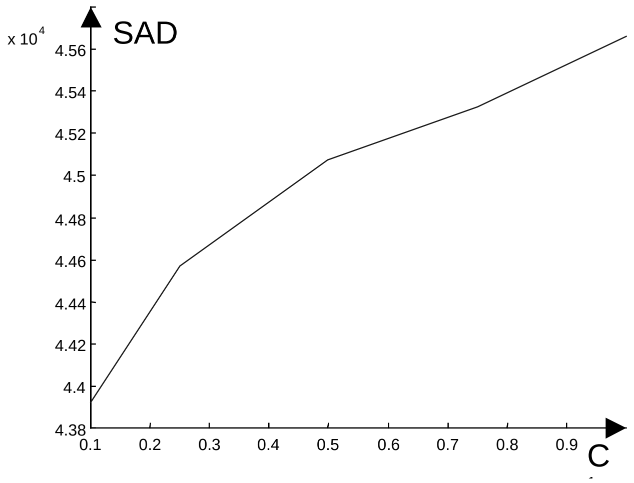


Figure 14. SAD vs. C parameter for 8x8 blocks.

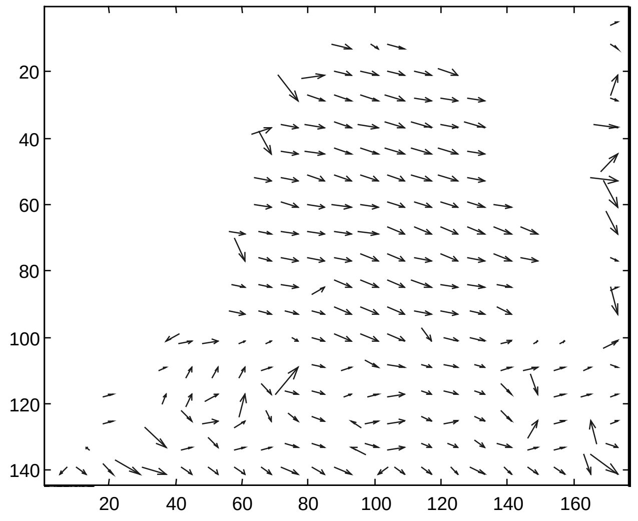


Figure 15. Motion field resulted from using 8x8 blocks and C=0.6
DV Number = 224.

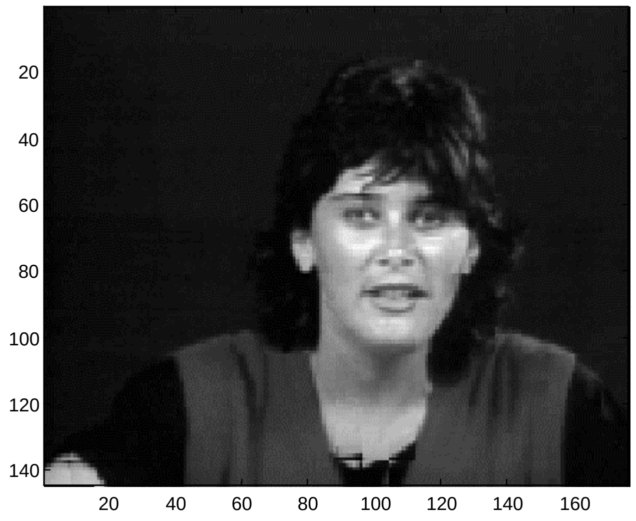


Figure 16. Motion compensated prediction frame resulted from using 8x8 blocks and C=0.6.

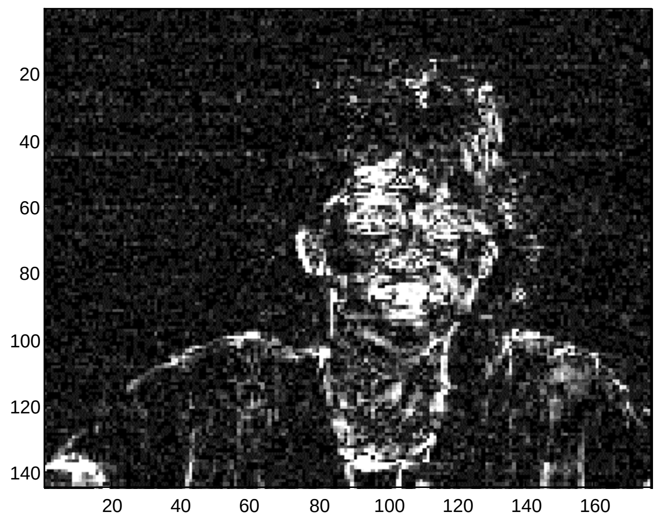


Figure 17. Frame difference with motion compensation resulted from using 8x8 blocks and C=0.6.

$Error_{mc\_8x8} = 45143$

Now, the residual error is: $Error_{mc\_8x8} = 45143$.

Let us summarize:

block size	C	error	DV’s Number	coment
no motion compesation	-	177859	-	-
16x16	0.25	55749	64	"best" MF for 16x16
8x8	0.25	44571	248	MF for 8x8
8x8	0.6	45143	224	"best" MF for 8x8

From this table and previous examples we can see, that with the motion compensation, we can predict one frame from previous one "very well" and so, reduce the nessesery information. By choosing parametar C, we can make "nice" MF and reduce unneeded DV’s that come from not uniqesness, of the solutions for DV and noise on the frames.

Bonus work

As we defined erlier, the block error is:

$$SAD_{B_i} = \sum_{\underline{n} \in B_i} \left| I_t(\underline{n}) - I_r(\underline{n} - \underline{v}) \right|,$$

it can be rewritten as:

$$SAD_{B_i} = \sum_{j=1}^{4} SAD_{B_{i,j}} \;,$$

where

$$SAD_{B_{i,j}} = \sum_{\underline{n} \in B_{i,j}} \left| I_t(\underline{n}) - I_r(\underline{n} - \underline{v}) \right|.$$

$B_{i,j}$ is defined as a partition of B_i without gaps and overlapping. Therefore, we can assume that we can compute the error for each sublock and by summing, we can get the error of B_i block. If the error is not uniformly distributed over the B_i, that is, $SAD_{B_{i,j}}$ has "big" variance, v is not good for the each subblock $B_{i,j}$ of the B_i block. If we divide B_i in sub blocks and compute DV for each of the $B_{i,j}$, maybe, we can get a better result. By not dividing the whole frame in the smaller blocks, we have less computations and faster algorithms.

Smirarly as for C, now we need to find the best value for Ro, which is defined as:

If $Ro * |B_i| < \max\left(SAD_{B_{i,j}} - \dfrac{SAD_{B_i}}{4} \right)$

then $v^* \leftarrow v_{B_i}$,

else we made subpartition of the B_i

The parameter Ro is a threshold, for the decision when to divide B_i, block into subblocks.

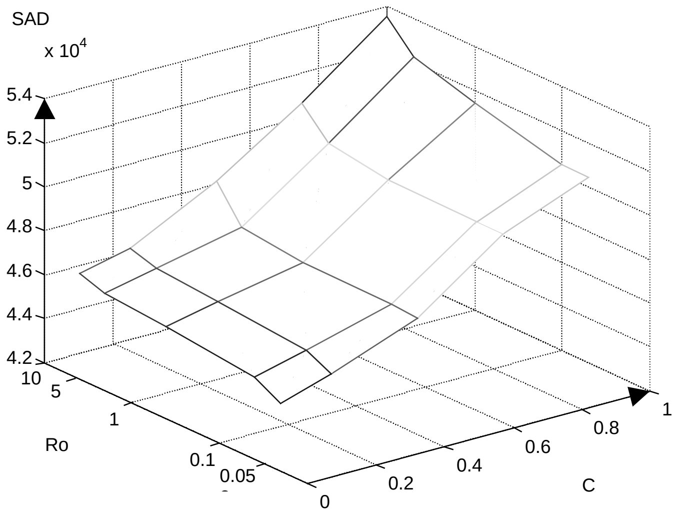


Figure 18. SAD vs. C, Ro parameters for adaptive block size.

By visual inspection, with task to minimize unneeded MF vectors and not to make SAD too big, I decided to pick 0.4 value for C and 6.5 for Ro. The results of the procedure by using these parameters are:

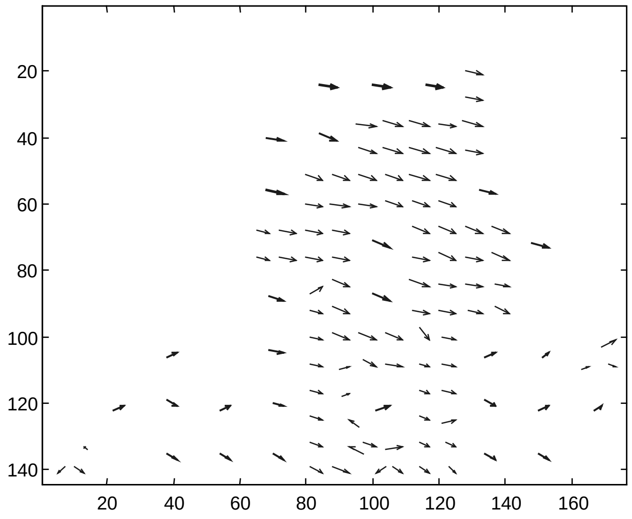


Figure 19. Motion field resulted from using adaptive block size, C=0.4 and Ro=6.5
DV Number = 181.

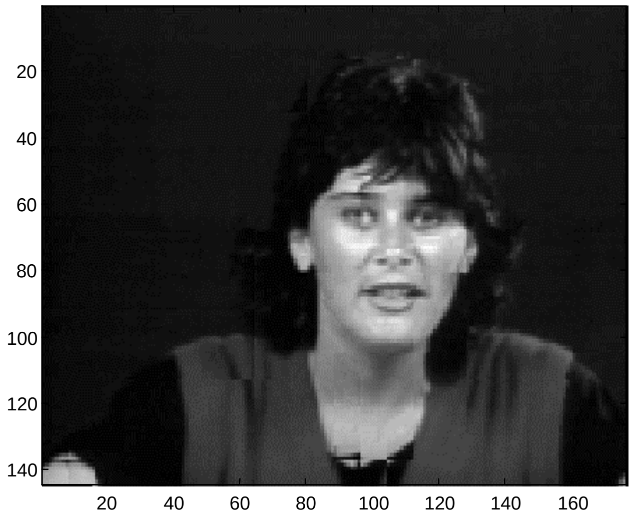


Figure 20. Motion compensated prediction frame resulted from using adaptive block size, C=0.4 and Ro=6.5.

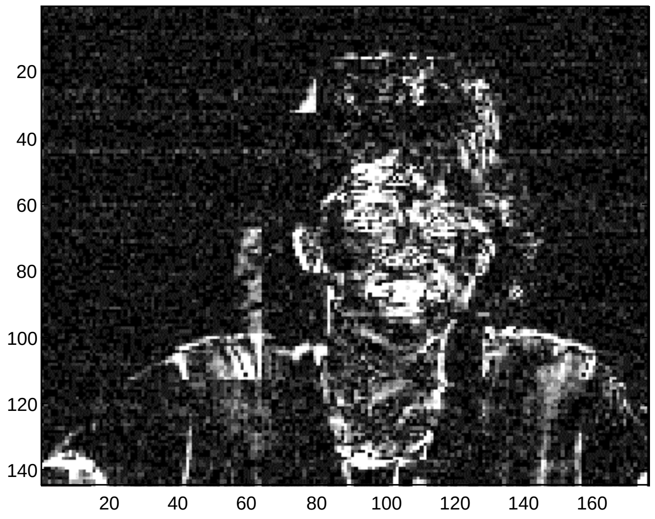


Figure 21. Frame difference with motion compensation resulted from using adaptive block size, C=0.4 and Ro=6.5.

$$Error_{mc\_adap} = 49827$$

We can see, from next table, that the error is larger than in the case when we used 8x8 block but still smaller then in the case when we used 16x16 blocks.

block size	computation time	error	parametars	DV Number
16x16	31.0630	55749	C=0.25	64
8x8	119.3900	45143	C=0.6	224
adaptive block size	61.8900	49827	C=0.4 Ro=6.5	181

We can see also, that we made the computation time half, compared to 8x8 block search algorithm, but we still have smaller error then in 16x16 block case. Also we reduced the number of DV's.

Bonus+ work

In trying to make MF using Spatio-temporal constranit method, from the handout p. 503-506, I did not have success. I think that my program for interpolation works properly, but I did not have enough time (or knowledge) to finish that task.

For interpolation I used this basis: $\phi_i \in$ { 1, x, y, t, x^2, y^2, xy, xt, yt }.

The function that was to be minimized in order to get a coeffcients of this interpolation is:

$$Error = \sum \sum_{(n_1,n_2,n_3)\in\psi} \sum \left(f(x,y,t) - \sum_{i=1}^{n} S_i \phi_i(x,y,t) \right)^2 \bigg|_{x=n_1T,\, y=n_2T,\, t=n_3T}$$

To minimize this function, with the repect to S_i coeffcients I used MATLAB function *fmin.*

*A*ll my programs for this thask have prefix *fit****.m.*

For the block size of 3x3 pixels, at the position (80,100) and initialy displacement of (8,8) I have:

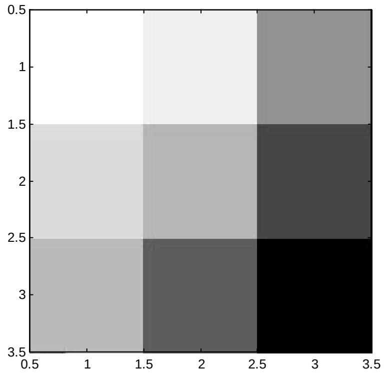


Figure 22. Block from reference frame at (80,100).

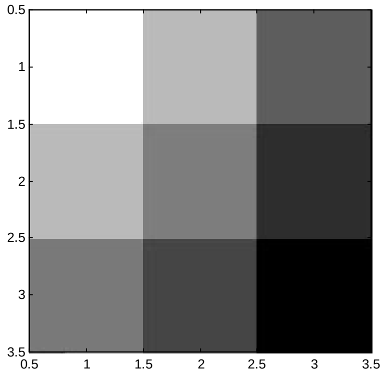


Figure 23. Interpolated reference block at (80,100).

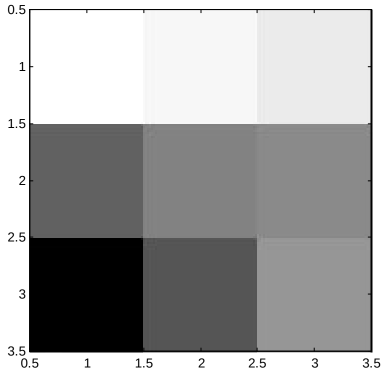


Figure 24. Block from target frame at (88,108).

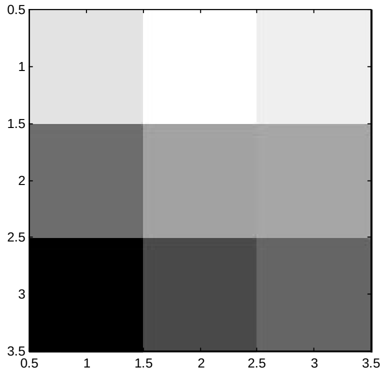


Figure 25. Interpolated target block at (88,108).

This works properly but as I said the rest does not. The next step would be to make a frame that is in time between the reference and target, using MF obtaind by BM algoritahms. Verifying the results of my programs in that task could help me to find what I did wrong.

Interpolated frame at the half of the way, in time, between the reference and target frame resulted from using $\phi_i \in \{ 1, x, y, t, x^2, y^2, xy, xt, yt \}$ basis and MF obtained by BM algorithms for 4x4 block and assuming that dispalcement is half for each 4x4 block . and C=0.4.

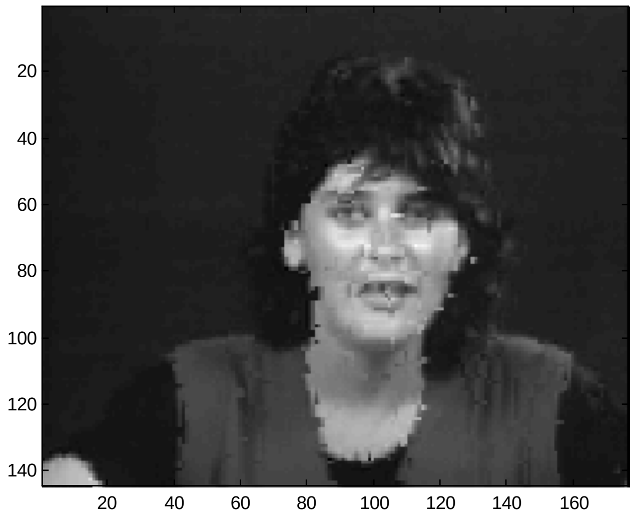


Figure 26. Interpolated at the half of the way, in time, between the reference and target frame.

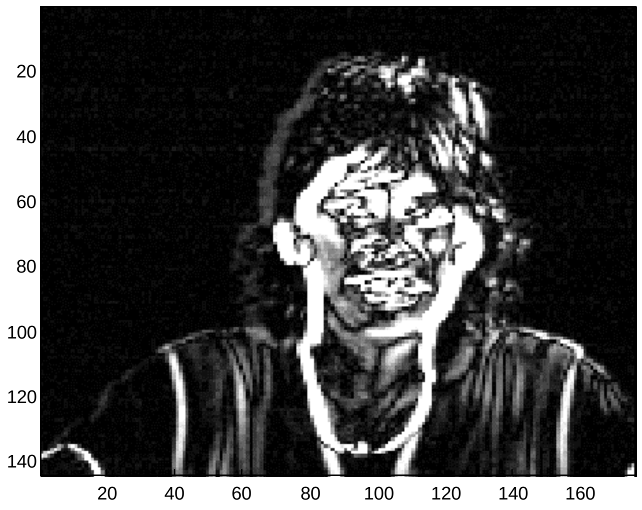


Figure 27. Frame difference between the referenc frame and the target frame.

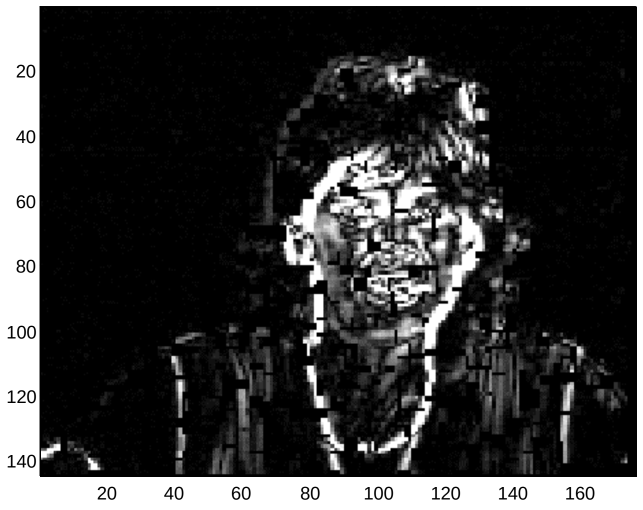


Figure 28. Difference between interpolated frame and the target frame.

I did not get the result as I expected. Using MF obtaind by BM algoritham, and dividing each DV by two gave better resul then this procedure.