Analisis *Inverse Kinematics* (IK) Menggunakan Vektor dan Rotasi dalam Animasi Prosedural

Rafael Marchel Darma Wijaya – 13523146[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523146@std.stei.itb.ac.id, [2]rafaelmarchel.dw@gmail.com

***Abstract*—This study explores fundamental concepts in procedural animation, including distance constraint, Forward Kinematics (FK), and Inverse Kinematics (IK), along with their application in creating dynamic and realistic procedural animations. Distance constraint is a method to limit the relative distance between points in a system, ensuring that the points remain within a defined distance. Forward Kinematics (FK) is used to calculate the position and orientation of the end effector in a kinematic system based on the rotation angles of joints, while Inverse Kinematics (IK) solves the problem by calculating the joint positions required to reach a target position. Three common IK methods are explored: Cyclic Coordinate Descent (CCD), Jacobian Inverse, and FABRIK, each with its advantages and disadvantages.**

***Keywords*—Distance Constraint, Forward Kinematics, Inverse Kinematics, Procedural Animation.**

I. Pendahuluan

Animasi dapat didefinisikan sebagai suatu teknik membuat pergerakan suatu hal dalam dua dimensi atau tiga dimensi [1]. Pergerakan ini biasanya dibuat dengan serangkaian gambar yang berurutan dan ditampilkan dengan cepat sehingga menciptakan efek gerak. Setiap gambar disebut sebagai *frame* dan ditampilkan dengan satuan kecepatan *frame* per detik (*frame per second*) yang menunjukkan banyaknya gambar yang ditampilkan dalam satu detik.

Animasi prosedural (*Procedural Animation*) adalah salah satu teknik dalam menciptakan suatu animasi. Berbeda dengan teknik animasi yang biasanya menggunakan *frame*, animasi prosedural membuat animasi secara dinamis menggunakan algoritma untuk menciptakan pergerakan yang realistis. Animasi prosedural diciptakan secara *real-time* berdasarkan algoritma yang digunakan dan perubahan masukan dalam parameter algoritma. Hal ini membuat animasi prosedural dapat beradaptasi dengan faktor-faktor eksternal seperti perubahan lingkungan dan masukan dari pengguna. Oleh karena itu, aplikasi dari animasi prosedural sering ditemui dalam pembuatan permainan digital (*video game*), robot, dan *virtual reality* yang memerlukan pemrosesan pergerakan secara langsung dan adaptif. Selain itu, kelebihan dari animasi prosedural adalah kemampuan dalam membuat simulasi yang membutuhkan pergerakan kompleks seperti *physics-based simulation*.

Dalam animasi prosedural, istilah *real-time* mengacu pada kemampuan untuk membuat animasi secara langsung tanpa membutuhkan serangkaian gambar (*frame*) yang telah dibuat sebelumnya yang biasanya ditemui pada animasi tradisional. Dalam permainan digital misalnya, animasi karakter dalam permainan tersebut harus dapat beradaptasi dengan lingkungan yang selalu berubah dan masukan dari pengguna yang tidak dapat diprediksi. Atau, dalam bidang robotika, pergerakan dari robot harus menyesuaikan posisi target atau rintangan. Animasi prosedural akan selalu menghitung pergerakan dan posisi suatu objek dengan algoritma.

Salah satu teknik penting dalam animasi prosedural adalah *Inverse Kinematics* (IK). IK adalah suatu teknik untuk menghitung posisi dan orientasi dari sendi (*joints*) dalam suatu sistem yang dibentuk dari banyak sambungan sendi seperti halnya tangan atau kaki pada manusia. IK sangat penting untuk menentukan posisi ujung (*end effector*) dari sistem. IK banyak digunakan dalam animasi karakter dan robotika yang memanipulasi gerakan bagian-bagian yang saling terhubung satu sama lain dengan sendi. IK menghitung bagaimana posisi dan sudut tiap sendi dalam sistem agar *end effector* dapat mencapai posisi yang diinginkan dengan tetap mempertahankan batasan sistem [2].

Inverse Kinematics berbeda dengan teknik animasi prosedural lain yang memiliki nama yang serupa yaitu *Forward Kinematics* (FK). FK menghitung posisi dari *end effector* ketika diketahui posisi dan sudut dari semua sendi dalam suatu sistem. FK memiliki perhitungan yang jauh lebih sederhana dari IK karena IK harus melakukan perhitungan untuk menentukan orientasi-orientasi yang tidak diketahui dari sendi-sendi dalam sistem. IK menjadi lebih rumit ketika berhadapan dengan sendi yang memiliki batasan rotasi (tidak fleksibel) untuk meniru pergerakan yang lebih natural. Beberapa algoritma telah dikembangkan untuk menyelesaikan permasalahan IK seperti CCD (*Cyclic Coordinate Descent*), *Jacobian Inverse*, dan FABRIK (*Forward and Backward Reaching Inverse Kinematics*).

Hal paling penting dalam algoritma IK adalah

penggunaan vektor. Vektor menyediakan cara yang paling efisien dalam merepresentasikan posisi, arah, dan besar dari setiap komponen dalam suatu sistem. Vektor memiliki perhitungan yang jelas dalam menentukan jarak dan sudut dari setiap sendi-sendi dan *end effector*. Dalam IK, vektor digunakan dalam menghitung besar sudut yang dibentuk setiap sendi dalam sistem agar *end effector* mencapai posisi target yang diinginkan. Penggunaan vektor juga memungkinkan transformasi dan rotasi dalam ruang tiga dimensi untuk mensimulasikan pergerakan yang lebih kompleks seperti pergerakan tangan atau kaki manusia dan pergerakan tangan robot [3].

Makalah ini bertujuan untuk melakukan eksplorasi dan analisis kegunaan vektor dalam *inverse kinematics* untuk pengaplikasian animasi prosedural. Makalah ini juga bertujuan sebagai pengantar dalam konsep fundamental animasi prosedural, *forward kinematics* (FK), dan *inverse kinematics* (IK), dan juga mempelajari peran vektor dalam menyelesaikan masalah IK. Dengan menjelaskan konsep-konsep ini dan menunjukkan bagaimana aplikasinya dalam algoritma, diharapkan pembaca dapat memiliki pemahaman tentang dasar animasi prosedural, aplikasi IK dalam sistem animasi, dan peran vektor dalam proses animasi prosedural.

Makalah ini pertama-tama akan mengenalkan konsep dasar dari animasi prosedural seperti *distance constraints* dan *angle constraints*. Kedua konsep ini penting dalam memastikan pergerakan yang dibuat dalam IK realistis secara fisik. Kedua konsep ini menciptakan pergerakan yang realistis dengan membatasi jangkauan pergerakan dari setiap sendi dan memberikan aturan dasar agar sistem animasi bergerak sesuai dengan yang diinginkan. Contohnya, dalam lengan manusia, bagian siku yang merupakan sendi tidak bisa berputar sepenuhnya dan tangan manusia tidak bisa bergerak terlalu jauh dari bahu.

Selanjutnya, makalah ini akan membahas sedikit tentang *forward kinematics* (FK). FK menentukan posisi dari *end effector* ketika semua orientasi sendi dalam sistem diketahui dan memiliki proses kalkulasi yang lebih sederhana daripada IK. Namun, FK cukup penting dalam menjadi dasar untuk memahami cara kerja IK. IK dapat dipahami dengan pendekatan mundur dari cara kerja FK.

Terakhir, makalah ini akan masuk ke analisis *inverse kinematics*. Algoritma IK menggunakan cara iteratif untuk menyesuaikan orientasi dari setiap sendi agar *end effector* dapat mencapai posisi yang diinginkan. Penggunaan vektor dalam proses ini menyederhanakan perhitungan matematis sebagai representasi dari posisi, jarak, dan sudut.

Tujuan dari makalah ini bukanlah untuk mengenalkan algoritma baru atau pun metode baru dalam penerapan animasi prosedural melainkan menyediakan penjelasan dalam teknik matematis yang digunakan di balik implementasi animasi prosedural dan *inverse kinematics*. Makalah ini diharapkan dapat menjadi bahan belajar bagi pembaca yang ingin mempelajari algoritma animasi prosedural dan pembaca yang berada dalam bidang animasi, robotika, pengembangan *video game*, atau bidang lain yang membutuhkan animasi prosedural.

II. Dasar Animasi Prosedural

Animasi prosedural adalah suatu teknik animasi yang memanipulasi pergerakan dan sifat suatu objek, karakter, atau elemen yang dibuat secara algoritmik ketimbang membuat animasi secara *frame by frame*. Teknik animasi ini dibuat berdasarkan model matematis, simulasi fisik, dan algoritma untuk menciptakan perilaku yang dinamis dan adaptif terhadap perubahan. Animasi prosedural sangat penting dalam aplikasi animasi yang membutuhkan performa *real-time* dan sistem yang kompleks dengan banyak variabel.

Animasi prosedural sangat bergantung kepada model matematika dan algoritma untuk menentukan sifat atau perilaku suatu objek. Algoritma ini termasuk menentukan posisi, kecepatan, atau pun gaya yang bekerja pada suatu objek. Pada simulasi fisika, animasi prosedural dapat mensimulasikan gravitasi, gesekan, atau gaya lain yang bekerja pada suatu objek yang menentukan perilaku gerak dari objek tersebut. Persamaan diferensial bisa menentukan kecepatan dan percepatan suatu objek. Lalu, persamaan kinematika bisa digunakan untuk menentukan posisi objek berdasarkan kecepatan dan percepatannya.

Salah satu contohnya adalah sistem *spring-mass-damper* yang biasa digunakan untuk mensimulasikan gerakan objek seperti tali, rambut, atau kain. Sistem ini dimodelkan dengan persamaan diferensial yang menjelaskan gaya dan gerak yang bekerja pada objek ketika dikenai gaya luar.

Selain itu, animasi prosedural sering kali menggunakan konsep batasan atau *constraints*. Batasan ini membuat gerak yang diciptakan terbatas dengan parameter tertentu agar lebih sesuai dengan gerak asli secara fisik. Contohnya, animasi prosedural pada lengan manusia harus diberi batas sudut pada bagian siku agar berputar sesuai dengan yang seharusnya. Batasan ini juga menjamin tangan pada bagian lengan tidak bisa memiliki posisi yang sangat jauh dari bahu. Beberapa contoh batasan adalah batasan jarak (*distance constraints*), batasan sudut (*angle constraints*), dan batasan kecepatan (*velocity constraints*). Batasan jarak adalah batasan yang mengharuskan objek atau bagian objek untuk harus selalu berada pada jarak yang sama (atau kurang) relatif terhadap suatu acuan. Batasan sudut adalah batasan yang mengharuskan objek atau bagian objek untuk harus selalu berada dalam batas rotasi relatif terhadap suatu acuan. Batasan kecepatan membatasi kecepatan objek atau bagian objek [4].

Salah satu metode paling terkenal dalam mensimulasikan gerak fisik yang realistis adalah *rigid body dynamics* yang mensimulasikan objek yang tidak mengalami deformasi dan *soft body dynamics* yang mensimulasikan objek yang mengalami deformasi.

Animasi prosedural sudah banyak digunakan pada beberapa industri saat ini. Salah satu industri yang banyak menggunakan animasi prosedural adalah industri

pengembangan permainan digital. Dalam permainan digital, animasi prosedural biasanya dipakai untuk animasi gerak karakter berdasarkan masukan yang diberikan oleh pemain. Misalkan, suatu karakter memiliki animasi yang berbeda ketika mendaki rintangan dengan elevasi yang berbeda. Selain itu, animasi *walking* suatu karakter bisa dibuat secara prosedural berdasarkan kecepatan karakter, arah gerak, dan posisi kaki. Hal ini sangat berguna terutama dalam permainan digital yang menggunakan *procedurally generated environments* yang tidak memungkinkan untuk animasi tradisional membuat semua kemungkinan animasi.

Selain dalam industri pengembangan permainan digital, animasi prosedural juga digunakan dalam bidang robotika. Dalam robotika, animasi prosedural digunakan untuk mengendalikan *robotic limbs* atau *manipulators*. Animasi prosedural memastikan pergerakan robot sesuai dengan lingkungan mereka dan efisien. Robot terkadang perlu bergerak dari satu titik ke titik lain atau mengambil suatu barang. Animasi prosedural dapat membantu menentukan pergerakan robot dalam menangani masalah tersebut dengan tetap memperhitungkan batasan dan menghindari tabrakan. Sebagai contoh, tangan dari suatu robot bisa saja digerakkan berdasarkan prinsip *inverse kinematics* yang menentukan posisi dan orientasi setiap *joints* agar *end effector* dari tangan robot tersebut dapat mencapai lokasi target.

Beberapa contoh lain dari aplikasi animasi prosedural adalah simulasi gerak kendaraan pada berbagai medan dan elevasi, simulasi lingkungan virtual untuk *augmented reality*, simulasi kerumunan untuk film atau animasi, dan simulasi fenomena natural seperti aliran air, api, atau asap.

Animasi prosedural memiliki beberapa kelebihan dibandingkan teknik animasi tradisional. Animasi prosedural dibuat secara *real-time*, memungkinkan faktor eksternal untuk memengaruhi animasi secara langsung. Animasi prosedural juga memungkinkan variasi animasi yang tidak terbatas sehingga tidak perlu membuat animasi untuk semua jenis kemungkinan. Animasi prosedural juga menciptakan animasi gerak yang organik dan lebih natural karena dibuat berdasarkan model matematika dan simulasi fisik.

Walaupun animasi prosedural terlihat memiliki banyak kelebihan, animasi prosedural juga memiliki beberapa batasan dan kelemahan. Animasi prosedural masih belum bisa mensimulasikan perilaku yang sangat kompleks seperti mimik muka manusia atau pun gerak ekspresi emosi suatu karakter. Untuk perilaku yang sangat kompleks seperti ini, diperlukan teknik animasi tradisional atau kombinasi antara teknik animasi tradisional dengan animasi prosedural. Selain itu, animasi prosedural berpotensi membutuhkan kekuatan komputasi yang sangat tinggi terutama dalam simulasi fisik dalam skala besar.

III. *Forward Kinematics* dan *Inverse Kinematics*

Forward Kinematics (FK) adalah proses untuk menghitung posisi dan orientasi dari *end effector* (biasanya ujung dari sistem rantai kinematik) berdasarkan parameter sudut atau rotasi dari setiap sendi dalam sistem tersebut. *Forward kinematics* digunakan untuk menentukan posisi akhir dari robot atau anggota tubuh dalam sistem koordinat berdasarkan parameter sudut atau jarak yang diterapkan pada masing-masing sendi.

Pada sistem kinematik terbuka yang terdiri dari beberapa segmen (misalnya lengan robot yang terdiri dari beberapa sendi), kita dapat menghitung posisi dari *end effector* dengan menggunakan serangkaian transformasi matriks. Misalnya, untuk sistem tiga dimensi dengan tiga sendi, posisi *end effector* dapat dihitung dengan mengalikan matriks transformasi yang berhubungan dengan sudut masing-masing sendi.

Forward kinematics sangat berguna dalam kontrol robot, karena dengan mengetahui posisi sudut pada sendi-sendi robot, kita dapat mengetahui posisi akhir dari *end effector* dan merencanakan gerakan robot dalam ruang. Namun, kekurangannya adalah tidak selalu mudah untuk mengontrol atau mencapai posisi target yang spesifik hanya dengan mengetahui parameter sendi awal.

Inverse Kinematics (IK) adalah proses untuk menentukan parameter sudut atau rotasi dari setiap sendi dalam sistem kinematik untuk mencapai posisi target yang diinginkan oleh *end effector*. Berbeda dengan *Forward Kinematics* yang menghitung posisi akhir berdasarkan sudut sendi, *Inverse Kinematics* mencoba untuk membalik proses tersebut dengan cara mengetahui posisi target yang diinginkan lalu menghitung sudut-sudut pada sendi yang diperlukan untuk mencapai posisi tersebut.

Inverse Kinematics sering digunakan dalam robotika, animasi karakter komputer, dan simulasi gerakan, di mana kita ingin menggerakkan bagian-bagian tubuh atau robot secara organik menuju suatu posisi tertentu. Salah satu tantangan utama dalam *Inverse Kinematics* adalah bahwa solusi untuk sudut sendi yang tepat tidak selalu unik, dan dalam beberapa kasus, bisa jadi tidak ada solusi yang valid, terutama jika posisi target berada di luar jangkauan sistem kinematik [3].

Ada beberapa metode yang digunakan untuk menyelesaikan masalah *Inverse Kinematics*, yaitu: CCD (*Cyclic Coordinate Descent*), *Jacobian Inverse*, dan FABRIK (*Forward And Backward Reaching Inverse Kinematics*).

Metode CCD melakukan pendekatan iteratif untuk menyelesaikan masalah *Inverse Kinematics*. Dalam metode ini, kita mulai dengan posisi *end effector* dan bekerja mundur ke arah sendi pertama, mencoba menyesuaikan sudut setiap sendi secara iteratif untuk mendekatkan posisi *end effector* ke posisi target.

Metode *Jacobian Inverse* menggunakan informasi dari matriks *Jacobian* untuk menghitung perubahan sudut sendi yang diperlukan untuk memindahkan *end effector* lebih dekat ke posisi target. Metode ini lebih akurat dalam mengatasi gerakan yang lebih kompleks, tetapi

memerlukan kalkulasi yang lebih berat, terutama untuk sistem dengan banyak sendi.

FABRIK adalah metode iteratif yang bekerja dengan pendekatan maju (*forward*) dan mundur (*backward*) untuk menemukan solusi *Inverse Kinematics*. Dalam metode ini, posisi *end effector* disesuaikan terlebih dahulu ke posisi target (*forward reaching*), lalu setiap sendi dihitung posisinya dalam urutan terbalik (*backward reaching*) untuk memastikan bahwa setiap segmen tetap berada pada panjang yang ditentukan.

IV. Vektor dan Rotasi dalam Animasi Prosedural

Salah satu konsep dasar dalam animasi prosedural adalah batasan jarak (*distance constraint*). Batasan jarak berfungsi untuk memberikan pembatasan terhadap jarak relatif antara dua titik dalam suatu sistem, sehingga titik-titik tersebut tetap berada pada jarak tertentu satu sama lain.

Untuk menggambarkan konsep ini, kita bayangkan sebuah bidang dua dimensi yang memiliki titik tertentu, yang kita sebut sebagai *anchor* (titik acuan). Dari titik *anchor* ini, sebuah jarak tertentu didefinisikan dan terdapat titik lain yang berada di luar radius jarak tersebut. Tujuan dari *distance constraint* ini adalah untuk membatasi pergerakan titik kedua, memastikan titik tersebut tidak melebihi jarak yang telah ditentukan dari titik *anchor*.

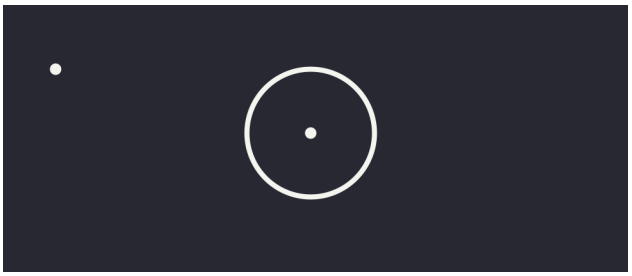

Fig. 1. Visualisasi titik *anchor*, radius jarak, dan titik lain.

Untuk mengimplementasikan konsep ini secara matematis, kita mulai dengan membangun sebuah vektor yang menghubungkan titik *anchor* dengan titik kedua yang berada di luar radius yang telah ditentukan.

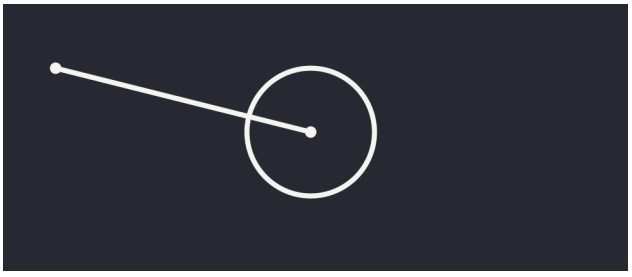

Fig. 2. Visualisasi vektor.

$$\vec{v} = \vec{P} - \vec{A}$$

$\vec{v}$ adalah vektor yang menghubungkan titik *anchor* $\vec{A}$ dengan titik kedua $\vec{P}$. $\vec{A}$ adalah posisi dari titik *anchor*. $\vec{P}$ adalah posisi dari titik kedua.

Setelah vektor didefinisikan, selanjutnya adalah menghitung besar vektor tersebut. Jika besar vektor melebihi batas jarak yang telah ditentukan, maka posisi titik kedua harus disesuaikan untuk memastikan bahwa titik tersebut tetap berada pada jarak yang ditentukan dari titik *anchor*. Penyesuaian ini dilakukan dengan cara menormalisasi vektor dan mengalikannya dengan jarak yang diinginkan, $\|\vec{d}\|$, sehingga vektor yang baru akan memiliki panjang yang sesuai dengan batasan jarak tersebut. Ini akan menghasilkan proyeksi titik ke jarak yang telah ditentukan, sehingga tetap memenuhi *distance constraint*.

$$\vec{P'} = \vec{A} + \hat{v} \cdot \|\vec{d}\|$$

$\vec{P'}$ adalah posisi baru dari titik kedua setelah penerapan batasan jarak. $\hat{v}$ adalah vektor satuan yang searah dengan $\vec{v}$. $\|\vec{d}\|$ adalah batasan jarak yang diinginkan.

Dengan demikian, titik kedua akan terikat untuk bergerak mengikuti jalur yang dibentuk oleh titik *anchor* dan batasan jarak. Ketika titik *anchor* bergerak, titik kedua akan menyesuaikan posisinya agar tetap berada pada jarak yang telah ditentukan, menciptakan gerakan yang dinamis dan realistis.

Fig. 3. Titik setelah diberi batasan jarak.

Keunggulan dari *distance constraint* adalah kemampuannya untuk menyebar dari satu titik ke titik lainnya, membentuk suatu struktur rantai (*chain*). Setiap titik yang terikat pada jarak tertentu dari titik anchor atau titik sebelumnya menciptakan rangkaian titik yang terhubung, atau *chain*.

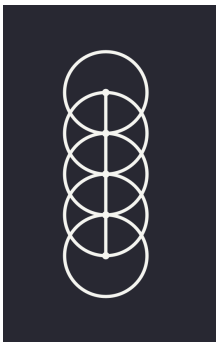

Fig. 4. Rantai titik yang diberi batasan jarak.

Contoh aplikasinya adalah suatu representasi animasi dari tubuh yang terdiri dari segmen-segmen, seperti ular atau lengan robot. Jika segmen pertama (*anchor*) dari

rantai tersebut dipindahkan, maka segmen-segmen berikutnya (titik-titik yang terikat) akan mengikuti gerakan tersebut dan menyesuaikan posisi mereka untuk mempertahankan jarak antar titik. Rantai ini bisa diperpanjang dengan terus menerapkan *distance constraint* antara setiap pasangan titik yang berdekatan, sehingga memungkinkan gerakan dinamis yang realistis.

Dengan demikian, sebuah rantai titik yang saling terhubung, seperti pada rantai atau segmen-segmen tubuh yang dianimasikan, dapat disimulasikan menggunakan *distance constraint*. Gerakan dari sistem ini dapat diamati mengalir dengan mulus mengikuti pergerakan titik anchor yang pertama. Ini memberikan gambaran bagaimana sistem yang saling terhubung dapat bergerak dengan cara yang alami dan realistis [4].

Fig. 5. Rantai titik dengan *anchor* yang digerakkan.

Lalu, dalam animasi prosedural, dikenal suatu teknik bernama *Forward Kinematics* (FK). FK adalah salah satu teknik dasar yang digunakan dalam animasi prosedural untuk menentukan posisi efektor akhir (*end effector*), seperti tangan atau kaki, berdasarkan posisi dan orientasi setiap sendi dalam sistem. Dalam metode ini, posisi dari setiap segmen dalam rantai kinematik dihitung berdasarkan sudut rotasi yang diberikan pada setiap sendi. Secara sederhana, FK digunakan untuk menghitung posisi dan orientasi objek dalam suatu sistem berdasarkan parameter input yang sudah diketahui, seperti panjang lengan atau sudut rotasi sendi. Dengan kata lain, FK adalah proses penghitungan posisi efektor akhir dari suatu rantai kinematik jika semua parameter posisi dan orientasi sendi diketahui.

Secara matematis, *Forward Kinematics* melibatkan penggunaan fungsi trigonometri dan matriks transformasi untuk menghitung posisi setiap segmen berdasarkan parameter yang diberikan. Dalam sistem kinematik dua dimensi, posisi setiap segmen dihitung dengan menggunakan persamaan trigonometri sederhana.

Misalkan kita memiliki sistem dua segmen: segmen pertama dan segmen kedua. Kedua segmen ini terhubung melalui sendi. Terdapat tiga sendi yang menghubungkan sendi ini yaitu: sendi pertama, sendi kedua, dan efektor akhir. Misalnya, panjang segmen pertama adalah l_1 dan panjang segmen kedua adalah l_2. Sudut rotasi pada sendi pertama adalah θ_1 dan pada sendi kedua adalah θ_2.

Untuk menghitung posisi efektor akhir dalam sistem koordinat dua dimensi, kita bisa menggunakan rumus trigonometri sebagai berikut:

Posisi sendi kedua

$$x_1 = l_1 \cdot cos(\theta_1)$$
$$y_1 = l_1 \cdot sin(\theta_1)$$

Di mana l_1 adalah panjang segmen pertama dan θ_1 adalah sudut yang dibentuk oleh segmen pertama terhadap sumbu horizontal.

Posisi efektor akhir

$$x_2 = x_1 + l_2 \cdot cos(\theta_1 + \theta_2)$$
$$y_2 = y_1 + l_2 \cdot sin(\theta_1 + \theta_2)$$

Di mana l_2 adalah panjang segmen kedua, dan θ_2 adalah sudut pada sendi kedua.

Misalkan kita memiliki sistem dengan n segmen kinematik yang terhubung oleh sendi. Kita dapat menghitung posisi setiap sendi dan efektor akhir (*end effector*) menggunakan panjang segmen dan sudut rotasi pada setiap sendi.

$$x_i = \sum_{j=1}^{i} l_j \cdot cos(\sum_{k=1}^{j} \theta_k)$$
$$y_i = \sum_{j=1}^{i} l_j \cdot sin(\sum_{k=1}^{j} \theta_k)$$

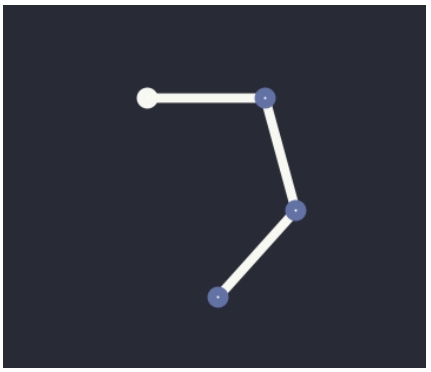

Fig. 6. Contoh sistem tiga segmen dengan sudut sendi pertama 0 (berwarna putih), sudut sendi kedua 1.3 radian, sudut sendi ketiga 1 radian, dan panjang segmen sama.

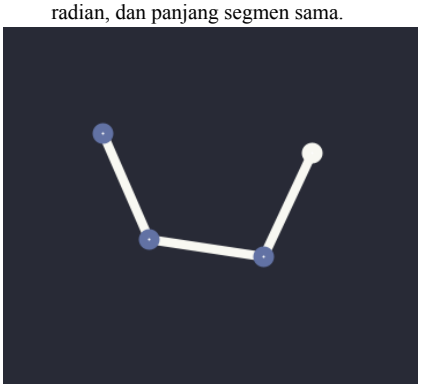

Fig. 7. Sistem tiga segmen yang sama dengan sudut sendi pertama 2 radian (berwarna putih), sudut sendi kedua 1.3 radian, sudut sendi ketiga 1 radian, dan panjang segmen sama.

Pada sistem kinematik tiga dimensi, setiap segmen kinematik dapat dipandang sebagai transformasi yang melibatkan dua elemen utama. Rotasi yang merupakan

perubahan arah segmen relatif terhadap sumbu koordinat dan Translasi yang merupakan perpindahan posisi segmen dalam ruang tiga dimensi. Untuk menggabungkan rotasi dan translasi dalam satu operasi, kita menggunakan matriks transformasi homogen 4x4. Matriks ini dapat mengubah koordinat suatu titik dalam ruang tiga dimensi dengan cara yang efisien.

Matriks transformasi homogen T_i dapat dituliskan sebagai:

$$T_i = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix}$$

R_i adalah matriks rotasi 3x3 yang menggambarkan rotasi pada segmen ke-i. t_i adalah vektor translasi 3x1 yang menggambarkan posisi segmen ke-i. Angka 0 di baris terakhir mewakili bahwa sistem menggunakan koordinat homogen. Angka 1 di bagian bawah kanan adalah elemen homogen.

Jika kita memiliki sistem dengan n segmen, kita dapat menghitung posisi dan orientasi efektor akhir dengan mengalikan matriks transformasi homogen dari setiap segmen. Misalnya, jika kita memiliki n segmen, matriks transformasi total T_{total} adalah hasil perkalian berturut-turut dari semua matriks transformasi T_1, T_2, …, , T_n:

$$T_{\text{total}} = T_1 \cdot T_2 \cdot \cdots \cdot T_n$$

Inverse Kinematics (IK) merupakan kebalikan dari *Forward Kinematics* (FK), di mana kita diberikan posisi akhir (end effector) dan ingin menghitung posisi dan orientasi sendi-sendinya untuk mencapai posisi tersebut. Dalam IK, kita harus menyelesaikan persamaan yang menghubungkan posisi end effector dengan parameter-parameter sudut sendi dalam sistem kinematik.

Ada beberapa metode untuk menyelesaikan masalah IK. Terdapat tiga metode populer dalam IK: CCD (*Cyclic Coordinate Descent*), *Jacobian Inverse*, dan FABRIK (*Forward and Backward Reaching Inverse Kinematics*).

CCD adalah metode iteratif untuk memecahkan masalah *inverse kinematics* pada sistem dengan banyak sendi. Pada dasarnya, CCD bekerja dengan mengoptimalkan sudut sendi satu per satu secara berulang-ulang untuk mendekatkan posisi *end effector* ke posisi targetnya. Metode ini iteratif dan cukup sederhana untuk diimplementasikan.

Misalkan kita memiliki sistem dengan n segmen dan n sendi, dengan *end effector* yang ingin kita tempatkan pada posisi target $\vec{p}_{\text{target}}$.

Pertama, tentukan posisi akhir *end effector*, $\vec{p}_{\text{end}}$, dari sistem kinematik berdasarkan nilai sudut sendi yang ada. Lalu, tentukan *error* atau selisih antara posisi *end effector* saat ini dan posisi target, $\vec{e} = \vec{p}_{\text{target}} - \vec{p}_{\text{end}}$. Setelah itu, untuk setiap sendi i, perbarui sudut θ_i sendi dengan cara memutar sendi tersebut untuk mengurangi *error*. Untuk segmen i, kita bisa mengatur sudut θ_i sedemikian rupa sehingga *error* di sepanjang sumbu i terkecil.

$$\theta_i' = \theta_i + \alpha \cdot \Delta\theta_i$$

α adalah laju pembelajaran (*learning rate*), dan $\Delta\theta_i$ adalah perubahan sudut yang dihitung untuk meminimalkan *error*. Ulangi langkah penentuan *error* dan pengaturan sudut beberapa kali hingga dirasa *error* cukup kecil (konvergen).

CCD memiliki beberapa kelebihan yaitu: sederhana dan mudah diimplementasikan, tidak memerlukan komputasi yang rumit, dan cocok untuk masalah IK dengan banyak sendi. Tetapi, CCD juga punya beberapa kekurangan yakni: proses yang iteratif, sehingga *error* tidak selalu konvergen dalam jumlah iterasi terbatas, dan tidak selalu memberikan solusi yang paling optimal.

Fig. 8. Bentuk sistem ketika target berada pada posisi (700, 500) dengan metode CCD.

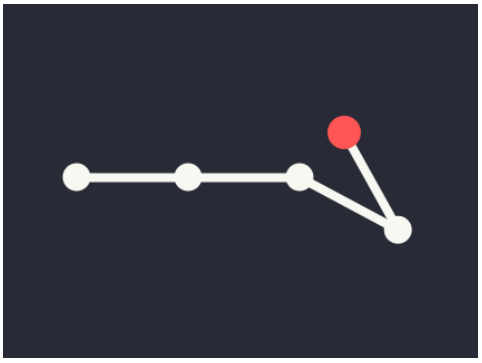

Fig. 9. Bentuk sistem ketika target berada pada posisi (1200, 500) dengan metode CCD.

Metode *Jacobian Inverse* adalah metode yang lebih matematis dan efisien untuk menyelesaikan masalah inverse kinematics. Dalam metode ini, kita menghitung perubahan kecil dalam sudut sendi dengan menggunakan *Jacobian matrix*, yang menghubungkan perubahan posisi end effector dengan perubahan sudut sendi.

Misalkan kita memiliki sistem kinematik dengan n segmen dan n sendi, dengan posisi *end effector* yang tergantung pada parameter sudut sendi, $\theta = [\theta_1, \theta_2, \ldots, \theta_n]^T$.

Pertama, tentukan posisi *end effector* saat ini, $\vec{P}_{\text{end}}$, berdasarkan sudut sendi saat ini. Posisi ini dihitung dengan menggunakan *forward kinematics*. Berikutnya, tentukan *error* atau selisih antara posisi target $\vec{P}_{\text{target}}$ dan posisi *end effector* $\vec{P}_{\text{end}}$, $\vec{e} = \vec{p}_{\text{target}} - \vec{p}_{\text{end}}$.

Lalu, hitung *Jacobian matrix* J, yang menghubungkan perubahan posisi end effector dengan perubahan sudut sendi

$$J = \frac{\partial \vec{P}_{\text{end}}}{\partial \theta}$$

Jacobian matrix ini adalah matriks yang berukuran $3 \times n$ untuk sistem tiga dimensi dan $2 \times n$ untuk sistem dua dimensi. Matriks ini memberikan informasi tentang bagaimana posisi end effector berubah dengan perubahan sudut sendi.

Kemudian, perbarui sudut sendi dengan menggunakan invers *Jacobian matrix* untuk menghitung perubahan sudut sendi yang diperlukan untuk mengurangi error.

$$\Delta\theta = J^{-1} \cdot \vec{e}$$

Lalu, perbarui sudut sendi.

$$\theta' = \theta + \Delta\theta$$

Ulangi langkah penentuan *error* hingga langkah memperbarui nilai sudut sendi hingga dirasa *error* cukup kecil (konvergen).

Jacobian Inverse memiliki beberapa kelebihan yaitu: efisien dan matematis, memberikan solusi yang lebih cepat dan lebih akurat dibandingkan metode iteratif seperti CCD, dan dapat digunakan untuk masalah IK dengan lebih sedikit iterasi. Tetapi, *Jacobian Inverse* juga punya beberapa kekurangan yakni: memerlukan perhitungan yang bisa saja rumit pada sistem kinematik yang sangat kompleks, dan memerlukan invers matriks, yang bisa menjadi masalah pada ketika *Jacobian matrix* tidak *invertible*.

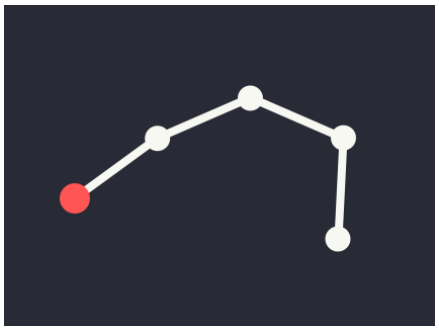

Fig. 10. Bentuk sistem ketika target berada pada posisi (700, 500) dengan metode *Jacobian Inverse*.

Fig. 11. Bentuk sistem ketika target berada pada posisi (1200, 500) dengan metode *Jacobian Inverse*.

Metode yang terakhir adalah FABRIK (*Forward and Backward Reaching Inverse Kinematics*). FABRIK adalah metode iteratif mirip seperti CCD. Metode ini bekerja dengan melakukan dua langkah utama: *Forward Reaching* dan *Backward Reaching*.

Misalkan kita memiliki sistem dengan n segmen dan n sendi, dengan posisi target $\vec{P}_{\text{target}}$. Langkah pertama, tentukan posisi awal untuk setiap sendi, yang biasanya dimulai dengan posisi *end effector* $\vec{P}_{\text{end}}$ pada posisi target $\vec{P}_{\text{target}}$. Langkah kedua (*Forward Reaching*), tentukan posisi sendi ke-1 hingga sendi $n-1$ dengan menggunakan pendekatan rekursif dari posisi terakhir (*end effector*). Perbarui posisi setiap sendi dengan menghitung posisi yang optimal berdasarkan panjang segmen:

$$\vec{P}_i = \vec{P}_{i+1} + L_i \cdot \frac{\vec{P}_{i+1} - \vec{P}_i}{\|\vec{P}_{i+1} - \vec{P}_i\|}$$

di mana L_i adalah panjang segmen antara sendi i dan sendi $i+1$. Langkah ketiga (*Backward Reaching*), tentukan posisi sendi pertama dengan cara bergerak mundur dari *end effector*. Perbarui posisi setiap sendi berdasarkan posisi yang baru dihitung pada langkah sebelumnya. Langkah keempat, ulangi langkah kedua dan ketiga hingga konvergen atau *error* cukup kecil.

FABRIK memiliki beberapa kelebihan yaitu: stabil dan efisien, tidak memerlukan komputasi matriks yang rumit seperti pada *Jacobian Inverse*, dan cocok untuk sistem kinematik dengan banyak sendi dan segmen. Tetapi, FABRIK juga punya beberapa kekurangan yakni: metode ini tidak selalu memberikan solusi optimal, tetapi lebih stabil dibandingkan CCD, dan terkadang membutuhkan lebih banyak iterasi untuk mencapai konvergensi.

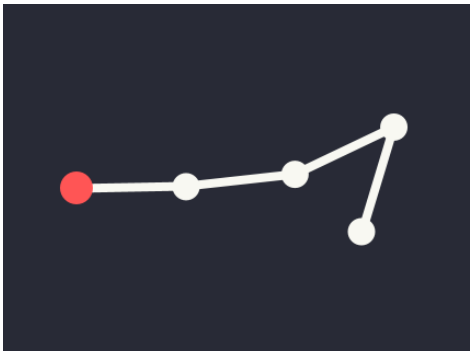

Fig. 12. Bentuk sistem ketika target berada pada posisi (700, 500) dengan metode FABRIK.

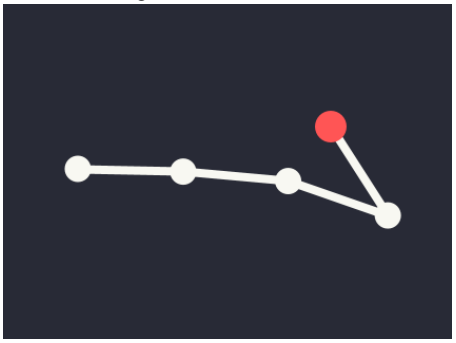

Fig. 13. Bentuk sistem ketika target berada pada posisi (1200, 500) dengan metode FABRIK.

V. Konklusi

Distance Constraint menunjukkan pembatasan jarak antar titik dalam sistem dapat menciptakan struktur dinamis yang saling terhubung, seperti rantai titik yang bergerak mengikuti pergerakan titik *anchor*. Ini memberikan gambaran bagaimana animasi objek yang saling terhubung, seperti tubuh atau segmen-segmen robot, dapat bergerak secara realistis dan mulus. Teknik ini sangat berguna dalam simulasi objek yang mengikuti jalur terikat dan mempertahankan jarak yang tetap.

Forward Kinematics (FK), sebagai teknik dasar, memungkinkan kita untuk menghitung posisi dan orientasi *end effector* dalam sistem kinematik berdasarkan sudut rotasi sendi. Dengan menggunakan persamaan trigonometri, kita dapat menentukan posisi *end effector* berdasarkan parameter sudut dan panjang segmen.

Inverse Kinematics (IK), sebagai kebalikan dari FK, menyelesaikan masalah dengan mengoptimalkan posisi dan orientasi sendi untuk mencapai posisi akhir yang diinginkan. Metode yang digunakan dalam IK, seperti *Cyclic Coordinate Descent* (CCD), *Jacobian Inverse*, dan FABRIK, masing-masing memiliki kelebihan dan kekurangan. CCD menawarkan solusi yang mudah diterapkan namun tidak selalu konvergen dalam jumlah iterasi terbatas. Sementara itu, *Jacobian Inverse* lebih efisien dan cepat, namun memerlukan perhitungan yang rumit dan invers matriks. FABRIK memberikan metode yang lebih stabil dan efisien tanpa memerlukan komputasi matriks yang kompleks, meskipun membutuhkan lebih banyak iterasi untuk mencapai konvergensi.

Dari hasil eksplorasi ini, dapat disimpulkan bahwa meskipun masing-masing metode memiliki kekuatan dan kelemahannya, kombinasi dari berbagai teknik ini dapat menghasilkan animasi yang lebih dinamis dan realistis. Pemilihan metode yang tepat sangat bergantung pada kebutuhan spesifik sistem yang dianimasikan, seperti kompleksitas model dan kecepatan konvergensi yang diinginkan. Penggunaan *distance constraint* untuk menghubungkan titik-titik dan mengendalikan jarak antar segmen memberikan kontrol lebih lanjut atas interaksi objek dalam ruang, sementara teknik kinematik maju dan mundur seperti FK dan IK memungkinkan penentuan posisi objek dan efektor dalam sistem yang lebih kompleks.

VI. Appendix

Sebagai tambahan material, di bawah ini merupakan *link* GitHub *repository* yang berisi kode yang digunakan untuk visualisasi animasi prosedural menggunakan *java processing library*:

https://github.com/V-Kleio/Procedural-Animation

References

[1] E. C. Danny Alberto, X. Luo, A. A. Navarro Newball, C. Zúñiga and C. Lozano-Garzón, "Realistic Behavior of Virtual Citizens through Procedural Animation," 2019 International Conference on Virtual Reality and Visualization (ICVRV), Hong Kong, China, 2019, pp. 243-247, doi: 10.1109/ICVRV47840.2019.00057. keywords: {Virtual reality;Visualization;procedural animation;virtual characters;simulation},

[2] P. Srisuk, A. Sento and Y. Kitjaidure, "Inverse kinematics solution using neural networks from forward kinematics equations," 2017 9th International Conference on Knowledge and Smart Technology (KST), Chonburi, Thailand, 2017, pp. 61-65, doi: 10.1109/KST.2017.7886084. keywords: {Kinematics;Manipulators;Mathematical model;Robot kinematics;Biological neural networks;forward kinematics;inverse kinematics;neural network;robotic arm},

[3] Aristidou, Andreas & Lasenby, Joan & Chrysanthou, Yiorgos & Shamir, Ariel. (2018). Inverse Kinematics Techniques in Computer Graphics: A Survey. Computer Graphics Forum. 37. 35-58. 10.1111/cgf.13310.

[4] argonaut. "A simple procedural animation technique," YouTube, May 19, 2024. [Video file]. Available: A simple procedural animation technique. [Accessed: January 1, 2025].

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 27 Desember 2024

Rafael Marchel Darma Wijaya 13523146