CAME: Cloud-Assisted Motion Estimation for Mobile Video Compression and Transmission

Yuan Zhao, Lei Zhang, Xiaoqiang Ma
Jiangchuan Liu, Hongbo Jiang
Outline

• Background
• Challenges
• Solutions
• Our Approach
• Evaluation
• Conclusions
Background

• Video streaming contributes a dominant fraction of the internet traffic

• Mobile devices as important end devices for internet video apps
  – YouTube mobile: 400M views a day, 13% of total

• Mobile device users can watch or shot high-quality videos anywhere anytime
Challenges

• Limited energy on mobile devices
  – A challenging task to compress high-quality videos

• Expensive bandwidth cost
  – Unaffordable to transmit raw video from both energy and bandwidth perspective
Solutions

• Do nothing, upload raw video: high bandwidth cost and high transmission energy consumption

• Conduct video compression before uploading: high computation cost, or high energy consumption

• Conduct simpler video compression before uploading: not effective enough

• Transfer video to PC, compress and upload: not convenient, discourage user to share video anywhere anytime.
CAME Approach

• Leverage cloud server resources for motion estimation, which is known to be the most computation-intensive step.
• No need to upload the whole video stream to cloud server.
• Derived from mesh-based motion estimation, which is known to be highly effective.
• Mobile devices only selects and uploads the key information
CAME: A Big Picture

Anchor Frame + Predictive Frames → Cloud Server → Motion Vectors

Cloud Server → Compressed Video → Mobile Device
CAME: System and Algorithms

• 1. On mobile device, raw video is divided to macro blocks (MBs). A reference frame and a mesh for each successive P-frame are extracted.
• 2. Mobile device uploads the reference frame and the meshes.
• 3. Cloud server conducts mesh-based motion estimation, generated motion vectors (MVs) are pushed back to mobile devices.
• 4. Mobile device finishes video compression and uploads the compressed video.
Standard Mesh-based Motion Estimation

• Standard mesh motion estimation
  – Mesh nodes selected on reference frame
  – Motion vectors calculated from the single mesh on each successive P-frame

• Mesh structures:
  – Regular: triangular or rectangular
  – Object-based
  – Hierarchy structure
Standard Mesh-based Motion Estimation

\[ x_1, x_2, x_3, x_4 \]

\[ d_1, d_2, d_3, d_4 \]
Reversed Mesh-based Motion Estimation

• Reversed mesh-based motion estimation
  – Mesh nodes selected on each successive P-frame
  – Motion vectors calculated from reference frame and mesh
  – Regular triangular mesh structure

• Reference frame and mesh nodes uploaded
  – Energy and transmission cost reduced

• Mesh motion estimation performed on the cloud server
Algorithm 2 Mobile-Video-Compression

**INPUT:** Complete raw video $V$  
Divide $V$ to MacroBlocks (MBs) denoted by $\{MB_i\}$  
for each $MB_i$ in $\{MB_i\}$ do  
    Calculate $\{R, M_{f,i}\}$, anchor frame $R$ and fraction mesh nodes on $P$  
    Upload $\{R, M_{f,i}\}$ to server for mesh node motion estimation  
end for  
$MV_{video} = \emptyset$  
On receiving $\{MV_{mesh}\}$ from server:  
for each $MV$ in $\{MV_{mesh}\}$ do  
    Local search to calculate sub-block $MV_{sub}$ for current frame  
    Interpolate $MV_{sub}$ into $MV$ to get $MV_f$ for frame  
    $MV_{video} = MV_{video} \cup MV_f$  
    Calculate video compensation for current frame  
end for  
if $MV_{video}$ is complete then  
    Finish video compression process to get $V_{comp}$  
    Anchor frames are already deducted from $V_{comp}$  
    Upload $V_{comp}$  
end if
Algorithm 1 CloudServer-Motion-Estimation

**INPUT:** \( \{R, M_{fi}\} \), anchor frame \( R \) and mesh nodes on \( P \)
\[
MV_s = \emptyset
\]
for each \( M_{fj} \) in \( \{R, M_{fi}\} \) do
  Full search to calculate \( MV \) from \( R \) and \( M_{fj} \)
  \[
  MV_s = MV_s \cup MV
  \]
end for

**OUTPUT:** Push motion vector \( MV_s \) back to the mobile device
Cost analysis

- Total transmission cost for mesh uploading:

\[
C_m = \sum_{i=1}^{m} (C_R + \sum_{i=1}^{n} C_i \cdot f)
\]

where \(m\) is the number of MBs, \(C_R\) is the cost of reference frame, and \(n\) is the number of P-frames in a single MB.
Cost analysis

• Cost of sub-Block motion estimation

\[ C_f = \sum_{j=1}^{n} \sum_{i=1}^{m} C_i \cdot f \]

where \( n \) is number of P-frame in a single MB, \( m \) is sub-block number inside a single P-frame.
Total Cost

\[ C = C_m + C_d + C_f + C_c + C_o + C_u \]

where
- \( C_m \) is the total cost for uploading all MBs
- \( C_d \) is MV downloading cost
- \( C_f \) is the total cost for sub block motion estimation
- \( C_c \) is the motion compensation cost
- \( C_o \) is all other costs for video compression on the mobile device after motion estimation
- \( C_u \) is the final step mobile video uploading cost
Evaluation

• Total cost is evaluated as equivalent CPU cycles
  – Video compression computation cost is recorded directly as CPU cycles
  – Transmission cost is converted to equivalent CPU cycles based on [balasubramanian, SIGCOMM 2009] and [yuan, TOCS 2006]

• Total cost is evaluated on a simulated system
  – More generic, testing result not specific to individual mobile devices
Total Energy Consumption

- **Foreman**
  - All on Mobile: 14
  - CAME: 10

- **Mother**
  - All on Mobile: 12
  - CAME: 10

- **Flower**
  - All on Mobile: 8
  - CAME: 6
Total Transmission

- Foreman
- Mother
- Flower

- All on Mobile
- Raw Uploading
- CAME
Conclusions

• The proposed approach can significantly reduce the complexity of video compression on mobile devices

• Better balance between energy and bandwidth consumption

• One limitation is the closed loop design of the system which may introduce unexpected delays.
Thanks

Q & A