Developing Compression Technique for Computer Vision Applications

Basar Turgut,
Institute of Computer Science
University of Tartu
turgutbasar@gmail.com

Abstract— **The significant progress in developing distributed computer vision systems leads to increasing data streams in machine-to-machine communications. Efficiency of data transmission in these systems can be improved by using compression techniques modified in order to satisfy specific requirements of computer vision algorithms.**

Motion compensation is one of the key methods used in video compression area. It implies prediction of motion between consecutive frames. A current frame is represented by motion vectors and a difference frame. In this study, we modify the standard block matching motion compensation to tailor it according to needs of computer vision algorithms. An iterative algorithm for constructing object maps is considered.

I. INTRODUCTION

Computer vision is a rapidly developing scientific research area which aims at automating functionalities of biological visionary systems. In this field, imaging sensors play the same role as eyes in biological visionary systems. The global market size of surveillance cameras is expected to reach 43,200 thousand by 2018 [1]. Captured images and video sequences are processed by computer systems modeling image and video processing abilities similar to human brains. IoT, smart surveillance cameras, machine learning approaches, machine oriented communication techniques and other computer vision related topics are nowadays more important than ever.

High computational complexity of computer vision algorithms leads to a necessity of distributing vision applications. Distributed computer systems with cloud based solutions for parallelization of heavy computations become more and more popular for such applications. In such systems, clients feed video packages to cloud hosted algorithms. However, cloud based architectures with non-compressed video packages require high bandwidths of communication channels. Besides that, a demand to speed up processing is growing day by day [2]. We can reduce both the required bandwidth of communication mediums and processing time by using compression techniques tailored to the needs of computer vision techniques. Due to well designed coding techniques, we can add pre-processing techniques used for both compression and visionary algorithm benefits.

Processing algorithms include extracting different image (video) features, image restoration, motion tracking, storing and transferring images (video) or their selected features between parties.

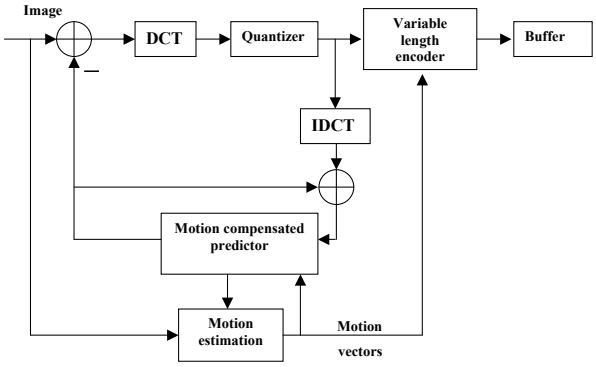

Fig. 1. Motion Compensation Block Scheme [3]

We will focus on motion related compression and transfer techniques. While we are focusing compression, proposing method will fit with needs of motion oriented computer vision techniques. We study motion compensation approach, in particular, foreground detection, differential motion vector based representation and lossless compression techniques as a baseline for the proposed method. A typical video coder with motion compensation technique is shown in Fig.1. It performs motion compensation which results in two motion vectors (vectors of shifts over abscissa and ordinate). A shifted version of the previous frame (prediction) is obtained in motion compensation predictor. The difference between the current frame and the prediction is subjected to discrete cosine transform (DCT), transform coefficients are quantized. Then zig-zag scanning from JPEG standard and lossless RLE (Run-Length Encoding) are applied to the quantized matrix.

II. BACKGROUND

Our main goal is developing motion compensation based compression compatible with needs of computer vision algorithms. In particular, information of important regions (regions of interest) should be compressed without visible distortions [3].

Although we mainly study motion compensation approach, we start with a short overview of standard video

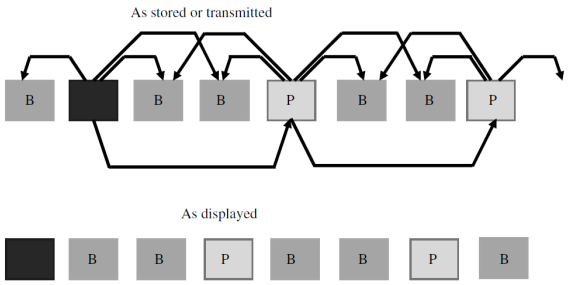

Fig. 2. Frame Prediction [3]

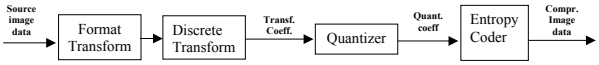

Fig. 3. Lossy Compression of Frames [3]

coding techniques. Typically, all video frames can be split into so-called intra (key) frames and inter frames (Fig.2), to which motion compensation is applied. A standard video coder for key frames is shown in Fig.3. It includes color format transform (RGB-to-YUV), discrete transform (discrete cosine transform (DCT)), quantization and lossless (entropy) coding [3].

A. Color Space Transform

We mentioned that representations of visual content are tailored for human vision system. In this context, most of capture libraries support RGB or BGR color space as default. However, YUV will be a better choice if we want to focus on motion and object details than color. We can apply motion concerned compression techniques on Y channel and use simpler lossy techniques to compress UV channels. Considering that, UV channels do not contain crucial information for motion related computer vision applications [3].

$$Y = 0.299R + 0.587G + 0.114B \tag{1}$$

$$U = (B - Y)0.5643 \tag{2}$$

$$V = (R - Y)0.7132 \tag{3}$$

B. Discrete Cosine Transform

An image component (Y or U, V) is split into blocks of given size $M \times N$, say. Typically, $M = N = 8$. Coefficients of 2D DCT are computed for each block as follows.

$$y_{k,l} = c_k c_l \sqrt{\frac{2}{M}} \sqrt{\frac{2}{N}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{m,n} T \tag{4}$$

where T is,

$$T = \cos\left(\frac{\pi k\,(n + 1/2)}{N}\right) \cos\left(\frac{\pi l\,(m + 1/2)}{M}\right) \tag{5}$$

and Where $y_{k,l}$ is the (k,l)-th element of the matrix of transform coefficients, $x_{m,n}$ is the (m,n)-th pixel value.

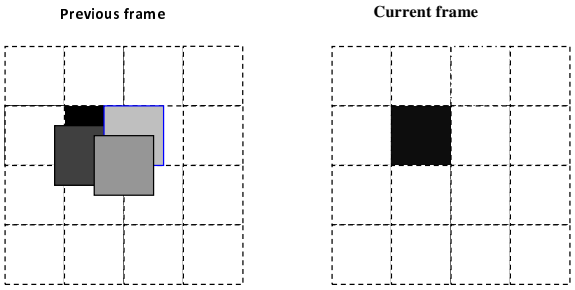

Fig. 4. Block Matching [3]

$$c_i = \begin{cases} 1/\sqrt{2}, i = 0 \\ 1, otherwise \end{cases} \tag{6}$$

C. Quantization

Quantization performed according to the formula,

$$q_{k,l} = \left\lfloor \frac{y_{k,j}}{step} \right\rfloor \tag{7}$$

where $q_(k,l)$ denotes the quantized value, $\lfloor\rfloor$ denotes rounding and step is quantization step. Reconstruction is performed as,

$$y_{k,j} = q_{k,j} step \tag{8}$$

D. Motion Compensation

The main idea behind motion compensation technique is estimating motion between two neighboring frames in order to reduce their difference. In fact, such prediction can be performed in different ways and use of different searching techniques. In what follows, we work on fixed block matching motion compensation approach [4].

Basically, we can use any block matching algorithm with selected metrics as similarity measure. In our experiments, RMSE metric was used. We use exhaustive search to estimate frame differences and motion vectors. Using generated motion vectors, we can construct an approximation of any frame in the form of block shifted version of the previous frame. However, approximation will still differ from the original frame. We should calculate the difference of approximation and the original frame to correct prediction errors while reconstructing frames. Difference between the current frame and the prediction can be compressed by standard compression techniques (i.e.JPEG).

We use fixed MXN sized moving blocks to estimate motion. Basically, it works as searching shift of the block in its search boundaries. We define this boundaries as $-M/2 < IndexX < M/2$, $-M/2 < IndexY < M/2$. For each block in the current frame we consider all possible shifts of the co-sited block in the previous frame as shown in Fig.4. The coordinates X and Y corresponding to the best value of the search criterion for this block are stored in two motion vectors which contain the corresponding coordinates for all

blocks of the frame. Obviously, exhaustive search over all possible shifts, which is performed for each block, needs a considerable amount of computational power. However, complexity can be reduced by applying different kinds of logarithmic search [6].

III. PROPOSED METHOD

$MapObj \Leftarrow zeros(NumOfBlocks)$
$MVecX(1:L) \Leftarrow 0$
$MVecY(1:L) \Leftarrow 0$
$Obj \Leftarrow CurrFrame$
$ObjP \Leftarrow PrevFrame$
$Iter \Leftarrow 1$
while $Iter < L$ **do**
  $[X, Y, Err] \Leftarrow GlobalMotionEst(Obj, ObjP)$
  while $IterObjBlock < NumOfBlocksObj$ **do**
    if $abs(Err(Block)) > Threshold$ **then**
      $MapObj(BlockNumber) \Leftarrow NewObj$
    end if
  end while
  $Obj \Leftarrow CurrentFrame(MapObj(NewObj))$
  $ObjP \Leftarrow Prev_Frame(MapObj(NewObj))$
  $MVecX(iter:L) \Leftarrow MVecX(iter:L) + X$
  $MVecY(iter:L) \Leftarrow MVecY(iter:L) + Y$
  $Iter \Leftarrow Iter + 1$
end while
while $IterCurBlocks < NumOfBlocksCur$ **do**
  $[X, Y, Err] \Leftarrow BlockMotionEst(CurrFrameBlock, PrevFrameBlock)$
  $MVecXBlock \Leftarrow MVecX(ObjectNum) + X$
  $MVecYBlock \Leftarrow MVecY(ObjectNum) + Y$
end while

Fig. 5. Pseudocode of the proposed method

A variety of computer vision algorithms require motion compensation techniques tailored to specific goals of these systems [5]. They focus on tracking movement of region of interest. An image (frame) can be split into different importance components. Then different compression techniques can be applied to these components. In such a way one can significantly reduce the bandwidth required for transmission without visual distortions of the important parts of frame.

Commonly used in video compression standards, block matching motion compensation method was considered in Section II. It implies that an image is partitioned into rectangular blocks of a fixed size and motion compensation is performed for each block. In this case the corresponding components of motion vectors represent the difference in position between the block from the current frame and the displaced block from the reference frame. The global motion compensation[3] is one more motion compensation technique which efficiently takes into account an entire

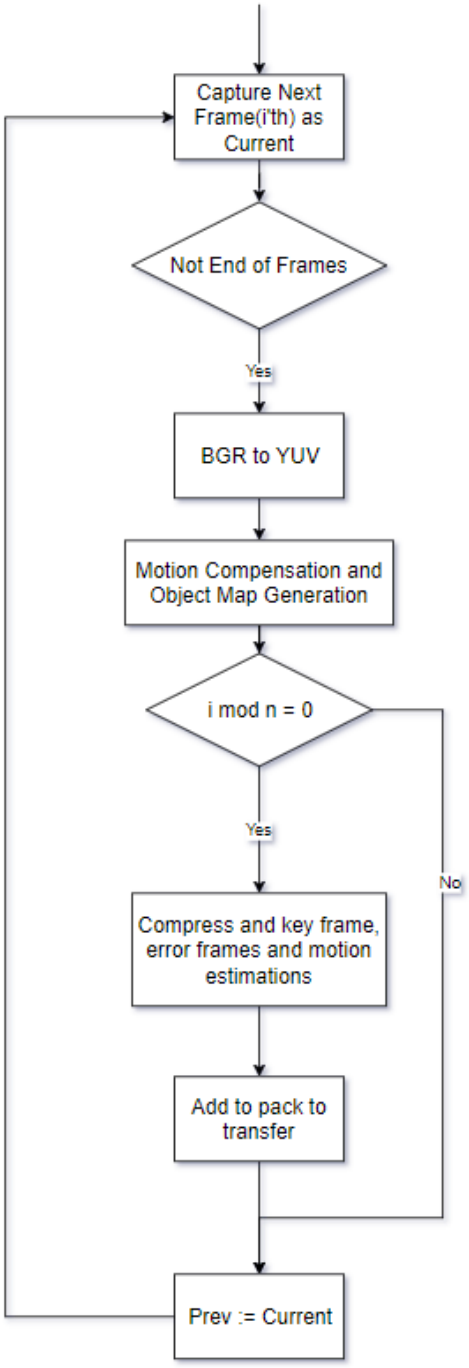

Fig. 6. Flow Chart of the proposed method

frame displacement due to the camera movement. However, typically, we are interested in tracking movement only for some important parts of the video sequence called regions of interest or objects.

For such an application, object based motion compensation techniques allow for to significantly reduce the required number of bits for representation of motion vectors and differences between frames. There are a number of object

based motion compensation methods. Some of them are based on the assumption that objects are extracted using only information of the current frame (for example, it can be done by analysis of luminance and/or color changes), without taking into account object movements. Another object based motion method is based on finding objects by analysis of the difference between the current and the reference frames. If the difference for an area is large enough the corresponding area is called an object. This method has rather low computational complexity but it does not take into account the camera movement. Moreover such technique is not capable to find a plurality of objects.

In our method, we should identify movements between two consecutive frames to estimate motion and identify multiple objects. We can apply motion compensation recursively to find similarities between same level of blocks and combine them to define higher level objects.More precisely, we split the frame into blocks of the fixed size. At each step of the motion compensation procedure we perform global motion compensation for the object (group of blocks or the entire frame) selected at the previous step. For each object the absolute value of the obtained difference between the current object and the shifted object in the previous frame is compared with a predetermined threshold. If absolute value of the difference exceeds the threshold then the corresponding block is classified as a new object. The new block is marked in the object map. When all objects are found the conventional block matching motion compensation is performed to generate relative block motion compensation vectors . While recursively applying motion compensation, we need to parameterize depth of recursion to control level of details.

When we merge blocks to represent an object, motion vector coordinates for each block can be represented as the sum of motion vector coordinates of previous level, additional motion vector coordinates of the current level and motion vector coordinates of current block. In other words, for each of the blocks in the L-th object coordinates X and Y in motion vectors are equal to the following sums

$$X_{Global} + \sum_{i=1}^{L} X_i + X_b \tag{9}$$

$$Y_{Global} + \sum_{i=1}^{L} Y_i + Y_b \tag{10}$$

where L is the object number, X_i and Y_i are the best shifts for the i-th level (object), X_{Global} and Y_{Global} are the best shifts for global motion and X_b and Y_b are the best shifts for the block.

While estimating motion vectors, we make assumptions that our predictor works perfectly. Practically, it predicts frames with potential errors. To eliminate errors, algorithm

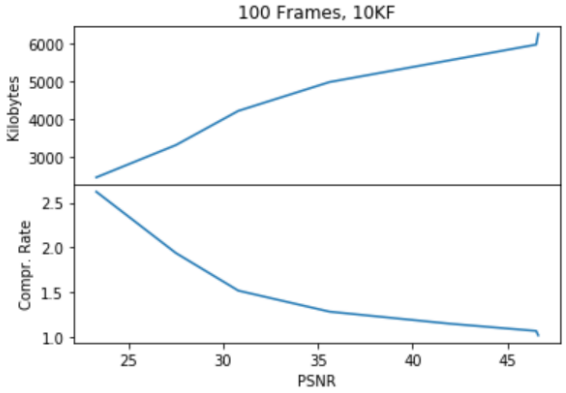

Fig. 7. Size, Compression Rate and PSNR

TABLE I
SIZE COMPARISON

d/d	Size(10 F)	Size(20 F)	Size(50 F)	Size(100 F)
XVID	50K	132K	435K	707K
IYUV	963K	1.9M	4.8M	9.6M
RAW	1.9M	3.8M	9.6M	19.2M
OURS(QR=2)	200K	450K	1.2M	2.4M
OURS(QR=16)	450K	1M	2.2M	4.5M

stores difference of predicted frames and original frames. We can compress the obtained difference frame by using mentioned coding method. Algorithm keeps key frames to eliminate explosion of error rates while reconstructing frames. Period of keeping key frames, can be considered as a hyper parameter for our method. We can compress key frames using frame compression method from section II. For key frames, coder shown in Fig.3 (the same coder as for difference frames) can be used. The quality of the reconstructed key frames should be as high as possible.

Finally, we will handle U and V channels in addition to Y channel. As we mentioned, U and V channel carries chromatic information. Color information is not crucial in most of applications. Due to this assumption, we will use down-sampling and lossy compression techniques on these channels to compress.

IV. EXPERIMENTS

We used python programming language to implement coder. In experiments, we used single core of i7 processor. There were three different videos. We compared our method with XVID, IYUV, RAW. We use python implementation of this standards. In addition, we define Quantization Rate parameter. QR(Quantization Rate) is a divider to make quantization matrix elements smaller. QR divides every element of quantization matrix before quantization step.

As we can see it is not easy to compete with even early industrial standards. Using larger quantization steps gives PSNR values smaller than 30 dB. Using smaller quantization steps gives better PSNR values but leads to larger sizes

of the compressed video file (Fig.7). However, there are a lot of room to improve in the method described here. Omitting non-foreground regions, region of interest selection, better lossy compression with different parameters for U and V channel are some major improvements can be done to improve results.

V. CONCLUSION

Development of computer vision applications gained momentum during last decade. We have a lot of breathtaking inventions about Computer Vision, Machine Learning and AI applications. This development comes with massive amount of processing needs and need for distributed architectures.

In this report, we study on how to define motion oriented compression which is tailored for machine-to-machine communication and computer vision applications. Compression technique we have studied still has room for performance and accuracy improvement. Compression topic is a well studied topic over years by a lot of groups including big industrial companies. However, in our study we try to scratch the surface of techniques which can be developed and improved to become more machine-to-machine friendly compression techniques.

VI. ACKNOWLEDGEMENTS

Special thanks for Irina Bocharova and Coding and Information Transmission group for sharing their valuable time.

REFERENCES

[1] Yano Research Institute,Global Surveillance Camera Market: Key Research Findings 2015,https://www.yanoresearch.com/press/pdf/1420.pdf, 2015

[2] DeepMind, IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures,https://arxiv.org/pdf/1802.01561.pdf, 2018

[3] Irina Bocharova, Compression for Multimedia, Cambridge University Press, 2010

[4] Warwick - Computer Science - Multimedia Coding Group, Block-Matching Motion Compensation, http://www.dcs.warwick.ac.uk/research/mcg/bmmc/index.html, 09.05.2018 - 04:59

[5] Christian Schmaltz, Joachim Weickert, Video Compression with 3-D Pose Tracking, PDE-based Image Coding, and Electrostatic Halftoning,http://www.mia.uni-saarland.de/Publications/schmaltz-dagm12.pdf, 2012

[6] 1 SALONI R. MISTRY, 2HENI S. MODI, 3RAHUL N. GONAWALA, LOGARITHMIC SEARCH FOR MOTION ESTIMATION,International Journal of Industrial Electronics and Electrical Engineering, Volume-2, Issue-6, June-2014