Rancang Bangun Aplikasi Desktop Simulasi Respon Hewan Peliharaan Berbasis *Leap Motion*

Andrean Cenjaya[1], Caecilia Citra Lestari[2], Nehemia Sugianto[3]

***Abstrak*— Memiliki hewan peliharaan adalah salah satu dari keinginan setiap orang. Dengan memiliki hewan peliharaan, orang dapat menikmati kelucuan atau keindahan dari hewan peliharaan tersebut. Namun tidak sedikit pula orang dengan keinginan memiliki hewan peliharaan tidak terkabul. Hal tersebut disebabkan beberapa faktor seperti kesehatan, biaya perawatan, dan tempat tinggal. Oleh karena itu, akan dilakukan rancang bangun aplikasi simulasi hewan peliharaaan dengan integrasi *Leap motion*, yang mampu memenuhi keinginan mereka yang tidak sanggup memiliki hewan peliharaan, dengan fitur interaktif dan efek realisme dengan grafik 3D yang dibangun menggunakan Unity. Tujuan dari penelitian ini adalah merancang dan membangun aplikasi simulasi hewan peliharaan berbasis desktop. Pembuatan aplikasi menggunakan Unity dengan bahasa pemograman C#. Aplikasi menggunakan *Leap motion* sebagai alat input penangkap gerakan tangan. Pada rancang bangun aplikasi ini, model hewan peliharaan yang digunakan adalah kucing.Simulasi pada aplikasi berpusat pada interaksi antara pengguna dengan hewan. Interaksi tersebut dibagi menjadi dua bagian, masukan dari pengguna dan reaksi dari hewan terhadap masukan. Masukan akan diklasifikasi menggunakan metode Dynamic Time Warping untuk menentukan jenis gerakan. Reaksi dari hewan akan diklasifikasi menggunakan metode Naïve Bayes. Data pelatihan untuk reaksi hewan didapat secara eksperimental. Pengujian aplikasi difokuskan pada pengukuran akurasi dari kedua metode klasifikasi yang digunakan, dengan data pengujian yang disediakan. Hasil pengujian menunjukan akurasi untuk klasifikasi gerakan tangan adalah 100% dan untuk klasifikasi respon hewan adalah 43,75%. Kesimpulan yang ditarik adalah model data pelatihan masih underfitting sehingga perlu penambahan sampel untuk data pelatihan.**

***Kata Kunci*: Aplikasi, Dekstop, Simulasi, Hewan Peliharaan, Leap Motion, Unity.**

***Abstract*— Having a pet is one of the desire for every person. By having a pet, people can enjoy the cuteness or the beauty of the pet. Having a pet, other than as a means of entertainment, also has a specific function. But that desire to have a pet can't always be fulfilled. This is due to several factors such as health, cost of care, and current living place. Therefore, pet simulation application will be build, with the integration of *Leap motion*, which is able to meet the expectation of those who can not have pet, with interactive features and realism from 3D graphics that built using Unity. The purpose of this research is to designs and develops pet simulation desktop application. Unity with C# as programming language will be uses for this purpose. This application uses *Leap motion* as input device for hand gesture. In this research, pet that will be uses as model is cat. Simulation from this application focused on interaction between user and pet. Interaction divide into two parts, input from user and input based pet reaction. Input from user will be classify using Dynamic Time Warping method, to know which kind of the gesture is detected. Pet reaction will be classify using Naïve Bayes method. Data training for pet reaction obtained experimentally. Application testing focused on calculating accuration of both classification used, using prepared data test. Test result shows that the accuracy for gesture classification is 100% and the accuracy for response classification is 43,75%. Given the result, the conclusion is the data training model is underfitting that another amount of new samples is needed to increase data training size.**

Keywords: Application, Desktop, Simulation, Pet, Leap Motion, Unity.

I. PENDAHULUAN

A. Latar Belakang

Memiliki hewan peliharaan adalah salah satu hobi yang orang biasa lakukan. Hewan-hewan peliharaan seperti anjing dan kucing adalah salah satu yang paling sering dipelihara dan kita bisa lihat di kehidupan sehari-hari. Banyak sekali yang bisa dijadikan alasan untuk memelihara hewan tersebut seperti kelucuan dari hewan tersebut atau keinginan untuk memiliki pendamping dan penjaga di rumah.

Terlepas dari hal di atas, tidak sedikit juga jumlah orang yang ingin memelihara hewan-hewan tersebut, tetapi tidak sanggup untuk memilikinya. Sebuah survei yang dilakukan oleh American Humane Association's Animal Welfare Research Institute [1] kepada 500 orang yang pada saat survey dilakukan, tidak memiliki hewan peliharaan. Hasil survey menunjukkan bahwa alasan tidak memiliki hewan peliharaan dikarenakan biaya pemeliharaan yang terlalu tinggi, tidak memiliki waktu untuk memelihara, atau alasan kesehatan seperti alergi. Selain dari tiga alasan utama tersebut, alasan lainnya yang dipaparkan Becker [2] dikarenakan adanya perasaan kehilangan dari hewan peliharaan sebelumnya atau keinginan untuk memlihara hewan lainnya yang tidak lazimnya untuk dipelihara seperti

[1] *Mahasiswa, Program Studi Teknik Informatika, Fakultas Industri Kreatif Universitas Ciputra, UC Town, Citraland, Surabaya 60291 INDONESIA (telp: 0857-4051-4777; e-mail: acenjaya@student.ciputra.ac.id)*

[2] *Dosen, Program Studi Teknik Informatika, Fakultas Industri Kreatif Universitas Ciputra, UC Town, Citraland, Surabaya 60291 INDONESIA (telp: 031-745 1699; fax: 031-745 1698; e-mail: caecilia.citra@ciputra.ac.id)*

[3] *Dosen, Program Studi Teknik Informatika, Fakultas Industri Kreatif Universitas Ciputra, UC Town, Citraland, Surabaya 60291 INDONESIA (telp: 031-745 1699; fax: 031-745 1698; e-mail: nsugianto@ciputra.ac.id)*

ular atau hewan-hewan dilindungi lainnya.

Pada era 1990, telah muncul sebuah permainan simulasi yang disebut Digital Pet. Digital Pet merupakan salah satu dari Artificial Human Companion (AHC), atau pendamping manusia buatan. AHC dapat berupa *hardware* (perangkat keras) maupun *software* (perangkat lunak). Tujuan dari AHC sendiri tidak lain untuk memberikan perhatian sekaligus sebagai pendamping bagi pemiliknya. Digital Pet menunjukkan popularitasnya pada tahun 1995-1996 dengan kehadiran Tamagotchi. Hingga sekarang, telah muncul banyak sekali permainan simulasi *digital pet* berserta pengembangannya.

Perkembangan teknologi saat ini memunculkan salah satu alat input yang menggunakan *natural interactivity* (interaktif alami), bernama *Leap motion*. *Leap motion* berkerja dengan cara menangkap gerakan tangan secara detil tanpa perlu memegang alat tersebut. Integrasi dengan *Leap motion* memungkinkan pengguna untuk memberikan input dalam bentuk gerakan.

Dengan melihat dari besarnya pasar dari *Digital Pet* dan peluang yang dilihat dari *Leap motion* tersebut serta peluang di masyarakat mengenai kebutuhan memiliki hewan peliharaan, maka terbentuklah sebuah gagasan untuk membuat aplikasi permainan simulasi hewan peliharaaan yang mengintegrasi *Leap motion* sebagai alat masukan (*input device*). Tujuan dari integrasi permainan dengan *Leap motion* adalah untuk menerapkan fungsi interaksi alami yang dapat dilakukan oleh *Leap motion* dengan melakukan gerakan tangan. Hal ini dapat memberikan efek realisme yang lebih baik dibandingkan dengan permainan simulasi yang masih menggunakan *keyboard* dan *mouse* sebagai alat masukan, seperti yang disebutkan Bouvier [3].

B. Tujuan Penelitian

Tujuan dari penelitian ini adalah membangun sebuah aplikasi permainan simulasi hewan peliharaan dengan integrasi *Leap motion* yang dapat memberikan respon hewan yang tepat.

C. Ruang Lingkup

1) Masukan diterima menggunakan *Leap motion*, yaitu dengan memanfaatkan sensor yang dimiliki alat tersebut untuk mendeteksi lokasi tangan serta pergerakan tangan.
2) Masukkan gerakan akan dimasukkan dan diklasifikasi menggunakan metode Dynamic Time Warping (DTW) untuk mengenai jenis gerakan yang dilakukan.
3) Respon hewan ditentukan dari jenis gerakan dan lokasi penerima gerakan pada tubuh hewan.
4) Jenis gerakan yang digunakan adalah gerakan mengelus dan menepuk yang dilakukan dengan tangan.
5) Lokasi penerima gerakan pada tubuh hewan adalah pada bagian kepala dan punggung.
6) Respon hewan yang dihasilkan dari input, adalah salah satu dari variasi beberapa jenis respon serupa yang ditentukan dengan menggunakan probabilitas Naïve Bayes.
7) Respon hewan yang dimiliki oleh objek terbagi menjadi diam, menjauh, dan menoleh.
8) Aplikasi permainan dibangun berbasis dekstop, dengan menggunakan Unity 5.4 sebagai game engine (software pembuatan permainan).
9) Hewan yang dijadikan model objek adalah kucing.

II. LANDASAN TEORI

A. *Digital Pet*

Digital Pet, yang juga dikenal sebagai *Virtual Pet*, merupakan salah satu dari *Artificial Human Companion* (AHC), atau pendamping manusia buatan, dimana AHC dapat ditemukan dalam wujud *hardware* (perangkat keras) maupun *software* (perangkat lunak). AHC bertujuan untuk memberikan perhatian sekaligus pendamping bagi pemiliknya. Dimuat dalam History of Virtual Pet [4], *Digital Pet* mengawali popularitasnya pada tahun 1995 dengan munculnya Dogz yang disusul dengan Catz pada tahun 1996 dan yang kemudian dikenal dengan *franchise* Petz. Hal ini juga diramaikan dengan kehadiran *franchise* Pokemon (Pocket Monster) dan Tamagotchi dengan kemunculan pertamanya pada tahun 1996. Keberhasilan pada era itu membuat *digital pet* berkembang dan memunculkan banyak ide baru. Keberhasilan *franchise* Pokemon juga disusul dengan kemunculan *franchise* Digimon (Digital Monster) pada tahun 1997 dalam bentuk Tamagotchi. Perkembangan lain dari *digital pet* juga memunculkan Ludobots, *virtual pet* dengan menggunakan robot sebagai objek permainan. Salah satu Ludobots yang terkenal adalah AIBO (*Artificial Intelligence Robot*) yang didesain dan dibuat oleh Sony dengan produk pertamanya pada tahun 1999.

Berdasarkan kutipan dari Miller [5], terdapat 2 elemen fitur yang dapat ditemukan dalam *digital pet*, yaitu:

1) Komunikasi, dimana dengan berkembangnya teknologi sekarang, *digital pet* pada umumnya tidak lagi menampilkan berbagai kebutuhannya dalam bentuk parameter atau pesan, tetapi pemiliknya dapat mengerti apa yang menjadi kebutuhan digital pet dari tingkah lakunya, memberikan rasa akan hubungan hewan dan pemiliknya.
2) Realisme, dimana *digital pet* pada umumnya memiliki tingkatan kemandirian tertentu. Sebagai contoh, *digital pet* dapat sakit atau bahkan mati jika berhari-hari tidak dirawat. Hal ini dapat meningkatkan rasa tanggung jawab dari pemiliknya.

Digital pet memiliki perkembangan baik dari segi pasar dan teknologi. Salah satu buktinya tertera pada laporan penjualan Bandai [6] menunjukan penjualan Tamagotchi yang berhasil menjual lebih dari 76 milyar barang ke seluruh penjuru dunia pada tahun 2010, kemunculan berbagai jenis *digital pet* dalam berbagai *platform*, yaitu basis *mobile* dan basis *web* seperti Neopets (1999) yang masih berdiri hingga sekarang, serta kemajuan desain grafis yang dari yang semula dalam wujud *dot-graphics* (grafis dalam wujud penggunaan pixel besar

berwarna hitam) hingga grafis 3D beranimasi seperti Nintedogs (2005) pada Nintendo DS *console*.

B. *Leap motion*

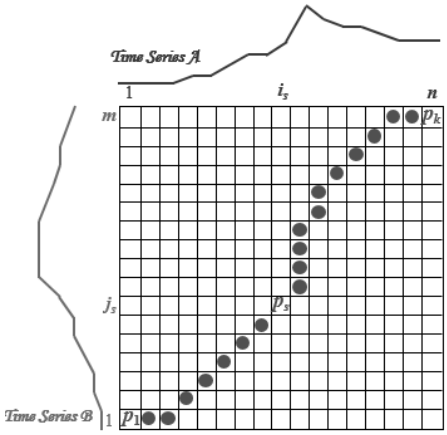
Leap motion controller adalah alat kecil dengan penghubung USB, yang penggunaannya didesain dengan meletakkan alat tersebut pada *desktop* atau tempat datar, dan menghadap ke atas. Dijelaskan oleh Colgan [7], dengan menggunakan 2 kamera *infrared* monokromatik dan 3 *infrared* LED, alat ini dapat melihat dan mendeteksi tangan dan gerakan dalam ruang lingkup setengah bola dengan jarak sejauh 60 cm. Data yang didapat dari hasil penglihatan dan deteksi dalam bentuk gambar per *frame*. Data tersebut kemudian dikirimkan ke komputer. Data selanjutnya akan dianalisa dengan menggunakan perhitungan matematika untuk menghasilkan data posisi 3D dari gambar 2D yang ditangkap dari 2 kamera.

Leap motion pertama kali dibuat oleh David Holz, pada tahun 2008. Setelah membangun perusahaan pada tahun 2010 dan beroperasi secara diam-diam, pada tahun 2010, *Leap motion* diumumkan secara publik dengan nama awal The Leap pada Mei 2012, disusul dengan *software developer*-nya pada Oktober 2012. *Leap motion* mendapatkan gelar *Best Buy in the States* pada 29 Juli 2014, dan kolaborasi dengan ASUS untuk membuat laptop yang terintegrasi dengan *Leap motion controller*. Berdasarkan *Leap motion* Facebook Statistic [8], jumlah penggemar dari LeapMotion di Facebook mengalami peningkatan dari 6 bulan terakhir menunjukan kenaikan sebesar 19 per hari, 88 per minggu, dan 268 orang per bulan. Pertumbuhan ini juga dapat dilihat dari sisi *developer*, dimana terjadi peningkatan jumlah dua kali lipat dengan lebih 25.000 unduh untuk SDK dari LeapMotion, seperti yang dikemukakan Emrich [9].

C. Dynamic Time Warping

Computer vision adalah sebuah bidang pengetahuan yang mempelajari bagaimana cara memproses dan menganalisa gambar untuk mendapatkan informasi yang berharga, seperti bentuk, pencahayaan dan distribusi warna, seperti dinyatakan oleh Szeliski [10]. Menurut Petty [11], pada sebuah artikel di marxentlabs menjelaskan, *gesture recognition* merupakan kemampuan sebuah komputer dalam menangkap dan menafsirkan gerakan manusia. Tujuan dari adanya *gesture recognition* adalah untuk mengetahui jenis gerakan apa yang dideteksi. Gerakan yang telah diidentifikasi tersebut kemudian akan diubah menjadi sebuah input yang dikenali oleh sistem. Cara kerja *gesture recognition* pada umumnya diawali dengan sebuah data gerakan yang ditangkap melalui kamera. Data tersebut diolah oleh *software* khusus dengan *library* dimana setiap *gesture* akan diteruskan menjadi perintah komputer. Di dalam *software* tersebut, setiap gerakan akan ditafsir dengan menggunakan *library* untuk menentukan sebuah arti dari gerakan tersebut. Ketika gerakan selesai ditafsir, *software* akan meneruskan hasil tafsiran ke komputer untuk dijalankan sebagai sebuah perintah.

Di sini penulis menggunakan Dynamic Time Warping sebagai *gesture recognition* yang akan digunakan. Menurut Tsikorpova [12], metode ini melakukan perhitungan jarak tiap point dari sebuah gerakan dengan data pelatihan, hingga menghasilkan sebuah nilai akhir normalisasi jarak perbandingan kedua data terhadap waktu. Nilai-nilai dari setiap perbandingan tersebut kemudian dicari nilai terkecilnya. Jika nilai terkecil didapat setelah membandingkan gerakan dengan semua data pelatihan, maka gerakan tersebut diklasifikasi sesuai dengan kelas dimana perhitungan nilai terkecil tersebut didapat pada data pelatihan. Gambar 1 adalah visualisasi dari perhitungan jarak pada DTW digambarkan sebagai berikut, dimana P merupakan *path* yang menunjukan perhitungan jarak terbaik dengan Pk merupakan hasil akhir dari perhitungan.



Gambar 1. Visualisasi Dynamic Time Warping

Path yang baik memiliki bentuk diagonal, tetapi tidak jarang sebuah perhitungan menghasilkan model path yang melenceng jauh dari bentuk diagonal, dikarenakan luasnya ruang model sehingga banyak bentuk path yang mungkin terbentuk. Untuk itu, metode ini memiliki beberapa *constraint* yang dapat digunakan untuk membatasi dan memperakurat perhitungan, salah satunya *Warping Window constraint*. Hal ini bertujuan menghasilkan path yang bagus tidak jauh dari bentuk diagonalnya.

D. Naïve Bayes

Klasifikasi merupakan salah satu model yang berada di dalam *data mining*, sebuah proses komputasi untuk menemukan sebuah informasi atau pola dari serangkaian data besar. Klasifikasi bekerja dengan cara menemukan pola struktur dari data yang pernah ada, untuk diterapkan pada data yang baru, untuk menentukan termasuk struktur manakah data tersebut. Salah satu metode yang menerapkan proses klasifikasi tersebut adalah Naïve Bayes.

Menurut Han [13], dijelaskan bahwa Naïve Bayes adalah metode klasifikasi yang memberikan prediksi probabilitas berdasarkan atribut dari kelas tertentu. Metode ini diawali dengan teorema dasar dimana jika terdapat X dan Y sebagai variabel faktor, maka probabilitas Y terhadap

X adalah,

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)} \quad (1)$$

dimana X mewakili set atribut dan Y mewakili kelas variabel. Teorema ini sangat berguna ketika Y hanya memiliki satu nilai saja. Tetapi ketika Y memiliki lebih dari satu nilai, maka teorema tersebut berkembang menjadi (2).

$$P(Y|X) = \frac{P(Y)\prod_{i=1}^{d} P(X_i|Y)}{P(X)} \quad (2)$$

Karena P(X) adalah tetap untuk semua kelas Y, maka perhitungan cukup dilakukan sebagai berikut,

$$P(Y|X) = P(Y)\prod_{i=1}^{d} P(X_i|Y = y) \quad (3)$$

Setelah semua P(Y|X) diketahui maka, hasil klasifikasi dapat ditentukan, yaitu mengambil kelas Y pada P(Y|X) terbesar..

E. Unity

Unity merupakan *cross-platform game engine*, yang diciptakan oleh Unity Technologies. *Game engine* adalah sebuah perangkat lunak *framework* yang dirancang untuk membuat dan mengembangkan *video game*. Unity digunakan untuk membuat *video game* pada PC, *console*, perangkat *mobile*, dan *website*.

Menurut Robertson [14], kemampuan Unity yang tidak dimiliki oleh *game engine* adalah kemampuan membuat game dalam banyak *platform*. Hal ini membuat sebuah projek yang dibuat melalui Unity dapat dibuat dalam berbagai *platform* tanpa perlu membuat game satu per satu dalam tiap *platform*. Selain itu, Unity juga mulai menambahkan integrasi dengan ruang lingkup *virtual reality*. Hal ini memungkinan kedepan, Unity mampu menciptakan sebuah game berbasis *virtual reality*.

Dijelaskan oleh MSFX [15], proses *scripting* (pembuatan script) oleh Unity dilakukan oleh Mono, sebuah aplikasi *open-source* (tanpa berbayar) yang sudah terintegrasi dengan Unity. Mono merupakan hasil implementasi dari .NET Framework. Terdapat pilihan bahasa pemrograman yang bisa digunakan dalam proses *scripting*. Programmer dapat memilih menggunakan UnityScript (berbasis JavaScript), C#, atau Boo (berbasis Python).

Pembuatan fitur *collision detection* pada rancang bangun ini telah dibantu Unity dengan adanya fitur Colliders. Unity telah menyiapkan berbagai macam bentuk *collider* yang dapat disesuaikan dengan objek 3D yang digunakan, mulai dari Box Collider, Sphere Collider, Capsule Collider, Mesh Collider dan Wheel Collider. Collider tersebut nantinya akan ditambahkan sebagai atribut pada objek 3D yang akan digunakan. Jika terdapat dua objek yang memiliki Collider, maka Collider akan segera menjalankan *event* berdasarkan kondisi tabrakan yang didapat. Kondisi tabrakan tersebut dibagi menjadi tiga, yaitu objek masuk (*enter*), berada di dalam (*occupy*), dan keluar (*leave*) dari objek lainnya.

III. ANALISA DAN DESAIN SISTEM

A. Analisa Sistem

Aplikasi menerima input gerakan tangan dari user untuk berinteraksi dengan model kucing di dalam aplikasi. Dengan demikian, bagian terpenting dari aplikasi tersebut adalah bagaimana aplikasi dapat menentukan hasil interaksi yang nantinya kelak diberikan. Oleh karena itu, hal yang pertama kali perlu dilakukan sebelum mendesain sistem aplikasi adalah melakukan pengumpulan data riil bagaimana interaksi manusia dengan kucing di dunia nyata. Hal ini bertujuan untuk memberikan hasil yang akurat kelak di dalam aplikasi.

Untuk mempersiapkan pengumpulan terlebih dahulu, ruang lingkup dan prosedur ditentukan dalam melakukan pengumpulan. Pengumpulan ini bersifat eksperimental. Ruang lingkup pengumpulan mencakup jenis gerakan yang dilakukan terhadap kucing, lokasi gerakan diberikan, kecepatan gerakan dan respon yang kemudian diterima. Jenis gerakan yang dilakukan mencakup 2 jenis, menepuk dan menelus. Kedua gerakan tersebut dilakukan dengan menempuh jarak ±10 cm. Satu kali gerakan ditentukan dari posisi awal tangan, yang kemudian bergerak menyentuh lokasi pada kucing, kemudian kembali lagi mendekati/sekitar posisi awal. Lokasi diberikan kedua gerakan tersebut adalah kepala atas dan punggung. Setiap set gerakan dan lokasi dilakukan dua kali dengan kecepatan yang berbeda, yaitu secara cepat dan lambat. Gerakan dikatakan cepat jika satu kali gerakan dilakukan kurang dari 1 detik. Sedangkan gerakan dikatakan lambat jika satu kali gerakan dilakukan lebih atau sama dengan 1 detik. Satu set gerakan, beserta lokasi dan kecepatannya, dilakukan berkisar 10 detik dan direkam sebagai satu rekaman. Dengan demikian, 8 rekaman dapat diambil dari masing-masing kucing. Respon yang diterima dibagi menjadi 3 kategori, yaitu diam, menoleh, dan menjauh. Respon dikategorikan diam, jika kucing tidak memberikan respon atau mengabaikan gerakan tangan. Respon dikategorikan menoleh, jika kucing memberikan respon dengan menggerakan kepala saja. Respon dikategorikan menjauh jika kucing berusaha menghindar dari gerakan tangan.

Setelah prosedur dan ruang lingkup ditentukan, penulis menentukan target populasi dari pengumpulan, yaitu semua sampel kucing yang penulis dapatkan di Surabaya. Dari hasil pengumpulan yang dilakukan, penulis mendapatkan 48 sampel rekaman dari 6 kucing. Tabel I. berikut adalah 10 data pertama dari hasil pengumpulan yang dilakukan.

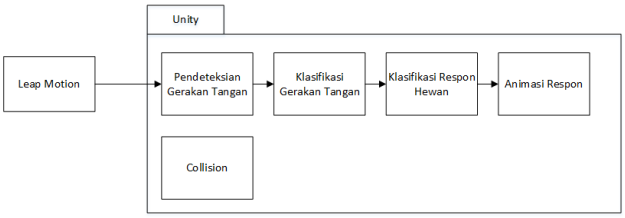
TABEL I
10 DATA PERTAMA PENGUMPULAN RESPON KUCING

No	Lokasi	Gerakan	Kecepatan	Respon
1	Kepala	Elusan	Cepat	Menoleh
1	Kepala	Elusan	Lambat	Menjauh
1	Kepala	Tepukan	Cepat	Menoleh
1	Kepala	Tepukan	Lambat	Menjauh
1	Punggung	Elusan	Cepat	Menjauh
1	Punggung	Elusan	Lambat	Menoleh
1	Punggung	Tepukan	Cepat	Menoleh
1	Punggung	Tepukan	Lambat	Menoleh
2	Kepala	Elusan	Cepat	Menjauh
2	Kepala	Elusan	Lambat	Menoleh

Dari 48 sampel tersebut, 33% digunakan pada pengujian, sedangkan 66% digunakan pada implementasi sistem. Dengan kata lain, 32 sampel digunakan pada implementasi, dan 16 sampel digunakan pada pengujian.

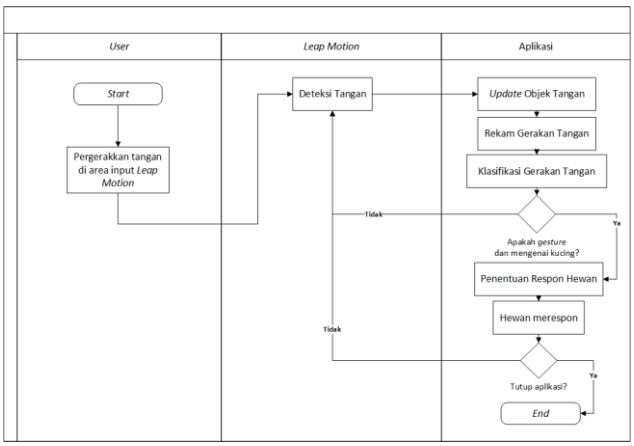
B. Desain Sistem

Secara garis besar, aplikasi ini menerima input dari *Leap motion* yang kemudian ditangkap oleh Unity untuk menggerakan model tangan di dalam aplikasi. Ketika model tangan itu berinteraksi dengan model kucing, maka model kucing tersebut akan memberikan respon sesuai dengan perhitungan yang dilakukan. Gambar 2 menjelaskan pembagian sistem aplikasi ini menjadi 5 bagian oleh penulis, meliputi pendeteksian gerakan tangan, klasifikasi gerakan tangan, klasifikasi respon hewan, animasi respon, dan *collision* (sentuhan model tangan dengan model hewan).



Gambar 2. Sistem Arsitektur *VirtuaPet*

Dari sistem arsitektur yang telah diperlihatkan pada gambar 2, gambar 3 adalah diagram alur dari aplikasi. Alur dimulai dari pengguna melakukan gerakan tangan pada area *input Leap motion*. *Leap motion* mendeteksi gerakan tangan kemudian diteruskan masuk ke Unity untuk menggerakkan model tangan. Model tangan yang bergerak direkam pergerakannya. Pergerakan kemudian diklasifikasikan untuk didapat jenis gerakannya. Gerakan kemudian ditentukan apakah mengenai model kucing atau tidak. Jika mengenai, maka klasifikasi respon dilakukan berdasarkan jenis dan kecepatan gerakan tangan, beserta lokasi sentuhan. Hasil klasifikasi diteruskan menjadi animasi respon pada model kucing, dan proses diulang kembali dengan *Leap motion* mendeteksi adanya gerakan tangan.



Gambar 3. Diagram Alur *VirtuaPet*

1) Pendeteksian Gerakan Tangan

Hal yang perlu dilakukan pada bagian ini adalah mengatur hubungan antar *Leap motion* dengan Unity agar Unity dapat menerima input dari *Leap motion* dan menggerakan model tangan di dalam aplikasi.

Untuk melakukan kedua hal tersebut, penulis menyiapkan 2 buat Asset Package untuk dimasukkan ke dalam aplikasi. Dua buah Asset Package tersebut adalah *Leap motion* Core Asset Orion 4.1.4. dan *Leap motion* Hand Module 2.0.0. Kedua Asset Package ini dapat diunduh langsung dari halaman *website Leap motion* Developer. Kedua *package* tersebut memiliki fungsi yang berbeda. Core Asset adalah *asset* utama yang mengatur semua input dari *Leap motion*, menyiapkan model tangan, dan menggerakannya. Sedangkan Hand Module merupakan *asset* tambahan untuk memberi tambahan model tangan yang dapat digunakan.

Ketika sebuah *scene* dibuat, Unity sudah menyiapkan 2 buat objek utama, yaitu *lighting* (pencahayaan) dan kamera. Kamera tersebut bertujuan untuk memberikan tampilan apa yang terjadi di dalam aplikasi berdasarkan apa yang ditangkap oleh kamera itu sendiri. Aplikasi menggunakan setting *first person viewpoint*, memberikan kesan bahwa apa yang ditampilkan oleh aplikasi tersebut adalah apa yang dilihat langsung oleh mata di dalam ruang aplikasi tersebut. Dengan setting demikian, maka model tangan akan menjadi anak dari objek kamera dengan jarak yang ditentukan, agar ketika model tangan yang bergerak dilihat sesuai dengan ketika melihat tangan sendiri bergerak di dunia nyata.

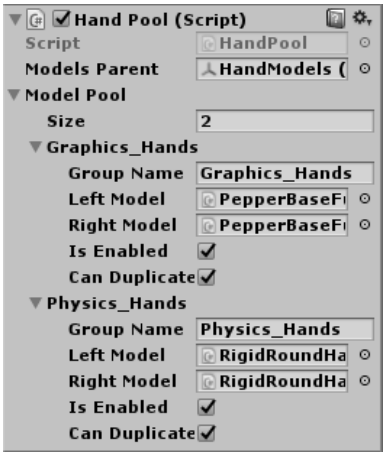
Selain mengatur model tangan, penulis juga memberikan tambahan objek sebagai tanah tempat model kucing nantinya berpijak. Untuk itu, penulis menggunakan plane sebagai tanah/landasan dan *box* sebagai tempat berpijak kucing tersebut. *Box* diletakkan diatas *plane*, membuat tempat model kucing lebih tinggi, dengan tujuan memudahkan model tangan untuk menggapainya. Semua objek yang tersebut kemudian dijadikan satu dalam sebuah objek *parent* yang diberi nama World.

Susunan akhir hirarki objek yang penulis gunakan di

dalam *scene* aplikasi tertera pada Gambar 4.

Gambar 4. Hirarki Objek pada Scene

Karena penulis menggunakan model tangan yang berbeda, maka salah satu *script* perlu diperbarui. *Script* tersebut adalah Hand Pool yang terletak di objek LeapHandController. Variabel yang diperbarui terletak pada Graphics_Hands, pada bagian Left Model dan Right Model, yang kemudian di isi dengan model yang ingin digunakan. Gambar 5 menunjukan hasil perubahan Hand Pool.

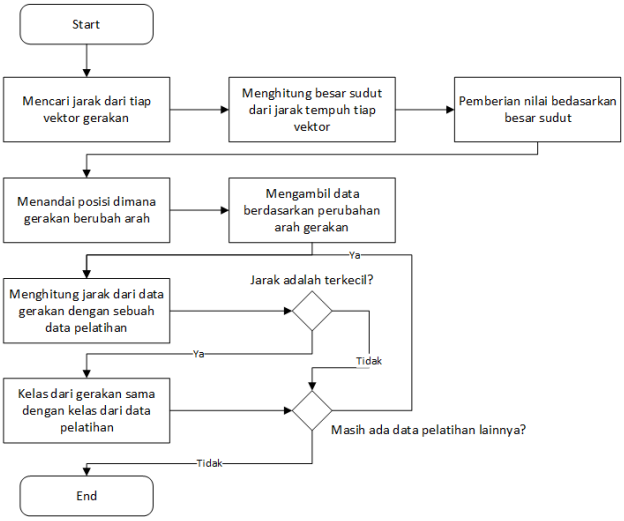


Gambar 5. Properties dari Hand Pool

Setelah *scene* selesai di *setup*, maka hal berikutnya adalah membuat *script* yang diletakkan di objek RigidRoundHand_R. *Script* ini yang bertugas untuk mencatat koordinat pergerakan tangan secara *realtime*. Catatan tersebut yang kemudian akan digunakan untuk melakukan klasifikasi gerakan tangan dan menghitung kecepatan model tangan.

2) Klasifikasi Gerakan Tangan

Klasifikasi gerakan tangan yang digunakan adalah Dynamic Time Warping (DTW). Sebelum data pergerakan model tangan tersebut diklasifikasi, data tersebut perlu diolah agar siap dimasukkan ke perhitungan klasifikasi. Gambar 6 menjelaskan rangkaian proses pada bagian ini.



Gambar 6. Diagram Alur Klasifikasi Gerakan Tangan

Pengolahan dilakukan dengan alasan DTW memiliki kelemahan bahwa jika jarak per *point* dari tiap gerakan berbeda jauh dengan apa yang ada di data pelatihan, maka DTW akan memberikan klasifikasi yang salah, meskipun pada kenyataannya gerakan yang dilakukan adalah benar. Pengolahan ini didasarkan pada jenis gerakan yang ingin ditangkap, yaitu menepuk dan mengelus. Kedua gerakan ini tergolong sederhana karena masing-masing bergerak lurus satu arah, sejajar pada sumbu Kartesius tertentu. Disini, gerakan menepuk bergerak sejajar pada *y-axis*, sedangkan mengelus bergerak sejajar selain *y-axis*. Dengan prinsip ini, data yang digunakan bukan lagi berupa jarak per *point* dari gerakan, tetapi sudut per *point* dari gerakan. Perhitungan sudut ini dilakukan dengan melakukan perhitungan *inverse tangent* terhadap *y-axis*.

Sudut yang didapat dari perhitungan tersebut kemudian diolah berdasarkan jarak sudut tersebut terhadap *y-axis*. Gerakan yang mendekati *y-axis*, positif maupun negatif, mempunyai nilai mendekati 0. Sedangkan gerakan yang mengelus akan cenderung mendekati nilai 90. Selain kedua gerakan tersebut, penulis juga memberikan sebuah gerakan, yaitu gerakan diam. Penambahan ini dilakukan karena tangan yang ‘diam’ oleh *Leap motion* akan tetap terdeteksi bergerak meskipun dengan nilai sangat kecil. Pendeteksian ini menyebabkan tangan tetap memberikan pergerakan, dimana yang seharusnya adalah diam. Penambahan ini dilakukan ini dengan memberikan *threshold* pada jarak per *point* dari gerakan. Jika jarak kurang dari *threshold*, maka jarak sama dengan 0. Jarak ini diberi nilai -90, sebagai nilai negatif.

Selain perhitungan sudut, pemotongan perlu dilakukan berdasarkan tiap titik balik dari gerakan. Gerakan menepuk mempunyai 3 titik balik, dimulai dari titik balik atas, titik balik bawah, dan kembali ke titik balik atas. Ketika kondisi ini terpenuhi, maka posisi dari titik balik atas awal hingga titik balik atas akhir akan diambil untuk nantinya diklasifikasikan. Hal ini juga berlaku pada gerakan

mengelus. Dengan diketahui titik balik awal dan titik balik akhir, maka kecepatan juga bisa dihitung sebagai selisih waktu titik balik akhir dengan titik balik awal.

Dengan pengolahan tersebut, klasifikasi dapat dilakukan dengan menggunakan catatan nilai dari sudut gerakan. Penulis juga menyiapkan data pelatihan yang berisikan 3 tipe data dengan masing-masing klasifikasi *vertical*, *horizontal*, dan *idle*. Pemberian klasifikasi ini dikarenakan gerakan yang sejajar dengan sumbu pada Kartersius. Gerakan mengelus diklasifikasikan sebagai *horizontal*. Sedangkan gerakan menepuk diklasifikasi sebagai *vertical*. Hasil perhitungan, yaitu *distance* akan digunakan untuk menentukan manakah dari data pelatihan yang memiliki jarak terdekat dengan *input* gerakan. Hasil klasifikasi, yang berupa jenis gerakan tangan, akan disimpan untuk digunakan pada klasifikasi respon.

3) Collision

Untuk mengetahui apakah model tangan menyentuh model kucing, maka *collider* ditambahkan pada model tangan dan model kucing. Masing-masing model dan ukuran collider disesuaikan dengan ukuran model. *Collider* untuk model tangan telah disediakan oleh *Leap motion*, yaitu didalam asset. Untuk bagian kepala, penulis menggunakan *Sphere Collider*, sedangkan untuk bagian badan, digunakan *Capsule Collider*. *Collider* inilah yang nantinya secara otomatis mendeteksi apakah sebuah *collision* terjadi.

4) Klasifikasi Respon

Perhitungan klasifikasi ini dilakukan terpisah karena klasifikasi hanya akan berjalan ketika sebuah gerakan menyentuh model kucing. Ketika hal itu terjadi, maka jenis gerakan, kecepatan gerakan, dan lokasi sentuhan akan diambil untuk diolah kembali untuk menentukan respon dari model kucing.

Perhitungan dari klasifikasi respon menggunakan Naïve Bayes. Pada metode ini, perhitungan dilakukan dengan membandingkan kelas manakah yang memiliki probabilitas *posterior* tertinggi. Probabilitas *posterior* didapat dari hasil perkalian semua *likehood* tiap atribut dengan probabilitas *prior* dari kelas tersebut. Pada hasil pengumpulan data, terdapat 3 kelas, yaitu diam, menoleh, dan menjauh. Probabilitas *prior* didapat dari hasil bagi jumlah munculnya kelas tersebut terhadap total keseluruhan data. Jumlah kemunculan kelas diam, menoleh dan menepuk adalah 7, 15 dan 10. Total keseluruhan data adalah 32. Dengan demikian, perhitungan probabilitas *prior* dapat dilakukan sebagai contoh, untuk kelas diam, probabilitas *prior* adalah 0,219, didapat dari hasil bagi 7 dengan 32. Tabel II. adalah daftar probabilitas *prior* untuk setiap kelas.

Perhitungan *likehood* didapat dari perbandingan jumlah kemunculan nilai dari atribut terhadap sebuah kelas, dengan jumlah data di kelas tersebut. Masing-masing dari kelas mempunyai 3 atribut, yaitu lokasi sentuhan, jenis gerakan dan kecepatan gerakan. Lokasi sentuhan mempunyai 2 nilai yaitu kepala dan punggung. Jenis gerakan mempunyai 2 nilai yaitu elusan dan tepukan. Kecepatan gerakan mempunyai 2 nilai cepat dan lambat. Sebagai contoh, *likehood* jenis gerakan elusan terhadap respon diam dengan jumlah data jenis gerakan elusan terhadap respon diam yaitu 4 data dan jumlah respon diam yaitu 7 data, memberikan hasil 0,571. Tabel III. adalah hasil perhitungan setiap *likehood*.

TABEL II
DAFTAR PROBABILITAS PRIOR SETIAP KELAS

P(Y)	Diam	Menoleh	Menjauh
	0.219	0.469	0.313

TABEL III
DAFTAR LIKEHOOD

P(X|Y)	Diam	Menoleh	Menjauh
Kepala	0.571	0.400	0.600
Punggung	0.429	0.600	0.400
Elusan	0.571	0.467	0.500
Tepukan	0.429	0.533	0.500
Cepat	0.143	0.600	0.600
Lambat	0.857	0.400	0.400

Setelah *likehood* dan probabilitas *prior* untuk setiap kelas diketahui, probabilitas *posterior* dapat mulai dilakukan. Sebagai contoh, terdapat sebuah kondisi dimana jenis gerakan adalah elusan dengan kecepatan lambat dan mengenai bagian kepala. Probabilitas *posterior* yang dihitung adalah 3 karena terdapat 3 jenis respon. Probabilitas *posterior* untuk respon diam didapat dari perkalian *likehood* gerakan elusan terhadap respon diam, *likehood* kecepatan lambat terhadap respon diam, *likehood* lokasi kepala terhadap respon diam, dan probabilitas *prior* untuk kelas diam. Hasil dari perhitungan tersebut adalah 0,571 x 0,571 x 0,857 x 0,219 = 0,061. Dengan cara yang sama, didapatkan probabilitas *posterior* untuk respon menoleh adalah 0,035 dan probabilitas *posterior* untuk respon menjauh adalah 0,038. Dengan demikian, karena respon diam memiliki probabilitas *posterior* tertinggi, maka respon yang diberikan untuk gerakan mengelus dengan kecepatan lambat dan mengenai kepala adalah diam.

5) Animasi Respon

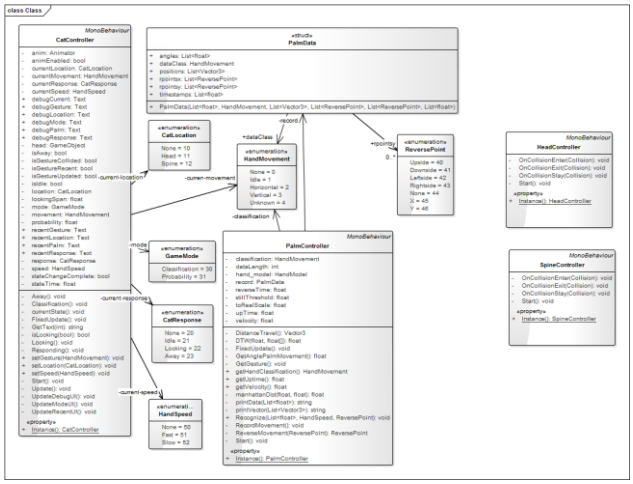
Animasi respon merupakan bagian terakhir dari desain sistem dari rancang bangun aplikasi ini. Penulis menggunakan *asset package* bernama Cu Cat 2, yang bisa dibeli pada Unity Asset Store. Aset ini sudah memiliki *file-file* yang dibutuhkan untuk membangun model kucing beserta animasinya. Setiap respon akan dibuat dengan memanipulasi model 3D menggunakan *Animation Controller*. Di dalam *controller* tersebut, terdapat banyak state, masing-masing berisikan animasi pada posisi tertentu. Perpindahan animasi dari kondisi satu ke kondisi lainnya dapat dilakukan dengan menghubungkan kedua kondisi menggunakan *Animation Transition*. Transisi tersebut dapat

diatur dengan memberi parameter untuk menentukan apakah transisi dilakukan atau tidak.

Seperti yang telah disebutkan sebelumnya, respon untuk model kucing terbagi menjadi 3. Masing-masing dari respon diwakili dengan state yang ada di dalam *controller*, pengecualian untuk respon menoleh. Penulis menggunakan respon diam sebagai posisi awal baik ketika tidak ada respon maupun respon dari hasil klasifikasi adalah diam. Ketika hasil klasifikasi bukan diam, maka kondisi akan berubah hingga animasi selesai dijalankan dan kembali ke posisi diam. Dengan demikian, parameter yang digunakan untuk menentukan transisi adalah satu, yaitu parameter diam dengan nilai *true* atau *false*. Sehingga ketika respon yang akan dijalankan bukan diam, maka parameter diam akan menjadi *false*, dan animasi akan bertransisi ke state sesuai dengan respon yang akan dijalankan.

6) Class Diagram

Gambar 7 merupakan *class diagram* dari rancang bangun aplikasi ini, dimana penulis menggunakan total 4 *class*, 6 *enumeration*, dan 1 *struct*.



Gambar 7. Class Diagram VirtuaPet

Pada gambar 7, klasifikasi gerakan tangan akan dilakukan oleh *class* PalmController dengan dua *method* utama, yaitu *method* RecordMovement untuk mencatat posisi model tangan dan *method* GetGesture untuk mendeteksi gerakan. Pendeteksian *collision* antara objek tangan dengan objek kucing akan dilakukan oleh *class* SpineController untuk bagian punggung dan *class* HeadController untuk bagian kepala, yang masing-masing mempunyai 3 *method* utama, yaitu OnCollisionEnter untuk mendeteksi awal *collision*, OnCollisionStay untuk mendeteksi *collision* yang sedang berlangsung, dan OnCollisionExit untuk mendeteksi *collision* sudah tidak terjadi. Pengolahan klasifikasi respon akan dilakukan oleh *class* CatController dengan dua *method* utama yaitu Classification untuk menentukan respon model kucing dan Responding untuk menjalankan animasi respon model kucing.

IV. PENGUJIAN

Pada bab ini, penulis akan melakukan pengujian hasil dari rancang bangun yang telah dibuat. Pengujian ini dibagi menjadi dua bagian, yaitu pengujian pada klasifikasi gerakan tangan dan klasifikasi respon hewan. Kedua pengujian ini bertujuan untuk mengukur akurasi dari hasil klasifikasi.

A. Pengujian Klasifikasi Gerakan Tangan

Pengujian ini bertujuan untuk mengukur seberapa besar akurasi dari hasil klasifikasi gerakan tangan. Pengujian ini dilakukan secara personal oleh penulis sendiri. Pengujian ini dilakukan dengan memberikan serangkaian set gerakan pada aplikasi. Setiap set gerakan akan dicatat berserta hasil klasifikasinya dan dibandingkan dengan hasil sebenarnya. Semua hasil perhitungan tersebut kemudian dihitung total akurasinya dalam bentuk persentase. Jumlah total gerakan yang diuji adalah 30 gerakan. Karena terdapat 2 jenis gerakan yang menjadi hasil klasifikasi, maka masing jenis gerakan akan dilakukan sebanyak 15 kali.

TABEL IV
HASIL PENGUJIAN KLASIFIKASI GERAKAN TANGAN

No	Jenis Gerakan	Nilai Jarak terhadap Data Pelatihan			Hasil Klasifikasi
		Vertikal	Horizontal	Diam	
1	Mengelus	716,326	73,631	1616,326	Horizontal
2	Mengelus	609,409	123,170	1509,410	Horizontal
3	Mengelus	785,953	22,809	1685,953	Horizontal
4	Mengelus	806,351	46,131	1706,351	Horizontal
5	Mengelus	703,015	88,435	1603,014	Horizontal
6	Mengelus	557,432	135,860	1457,432	Horizontal
7	Mengelus	692,153	77,312	1592,15	Horizontal
8	Mengelus	763,067	68,601	1663,067	Horizontal
9	Mengelus	812,563	29,011	1712,563	Horizontal
10	Mengelus	769,001	37,902	1669,002	Horizontal
11	Mengelus	858,648	10,253	1758,64	Horizontal
12	Mengelus	716,185	67,025	1616,186	Horizontal
13	Mengelus	519,684	133,651	1419,684	Horizontal
14	Mengelus	825,059	47,981	1725,059	Horizontal
15	Mengelus	658,237	108,475	1558,238	Horizontal
16	Menepuk	156,655	613,794	1056,655	Vertikal
17	Menepuk	200,540	595,764	1100,540	Vertikal
18	Menepuk	207,830	525,265	1107,839	Vertikal
19	Menepuk	222,785	559,352	1122,785	Vertikal
20	Menepuk	272,221	296,645	1172,221	Vertikal
21	Menepuk	240,637	554,708	1140,637	Vertikal
22	Menepuk	260,985	511,057	1160,986	Vertikal
23	Menepuk	269,819	533,661	1169,818	Vertikal
24	Menepuk	284,594	532,354	1184,594	Vertikal
25	Menepuk	243,858	536,476	1143,859	Vertikal

26	Menepuk	276,653	531,963	1176,653	Vertikal
27	Menepuk	202,541	563,490	1102,542	Vertikal
28	Menepuk	112,130	689,029	1012,131	Vertikal
29	Menepuk	130,101	685,280	1030,101	Vertikal
30	Menepuk	290,912	470,364	1190,912	Vertikal

Hasil pengujian yang tertera pada Tabel IV menunjukan bahwa semua gerakan mengelus memberikan hasil klasifikasi horizontal sedangkan semua gerakan menepuk memberikan hasil klasifikasi vertikal. Dengan demikian, hasil akurasi dari klasifikasi gerakan tangan adalah 100%.

B. Pengujian Klasifikasi Respon Hewan

Pengujian ini bertujuan untuk mengukur akurasi dari hasil klasifikasi respon hewan terhadap hasil input yang diberikan. Pengujian ini juga dilakukan secara personal oleh penulis sendiri. Pengujian ini dilakukan dengan memberikan serangkaian input yang nantinya akan dihitung perhitungan yang digunakan oleh aplikasi, yaitu Naïve Bayes. Setiap rangkaian input akan dicatat berserta hasil klasifikasinya dan dibandingkan dengan hasil sebenarnya. Semua hasil perhitungan tersebut akan dihitung total akurasinya dalam bentuk persentase. Penulis mengambil 33% dari jumlah data pelatihan yang penulis punya sebagai data pengujian.

TABEL V
HASIL PENGUJIAN KLASIFIKASI RESPON DENGAN 33% DATA PENGUJIAN

No	Lokasi Penerima	Jenis Gerakan	Kecepatan Gerakan	Respon	Hasil Klasifikasi
1	Kepala	Elusan	Cepat	Menjauh	Menjauh
2	Kepala	Elusan	Lambat	Menjauh	Diam
3	Kepala	Tepukan	Cepat	Menjauh	Menoleh
4	Kepala	Tepukan	Lambat	Menjauh	Diam
5	Punggung	Elusan	Cepat	Menjauh	Menoleh
6	Punggung	Elusan	Lambat	Menjauh	Menoleh
7	Punggung	Tepukan	Cepat	Menoleh	Menoleh
8	Punggung	Tepukan	Lambat	Menjauh	Menoleh
9	Kepala	Elusan	Cepat	Diam	Menjauh
10	Kepala	Elusan	Lambat	Diam	Diam
11	Kepala	Tepukan	Cepat	Menoleh	Menoleh
12	Kepala	Tepukan	Lambat	Menjauh	Diam
13	Punggung	Elusan	Cepat	Menoleh	Menoleh
14	Punggung	Elusan	Lambat	Menoleh	Menoleh
15	Punggung	Tepukan	Cepat	Menoleh	Menoleh
16	Punggung	Tepukan	Lambat	Menjauh	Menoleh

Hasil pengujian pada Tabel V menunjukan tingkat akurasi dari klasifikasi respon hewan adalah 43,75%. Karena akurasi yang kecil, maka penulis melakukan pengujian kedua, dengan melibatkan 100% data pelatihan dan data pengujian untuk melihat apakah model klasifikasi mengalami *underfitting*.

TABEL VI
HASIL PENGUJIAN KLASIFIKASI RESPON DENGAN 100% DATA PENGUJIAN DAN PELATIHAN

No	Lokasi Gerakan	Jenis Gerakan	Kecepatan Gerakan	Respon	Hasil Klasifikasi
1	Kepala	Elusan	Cepat	Menoleh	Menjauh
2	Kepala	Elusan	Lambat	Menjauh	Diam
3	Kepala	Tepukan	Cepat	Menoleh	Menjauh
4	Kepala	Tepukan	Lambat	Menjauh	Menjauh
5	Punggung	Elusan	Cepat	Menjauh	Menoleh
6	Punggung	Elusan	Lambat	Menoleh	Menoleh
7	Punggung	Tepukan	Cepat	Menoleh	Menoleh
8	Punggung	Tepukan	Lambat	Menoleh	Menoleh
9	Kepala	Elusan	Cepat	Menjauh	Menjauh
10	Kepala	Elusan	Lambat	Menoleh	Diam
11	Kepala	Tepukan	Cepat	Menjauh	Menjauh
12	Kepala	Tepukan	Lambat	Menjauh	Menjauh
13	Punggung	Elusan	Cepat	Menoleh	Menoleh
14	Punggung	Elusan	Lambat	Menoleh	Menoleh
15	Punggung	Tepukan	Cepat	Menoleh	Menoleh
16	Punggung	Tepukan	Lambat	Menoleh	Menoleh
17	Kepala	Elusan	Cepat	Diam	Menjauh
18	Kepala	Elusan	Lambat	Diam	Diam
19	Kepala	Tepukan	Cepat	Menoleh	Menjauh
20	Kepala	Tepukan	Lambat	Diam	Menjauh
21	Punggung	Elusan	Cepat	Menoleh	Menoleh
22	Punggung	Elusan	Lambat	Menjauh	Menoleh
23	Punggung	Tepukan	Cepat	Menjauh	Menoleh
24	Punggung	Tepukan	Lambat	Diam	Menoleh
25	Kepala	Elusan	Cepat	Menoleh	Menjauh
26	Kepala	Elusan	Lambat	Diam	Diam
27	Kepala	Tepukan	Cepat	Menjauh	Menjauh
28	Kepala	Tepukan	Lambat	Menoleh	Menjauh
29	Punggung	Elusan	Cepat	Menjauh	Menoleh
30	Punggung	Elusan	Lambat	Diam	Menoleh
31	Punggung	Tepukan	Cepat	Menoleh	Menoleh
32	Punggung	Tepukan	Lambat	Diam	Menoleh
33	Kepala	Elusan	Cepat	Menjauh	Menjauh
34	Kepala	Elusan	Lambat	Menjauh	Diam
35	Kepala	Tepukan	Cepat	Menjauh	Menjauh
36	Kepala	Tepukan	Lambat	Menjauh	Menjauh
37	Punggung	Elusan	Cepat	Menjauh	Menoleh
38	Punggung	Elusan	Lambat	Menjauh	Menoleh
39	Punggung	Tepukan	Cepat	Menoleh	Menoleh
40	Punggung	Tepukan	Lambat	Menjauh	Menoleh
41	Kepala	Elusan	Cepat	Diam	Menjauh

42	Kepala	Elusan	Lambat	Diam	Diam
43	Kepala	Tepukan	Cepat	Menoleh	Menjauh
44	Kepala	Tepukan	Lambat	Menjauh	Menjauh
45	Punggung	Elusan	Cepat	Menoleh	Menoleh
46	Punggung	Elusan	Lambat	Menoleh	Menoleh
47	Punggung	Tepukan	Cepat	Menoleh	Menoleh
48	Punggung	Tepukan	Lambat	Menjauh	Menoleh

Hasil pengujian pada Tabel VI menunjukan akurasi dari klasifikasi untuk respon hewan adalah 52.08%. Dari nilai akurasi tersebut, dapat disimpulkan bahwa model klasifikasi mengalami *underfitting*. Hal ini disebabkan karena data pelatihan yang kurang sehingga membuat model klasifikasi belum benar.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Bedasarkan hasil pengujian yang dilakukan pada rancang bangun aplikasi ini, diambilah kesimpulan sebagai berikut,

1) Aplikasi dapat mengklasifikasikan gerakan tangan yang terdeteksi dengan akurasi sebesar 100%.
2) Aplikasi dapat memberi hasil klasifikasi respon hewan dengan akurasi sebesar 43.75%.
3) Data pelatihan mengalami *underfitting*, dikarenakan jumlah sampel data yang masih kurang.

B. Saran

Bedasarkan kajian dan penelitian yang telah dilakukan oleh penulis, penulis memberikan saran yaitu,

1) Penambahan sampel untuk data pelatihan hingga data pelatihan tidak mengalami underfitting.
2) Penggunaan metode klasifikasi lainnya karena atribut respon yang bersifat kontinu.
3) Penambahan model gerakan tangan, respon, atau faktor pengukuran lainnya seperti kesehatan atau kondisi emosi yang berpengaruh terhadap hasil klasifikasi.

DAFTAR PUSTAKA

[1] American Humane Association. Keeping Pets (Dogs and Cats) in Homes. (2015, Maret 22). Tersedia: http://www.americanhumane.org/aha-petsmart-retention-study-phase-1.pdf.

[2] Becker. (2012). New Study Examines Why Some People Don't Have Pets.

[3] Bouvier, P., Sorbier, F. D., Chaudeyrac, P., & Biri, V.. (2008). Cross Benefits between Virtual Reality and Games. Dans International Conference and Industry Symposium on Computer Games; Animation, Multimedia, IPTV, Edutainment and Security (CGAT'08).

[4] History of Virtual Pet. (2015, Maret 31). Tersedia: http://www.encyclopediaonpets.com/history%20and%20beginig%20of%20virtual%20pet_main.php.

[5] Miller, D. (2013). Introduction to Collective Behavior & Collective Action (3rd ed., pp. 203-204). Waveland Press.

[6] Bandai. (2010). Tamagotchi iD L March 19th sale!. Bandai: 2010.

[7] Colgan, Alex. (2014). How Does the *Leap motion* Controller Work?.

[8] *Leap motion* Facebook Statistic. (2015, Mei 4). Tersedia: http://www.socialbakers.com/statistics/facebook/pages/detail/212179268798946-leap-motion.

[9] Emrich, Tom. (2013). *Leap motion* Celebrates One Million App Downloads in 3-weeks.

[10] Szeliski, Richard. (2010). Computer Vision: Algorithms and Applications. Springer.

[11] Petty, John. (2014). What is gesture recognition? Gesture recognition defined. (2015, Maret 22) Tersedia: http://www.marxentlabs.com/what-is-gesture-recognition-defined/

[12] Tsikorpova, Elena.. (2016, Mei 23). Dynamic Time Warping Algorithm for Gene Expression Time Series. Tersedia: www.psb.ugent.be/cbd/papers/gentxwarper/DTWAlgorithm.ppt

[13] Han, J., Kamber, M., & Pei, J. (2012). *Data mining* Concepts and Techniques 3rd Edition. Amsterdam: Elsevier/Morgan Kaufmann.

[14] Robertson, A.. (2015). Unity officially releases its new game engine: Unity 5.

[15] MSFX. (2011). Getting Started with Unity - Colliders & UnityScript.