Aplikasi Deteksi Orang Jatuh dengan Memanfaatkan Kinect

Jonathan Tjitrokusmo[1], Liliana[2], Kartika Gunadi[3]
Program Studi Teknik Informatika
Fakultas Teknologi Industri, UK Petra
Jln. Siwalankerto 121-131 Surabaya 60236
Telp. (031)-2983455, Fax. (031)-8417658
m26413041@john.petra.ac.id[1], lilian@petra.ac.id[2], kgunadi@petra.ac.id[3]

ABSTRAK

Karya tulis ini menyajikan metode untuk mendeteksi jatuh, karena jatuh merupakan masalah yang serius dan hal ini seringkali mengakibatkan cedera yang dapat membahayakan keselamatan orang tersebut. Oleh karena itu, pendeteksian orang jatuh sangatlah diperlukan.

Alat yang digunakan adalah *Kinect*, yang dapat juga digunakan untuk mendeteksi orang. Pendeteksiannya menggunakan bantuan *Microsoft Kinect SDK*. *Kinect* ini dapat mendeteksi orang yang berada di depannya dan memprosesnya dengan membuat *skeleton* dari orang tersebut.

Metode yang digunakan adalah mendapatkan sekumpulan data posisi dari orang tersebut. Setelah mendapatkan data tersebut, memperhitungkan kecepatan perubahan posisinya dengan rumus yang telah dimiliki. Data yang didapatkan dipilah untuk membedakan aktivitas yang dilakukan. Ketika terdeteksi jatuh, aplikasi dapat memberikan peringatan.

Kata Kunci: *Kinect, Microsoft Kinect SDK, skeleton*, jatuh, deteksi jatuh

ABSTRACT

This paper describes method for fall detection, because fall is a serious problem and often resulting to injury, which can endanger the safety of the person. Therefore, falling detection is crucially needed.

The device that being used is Kinect, which also could be used to detect people. The detection is using the help of Microsoft Kinect SDK. Kinect can detect a person in front of it and processing it to create a skeleton of the person.

The method which being used is to get a set of data on the position of the person. Next, the rate of change in position would be calculated with the available formulae. The data obtained would be selected, in order to distinguish the activities undertaken. When fall is detected, the application can provide the alert.

Keywords: *Kinect, Microsoft Kinect SDK, skeleton, fall, fall detection*

1. PENDAHULUAN

Jatuh bukanlah hal yang baru, tetapi masih banyak orang menganggap remeh jatuh. Jatuh dapat diartikan sebagai gerakan yang tidak sengaja yang membuat tubuh turun menuju lantai atau beberapa tingkat lebih rendah [1]. Jatuh ini biasanya berlangsung selama satu sampai dua detik [8]. Menurut *Centers for Disease Control and Prevention*, satu dari lima orang jatuh menyebabkan cedera yang serius seperti patah tulang atau cedera di kepala dan jatuh merupakan penyebab utama kematian bagi orang berusia lanjut [3]. Orang tua adalah yang paling perlu diwaspadai karena satu dari tiga orang tua terjatuh setiap tahunnya. Jatuh memang lebih sering terjadi pada anak-anak, tetapi dibandingkan dengan anak-anak, orang tua yang jatuh, sepuluh kali lebih mungkin untuk mendapatkan perawatan di rumah sakit dan delapan kali lebih mungkin untuk meninggal [4]. Jadi, jatuh tidak dapat dianggap remeh, meskipun tidak semua jatuh akan berakibat fatal, memberikan pertolongan pertama sesegera mungkin adalah hal yang sangat penting [6].

2. TINJAUAN PUSTAKA

Aplikasi yang dibuat ini memanfaatkan Kinect untuk mendeteksi orang tersebut. Kinect adalah salah satu produk Microsoft untuk Xbox 360 yang digunakan agar user dapat mengontrol dan berinteraksi dengan konsol. Kinect ini menggunakan kamera RGB-D, yang dapat menangkap gambar RGB seperti kamera biasa dan dapat menangkap Depth atau jarak orang tersebut. Menangkap Depth ini adalah sebuah kelebihan dari Kinect dan dengan menangkap Depth, maka gambar yang diterima memiliki posisi yang jelas. Posisi yang dimaksudkan adalah letak orang tersebut dalam sumbu X, sumbu Y, dan sumbu Z. Jarak minimum agar orang yang berada di depannya terdeteksi adalah 0.6 meter dan jarak maksimumnya mencapai 4 sampai 5 meter, dan pada 30 *frame* per detik [6]. Untuk memproses data yang ada di Kinect, diperlukan SDK Kinect. Memproses data yang dimaksud adalah seperti melakukan perhitungan dan pengolahan terhadap sumbu X, sumbu Y, dan sumbu Z yang telah ditangkap oleh Kinect.

Software Development Kit (SDK) adalah suatu set alat pengembangan perangkat lunak yang digunakan untuk mengembangkan atau membuat aplikasi untuk paket software tertentu. SDK Microsoft Kinect digunakan untuk mendapatkan dan mengolah data yang ada pada Kinect. SDK ini memungkinkan untuk mengakses data mentah dari sensor kamera Kinect. Terdapat juga Skeletal Tracking yang merupakan kemampuan untuk mendeteksi orang yang bergerak di area pendeteksian Kinect. Dengan menggunakan *skeletal tracking*, informasi yang didapatkan akan banyak bermanfaat bagi algoritma-algoritma yang berkaitan dengan *skeleton*, seperti mengakses data pada posisi kepala, pundak, tulang belakang, pinggul dan dapat mengolahnya [5].

3. METODE

Jatuh dapat terdeteksi dengan menghitung kecepatan perubahan orang tersebut pada lingkungannya [2]. Oleh karena itu, dapat dilakukan perhitungan terhadap posisi orang tersebut. Posisi *skeleton* yang didapatkan dijabarkan dalam nilai sumbu X, sumbu Y, dan sumbu Z sehingga dapat menghitung lebar, tinggi dan jarak (depth) orang tersebut. Menghitung lebar, tinggi dan jarak (depth) dengan menggunakan persamaan 1, persamaan 2, dan persamaan 3 [7] :

$$Width = |Xmin - Xmax| \quad (1)$$

$$Height = |Ymin - Ymax| \quad (2)$$

$$Depth = |Zmin - Zmax| \quad (3)$$

Orang yang jatuh akan mengalami perubahan tinggi, lebar atau jarak. Biasanya tinggi dari orang tersebut akan mengalami penurunan yang drastis. Orang tersebut juga mengalami pertambahan lebar atau jarak tergantung pada arah jatuhnya. Oleh karena itu, diperlukan perhitungan terhadap kecepatan perubahan tinggi dan lebar atau jaraknya.

Menghitung kecepatan perubahan tinggi dapat dimulai dengan menghitung ketinggian orang tersebut setiap frame. Setelah itu, menghitung kecepatan perubahan tinggi setiap frame dengan membagi ketinggian dengan waktu setiap frame. Dapat dirumuskan seperti pada persamaan 4 [2]:

$$V_h = \frac{h_k - h_{k-1}}{t_k - t_{k-1}} \quad (4)$$

V_h = kecepatan perubahan tinggi.

h_k = tinggi pada *frame* ke k.

t_k = waktu pada *frame* ke k.

Setelah mendapatkan kecepatan perubahan tinggi, dengan data lebar dan jarak yang dimiliki, dicari kecepatan perubahan lebar dan jaraknya juga. Karena tidak diketahuinya jarak atau lebar yang bertambah akibat jatuh, maka dilakukan normalisasi terhadap data jarak dan lebar tersebut. Hal ini dapat dirumuskan seperti pada persamaan 5 :

$$JL = \sqrt{J^2 + L^2} \quad (5)$$

J = jarak

L = lebar

Dari data JL tersebut digunakan untuk mencari kecepatan perubahan lebar dan jarak dengan membagi JL pada setiap frame dengan waktu setiap frame. Hal ini dapat dirumuskan seperti pada persamaan 6 :

$$v_{JL} = \frac{JL_i - JL_{i-1}}{t_i - t_{i-1}} \quad (6)$$

V_{JL} = kecepatan perubahan JL

JL_i = JL pada *frame* i

t_i = waktu pada *frame* i

Dari data kecepatan perubahan tinggi dan kecepatan perubahan jarak dan lebar, dapat dilihat jika terjadi kecepatan yang mencolok. Agar dapat memastikannya maka dibuat sebuah nilai acuan. Jika kecepatan perubahan tinggi, jarak dan lebar orang tersebut melebihi nilai acuan untuk kecepatan perubahan tinggi, jarak dan lebar yang ditentukan maka kemungkinan besar terjadi jatuh, tetapi bisa jadi orang tersebut tidak jatuh, melainkan sedang tidur, duduk, berolah raga atau melakukan hal lain. Untuk lebih memastikannya lagi, dilakukan perhitungan untuk posisi kepala, tangan, dan kaki dan kecepatan perubahan posisinya.

Untuk mencari kecepatan perubahan posisi kepala dapat dilakukan rumus seperti pada persamaan 7 :

$$v_{kepala} = \frac{kepala_i - kepala_{i-1}}{t_i - t_{i-1}} \quad (7)$$

V_{Kepala} = kecepatan perubahan posisi kepala

$Kepala_i$ = posisi kepala pada *frame* i

t_i = waktu pada *frame* i

Untuk mencari kecepatan perubahan posisi tangan dan kaki dapat dilakukan rumus seperti pada persamaan 8 dan persamaan 9:

$$v_{tangan} = \frac{tangan_i - tangan_{i-1}}{t_i - t_{i-1}} \quad (8)$$

V_{tangan} = kecepatan perubahan posisi tangan

$Tangan_i$ = posisi tangan pada *frame* i

T_i = waktu pada *frame* i

$$v_{kaki} = \frac{kaki_i - kaki_{i-1}}{t_i - t_{i-1}} \quad (9)$$

V_{kaki} = kecepatan perubahan posisi kaki

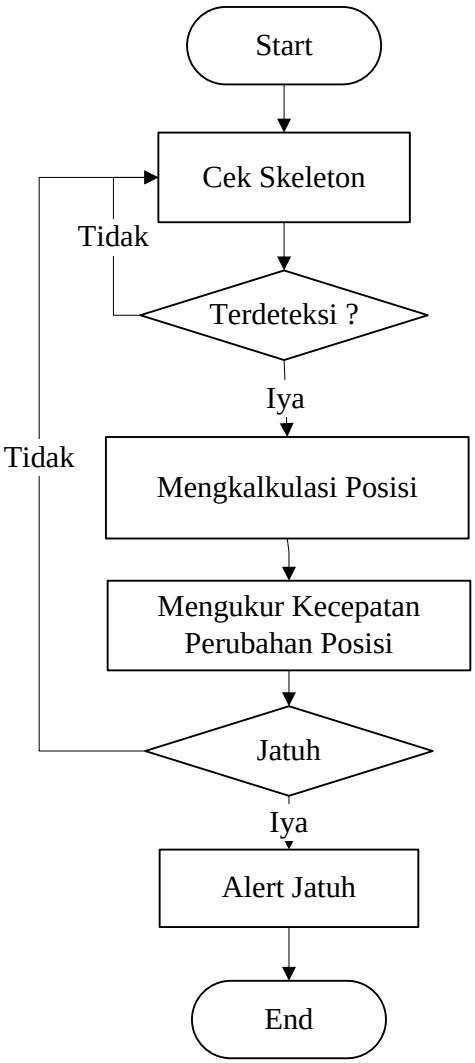
$kaki_i$ = posisi kaki pada *frame* i

T_i = waktu pada *frame* i

Untuk kecepatan perubahan posisi tangan dan kaki ini dilakukan perhitungan pada kedua tangan dan kaki.

Setelah didapatkan kecepatan perubahan posisi *height, width, depth,* kepala, tangan dan kaki, data kecepatan perubahan posisi tersebut dibandingan dengan nilai acuan atau nilai batasan untuk kecepatan perubahan posisi *height, width, depth,* kepala, tangan, dan kaki. Karena ketika jatuh terdapat sebuah perubahan posisi dan hal tersebut terjadi cukup cepat, maka terdapat sebuah nilai kecepatan yang berbeda dengan aktivitas sehari-hari Jika kecepatan perubahan posisi-posisi tersebut melebihi nilai acuan atau nilai batasan yang telah ditentukan maka dapat disimpulkan bahwa orang tersebut terjatuh.

Untuk lebih menjelaskannya dapat menggunakan *flowchart* pada Gambar 1. Pada awalnya, dilakukan pengecekan terlebih dahulu terhadap *Kinect*, jika *Kinect* telah terhubung maka langkah selanjutnya adalah mendeteksi orang yang berada didepan *Kinect.* Jika tidak ada orang yang terdeteksi, maka proses selanjutnya tidak dapat dimulai. Setelah itu Microsoft Kinect SDK akan membuatkan skeleton-nya yang akan menyimpan data untuk diolah pada langkah-langkah berikutnya.. Jika *skeleton* sudah terbentuk, proses selanjutnya adalah mengkalkulasi posisi, seperti *height, width, depth, joint* kepala, tangan, dan kaki dari *skeleton* tersebut. Dari data posisi yang telah didapatkan, dihitung kecepatan perubahan posisinya dengan persamaan sebelumnya. Hasil yang didapatkan dibandingkan dengan nilai batasan yang telah dimiliki. Jika nilai kecepatan yang telah dikalkulasikan melebihi nilai batasan yang telah ditentukan tersebut, maka dapat disimpulkan terjadi jatuh. Jika terjadi jatuh, maka aplikasi akan mengirimkan *alert* atau peringatan.

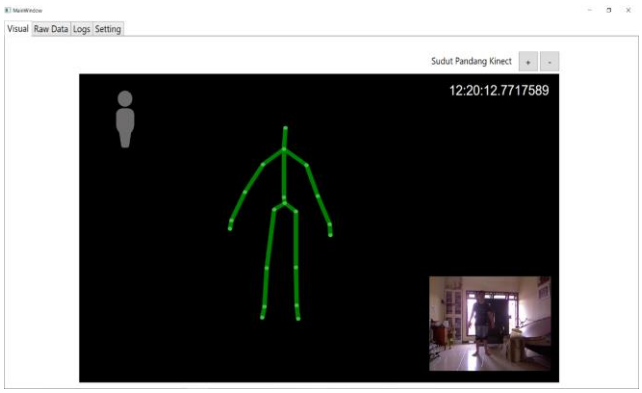


Gambar 1. *Flowchart* alur aplikasi

4. IMPLEMENTASI

Aplikasi ini membutuhkan dua *software* pendukung, yaitu WPF .Net yang menggunakan bahasa VB dan *Microsoft Kinect SDK*. Pemilihan dua *software* ini dikarenakan WPF .Net mendukung penggunaan *Microsoft Kinect SDK*, sehingga memungkinan untuk memproses hasil gambar *Kinect.*

Skeleton adalah faktor yang sangat penting pada aplikasi ini, karena data perhitungan posisi menggunakan data dari *skeleton* yang terbentuk. Data perhitungan posisi tersebut nantinya akan dilakukan untuk melakukan pengecekan jatuh. *Skeleton* yang terbentuk mengandung data posisi dari tiap *joint* yang telah digambarkan pada layar. Contoh gambaran dari *skeleton* yang terdeteksi dapat dilihat pada Gambar 2.



Gambar 2. *Skeleton* yang terbentuk

Data yang telah diekstrak dari *skeleton* tersebut digunakan untuk melakukan pengecekan jatuh. Hasil olahan data yang telah terdeteksi adalah *height, width, depth*, vh, vjl, y kepala, y tangan kanan, y tangan kiri, y kaki kanan, y kaki kiri, v kepala, v tangan kanan, v tangan kiri, v kaki kanan, dan v kaki kiri.

Height adalah ketinggian keseluruhan dari orang tersebut, sehingga jika mengangkat tangan, maka ketinggian orang tersebut dihitung sampai ke tangan yang terangkat tersebut. Dari *height* didapatkan vh, yaitu kecepatan perubahan data *height.* Vh diperlukan karena ketika jatuh, ketinggian dari orang tersebut mengalami penurunan, dan dengan menangkap ketinggian keseluruhan ini, membuat perhitungan lebih fleksibel. *Width* menyimpan lebar dari *skeleton* berdasarkan sumbu x. *Depth* menyimpan *depth* atau kedalaman dari *skeleton* berdasarkan sumbu z. Dari *depth* dan *width* ini didapatkan jl, yaitu normalisasi dari *depth* dan *width*. Dari jl, didapatkan vjl, yaitu kecepatan perubahan *width-depth* atau kedalaman dan lebar. Dengan menangkap vjl, ketika menganyunkan tangan ke depan atau ke samping, nilai *width* dan *depth* yang mengalami perubahan posisi akan terhitung menjadi satu, sehingga tidak perlu memastikan arah gerakan dari orang tersebut. Y kepala meyimpan posisi kepala pada sumbu y. Dari y kepala tersebut, didapatkan v kepala, yaitu kecepatan perubahan posisi kepala pada sumbu y. V kepala ini digunakan untuk mengukur penurunan posisi kepala atau kenaikkan posisi kepala ketika melakukan aktivitas. Y tangan kiri, y tangan kanan, y kaki kiri, dan y kaki kanan menyimpan posisi dari tangan kiri, tangan kanan, kaki kiri, dan kaki kanan pada sumbu y. Setelah itu didapatkan kecepatan perubahan posisi pada sumbu y dan didapatkan v tangan kiri, v tangan kanan, v kaki kiri, dan v kaki kanan. Kecepatan perubahan posisi dari anggota tubuh ini digunakan untuk mengkalkulasikan aktivitas yang sedang dilakukan, dan membandingkannya dengan aktivitas jatuh.

Jatuh dilakukan dengan posisi awal berdiri dan posisi akhir terbaring di lantai [8]. Hasil dari percobaan jatuh tersebut dapat dilihat pada Tabel 1 yang berisikan nomor *frame*, vh, vjl, dan v kepala. Nomor *frame* merupakan nomor dari perekaman data tersebut. Setiap nomor *frame* memiliki waktu 0,037 milidetik. Untuk v tangan kiri, v tangan kanan, v kaki kiri, dan v kaki kanan dapat dilihat pada Tabel 2, dengan aktivitas jatuh yang sama dengan Tabel 1.

Tabel 1. Pengujian jatuh menghasilkan vh, vjl dan v kepala

No. Frame	Vh	Vjl	V Kepala
1	-3	-6	4
2	-4	-5	5
3	-3	2	6
4	-3	-2	7
5	-5	4	10
6	-7	6	12
7	-6	7	15
8	-6	9	17
9	-5	10	19
10	-4	9	21
11	-3	6	23
12	-4	8	26
13	-10	14	27
14	-11	15	29
15	-11	24	31
16	-23	32	31
17	-41	29	33
18	-50	35	58
19	-47	26	59
20	-53	40	61
21	-50	37	57
22	-53	42	55
23	-56	18	57
24	-56	50	56
25	-59	48	54
26	-59	33	52
27	-61	50	50
28	-60	48	47
29	-52	44	45
30	-52	45	44
31	-55	41	43
32	-40	19	40
33	-20	1	37
34	1	1	12

Tabel 2. Data pengujian jatuh yang menghasilkan v tangan kiri, v tangan kanan, v kaki kiri, dan v kaki kanan

No. Frame	V Tangan Kiri	V Tangan Kanan	V Kaki Kiri	V Kaki Kanan
1	6	5	5	0
2	6	4	6	0
3	8	5	1	2
4	5	7	4	2
5	6	7	2	4
6	7	8	4	5
7	8	8	5	9
8	8	8	7	11
9	8	7	7	13
10	8	6	12	18
11	8	5	14	19
12	9	7	19	22
13	10	10	3	18
14	12	12	2	21
15	13	14	9	22
16	15	18	10	3
17	17	22	16	6
18	22	12	19	2
19	20	12	22	2
20	20	10	17	0
21	20	19	20	2
22	14	18	15	6
23	9	15	23	8
24	1	20	15	16
25	0	20	24	16
26	1	21	27	22
27	1	21	28	25
28	3	19	34	28
29	5	17	8	30
30	7	14	7	31
31	4	11	10	32
32	7	7	19	1
33	9	1	14	1
34	18	11	12	8

5. KESIMPULAN

Pada saat jatuh, *height*, *width*, dan *depth* mengalami perubahan. *Height* mengalami penurunan, *width* dan *depth* yang telah menjadi jl, mengalami pertambahan. Vh yang dihasilkan akan memiliki nilai negatif yang besar, karena mengalami perubahan *height* yang cukup besar dalam waktu yang singkat. Vjl juga akan memiliki nilai yang besar, karena bertambahnya *width* dan *depth*. Untuk kecepatan kepala, juga memiliki nilai yang besar, bahkan lebih besar daripada vh. Untuk kecepatan perubahan posisi tangan, pada *frame* yang sama, tangan kiri dan tangan kanan belum tentu memiliki nilai yang sama, tetapi sama-sama mengalami perubahan posisi yang cepat, tetapi nilainya tidak sebesar v kepala. Untuk kaki juga memiliki kecepatan dalam perubahan posisinya tetapi tidak sebesar v kepala, lebih mirip dengan kecepatan perubahan posisi dari tangan.

6. REFERENCES

[1] Arote, & Bhosale. 2015. Fall Detection System using Accelerometer Principals with Arduino Development Board. *International Journal of Advance Research in Computer Science and Management Studies Volume 3, Issue 9.*

[2] Bevilacqua, V., Nuzzolese, N., Barone, D., Pantaleo, M., Suma, M., D'Ambruoso, D., et al. 2014. Fall Detection in Indoor Environment with Kinect Sensor. *Innovations in Intelligent Systems and Applications (INISTA) Proceedings, 2014 IEEE International Symposium*, 319-324.

[3] *Falls Among Older Adults: An Overview*. 2016. Retrieved from Centers for Disease Control and Prevention: http://www.cdc.gov/HomeandRecreationalSafety/Falls/adultfalls.html

[4] Falls in the Elderly. 2016. Retrieved from American Family Physician: http://www.aafp.org/afp /2000/0401/p2159.html

[5] Kawatsu, C., Li, J., & Chung, C. 2013. Development of a Fall Detection System with Microsoft Kinect. *An Edition of the Presented Papers from the 1st International Conference on Robot Intelligence Technology and Applications*, 623-630.

[6] Kwolek, B., & Kepski, M. 2013. Fall Detection Using Kinect Sensor and Fall Energy Image. *In Hybrid Artificial Intelligent Systems* (pp. 294-303). Berlin: Springer Berlin Heidelberg.

[7] Mastorakis, G., & Makris, D. 2014. Fall Detection System Using Kinect's Infrared Sensor. *Journal of Real-Time Image Processing 9: 635. doi:10.1007/s11554-012-0246-9*, 635-646.

[8] Yu, X. 2008. Approaches and Principles of Fall Detection for elderly and patient. *e-health Networking, Applications and Services, 2008. HealthCom 2008. 10th International Conference on.* 42-47.