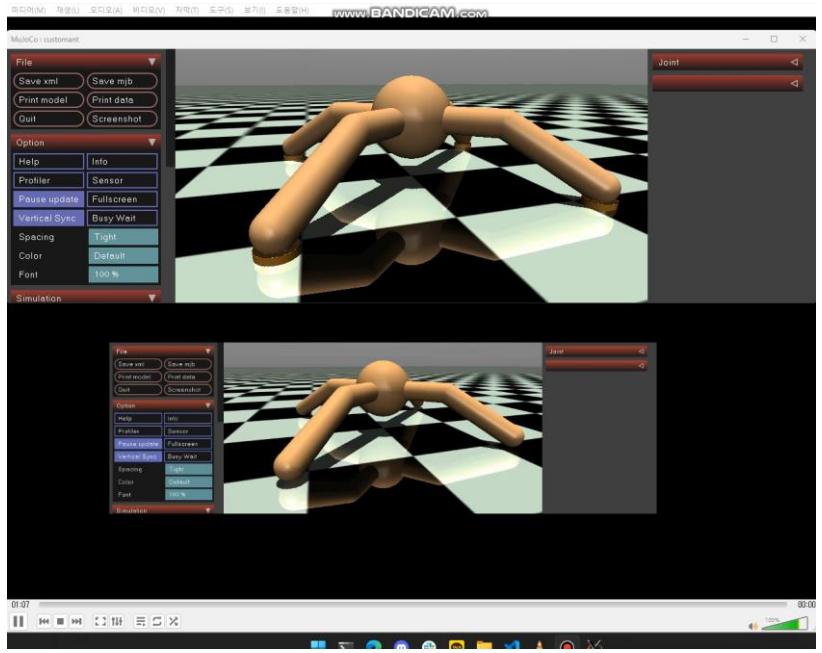
2024/07/25 Research Report

RTCL Lab Meeting
Presenter: Chaehoon Park

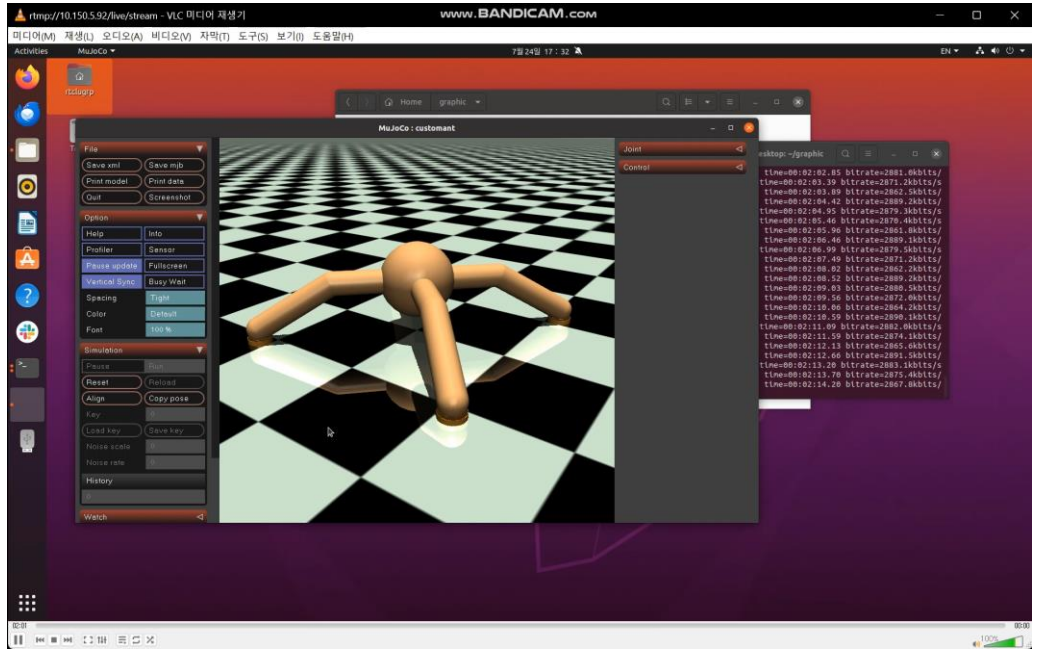
Server-Client Communication

Baseline Demo



Local to Local

5 sec delay



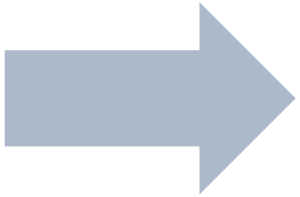
Remote to Local

8 sec delay

Impact of Camera Speed

Motion complexity
(Camera motion &
Peripheral Object motion)

?



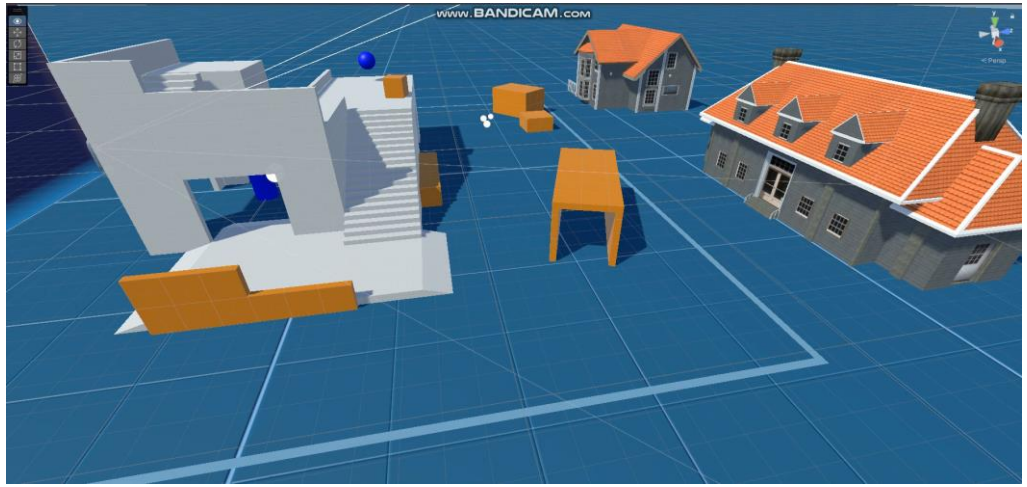
Decoding time

Impact of Camera Speed

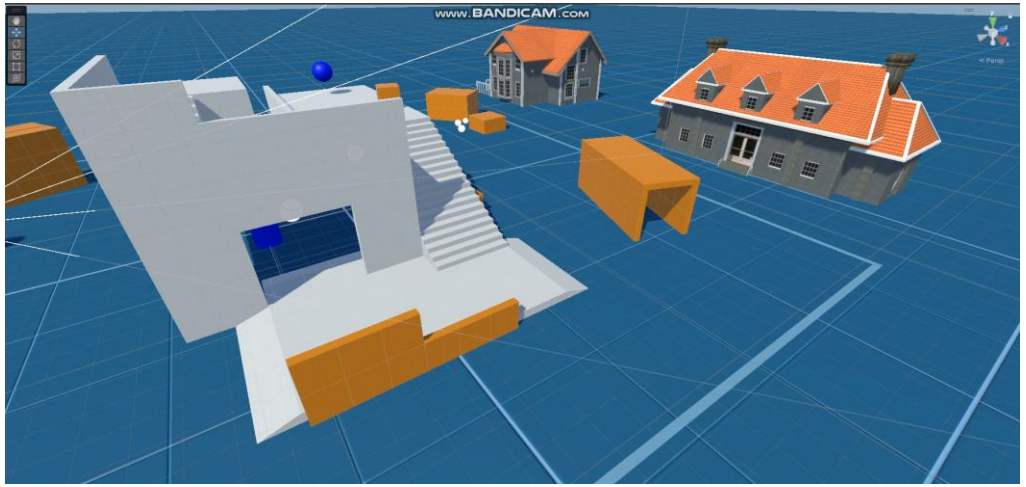
- Simple Experiment

Video Recording with different camera speed

both 10 sec length video



Slow Camera Speed



Fast Camera Speed

Impact of Camera Speed

- Simple Experiment

Measure of Total Decoding Time

Each repeated 10 times

```
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/slow.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=2.465s stime=0.153s rtime=0.485s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/slow.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=2.459s stime=0.112s rtime=0.497s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/slow.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=2.496s stime=0.213s rtime=0.526s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/slow.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=2.483s stime=0.163s rtime=0.490s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/slow.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=2.466s stime=0.170s rtime=0.506s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/slow.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=2.597s stime=0.151s rtime=0.515s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/slow.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=2.577s stime=0.103s rtime=0.494s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/slow.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=2.658s stime=0.161s rtime=0.521s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/slow.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=2.780s stime=0.158s rtime=0.547s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/slow.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=2.450s stime=0.155s rtime=0.497s
```

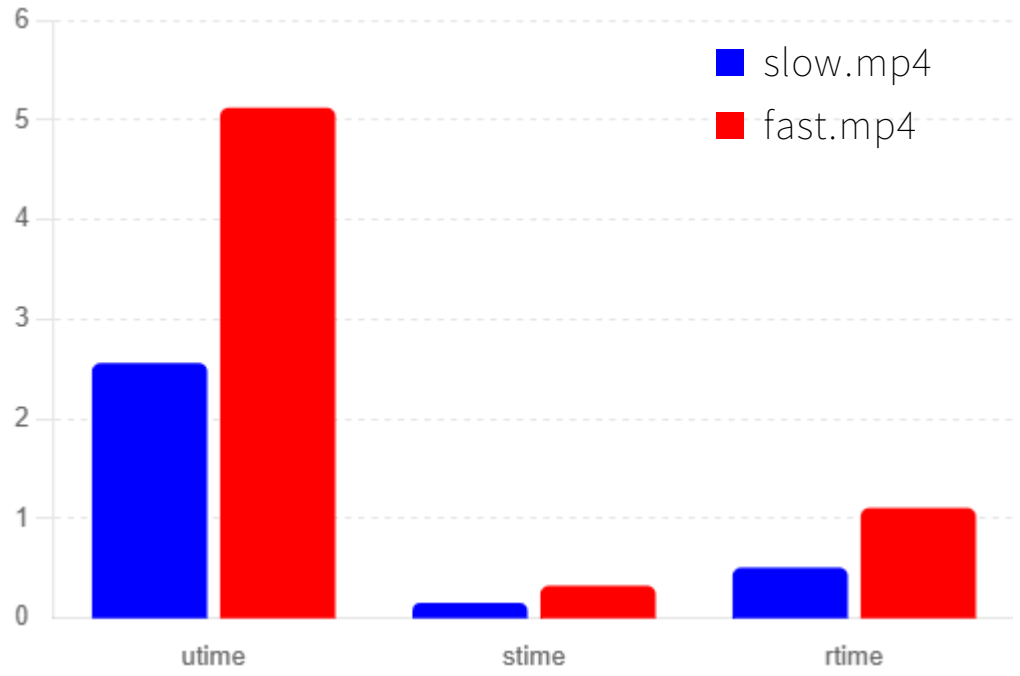
Slow Camera Speed

```
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/fast.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=3.268s stime=0.323s rtime=0.733s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/fast.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=3.426s stime=0.144s rtime=0.745s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/fast.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=3.549s stime=0.258s rtime=0.781s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/fast.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=4.794s stime=0.342s rtime=1.048s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/fast.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=7.827s stime=0.513s rtime=1.678s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/fast.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=5.454s stime=0.353s rtime=1.202s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/fast.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=6.041s stime=0.392s rtime=1.291s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/fast.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=5.710s stime=0.318s rtime=1.219s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/fast.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=5.129s stime=0.348s rtime=1.109s
chek@ChekPC:~/ffmpeg$ ffmpeg -i src/fast.mp4 -f null -benchmark - 2>&1 | grep 'bench: utime='
bench: utime=6.045s stime=0.293s rtime=1.281s
```

Fast Camera Speed

Impact of Camera Speed

- Simple Experiment



- Average Decoding Times for "slow.mp4"
  - utime: 2.5626 seconds
  - stime: 0.1558 seconds
  - rtime: 0.5102 seconds

- Average Decoding Times for "fast.mp4"
  - utime: 5.1243 seconds
  - stime: 0.3284 seconds
  - rtime: 1.1387 seconds

Multithreading → user and system times can accumulate faster than real time

Impact of Camera Speed

If camera speed is fast...

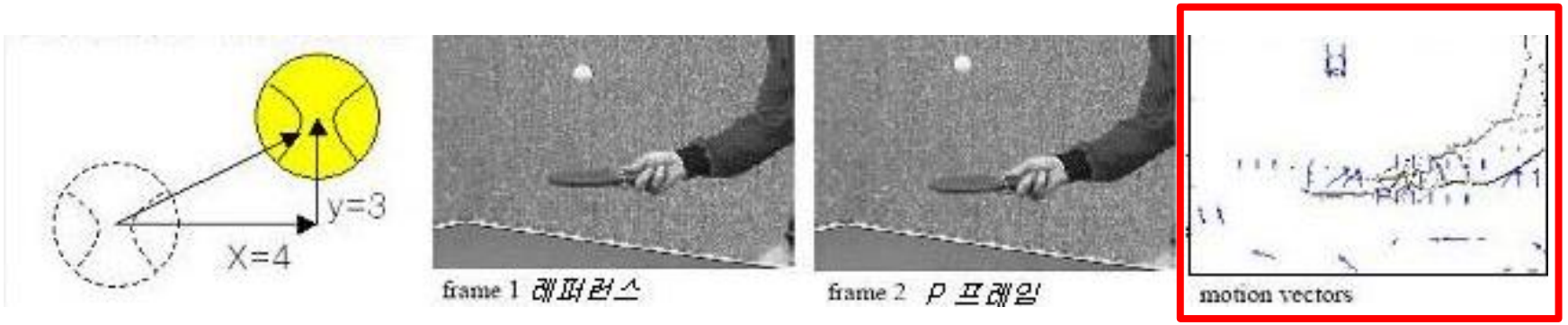
- Increased Motion Vectors: When the camera moves quickly, the objects within the video frame move more between consecutive frames. This leads to a greater number of motion vectors that the codec must calculate and encode.
- Fast motion can introduce motion blur or artifacts that require additional processing to correct during decoding.

In Perspective of Temporal and Spatial Redundancy:

- Temporal Redundancy: With faster movements, exploiting temporal redundancy (similarities between consecutive frames) becomes challenging, leading to less efficient compression and longer decoding times.
- Spatial Redundancy: High-speed movements can reduce spatial redundancy (similarities within a frame), requiring more detailed encoding and decoding efforts.

Impact of Camera Speed

Motion Vector



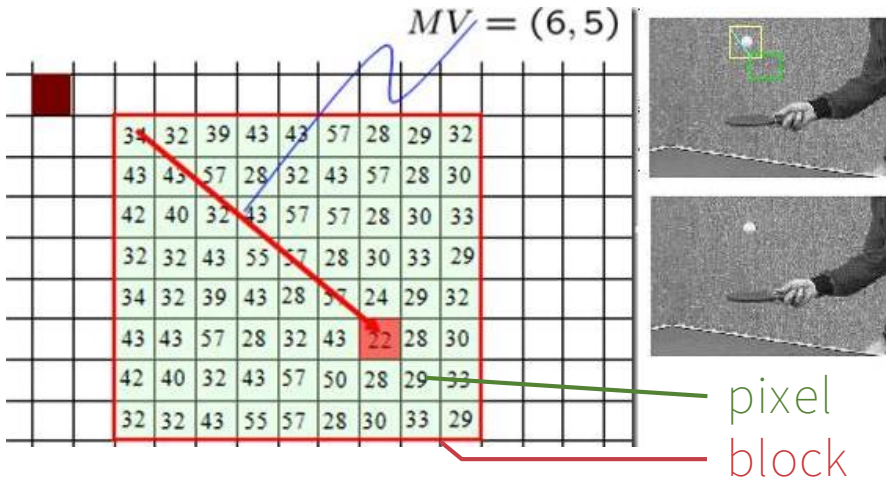
Values that indicate direction and magnitude of motion in a video
Fundamental in compression (e.g., MPEG, H.264)

When comparing successive frames, most of the background remains static while only moving objects change. By storing only the changes between frames, significant compression can be achieved.

Impact of Camera Speed

Motion Vector

Block-based Motion Estimation: divide each frame into blocks and search for the corresponding part of the previous frame within these blocks.

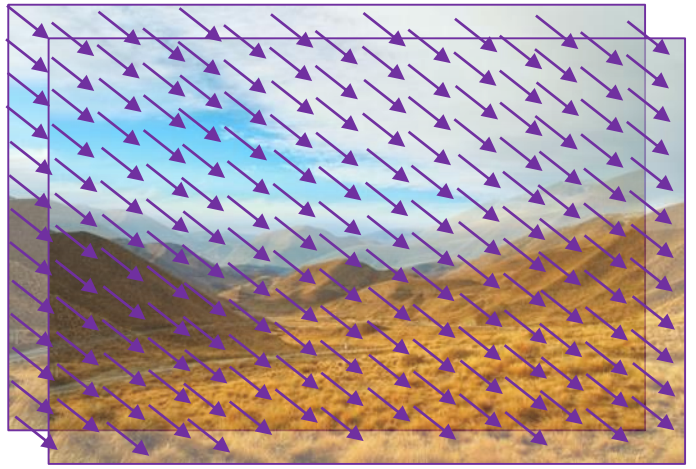


Find Minimum $SAD(i,j) = \sum_{x=0}^{x=4} \sum_{x=0}^{x=4} |B_t(x,y) - B_{t-1}(x,y)|$ in a block

What I am implementing...

- Global Motion Vector

Frame N



Frame N+1

Camera move makes most pixels to move with same motion vector

→ Set “ Global Motion Vector” which matches with camera movement, and only leave motion vectors that are not same with global motion vector

→ Even more compression → Faster Decoding

What I am implementing...

- Global Motion Vector

motion_est.c conducts motion estimation in FFMPEG
This is the function that calculates SAD value

```
libavcodec > C motion_est.c > sad_hpel_motion_search(MpegEncContext *, int *, int *, int, int, int, int, int)
408  static int sad_hpel_motion_search(MpegEncContext * s,
411                                    int size, int h)
412  {
413      MotionEstContext * const c= &s->me;
414      const int penalty_factor= c->sub_penalty_factor;
415      int mx, my, dminh;
416      const uint8_t *pix, *ptr;
417      int stride= c->stride;
508      *mx_ptr = mx;
509      *my_ptr = my;
510      return dminh;
511  }
```

Added compress_motion_vectors function to the code

```
void compress_motion_vectors(MpegEncContext *s, AVMotionVector global_motion, MotionVectorTable *table) {
    MotionEstContext *c = &s->me;
    int width = s->width;
    int height = s->height;

    for (int y = 0; y < height; y += 16) {
        for (int x = 0; x < width; x += 16) {
            int mb_index = (y / 16) * s->mb_stride + (x / 16);
            int16_t (*mv_table)[2] = s->cur_pic.motion_val[0];

            int mv_x = mv_table[mb_index][0];
            int mv_y = mv_table[mb_index][1];

            if (mv_x != global_motion.motion_x || mv_y != global_motion.motion_y) {
                // Store the motion vector in the hash table
                insert_motion_vector(table, x, y, mv_x, mv_y);
            }
        }
    }
}
```

End of Presentation