Clustering Titik Fitur Model Wajah 3D Menggunakan *K-Nearest Neighbour*

Matahari Bhakti Nendya[1], Eko Mulyanto Yuniarno[2], Surya Sumpeno[3]

***Abstrak*— Langkah awal dalam proses transfer animasi menggunakan *data motion capture* kepada model wajah 3D adalah menentukan titik fitur wajah dan keterkaitan antar titik tersebut supaya membentuk sistem gerak model wajah 3D. Penelitian ini berfokus pada pengelompokan titik fitur wajah dimana sudah ditentukan 33 titik pusat (centroid) dan mencari keterkaitannya dengan titik fitur lainya. Model wajah 3D yang digunakan berupa model wajah karakter humanoid yang mana mempunyai kemiripan dengan model wajah 3D manusia. Hasil yang didapatkan berupa sebaran titik fitur wajah yang akan digunakan sebagai acuan dalam proses *mesh deformation* menggunakan *linear blend skinning*.**

***Kata Kunci*: titik fitur, model wajah 3d, *clustering*, k-nearest neighbour, *mesh deformation*, *linear blend skinning*.**

***Abstract*—The first step in the process of transferring animation using motion capture data to a 3D face model is to determine the facial feature points and the relationship between these points to form a 3D facial model motion system. This study focuses on grouping facial feature points where 33 centroids have been determined and looking for their association with other feature points. The 3D face model used is a humanoid character face model which is similar to a human 3D face model. The results obtained are the distribution of facial feature points that will be used as a reference in the mesh deformation process using linear blend skinning**

***Keywords: feature point, 3d face models, clustering, k-nearest neighbour, mesh deformation, linear blend skinning*.**

I. PENDAHULUAN

Animasi wajah merupakan aspek penting di lingkungan virtual 3D yang menghadirkan keberadaan karakter, baik manusia maupun yang menyerupainya. Pemanfaatannya meliputi industri game 3D, perangkat lunak interaktif, dan film animasi 3D. Bagaimanapun juga untuk menghasilkan animasi wajah yang baik diperlukan waktu yang lama bagi seorang animator ahli dikarenakan kompleksitas ekspresi wajah manusia yang merupakan bagian penting dalam pergerakan secara alamiah [1]. Bahkan sampai saat ini perkembangan penelitian animasi wajah masih banyak dihasilkan, terutama untuk dua aspek penting: proses rigging wajah dan transfer animasi antar dua buah karakter virtual 3D atau antar manusia ke karakter 3D.

Metode pembuatan ekpresi wajah 3D secara prinsip ada 2 buah, yaitu: dengan penggunaan data hasil pemindai 3D atau hasil dari pahatan seniman [2], [3]. (Pada pembuatan animasi menggunakan *blendshapes*, setiap bentuk ekpresi wajah membutuhkan kesamaan geometri model wajah pada posisi netral. Hal ini membutuhkan waktu 2-3 minggu dan lebih dari 100 *shapes* dibuat untuk mewujudkan ekpresi wajah dan fonem dari sebuah karakter yang kompleks. Hal ini dapat diatasi dengan melakukan transfer otomatis bentuk dari model sumber ke model target yang dikendaki [4].

Sebelum memulai proses transfer animasi terlebih dahulu perlu disiapkan model wajah 3D yang telah ditentukan titik fiturnya. Hal ini supaya proses perpindahan animasi mudah untuk dilakukan. Penelitian ini akan berfokus pada tahapan mempersiapkan titik fitur model wajah 3D dan mengelompokkannya berdasrakan titik gerak terdakat yang berpengaruh dalam pembentukan animasi wajah. Titik gerak terdekat inilah yang akan menjadi acuan dalam proses akhir transfer animasi yaitu *mesh deformation* menggunakan *linear blend skinning*.

II. TINJAUAN PUSTAKA

A. *Feature Point*

Titik merupakan fitur yang penting dalam sebuah gambar. Titik berperan untuk mengetahui koordinat dari sejumlah titik yang saling berkorespondensi pada dua gambar. Titik juga berperan dalam proses transformasi seperti resample gambar geometri lain. Titik fitur juga dikenal sebagai *interest point, point landmarks, corner point* dan *control point*. Titik fitur merupakan pusat dari lingkungan (*neighborhoods*) yang unik dan berisi informasi tentang gambar yang cukup besar. Teknik pengukuran yang berbeda menyebabkan kadar keunikan dan informasi dari neighorhoods berbeda pula.

Neighborhoods dengan jumlah piksel dengan nilai gradien yang tinggi sangat informatif dan memungkinkan untuk terjadinya pembentukan pola yang unik, seperti sudut sehingga kemudian *neigborhoods* menjadi unik pula. Berbagai metode telah dikembangkan untuk mendeteksi sudut dalam sebuah gambar. Deteksi tepi dari sebuah gambar dilakukan dengan menempatkan titik yang

[1] *Dosen Program Studi Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana, Jl. Dr Wahidin Sudirohusodo 5-25 Yogyakarta 552244 INDONESIA e-mail: didanendya@ti.ukdw.ac.id*

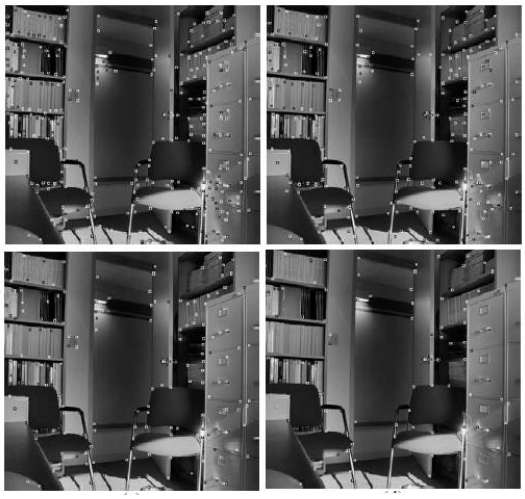
[2] *Dosen Departemen Teknik Elektro, Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo Surabaya 60110 INDONESIA e-mail: ekomulyanto@ee.its.ac.id*

[3] *Dosen Departemen Teknik Elektro, Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo Surabaya 60110 INDONESIA e-mail: surya@ee.its.ac.id*

memiliki nilai maksimul lokal yang tinggi. Sebagai asumsi $I_x(x,y)$ dan $I_y(x,y)$ adalah gradient dari gambar $I(x,y)$ terhadap x dan y dimana (x,y), $\overline{I_x(x,y)}$ dan $\overline{I_y(x,y)}$ adalah gradient rata-rata terhadap x dan y pada jendela kecil yang berpusat di (x,y) dan

$$C(x,y) = \begin{bmatrix} \overline{I_x(x,y)I_x(x,y)} & \overline{I_x(x,y)I_y(x,y)} \\ \overline{I_x(x,y)I_x(x,y)} & \overline{I_y(x,y)I_y(x,y)} \end{bmatrix} \quad (1)$$

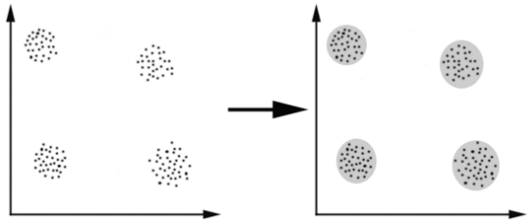
kemudian *det(C)* dapat digunakan untuk melakukan pengukuran tepi. *C* disebut dengan matriks *inertia*. Pengukuran tepi ini kemudian dilakukan proses normalisasi dengan menggunakan *det(C) / tr(C)* untuk mendapatkan deteksi tepi dari sebuah gambar. *det(C)* dan *tr(C)* menunjukkan determinasi dan jejak dari matriks.
Nilai *eigen* dari matriks *C* merupakan indikator kekuatan gradien dalam arah normal satu sama lain pada sebuah *window*. Ketika nilai *eigen* yang didapat besar, dapat disimpulkan bahwa terdapat tepi yang kuat pada *window*. Hal ini cukup untuk menemukan nilai terkecil dari kedia nilai eigen yang digunakan untuk menemukan tepi gambar. Tomasi dan Kanade [5]menggunakan pendekatan ini untuk menemukan tepi dari sebuah gambar. Deteksi tepi terkadang tidak diperlukan, namun dalam rangka mempercepat proses, deteksi tepi pada gambar dapat digunakan untuk membatasi ruang pencarian.



Gambar 1 Deteksi Titik Fitur Gambar

B. *Clustering*

*Cluster*ing dapat dianggap sebagai salah satu cara untuk mempelajari sebuah kasus yang berbuhungan dengan pencarian struktur dalam kumpulan data yang berlabel. *Cluster*ing sendiri dapat diartikan sebagai proses pengorganisasian obyek kedalam kelompok-kelompok tertentu yang anggotaya serupa dengan berbagai macam cara. *Cluster* merupakan kumpulan obyek-obyek yang mempunyai persamaan dan mempunyai perbedaan dengan obyek *cluster* lain. Bentuk *cluster* dapat digambarkan sebagai berikut

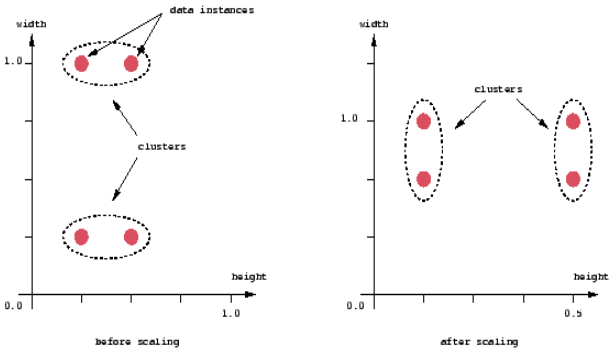


Gambar 2 Clustering

Pada kasus ini, dapat dengan mudah dilakukan identifikasi 4 *cluster* menjadi data yang dapat dibagi berdasarkan kriteria persamaan jarak. Dua atau lebih obyek termasuk dalam satu *cluster* yang sama berdasarkan jarak (jarak *geometry*) yang sama sehingga dapat disebut sebagai *distance-based cluster*ing. Tipe lain dari *cluster*ing adalah konseptual *cluster*ing dengan dua atau lebih obyek yang memiliki *cluster* sama. Obyek dikelompokkan sesuai dengan deskripsi konsep yang memiliki kesamaan.

C. *Pengukuran Jarak*

Komponen utama dalam algoritma *cluster*ing adalam pengukuran jarak antar point data. Data dalam bentuk vektor yang sama secara bentuk physical dapat menggunakan pendekatan matriks Euclidean distance utuk melakukan jarak antar point data. Akan tetapi, dalam kasus tertentu Euclidean distance dapat membingungkan. Gambar dibawah ini menampilkan contoh dari pengukuran lebar dan tinggi dari suatu obyek. Meskipun kedua pengukuran berasal dari unit physical yang sama, keputusan harus dibuat berdasarkan skala relatif. Seperti yang ditunjukkan pada Gambar 2.5 perbedaan skala dapat menyebabkan perbedaan *cluster*.



Gambar 3 Perbedaan Skala Pada Clustering

Perlu menjadi perhatian bahwa permasalahan bukan hanya terjadi pada masalah grafis, masalah lain yang muncul adalah penggunaan model matematika yang dikombinasikan untuk melakukan pengukuran komponen tunggal dari vector data feature kedalam bentuk unik dari pengukuran jarak yang dapat digunakan untuk keperluan *cluster*ing. Formula matematika yang berbeda dapat mengakibatkan proses *cluster*ing yang berbeda pula.

Domain knowledge dari suatu permasalahan yang diambil harus digunakan untuk mmbentuk formula yang cocok untuk melakukan pengukuran jarak pada kasus tertentu.

D. K-Nearst Neighbour

Algoritma *k-nearest neighbor* (k-NN atau KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. *k-Nearest Neighbor* berdasarkan konsep '*learning by analogy*'. Data *learning* dideskripsikan dengan atribut numerik *n*-dimensi. Tiap data *learning* merepresentasikan sebuah titik, yang ditandai dengan *c*, dalam ruang *n*-dimensi. Jika sebuah data *query* yang labelnya tidak diketahui diinputkan, maka *k-Nearest Neighbor* akan mencari *k* buah data *learning* yang jaraknya paling dekat dengan data *query* dalam ruang *n*-dimensi. Jarak antara data *query* dengan data *learning* dihitung dengan cara mengukur jarak antara titik yang merepresentasikan data *query* dengan semua titik yang merepresentasikan data *learning* dengan rumus e*uclidean distance*.

Pada fase *training*, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi data *training* sample. Pada fase klasifikasi, fitur–fitur yang sama dihitung untuk testing data (klasifikasinya belum diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor *training* sample dihitung, dan sejumlah *k* buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik – titik tersebut. Nilai *k* yang terbaik untuk algoritma ini tergantung pada data; secara umumnya, nilai *k* yang tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Nilai *k* yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan *cross-validation*. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat (dengan kata lain, *k* = 1) disebut algoritma *nearest neighbor*.

Ketepatan algoritma k-NN ini sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan, atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur, agar performa klasifikasi menjadi lebih baik.

k buah data learning terdekat akan melakukan voting untuk menentukan label mayoritas. Label data *query* akan ditentukan berdasarkan label mayoritas dan jika ada lebih dari satu label mayoritas maka label data query dapat dipilih secara acak di antara label-label mayoritas yang ada.

Nilai *k* yang terbaik untuk algoritma ini tergantung pada data. Secara umum, nilai *k* yang tinggi akan mengurangi efek *noise* pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi semakin kabur. Nilai *k* yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan *crossvalidation*. Kasus khusus dimana klasifikasi diprediksikan berdasarkan *training* data yang paling dekat (dengan kata lain, *k* = 1) disebut algoritma *nearest neighbor*. Ketepatan algoritma k-NN sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur agar performa klasifikasi menjadi lebih baik.

KNN memiliki beberapa kelebihan yaitu ketangguhan terhadap *training* data yang memiliki banyak *noise* dan efektif apabila *training* data-nya besar. Sedangkan, kelemahan KNN adalah KNN perlu menentukan nilai dari parameter *k* (jumlah dari tetangga terdekat), *training* berdasarkan jarak tidak jelas mengenai jenis jarak apa yang harus digunakan dan atribut mana yang harus digunakan untuk mendapatkan hasil terbaik, dan biaya komputasi cukup tinggi karena diperlukan perhitungan jarak dari tiap *query instance* pada keseluruhan *training* sample.

Terdapat beberapa jenis algoritma pencarian tetangga terdekat, diantaranya:

1. *Linear scan*
2. Pohon kd
3. Pohon *Balltree*
4. Pohon metrik
5. *Locally-sensitive hashing* (LSH)

Algoritma k-NN ini memiliki konsistensi yang kuat. Ketika jumlah data mendekati tak hingga, algoritma ini menjamin error rate yang tidak lebih dari dua kali Bayes *error rate* (*error rate* minimum untuk distribusi data tertentu).

E. Deformasi Mesh

Pendekatan teknik animasi wajah yang paling umum adalah *linear weighting* dengan model *blendshape* [6] dan *linear blend skinning* berbasis rangka [7]. Pada teknik ini animator membuat bentuk acuan dan secara linier dilakukan penggabungan dengan bentuk tersebut untuk menghasilkan sebuah aliran gerakan. Namun hal ini membutuhkan waktu yang lama dikarenakan ekspresi manusia sangatlah kompleks dan beragam. *Linear blend skinning* juga dikenal sebagai teknik *skeleton subspace*, yaitu proses penyatuan *mesh* wajah ke kerangka tulang. Setiap titik diasosiasikan dalam sebuah himpunan titik sendi yang terpengaruh dengan nilai bobotnya. Perubahan bentuk *mesh* terjadi jika setiap titik secara konstan berubah sesuai dengan pengaruh titik-titik yang berhubungan berdasarkan bobot perubahan setiap titiknya. Metode *skinning deformation* masih merupakan metode paling popular yang digunakan pada praktek animasi karakter secara *real-time* dikarenakan kesederhanaan proses dan efisiensi kerja komputasinya.

Proses pembentukan animasi *mesh* melibatkan teknik *mesh deformation*. Batasan yang diacu pada konteks ini adalah: pertama, proses komputasi harus dapat dilakukan secara singkat atau mendekati *real-time*. Dan kedua, perubahannya harus dipandu oleh deformasi beberapa titik

fitur dikarenakan metode yang diacu melandasi proses *retargeting* animasi dengan beragam sumber yang dimungkinkan, seperti penjejakan secara 2D/3D dan animasi 3D. Berdasarkan hal ini, maka digunakanlah teknik deformasi berbasis kerangka/skeletal: *linear blend skinning*.

Linear blend skinning (LBS) mengubah *mesh* menurut transformasi lokal dari rangka tulang. Untuk mengadaptasi teknik *skinning deformation* dengan perubahan letak titik fitur, maka dipertimbangkan juga posisi titik fitur sebagai sebuah titik sendi gerak. Bobot untuk setiap titik didefinisikan $\sum w_i = 1$ dengan bobot tulang ke-i $w_i \in [0,1]$. Bobot dengan nilai 0 berarti bahwa titik fitur tersebut tidak mempengaruhi titik vertek pada *mesh* tersebut, sedangkan jika bobot bernilai 1 berarti bahwa titik vertek pada *mesh* tersebut dipengaruhi hanya oleh titik fitur itu saja. Posisi vertek $\vec{v}$ pada *frame f* didefinisikan oleh:

$$\overrightarrow{v_f} = \overrightarrow{v_0} + \sum_{i=1}^{N} w_i \cdot \overrightarrow{d_{if}} \quad (2)$$

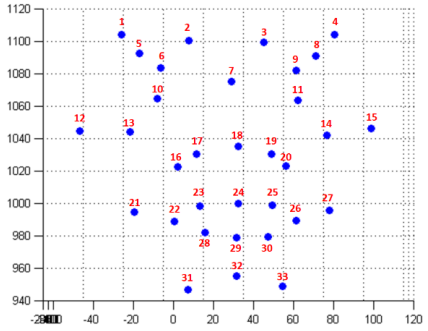
Dimana N adalah banyaknya titik fitur dan $\overrightarrow{d_{if}}$ adalah perubahan letak titik fitur ke-i pada *frame f*.

III. HASIL DAN PEMBAHASAN

Penelitian ini menggunakan model wajah 3D humanoid. Pada tahapan ini digunakan 33 titik fitur wajah. Pengamatan dilakukan dengan pendekatan sintesis mengelompokkan langsung simpul permukaan pada model wajah 3D humanoid dengan 33 titik fitur tadi sebagai titik pusat gerak.

Facial rigging merupakan proses pembuatan kendali wajah untuk melakukan pembentukan animasi yang dilakukan oleh animator. Pada penelitian ini proses *facial rigging* dilakukan pada tiap model dari karakter wajah yang digunakan. Proses pembentukan *facial rigging* berupa pemberian titik fitur pada topeng wajah yang mengacu pada pendekatan FACS yang digunakan dalam *optitrack motion capture system*.

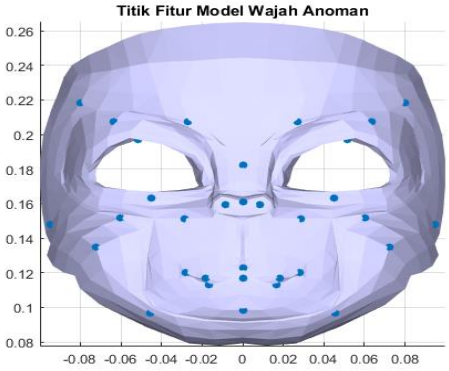
Untuk mempermudah dalam proses penggunaan ulang animasi pada model wajah serta memberikan analisa terdapat perpindahan titik fitur yang ada, lokasi titik fitur wajah diberi nomor secara manual. Pemberian nomor tersebut juga bertujuan untuk mempermudah dalam melakukan pengaksesan data koordinat sehigga ketika proses perindahan titik fitur tidak terlihat secara visual masih dapat dilakukan pengamatan terhadap perubahan data yang terjadi.



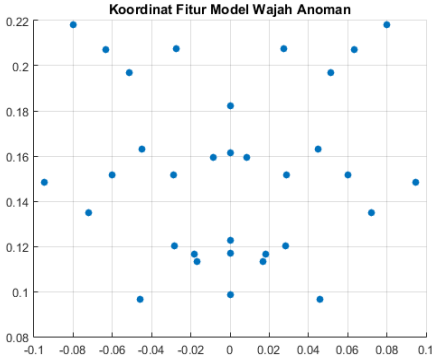
Gambar 4. Pemberian nomor pada titik fitur wajah

Proses *facial rigging* ini dilakukan secara manual pada model wajah 3D. Hasilnya dari proses *facial rigging* ini berupa data koordinat titik fitur topeng wajah dari model avatar yang digunakan. Koordinat titik fitur topeng wajah tersebut akan digunakan sebagai centroid dalam proses *clustering* dan digunakan dalam proses *retargeting*.

Proses *facial rigging* pada model wajah kaakter anoman dilakukan dengan mengambil salah satu vertek yang posisinya sesuai dengan acuan dalam *OptiTrack motion capture system* sebagai titik fitur wajah. Tahapan selanjutnya adalah proses registrasi (penyamaan koordinat) titik fitur wajah pada model wajah karakter anoman.



(a)



(b)

Gambar 5. Titik Fitur Model Wajah Anoman (a) Lokasi titik fitur (b) titik fitur dalam sistem grafik

Hasil yang didapatkan dalam proses registrasi pada model wajah karakter anoman berupa data koordinat titik fitur model wajah karakter anoman yang disajikan dalam table 1.

Data koordinat titik fitur model wajah karakter anoman ini kemudian digunakan sebagai data masukan dalam proses *clustering* dan *retargeting*.

TABEL 1

KOORDINAT TITIK FITUR PADA MODEL TOPENG WAJAH ANOMAN

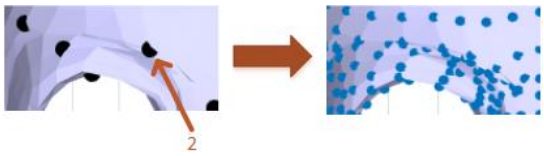
Titik Fitur	Koordinat		
	x	y	z
1	-0.08006	0.2182	0.07035
2	-0.02728	0.2075	0.09957
3	0.02728	0.2075	0.09957
4	0.08006	0.2182	0.07035
5	-0.06334	0.2072	0.08352
6	-0.05143	0.197	0.08354
7	0	0.1821	0.1001
8	0.06334	0.2072	0.08352
9	0.05143	0.197	0.08354
10	-0.04483	0.1632	0.09349
11	0.04483	0.1632	0.09349
12	-0.09451	0.1482	0.05082
13	-0.06013	0.1518	0.09318
14	0.06013	0.1518	0.09318
15	0.09451	0.1482	0.05082
16	-0.02868	0.1515	0.1003
17	-0.008673	0.1592	0.1074
18	0	0.1613	0.1078
19	0.008673	0.1592	0.1074
20	0.02868	0.1515	0.1003
21	-0.0723	0.1347	0.08973
22	-0.02826	0.1202	0.1021
23	-0.01829	0.1167	0.1063
24	0	0.1226	0.1086
25	0.01829	0.1167	0.1063
26	0.02826	0.1202	0.1021
27	0.0723	0.1347	0.08973
28	-0.01678	0.1132	0.1046
29	0	0.117	0.1064
30	0.01678	0.1132	0.1046
31	-0.0458	0.09664	0.08959
32	0	0.0984	0.1011
33	0.0458	0.09664	0.08353

Proses *clustering* yang digunakan dalam penelitian ini mengacu pada algoritma clustering *k-nearest neighbor*. Algoritma *k-nearest neighbor* (k-NN atau KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Dalam hal ini, data pembelajaran merupakan data verteks yang lokasinya dekat dengan titik fitur.

Komponen utama dalam algoritma *cluster*ing adalam pengukuran jarak antar verteks dan titik fitur. Data dalam bentuk vektor yang sama secara dapat menggunakan pendekatan matriks *euclidean distance* utuk melakukan jarak antar verteks dan titik fitur.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (3)$$

Proses *clustering* pada model wajah karrakter anoman melibatkan seluruh verteks yang ada didalam topeng wajah model anoman yaitu sebanyak 4416 buah. Jumlah verteks tersebut dibagi kedalam 33 titik fitur yang telah ditentukan dalam proses *facial rigging*. Nilai *k* ditentukan sebagai koordinat verteks yang berasosisi dengan titik fitur wajah. Penentuan anggota *cluster* mengguakan perhitungan jarak terpendek *euclidean distance*.



Gambar 6. Titik Fitur Nomor 2

cluster yang dipilih adalah pusat *cluster* yang berasosiasi dengan titik fitur nomor 2 yang memiliki koordinat -0.02728, 0.2075, 0.09957, maka koordinat verteks yang termasuk anggota *cluster* titik fitur nomor 2 disaijkan dalam tabel 2

TABEL 2.

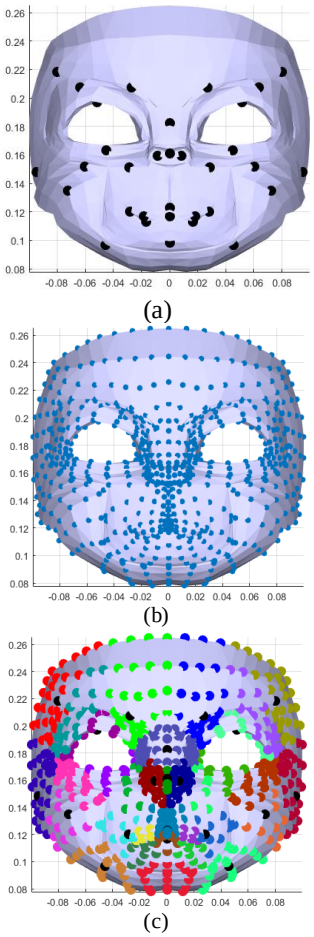
HASIL PERHITUNGAN KEANGGOTAAN CLUSTER NOMOR 2

Cluster	Koordinat Vetreks			Jarak Terpendek
	x	y	z	
2	0	0.26502	0.090776	0.064266
	0	0.26502	0.090776	0.064266
	0	0.244534	0.102285	0.046077
	0	0.226004	0.103402	0.033186
	0	0.210119	0.100996	0.027443
	0	0.210119	0.100996	0.027443
	0	0.244534	0.102285	0.046077
	0	0.226004	0.103402	0.033186
	0	0.226004	0.103402	0.033186
	-0.03943	0.261193	0.085003	0.056946
	-0.03943	0.261193	0.085003	0.056946
	-0.03601	0.242239	0.096956	0.035914
	-0.02425	0.242745	0.099292	0.035376
	-0.0266	0.263165	0.087684	0.056924
	-0.03943	0.261193	0.085003	0.056946

Hasil perhitungan keanggotan *cluster* nomor 2 secara keseluruhan dapat dilihat pada bagian lampiran. Banyaknya jumlah masing-masing anggota *cluster* berbeda-beda bergantung pada banyaknya jumlah *mesh* yang digunakan dalam tahap pembuatan model karakter. Semakin banyak *mesh* yang digunakan semakin banyak pula kenaggotaan tiap *cluster* dan tentu saja akan memberikan beban komputasi yang lebih berat.

Secara keseluruhan, proses *cluster*ing pada model wajah karakter anoman dapat dilihat pada gambar 8. Proses

clustering pada model wajah karakter anoman memberikan keanggotan bagi verteks terhadap titik fitur tertentu.



Gambar 7. Proses Clustering pada Topeng Wajah Anoman (a) Posisi pusat cluster yang (b) Vertex pada model wajah anoman (c) Hasil Clustering

Derajat keanggotan ini yang akan menjadi acuan dalam proses deformasi untuk mengetahui relasi antara *mesh* wajah dan titik fitur dalam pembentukan animasi ekspresi wajah terutama pada model wajah karakter anoman.

IV. KESIMPULAN

Dari penelitian ini dapat disumpulkan bahwa pembentukan cluster animasi pada titik fitur gerak wajah karakter dapat membantu dalam proses pembentukan animasi ekspresi wajah. Hasil yang didapatkan berupa sebaran titik fitur wajah yang akan digunakan sebagai acuan dalam proses *mesh deformation* menggunakan *linear blend skinning*.

DAFTAR PUSTAKA

[1] F. I. Parke, "Computer generated animation of face," *Proc. ACM Annu. Conf.*, vol. 1, pp. 451–457, 1972.

[2] G. Faigin, *The artist's complete guide to facial expressions 2nd Edition*. New York: Watson-Guptill Publications, 2008.

[3] P. Joshi, W. C. Tien, M. Desbrun, and F. Pighin, "Learning controls for blend shape based realistic facial animation," *SIGGRAPH 2006 - ACM SIGGRAPH 2006 Courses*, 2006, doi: 10.1145/1185657.1185857.

[4] V. C. Orvalho, E. Zacur, and A. Susin, "Transferring the rig and animations from a character to different face models," *Comput. Graph. Forum*, vol. 27, no. 8, pp. 1997–2012, 2008, doi: 10.1111/j.1467-8659.2008.01187.x.

[5] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *Int. J. Comput. Vis.*, vol. 9, no. 2, pp. 137–154, 1992, doi: 10.1007/BF00129684.

[6] Z. Deng, P. Y. Chiang, P. Fox, and U. Neumann, "Animating blendshape faces by cross-mapping motion capture data," *Proc. Symp. Interact. 3D Graph.*, vol. 2006, no. March, pp. 43–48, 2006, doi: 10.1145/1111411.1111419.

[7] J. P. Lewis, M. Cordner, and N. Fong, "Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation," *Proc. ACM SIGGRAPH Conf. Comput. Graph.*, pp. 165–172, 2000.