Perancangan Codec Berbasis Algoritma Kompresi H.264 untuk Aplikasi Konferensi Video

Fairuz Azmi[1], Budhi Irawan[2], Gelar Budiman[3]
Gedung Barung Ruang TE1.02.08, Universitas Telkom, Jl. Telekomunikasi No.1, Bandung 40257
[1,2,3] Program Studi Sistem Komputer, Fakultas Teknik Elektro – Universitas Telkom, Bandung
[1]**worldliner@telkomuniversity.ac.id**, [2]budhiirawan@telkomuniversity.ac.id,
[3]gelarbudiman@telkomuniversity.ac.id

Abstrak

Dewasa ini, banyak aplikasi yang dibuat dengan melibatkan fitur multimedia. Salah satu aplikasi tersebut adalah aplikasi konferensi, yaitu komunikasi yang dilakukan oleh suatu group yang terhubung pada suatu jaringan dengan setiap anggota group dapat saling melihat anggota yang lain melalui fasilitas video yang direkam menggunakan sebuah piranti webcam. Dalam komunikasi video, dibutuhkan bandwidth yang cukup besar karena ukuran data video yang jauh lebih besar dibandingkan data suara. Untuk menyiasati hal tersebut, maka dilakukan kompresi pada video sebelum ditransmisikan. Dengan adanya kompresi, maka ukuran data video yang akan ditransmisikan menjadi lebih kecil sehingga dapat menghemat bandwidth saat transmisi data berlangsung. Dalam penelitian ini dirancang codec yang mengadopsi standar kompresi video H.264. Selanjutnya dianalisis kualitas video hasil kompresi meliputi : rasio kompresi, Peak Signal-to-Noise Ratio (PSNR), dan Mean Opinion Score (MOS). Selain itu, juga dianalisis pengaruh kompresi video terhadap kinerja aplikasi konferensi video. Dari hasil penelitian yang dilakukan, didapatkan hasil bahwa dengan adanya kompresi H.264 menghasilkan bitrate video yang rendah dengan penurunan mencapai 98.7% (1:77) hingga 99.52% (1:208) dengan PSNR minimal 30,90 dB.

Kata kunci— *kompresi video, H.264, konferensi video*

Abstract

Nowadays, many applications which use multimedia features. One of the applications is video conferencing application, which enable a group of people to communicate by viewing each other using a webcam as capture device. On video communication, it need a large bandwidth to transmit the video data. To prevent a huge bandwidth usage due to large data size of video, the video data need to be compressed prior to be transmitted. By using compression, video data will have a smaller size of bitstream which give a small cost of network bandwidth during communication session. In this research, was designed a codec which adopts H.264 compression standard. Further analysis includes the compressed video quality: compression ratio, Peak Signal-to-Noise Ratio (PSNR), and Mean Opinion Score (MOS). In addition, it also analyzed the effect of video compression to the application performance. By the result of the research showed that use of H.264 compression can produce lower video bitrate with the reduction reached 98.7% (1:77) up to 99.52% (1:208) with minimum PSNR is 30,90 dB.

Keywords— *video compression, H.264, video conference*

1. PENDAHULUAN

Dewasa ini, banyak aplikasi yang dibuat dengan melibatkan fitur multimedia. Salah satu aplikasi tersebut adalah aplikasi konferensi video (*video conference*), yaitu komunikasi yang dilakukan oleh suatu group yang terhubung pada suatu jaringan dengan setiap anggota group dapat saling melihat anggota yang lain melalui fasilitas video yang direkam menggunakan sebuah *capture device*.

Pada komunikasi video, dibutuhkan *bandwidth* yang cukup besar karena ukuran data video yang jauh lebih besar dibandingkan data suara[1]. Hal ini dikarenakan data video yang terdiri dari beberapa buah gambar (*frame*) yang ditampilkan secara sekuensial dalam satu waktu sehingga gambar terlihat bergerak. Untuk menyiasati ukuran data video yang cukup besar, maka dilakukan kompresi pada data video yang akan ditransmisikan. Dengan adanya kompresi, ukuran data video yang akan ditransmisikan menjadi lebih kecil sehingga dapat menghemat *bandwidth* jaringan yang digunakan.

Saat ini, banyak algoritma yang dapat digunakan untuk melakukan kompresi video. ITU-T (International Telecommunications Union - Telecommunication Standardisation Sector) melakukan standarisasi beberapa buah codec yang cocok digunakan untuk komunikasi video, antara lain : H.261, H.263, dan H.264. Algoritma yang saat ini populer digunakan adalah H.264[2,3]. Dalam penelitian ini, dirancang sebuah *codec* yang mengimplementasikan algoritma H.264 untuk sebuah aplikasi konferensi video.

2. TINJAUAN PUSTAKA

2.1. Kompresi Video H.264

H.264 adalah suatu metode untuk melakukan kompresi terhadap video digital menjadi suatu format video dengan kapasitas yang lebih kecil untuk disimpan atau ditransmisikan[4]. Standar H.264/AVC (*Advance Video Coding*) pertama kali diterbitkan pada tahun 2003, dengan beberapa revisi yang kemudian diterbitkan sejak saat itu. Standar tersebut dibangun berdasarkan pada konsep standar sebelumnya seperti MPEG-2 dan Visual MPEG-4 dan menawarkan potensi untuk efisiensi kompresi yang lebih baik, yaitu kompresi video yang lebih berkualitas, dan fleksibilitas lebih besar dalam kompresi, transmisi dan penyimpanan video [3,5].

H.264 terdiri dari bagian *video encoder* dan *video decoder*. *Video encoder* H.264 melakukan prediksi, transformasi dan proses pengkodean untuk menghasilkan *bitstream* H.264 terkompresi. *Video decoder* H.264 melaksanakan proses decoding, invers transformasi dan rekonstruksi untuk menghasilkan urutan video. Urutan frame video asli dikodekan ke dalam format H.264 menjadi serangkaian bit yang merupakan video dalam bentuk terkompresi. *Bitstream* yang dikompres ini kemudian disimpan atau ditransmisikan dan dapat diterjemahkan untuk merekonstruksi kembali urutan video. Hasil dekode pada umumnya tidak identik dengan urutan asli karena H.264 bersifat kompresi *lossy*, yaitu terdapat beberapa kualitas gambar yang hilang selama kompresi[4].

2.2. Konferensi Video

Konferensi video adalah seperangkat teknologi telekomunikasi interaktif yang memungkinkan dua pihak atau lebih di lokasi berbeda dapat berinteraksi melalui pengiriman dua arah audio dan video secara bersamaan. Konsep dasar konferensi video sama seperti percakapan sederhana antara dua orang atau lebih, tetapi dilakukan pada lokasi yang berbeda antara para pembicara yang terlibat. Teknologi inti yang digunakan dalam konferensi video adalah sistem kompresi digital audio dan video *stream* yang memungkinkan pengiriman data audio dan video dengan *bandwidth* yang kecil[1].

3. METODE PENELITIAN

Dalam konferensi video, terdapat proses transmisi data video antar pengguna yang terhubung dalam konferensi. Untuk itu, perlu adanya alat yang dapat digunakan untuk merekam video serta alat untuk menampilkan video tersebut sehingga dapat dilihat oleh setiap pengguna yang terhubung dalam konferensi.

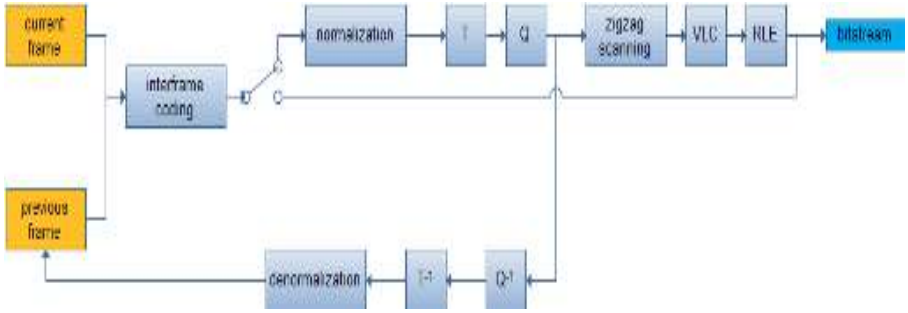
Konsep dasar transmisi video ialah proses *capturing* video menggunakan *capture device*, misalnya *webcam*. Selanjutnya data video dikompresi menggunakan *H.264 encoder*, setelah itu data dikirimkan ke jaringan. Pada bagian penerima, data video didekompresi menggunakan *H.264 decoder*. Kemudian video tersebut ditampilkan pada layar (display) komputer penerima[4]. Dalam sistem yang dirancang, *codec* yang digunakan dirancang dengan mengadopsi standar H.264/AVC tanpa menggunakan subsistem *intraframe coding* untuk menghasilkan *codec* yang membutuhkan bitrate

rendah dengan kualitas video yang masih dapat ditoleransi pengguna sesuai penggunaannya untuk aplikasi konferensi video.

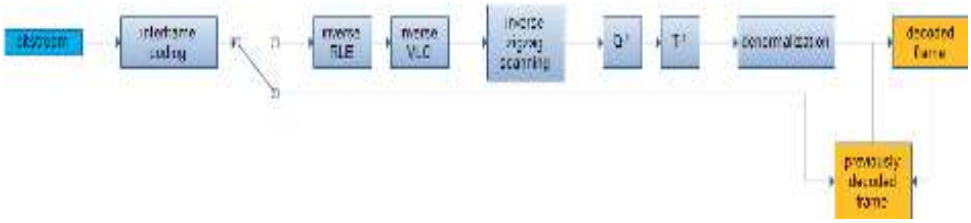
Gambar 1 Pemodelan Transmisi Video

3.1. Arsitektur Video Codec

Video codec yang dirancang adalah sebuah set *encoder* dan *decoder* yang mengadopsi proses algoritma H.264 dengan beberapa modifikasi pada subsistemnya. *Video codec* dirancang menggunakan bahasa C# (*C sharp*). Sebelum memasuki proses kompresi, *frame* yang berupa citra ARGB (*Alpha Red Green Blue*) diubah menjadi format YCbCr (*Luminance Chrominance blue/red*). Hal ini dilakukan karena mata manusia lebih peka terhadap komponen kecerahan (*luma*) dibandingkan komponen warna (*chroma*), sehingga nantinya dapat dimanfaatkan dalam proses kompresi jika menggunakan format ini. Selanjutnya, frame dibagi menjadi *macroblock-macroblock* yang masing-masing berukuran 4x4 *pixel*. Skema proses dari *video codec* ditunjukkan dalam Gambar 2 dan Gambar 3.



Gambar 2 Skema *Encoder*



Gambar 3 Skema *Decoder*

3.1.1. Interframe Coding

Proses *interframe coding* menentukan apakah suatu *macroblock* akan melalui proses H.264 atau tidak dengan memanfaatkan redundansi temporal antara *frame* saat ini dengan *frame* sebelumnya. Pengambilan keputusan dilakukan berdasarkan nilai *Mean Absolute Error* (MAE) antara *macroblock* pada *frame* saat ini dengan *macroblock* pada posisi yang sama pada *frame* sebelumnya. Pada sistem ini, jika MAE bernilai kurang dari nilai *threshold* yang ditetapkan, maka *macroblock* tersebut tidak akan melalui proses H.264 dan akan langsung dikodekan menjadi x00. Selanjutnya, pada bagian *decoder*, cukup diperiksa kode yang diterima. Jika kode yang diterima adalah x00, maka keluaran *decoder* adalah salinan dari macroblock pada frame yang didekodekan sebelumnya, sehingga tidak perlu melalui proses *decoding*.

3.1.2. Normalisasi

Proses normalisasi dilakukan untuk mengubah jangkauan dari nilai-nilai pada tiap *pixel* dalam suatu *macroblock*. Pada proses *encoding*, dilakukan pengurangan jangkauan yang semula dari 0 sampai

255 menjadi 0 sampai 127. Tujuan pengurangan jangkauan ini adalah supaya masukan dari proses transformasi memiliki nilai yang lebih kecil sehingga dapat menghasilkan keluaran yang diharapkan untuk proses selanjutnya. Proses normalisasi hanya dilakukan pada *luma macroblock*, sedangkan pada *chroma macroblock*, setiap nilai pada *macroblock* dikurangi 128.

3.1.3. Transformasi dan Kuantisasi

Proses transformasi pada H.264 adalah berbasis *discrete cosine transform* (DCT). Perbedaan utama antara transformasi pada H.264 dibandingkan dengan DCT yang asli adalah pada penggunaan *integer transform* yang menyebabkan proses perkalian matriks menjadi lebih cepat pada H.264 [6]. Selain itu, pada H.264 transformasi dilakukan pada *macroblock* 4x4, berbeda dengan *codec-codec* sebelumnya yang melakukan transformasi pada *macroblock* 8x8 [6].

Dengan X merupakan *macroblock* yang akan ditransformasi, dan Y merupakan *macroblock* hasil transformasi, proses transformasi dapat dilihat di Persamaan 1. Sedangkan proses invers transformasi dapat dilihat di Persamaan 2.

$$Y = (C \cdot X \cdot C_T) \otimes E \quad (1)$$

$$X = C_T (Y \otimes E) C \quad (2)$$

Dimana C dan E adalah matriks koefisien transformasi dengan nilai:

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & 1 \end{bmatrix}$$

$$E = \begin{bmatrix} 0.2500 & 0.1581 & 0.2500 & 0.1581 \\ 0.1581 & 0.1000 & 0.1581 & 0.1000 \\ 0.2500 & 0.1581 & 0.2500 & 0.1581 \\ 0.1581 & 0.1000 & 0.1581 & 0.1000 \end{bmatrix}$$

Pada keluaran proses transformasi, nilai pada posisi (0,0) merupakan koefisien pada frekuensi terendah dari matriks masukan. Koefisien ini dinamakan komponen DC, yang paling menentukan dari blok karena merupakan nilai rata-rata dari blok. Nilai-nilai lainnya disebut komponen AC, yang menerangkan jumlah daya spektral yang terdapat pada masing-masing frekuensi spasial.

Setelah dilakukan proses transformasi, selanjutnya data hasil transformasi dikuantisasi untuk memotong nilai-nilai hasil transformasi sehingga saat dikodekan akan menghasilkan kompresi yang optimal. Prinsip dasar dari kuantisasi adalah membagi nilai-nilai pada matriks hasil transformasi dengan suatu nilai Q_{step} yang menyatakan ukuran *step* dari *quantizer*[6]. Semakin besar nilai Q_{step}, maka akan semakin kecil ukuran data yang dikompres. Proses kuantisasi dapat dilihat di Persamaan 3.

$$Z = round\left(\frac{1}{Q_{step}} Y \cdot E\right) \quad (3)$$

Nilai Q_{step} diperoleh dari nilai *Quantization Performance* (QP) yang dapat dilihat pada *lookup table* yang disediakan H.264. Karena mata manusia lebih peka terhadap perubahan kecerahan dibanding perubahan warna, maka pada bagian *chroma* nilai QP yang diberikan dapat lebih besar dari yang diberikan untuk bagian *luma*. Pada sistem ini, QP yang diberikan untuk bagian *chroma* adalah QP+6 jika dibandingkan dengan yang diberikan untuk *luma*.

3.1.4. Variable Length Coding

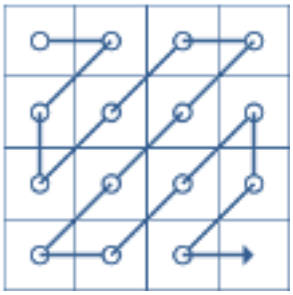
Variable Length Coding (VLC) adalah teknik pengkodean *loseless* yang digunakan sebagai pengkodean entropi pada H.264. Pada bagian inilah, proses kompresi terjadi. VLC melakukan kompresi dengan memanfaatkan keluaran *quantizer* yang memiliki sifat-sifat sbb[6]:

1. Setelah proses normalisasi, transformasi dan kuantisasi, *macroblock* umumnya menghasilkan nilai-nilai yang sebagian besar bernilai nol.
2. Koefisien *non-zero* tertinggi setelah dilakukan *zigzag scanning* sering bernilai ±1 atau biasa disebut *trailing ones*.
3. Koefisien *non-zero* yang bernilai lebih besar umumnya hanya terletak di bagian kiri setelah dilakukan *zigzag scanning* (pada bagian DC), kemudian nilai akan terus mengecil pada komponen AC.

Dari sifat-sifat tersebut, maka dilakukan proses VLC dengan langkah-langkah sbb:

1. Inisialisasi kode berdasarkan jumlah koefisien *non-zero*.
2. Pengkodean *trailing ones*.
3. Pengkodean sisa koefisien *non-zero*.
4. Pengkodean jumlah nilai nol sebelum koefisien terakhir.
5. Pengkodean *run-zero*.

Sebelum dilakukan proses VLC, *macroblock* yang semula berbentuk dua dimensi diubah menjadi satu dimensi menggunakan metode *zigzag scanning*. Metode *zigzag scanning* melakukan pengurutan dari koefisien pada frekuensi terendah (DC) hingga frekuensi tertinggi (AC). Proses *zigzag scanning* ditunjukkan di Gambar 4.



Gambar 4 ***Zigzag scanning***

Setelah dilakukan *zigzag scanning*, dilakukan pengkodean yang menunjukkan total dari koefisien *trailing ones* dan koefisien *non-zero* (selain *trailing ones*). Karena *macroblock* yang digunakan berukuran 4x4, koefisien *non-zero* dapat bernilai dari 0 (tidak ada koefisien *non-zero* dalam *macroblock*) hingga 16 (semua koefisien adalah *non-zero*). Sedangkan *trailing ones* dapat bernilai dari 0 hingga 3. Jika lebih dari tiga *trailing ones* dalam *macroblock*, maka hanya tiga yang dikodekan sebagai *trailing ones*, sisanya dikodekan sebagai koefisien *non-zero*. Hal ini dilakukan untuk efisiensi panjang kode inisial sehingga dapat menghasilkan kode yang efisien. Kode inisial yang dihasilkan didapat dari *VLC lookup table* yang sesuai dengan jumlah *trailing ones* dan koefisien *non-zero*.

Selanjutnya, untuk setiap koefisien *trailing ones*, tanda (*sign*) dari koefisien dikodekan dengan satu bit, yaitu "0" jika koefisien bernilai positif, dan "1" jika koefisien bernilai negatif. Pengkodean dimulai dari koefisien dengan frekuensi tertinggi (sebelah kanan). Sedangkan koefisien *non-zero* selain *trailing ones* dikodekan menggunakan *Exponential Golomb Coding* (EGC) yang dimulai dari koefisien dengan frekuensi tertinggi. EGC merupakan teknik pengkodean *integer* yang akan menghasilkan kode dengan panjang yang minimum untuk bilangan yang kecil, dan akan menghasilkan kode yang lebih panjang jika bilangan yang dikodekan bernilai lebih besar. Secara umum, EGC mengodekan bilangan dengan format yang ditunjukkan dalam Persamaan 6. Untuk mengodekan bilangan bertanda, maka digunakan format yang ditunjukkan di Persamaan 7.

$$M = \lfloor \log_2 (num + 1) \rfloor \quad (4)$$

$$INFO = num + 1 - 2^M \quad (5)$$

$$kode = [M_{zeros}][1][INFO] \quad (6)$$

$$kode = \begin{cases} [M_{zeros}][1][INFO], num > 0 \\ [M_{zeros}][0][INFO], num < 0 \end{cases} \quad (7)$$

Setelah dilakukan *zigzag scanning* pada keluaran *quantizer*, idealnya matriks akan berisi koefisien-koefisien yang terurut mengecil dari koefisien tertinggi pada komponen DC hingga urutan nilai nol pada komponen AC. Akan tetapi, ada kalanya di antara koefisien-koefisien *non-zero* juga akan terdapat koefisien bernilai nol, sehingga harus ikut dikodekan. Koefisien tersebut dikodekan berdasarkan *VLC lookup table* sesuai dengan jumlah koefisien bernilai nol yang ada pada *macroblock*. Setelah menghitung jumlah koefisien nol di antara koefisien-koefisien *non-zero*, selanjutnya adalah menentukan kode yang menunjukkan posisi dari koefisien-koefisien nol tersebut di antara koefisien-koefisien *non-zero*.

Pada bahasa pemrograman yang digunakan, yaitu Visual C#, sebuah variabel dapat menampung data dengan ukuran minimum 1 byte (8 bit). Karena keterbatasan tersebut, hasil pengkodean VLC harus berjumlah sebanyak kelipatan delapan. Jika kurang, maka harus ditambahkan *padding bit* berupa bit "0" sehingga dapat disimpan pada variabel bertipe byte.

3.1.5. Run Length Encoding

Run Length Encoding (RLE) adalah teknik pengkodean *loseless* yang sederhana yang bekerja dengan memanfaatkan urutan data yang sama dalam suatu kelompok data[7]. Dengan RLE, sebuah kumpulan data yang sama cukup disimpan dalam sebuah nilai data tersebut dan sebuah nilai yang menyatakan jumlah data tersebut. Dalam sistem yang dirancang, RLE digunakan untuk mengompres urutan byte keluaran *encoder* yang sebagian besar terdiri dari urutan 0 (nol) karena pengaruh *interframe coding*.

3.1.6. Proses Transmisi

Proses transmisi pada aplikasi konferensi video yang dirancang memanfaatkan protokol transpor UDP, yaitu sebuah protokol transpor yang lebih mengedepankan efisiensi waktu pengiriman paket dibanding reliabilitas. Hal ini dikarenakan pada proses *streaming* data video tidak menghendaki adanya retransmisi yang akan berakibat pada penambahan waktu tunda serta menjadi sulit dimengerti pesan yang disampaikan dalam data tersebut. Untuk pengalamatan, aplikasi ini menggunakan pengalamatan IPv4.

4. HASIL DAN PEMBAHASAN

Hasil dari penelitian ini berupa aplikasi konferensi video yang menerapkan algoritma kompresi berbasis *H.264 viudeo coding*. Aplikasi terdiri dari dua bagian, yaitu bagian *server* (*host*) dan bagian *client*. Antarmuka aplikasi dapat dilihat di Gambar 5.

Gambar 5 Antarmuka Aplikasi

Dengan menggunakan aplikasi yang telah dirancang, selanjutnya diuji parameter kualitas hasil kompresi. Parameter kualitas yang diukur meliputi rasio kompresi, *Peak Signal-to-Noise Ratio* (PSNR), dan *Mean Opinion Score* (MOS).

4.1 Rasio Kompresi

Rasio kompresi merupakan perbandingan ukuran antara data hasil kompresi dengan data sebelum dilakukan kompresi. Semakin kecil nilai rasio kompresi menunjukkan bahwa kemampuan codec untuk mengompres data semakin baik. Berdasarkan nilai QP yang diberikan, hasil pengukuran rasio kompresi dapat dilihat di Tabel 1.

Tabel 1 Rasio Kompresi berdasarkan Nilai QP

QP	Rasio Kompresi
4	2,00 %
10	1,09 %
16	0,68 %
22	0,53 %

4.2 Peak Signal-to-Noise Ratio

Peak Signal-to-Noise Ratio (PSNR) menyatakan perbandingan antara kemungkinan terbesar dari daya sebuah sinyal terhadap *noise* yang mempengaruhi sinyal tersebut[10]. Semakin tinggi nilai PSNR menunjukkan bahwa kualitas data setelah dilakukan kompresi semakin baik. Hasil pengukuran PSNR dapat dilihat di Tabel 2.

Tabel 2 PSNR berdasarkan Nilai QP

QP	PSNR	PSNR (dB)
4	2865,80	34,57
10	2246,70	33,52
16	1846,30	32,66
22	1231,00	30,90

Tabel 2 menunjukkan nilai PSNR yang semakin menurun seiring penambahan nilai QP yang berarti kualitas video juga semakin buruk. Penurunan nilai PSNR tidak terlalu signifikan, yaitu 3,04% saat QP ditingkatkan dari 4 menjadi 10, 2,57% saat QP ditingkatkan lagi menjadi 16, dan 5,39% saat QP ditingkatkan lagi menjadi 22. Pada data tersebut, kualitas video masih memenuhi standar yang ditetapkan.

4.3 Mean Opinion Score

Mean Opinion Score (MOS) adalah nilai rata-rata yang diambil dari pendapat orang-orang yang diminta untuk menilai kualitas data setelah dilakukan kompresi berdasarkan penglihatan mereka. Tujuan pengukuran MOS adalah untuk mengetahui tingkat kepuasan dari orang-orang berdasarkan apa yang mereka lihat sehingga dapat menentukan apakah *codec* yang dirancang dapat diterima oleh pengguna. Pengukuran dilakukan dengan memberikan penilaian dari 1 (sangat buruk) hingga 5 (sangat baik) dari video yang diuji oleh tiga puluh orang responden. Hasil pengukuran dapat dilihat di Tabel 3.

Tabel 3 *Bitrate* berdasarkan Nilai QP

QP	Nilai MOS
4	3,60
10	3,63
16	2,63
22	1,43

4.4 Pengaruh Kompresi terhadap Penggunaan Bandwidth

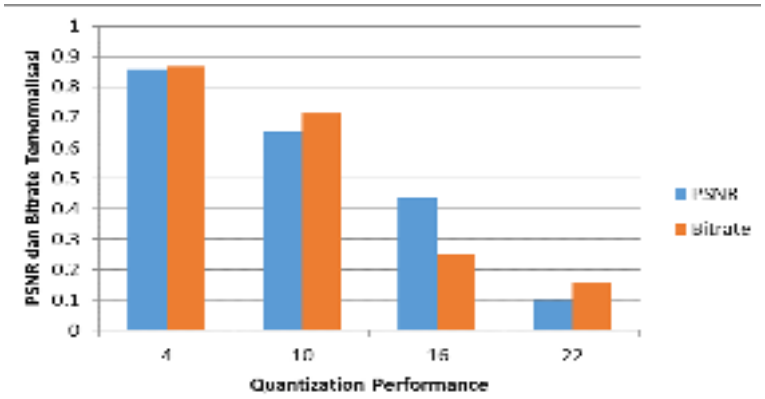
Penggunaan *bandwidth* diukur dengan mengamati jumlah *bitrate* yang ditransmisikan oleh aplikasi. Tujuan pengukuran *bitrate* adalah untuk mengetahui banyaknya data yang dikirimkan ke jaringan komunikasi setiap satuan waktu (dinyatakan dengan bit/s), sehingga dapat diketahui *bandwidth* minimal yang dapat digunakan untuk konferensi video yang menggunakan *codec* yang dirancang. Sebelum dilakukan kompresi, *bitrate* untuk video ARGB berukuran QCIF dengan *framerate* 20 fps adalah 16 Mbit/s. Setelah dilakukan kompresi, *bitrate* berkurang sebagaimana ditunjukkan dalam Tabel 4.

Tabel 4 *Bitrate* berdasarkan nilai QP

QP	*Bitrate* (Kbit/s)
4	207,46
10	173,23
16	97,08
22	76,87

Berdasarkan data pada Tabel 2, *bitrate* yang dihasilkan mengalami penurunan yang cukup signifikan dibandingkan jika tanpa dilakukan proses kompresi, yaitu antara 98,70% hingga 99,52%. Hal ini tentu saja memberikan pengaruh yang cukup besar terhadap penggunaan bandwidth saat transmisi video dilakukan.

Dari hasil pengukuran *bitrate* yang dilakukan, dapat dibandingkan dengan PSNR sehingga diperoleh *trade-off* yang menghasilkan parameter algoritma yang paling optimal. Hasil perbandingan dapat dilihat di Gambar 6. Dari grafik yang ditampilkan pada Gambar 6, performa terbaik didapat saat QP bernilai 16. Performa terbaik dilihat pada penurunan *bitrate* yang cukup drastis sebesar 43.96%, dengan nilai PSNR yang masih memenuhi standar.



Gambar 6 Grafik Penurunan PSNR dan *Bitrate* sesuai Nilai QP yang diberikan

5. KESIMPULAN

Dari pengujian dan analisis yang telah dilakukan pada perancangan aplikasi konferensi video yang dibuat, dapat diambil kesimpulan sebagai berikut:

1. Penurunan kualitas video hasil kompresi setiap kenaikan nilai Quantization Performance (QP) cenderung konstan, yaitu sebesar 3.04% (1.05 dB) saat QP ditingkatkan dari 4 menjadi 10, 2.57% (0.86 dB) saat QP ditingkatkan dari 10 menjadi 16, dan 5.39% (1.76 dB) saat QP ditingkatkan dari 16 menjadi 22.
2. Bitrate yang dihasilkan setelah dilakukan kompresi mengalami penurunan sebesar 98.70% (1:77) hingga 99.52% (1:208).
3. Penurunan bitrate setiap kenaikan nilai QP adalah sebesar 16.5% saat QP ditingkatkan dari 4 menjadi 10, 43.96% saat QP ditingkatkan dari 10 menjadi 16, dan 20.82% saat QP ditingkatkan dari 16 menjadi 22.

4. Berdasarkan pengukuran objektif, kinerja terbaik diperoleh codec saat diberikan nilai QP sebesar 16 dengan bitrate 97,08 kbps dan PSNR 32,66 dB.
5. Berdasarkan pengukuran subjektif, kinerja terbaik diperoleh codec saat diberikan nilai QP sebesar 10 yaitu dengan nilai 3,63 dari skala 5.
6. Berdasarkan pengukuran subjektif, dapat disimpulkan kualitas video masih dapat ditoleransi oleh pengguna saat menggunakan parameter QP bernilai 16, dengan nilai MOS 2,63 dari skala 5.

6. SARAN

Untuk pengembangan penelitian ini, dapat dilakukan dengan penerapan algoritma kompresi menggunakan standar *High Efficient Video Coding* (HEVC) dan menggunakan metode *macroblock prediction* yang lebih kompleks sesuai dengan standar HEVC dari ITU-T, sehingga diharapkan dapat diperoleh *bitrate* yang lebih rendah dengan kualitas yang lebih baik.

DAFTAR PUSTAKA

[1] Sullivan, Gary J., and Wiegand, Thomas, 2005, Video Compression—From Concepts to the H.264/AVC Standard, *Proceedings of the IEEE*, Vol. 93, hal 18-31.

[2] Wiegand, Thomas, Sullivan, Gary J., Bjøntegaard, Gisle, and Luthra, Ajay, 2003, Overview of the H.264/AVC Video Coding Standard, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, hal 560-576.

[3] Hanzo, L., P. Cherriman, and J. Streit, 2007, *Video Compression and Communications*, Wiley, United Kingdom.

[4] Yonata, Yosi, 2002, *Kompresi Video*, Elex Media Komputindo, Jakarta.

[5] Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhammer, T., and Wedi, T., 2004, Video coding with H.264/AVC: Tools, Performance, and Complexity, *IEEE Circuits and Systems Magazine*, Vol. 4, hal 7-28.

[6] Richardson, Iain E. G., 2003, *H.264 and MPEG-4 Video Compression*, Wiley, United Kingdom.

[7] Sayood, Khalid, 2012, *Introduction to Data Compression, Fourth Edition*, Morgan Kaufmann, San Francisco.