# Analisis dan Implementasi Object Tracking Menggunakan Metode ASIFT dan Mean Shift

Andrian Wijayana [#1], Tjokorda Agung Budi W, ST,. MT[#2] Siti Sa'adah, ST,. MT[#3]

[#] *Departemen Informatika, Fakultas Teknik, Universitas Telkom*
*Bandung, Indonesia*

***Abstrak*—** Semakin tingginya kebutuhan manusia terhadap sistem keamanan berbasis *tracking* yang dapat bekerja secara otomatis, membuat bermunculan metode dan teknik baru guna memenuhi kebutuhan tersebut. Pada *object tracking*, ekstraksi ciri menjadi salah satu tugas utama dalam melacak sebuah objek dimana ciri yang digunakan harus tahan terhadap berbagai kondisi karena objek selalu bergerak bebas dalam video. Oleh karena itu, ektraksi ciri yang tahan terhadap segala bentuk transformasi atau *fully invariant* sangat dibutuhkan pada *object tracking*. Salah satu metode ekstraksi ciri yang *fully invariant* adalah metode *affine scale invariant feature transform* (ASIFT). Hasil penelitian menunjukkan bahwa kondisi lingkungan objek berpengaruh terhadap banyaknya kemungkinan objek terdeteksi dengan tepat. Kondisi lingkungan terkontrol cenderung memiliki hasil yang lebih baik dari lingkungan yang tidak terkontrol. Selain itu, nilai threshold dan radius yang digunakan juga sangat mempengaruhi hasil *matching* objek. Berdasarkan penelitian, threshold dengan nilai 0.9 dan radius 10 % memiliki kecenderungan dapat mendeteksi objek dengan tepat. Hasil penelitian juga menunjukkan bahwa metode ASIFT-Mean Shift dapat mengatasi permasalahan perubahan *point of view* yang terjadi pada objek dengan akurasi 30%.

## I. PENDAHULUAN

Meningkatnya kebutuhan manusia akan alat pengamanan berbasis video yang dapat bekerja secara otomatis membuat ilmuwan mulai menciptakan berbagai macam teknologi dan metode guna memenuhi kebutuhan tersebut contohnya adalah implementasi *object tracking*. *Object tracking* adalah proses mengikuti perubahan posisi suatu objek pada suatu interval waktu. Dalam *object tracking*, proses ekstraksi ciri objek merupakan hal yang sangat penting karena ciri-ciri yang didapat dari suatu objek tersebutlah yang akan kita gunakan sebagai perbandingan dalam melakukan pelacakan.

Huiyou Zhou et al[4] mengkombinasikan metode SIFT dan *mean shift* untuk melakukan *object tracking* dimana metode SIFT digunakan untuk melakukan proses ekstraksi ciri dan metode *mean shift* digunakan untuk proses *tracking*. Metode SIFT merupakan metode ekstraksi ciri berupa *point* yang *invariant* terhadap empat dari enam buah parameter affine, yaitu *scale*, *rotation*, dan *translation*. Dengan sifatnya yang *invariant* terhadap empat buah parameter tersebut, SIFT merupakan metode yang baik untuk digunakan pada *tracking*, namun SIFT kurang efektif pada kondisi terdapat perubahan sudut pandang (*point of view*) gambar sehingga ciri yang diekstrak kurang stabil apabila ada perubahan *point of view*. Sedangkan metode *mean shift* merupakan sebuah metode *tracking* yang sederhana dan *low cost* pada penerapannya sehingga cocok digunakan dengan metode SIFT yang sifatnya kompleks.

Berdasarkan penjelasan diatas, dapat dilihat bahwa permasalahan dalam kasus *object tracking* adalah metode pengenalan ciri terhadap objek yang ingin kita lacak kurang tahan atau kurang *invariant* terhadap berbagai perubahan kondisi objek dimana objek selalu bergerak secara bebas. Dibutuhkan sebuah metode yang *fully invariant* dan memenuhi kriteria *affine transformation* dalam permasalahan kali ini. Selain itu, akan dibutuhkan juga sebuah metode *tracking* yang sederhana namun menghasilkan hasil yang baik. Oleh karena itu, pada tugas akhir ini, penulis akan membuat sebuah sistem *object tracking* dengan menerapkan metode *fully invariant*, yaitu metode ASIFT dan *mean shift*. Selain itu, penulis juga akan mengukur performansi metode ASIFT dan *mean shift* dari tingkat akurasi dan tingkat keberhasilannya dalam melacak sebuah objek

.

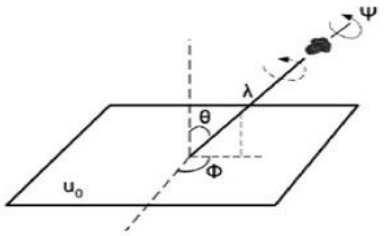
## II. LANDASAN TEORI

### A. Ekstraksi Fitur

ASIFT merupakan algoritma yang dikembangkan dari algoritma pendahulunya yaitu algoritma SIFT. Algoritma SIFT ditemukan oleh Lowe pada tahun 1994. Algoritma SIFT merupakan algoritma yang *invariant* terhadap empat buah parameter dari enam parameter yang terdapat pada *affine transformation*, yaitu *scale*, *translation*, *rotation*, *shear*, *longitude*, *dan latitude*. Pada algoritma ASIFT ditambahkan dua buah parameter yang sebelumnya belum ada pada algoritma SIFT, yaitu *longitude* dan *latitude*. Kedua parameter tersebut diimplementasikan agar algoritma dapat beradaptasi terhadap perubahan sudut pandang atau sudut ambil sebuah *image*. Dengan demikian, algoritma ASIFT merupakan salah satu algoritma yang dapat dikatakan sebagai algoritma yang *fully invariant*. Secara matematis, metode ASIFT yang *fully invariant* pun dapat dibuktikan kebenarannya [9]. Terdapat 5 tahapan utama dalam algoritma ASIFT.

## 1. Affine Camera Model

Pada tahapan ini, algoritma akan memproses distorsi gambar yang berupa sudut pandang dari sebuah gambar. Perubahan dari sudut pandang pada *affine transformation* dapat diilustrasikan sebagaimana pada gambar 2.1.




Gambar 2.1 Ilustrasi pergerakan kamera[6]

Gambar 2.1 merepresentasikan perubahan sudut pandang dimana λ merepresentasikan faktor *zoom* atau *scaling* dari kamera menuju objek, sedangkan θ dan Φ merepresentasikan posisi *latitude* dan *longitude* berdasarkan sudut yang dibuat oleh kamera dari objek. Untuk Ψ merupakan besarnya sudut rotasi kamera pada saat pengambilan gambar. Yang dihasilkan dari proses adalah gambar yang telah mengalami normalisasi terhadap perubahan *point of view* kamera.

## 2. Scale Space Extremum Detection

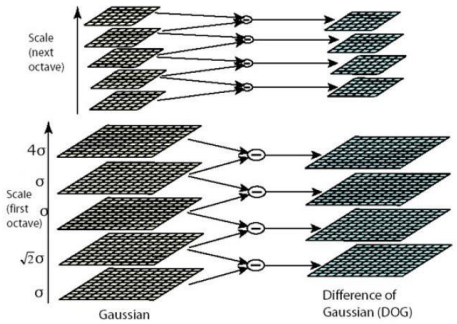
Pada tahap ini akan dicari titik atau piksel (*keypoint*) yang *invariant* terhadap perubahan ukuran gambar atau *scale*. Langkah awal pada tahap ini adalah mencari letak titik yang *invariant* terhadap perubahan skala dengan menggunakan fungsi ruang skala, dalam hal ini yang digunakan adalah fungsi Gaussian. Kemudian ruang skala pada gambar didefinisikan sebagai L(x,y,σ) dimana L merupakan *blurred image*. Nilai fungsi L(x,y,σ) didapatkan setelah dilakukan konvolusi terhadap image awal, I(x,y), dengan fungsi Gaussian (I) pada skala yang berbeda. Skala yang dimaksud adalah skala *blurred* atau *smoothed*, yaitu dari σ sampai *k*σ. Proses konvolusi akan berulang beberapa kali atau beberapa oktaf dimana pada saat konvolusi selesai, resolusi gambar awal akan dikurangi setengahnya dan dimulai lagi dari awal proses konvolusi tersebut untuk gambar awal yang sudah dikurangi resolusinya. Banyaknya proses konvolusi dalam satu oktaf adalah $S = s + 3$. Nilai 3 didapat agar pada saat proses penghitungan nilai DoG tidak diperoleh nilai dibawah
3. Dengan kata lain, jumlah konvolusi tiap oktaf minimal adalah 3. Sedangkan nilai *k* itu sendiri adalah $k = 2^{1/s}$ dan nilai σ berdasarkan percobaan yang dilakukan oleh Lowe adalah sebesar 1.6 yang menghasilkan nilai yang optimal.

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad (2.1)$$

$$D(x,y,\sigma) = \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) * I(x,y) \quad (2.2)$$

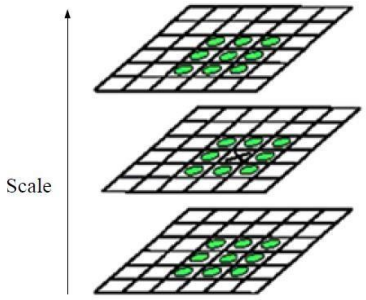
$$= L(x,y,k\sigma) - L(x,y,\sigma)$$

Untuk dapat mendeteksi *keypoint* secara efisien, Lowe mengajukan penggunaan fungsi difference-of-Gaussian (DoG) (2.2). Nilai pada fungsi ini dapat diperoleh dengan melakukan pengurangan terhadap gambar Gaussian yang saling berdekatan yang dibedakan dengan faktor skala *k* pada variable σ. Gambar 2.2. menunjukkan representasi dari proses pencarian nilai DoG.




Gambar 2.2 Representasi *Difference-of-Gaussian*[7]

Proses selanjutnya adalah pencarian nilai *extrema local* dengan melakukan pengecekan sebuah sampel piksel dengan setiap tetangganya yang berjumlah delapan untuk mengetahui apakah nilai sampel pixel tersebut minima atau maxima terhadap nilai tetangganya. Agar didapat akurasi yang baik, pengecekan dapat dilakukan dengan membandingkan sampel pixel dengan tetangga yang berada diatas dan dibawahnya. Sebuah titik akan terpilih sebagai kandidat *keypoint* apabila nilai titik tersebut lebih kecil atau lebih besar dari nilai semua tetangganya yang berjumlah 26. Gambar 2.3. menunjukkan pengecekan nilai *extrema local*.




Gambar 2.3 Pencarian nilai extrema local dengan membandingkan sampel pixel dengan tetangga-tetangganya[7]

## 3. Keypoint Localization

Setelah didapatkan kandidat-kandidat *keypoint* pada proses sebelumnya, tahapan selanjutnya adalah mengeliminasi kandidat keypoint yang tidak stabil. Biasanya *keypoint* yang tidak stabil itu adalah *keypoint* yang nilainya berbeda dari nilai sekitarnya pada matriks yang mungkin disebabkan oleh adanya *noise*. Selain itu, kandidat yang tidak stabil tersebut dapat juga berupa kandidat *keypoint* yang memiliki kontras

yang rendah dan posisinya berada pada tepian gambar. Proses eliminasi pertama akan dihilangkan *keypoint* yang memiliki *low contrast* dengan menggunakan *Quadratic Taylor* dari DoG. Dimana nilai perluasan Quadratic Taylor didapat dari persamaan (2.3).

$$D(X) = D + \frac{\partial D^T}{\partial x}X + \frac{1}{2}X^T\frac{\partial^2 D}{\partial x^2} \quad (2.3)$$

Dimana D merupakan turunan dari gambar berdasarkan nilai DoG dan nilai $x = (x,y,\sigma)^T$ merupakan nilai offset dari titik tersebut. Selanjutnya lokasi extremum, *x-hat*, ditentukan dengan mengambil turunan dari fungsi ini sehingga dihasilkan.

$$xhat = -\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x} \quad (2.4)$$

Untuk mengeliminasi keypoint dengan kontras rendah dapat diperoleh dengan mensubstitusikan persamaan (2.4) dengan persamaan (2.3) sehingga didapatkan nilai dari D(*xhat*). Semua nilai extrema (|D(*xhat*)| ) dibawah 0.03 akan dieliminasi. Selanjutnya adalah mengeliminasi keypoint yang berada pada tepian dengan menggunakan *Hessian matrix* (2.5).

$$H = \begin{bmatrix} Dxx & Dxy \\ Dxy & Dyy \end{bmatrix} \quad (2.5)$$

Penggunaan *Hessian Matrix* dimaksudkan untuk mencari nilai *principal curvature* dimana sebuah gambar dianggap sebagai dataran dua dimensi yang terdapat lekukan-lekukan. Nilai *principal curvature* sebanding dengan nilai eigen dari *Hessian matrix* yang diasumsikan α adalah nilai eigen dengan *magnitude* terbesar dan β adalah nilai eigen dengan *magnitude* yang kecil. Sehingga didapatkan persamaan sebagai berikut :

$$Tr(H) = Dxx + Dxy = \alpha + \beta \quad (2.6)$$

$$Det(H) = DxxDxy - (Dxy)^2 = \alpha\beta \quad (2.7)$$

Proses eliminasi keypoint pada tepian dilakukan dengan membandingkan rasio $Tr(H)^2/Det(H)$ (rasio *principal curvature*) dengan nilai $(r + 1)^2/r$, dimana nilai r merupakan nilai batasan threshold terhadap principal curvature. Nilai r bernilai sama dengan 10 seperti yang digunakan oleh David G. Lowe pada percobaannya[4]. Nilai rasio principal curvature yang berada diatas threshold akan dieliminasi karena berada pada tepian.

## 4. Orientation Assignment

Proses selanjutnya adalah menentukan orientasi dari setiap *keypoint*. Penentuan orientasi ini dilakukan dengan cara memilih *keypoint* dan tetangganya untuk dihitung magnitude dan orientasinya. *Magnitude* direpresentasikan dengan *m* dan *orientation* direpresentasikan dengan θ. *Magnitude* dapat dicari dengan menggunaan persamaan (2.8) dan *orientation* dapat dicari dengan menggunakan (2.9).

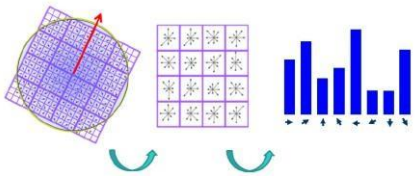
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.8)$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \quad (2.9)$$

Untuk mengetahui orientasi gambar, dibuat sebuah histogram dimana bobotnya adalah nilai *magnitude* pada perhitungan sebelumnya. Histogram dibentuk dengan mengkuantisasi orientasi pada skala 360 derajat sehingga tercipta 36 bin atau batang histogram. Arah orientasi yang dominan akan didapatkan dengan melihat bin yang memiliki nilai tertinggi pada histogram.

## 5. Keypoint Descriptor

Langkah terakhir dalam ASIFT adalah mendefinisikan *keypoint descriptor* pada area lokal gambar. Descriptor ini berbentuk area 16x16 yang dibagi ke dalam 4x4 sub area. Dalam setiap sub area, terdapat 8 arah gradient *keypoint*. Pada setiap sub area, dihitung nilai magnitude setiap keypoint untuk menentukan arah yang dominan. Sebelum dilakukan perhitungan magnitude untuk descriptor, terlebih dahulu magnitude yang dihasilkan pada proses sebelumnya dilakukan konvolusi Gaussian dengan sigma 1.5. Ini dimaksudkan agar semakin jauh dari sebuah ciri, nilai magnitudenya semakin kecil. Setelah itu dilakukan penentuan arah dengan membuat histogram yang sama seperti pada tahap sebelumnya dengan jumlah bin sama dengan jumlah arah gradient, yaitu 8 bin. Dengan demikian, jumlah atau dimensi vektor dari ekstraksi ciri ASIFT adalah 128 dengan perhitungan 4x4x8 = 128. Dengan digunakannya arah gradient gambar sebagai descriptor, maka ini akan membuat keypoint invariant terhadap perubahan kontras dan illuminasi cahaya yang tidak terlalu tinggi[6]. Setelah proses perhitungan nilai vector ciri untuk setiap keypoint selesai, nilai magnitude atau vector ciri yang masuk terlebih dahulu melalui proses normalisasi. Hal ini dilakukan untuk mengurangi efek perubahan kontras dan perubahan cahaya yang tidak terlalu ekstrim. Selain itu, untuk mengatasi pengaruh adanya gradient yang bernilai besar yang dikarenakan oleh saturasi dari kamera atau perubahan iluminasi sebagai akibat pantulan cahaya, kita lakukan thresholding sebesar 0.2. Jadi nilai gradient yang diatas 0.2 akan diubah menjadi sama dengan 0.2 lalu dilakukan proses normalisasi dengan vektor unit kembali.



Gambar 5.1. Keypoint Descriptor[7]

### B. *Tracking*

Mean shift adalah sebuah metode object tracking berbasis kepadatan atau density sebuah ciri. Dalam penelitian kali ini, mean shift menggunakan warna sebagai acuan tracking-nya. Mean shift banyak digunakan untuk melakukan tracking terhadap objek yang bersifat non rigid atau dengan kata lain bentuk dari objek senantiasa berubah. Pada mean shift objek non rigid dideskripsikan dengan histogram warna sehingga objek dapat tetap terdeteksi walaupun objek berubah bentuk sekalipun.

Proses kerja pada mean shift adalah melakukan konvolusi gambar atau frame video dalam window tertentu yang dipilih menggunakan sebuah window yang dinamakan parzen window. Parzen window atau disebut juga kernel density estimation adalah sebuah kernel yang digunakan untuk perhitungan density dimana sebelumnya dilakukan perataan atau smoothing histogram warna. Smoothing histogram tersebut dilakukan agar pada saat pencarian density tersebut tidak didapatkan kepadatan yang salah. Konvolusi pada parzen window dapat menggunakan banyak fungsi seperti Gaussian kernel, Uniform kernel, Epanechnikov kernel, dan lain-lain. Dalam penelitian ini, digunakan Gaussian kernel untuk konvolusinya. Perhitungan density menggunakan persamaan (2.10) dimana K adalah parzen window dan h adalah ukuran window yang digunakan[1].

$$f(x) = \frac{1}{nh^d} + \sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right) \quad (2.10)$$

Setelah didapatkan nilai density pada frame tersebut, selanjutnya adalah menghitung nilai gradient density yang sama, namun bedanya adalah perhitungan dilakukan pada frame selanjutnya. Setelah didapatkan nilai gradient density pada kedua frame (frame i dan frame i+1), kedua nilai tersebut dihitung kemiripannya atau similarity kedua nilai tersebut. Perhitungan similarity tersebut menggunakan persamaan (2.11) dimana $p_u$ adalah nilai gradient density pada frame kedua dan $q_u$ adalah nilai gradient density pada frame awal.

$$\rho(y) = \cos\theta_y = \sum_{u=1}^{m} \sqrt{p_u(y) q_u} \quad (2.11)$$

Untuk menentukan bahwa gradient density tersebut mirip atau tidak kita gunakan jarak Bhattacharyya Coefficient ( $\rho(y)$ ) yang berarti semakin kecil jarak yang dihasilkan maka akan semakin mirip kedua gradient density tersebut. Perhitungan jarak Bhattacharyya dapat menggunakan persamaan (2.12).

$$d(y) = \sqrt{1 - \rho(y)} \quad (2.12)$$

Apabila nilai jarak Bhattacharyya kurang dari threshold yang digunakan maka tracking akan di-update pada posisi yang menghasilkan similarity terdekat tersebut. Proses berjalan terus menerus sampai pada frame terakhir.

### C. *Image Matching*

Best bin first (BBF) merupakan variansi dari kNN dimana proses kerja dari BBF sangat mirip dengan kNN. Perbedaan antara BBF dan kNN hanya terletak dari perhitungan jarak antar ciri. Apabila kNN menggunakan Euclidean Distance sebagai tolak ukur kedekatan antar dua ciri maka, BBF menggunakan jumlah selisih tiap bin antara dua ciri atau dalam tugas akhir ini keypoint yang akan dihitung kedekatannya. Pada proses perhitungannya, sebelum selisih tiap bin dijumlahkan terlebih dahulu dilakukan operasi kuadrat. Hal ini dimaksudkan agar tidak dihasilkan nilai negatif pada penjumlahan nantinya. Setelah nilai selisih bin dijumlahkan, maka akan dilakukan perbandingan nilai. Nilai yang paling kecil akan dianggap sebagai tetangga terdekat. Berikut pseudocode dari best bin first.

```
matches = []
For each descriptor k1 in image 1:
    closest_match_distance = Infinity
    second_closest_match_distance = Infinity
    best_match = None
    For each descriptor k2 in image 2:
        distance_squared = d(k1, k2)
        if (distance_squared < closest_match_distance):
            second_closest_match_distance = closest_match_distance
            closest_match_distance = distance_squared
            best_match = k2
    If (threshold * closest_match_distance <
      second_closest_match_distance AND best_match != None):
        matches.Insert((k1, best_match, closest_match_distance))
return matches
```

Gambar 2.6 Pseudocode Best Bin First

## III. GAMBARAN SISTEM DAN SKENARIO PENGUJIAN

Dalam implementasinya, sistem akan diuji dalam dua tahap yaitu tahap *matching* dan tahap *tracking*. Tahap *matching* digunakan untuk mencari parameter-parameter yang optimal dalam melakukan *matching* objek ke frame video. Pengujian *matching* menggunakan dua buah gambar yang akan dicocokan. Tahap *tracking* digunakan untuk melihat performansi metode yang digunakan dengan melacak pergerakan objek yang membentuk huruf. *Tracking* dianggap berhasil apabila *bounding box* menghasilkan jalur yang sesuai dengan bentuk huruf yang diharapkan. Pengujian tersebut dilakukan dengan tujuan mengetahui apakah metode ASIFT dan mean shift yang diterapkan bisa melacak objek dalam keadaaan yang beraneka macam. Secara umum tahapan proses sistem yang dibuat dapat dilihat pada gambar 3.1.

### 3.1. Preprocessing

Pada tahapan ini, inputan sistem yang berupa gambar akan melalui proses pengolahan sebelum siap untuk diolah lebih lanjut pada *feature extraction*. sistem akan melakukan ekstraksi ciri dan pencocokan ciri pada frame video yang sedang diputar. Hal ini bertujuan agar objek yang dipilih oleh

user dapat selalu terlacak dengan proses yang berulang-ulang tersebut.

1. *Grayscaling* Gambar

Sebelum gambar masuk ke proses *feature extraction*, gambar akan diubah terlebih dahulu menjadi gambar *grayscale*. Hal ini bertujuan agar mempermudah dalam pengolahan nilai-nilai yang berada pada matriks gambar tersebut. Proses ini menggunakan persamaan sebagai berikut.

$$Y = 0.2126R + 0.7152G + 0.0722B \quad (3.1)$$

Persamaan (3.1) merupakan persamaan yang merepresentasikan kinerja mata manusia (*trichromat humans eyes*). Dimana mata manusia sangat sensitif terhadap warna hijau dan kurang sensitif terhadap warna biru. Sehingga gambar *grayscale* yang dihasilkan dari persamaan (3.1) merupakan hasil yang cukup untuk digunakan pada proses selanjutnya.

2. *Histogram Equalization*

Ekualiasasi histogram adalah sebuah cara untuk mengatur intensitas dan meningkatkan kontras sebuah gambar. Teknik yang digunakan pada histogram ekualisasi adalah menyetarakan histogram sebuah gambar berdasarkan sebuah garis linier. Histogram ekualisasi menggunakan fungsi distribusi kumulatif sebagai perhitungan perataan nilai histogram.

### 3.2. Feature Extraction

Pada proses ini, gambar grayscale akan dicari cirinya dengan metode ASIFT. Metode ini mempunyai beberapa tahapan mulai dari deteksi ciri sampai dihasilkan vektor ciri. Simulasi *Tilt* dan simulasi *Orientation*

Pada proses ini, sistem akan melakukan simulasi segala perubahan sudut pandang yang memungkinkan. Simulasi tersebut dilakukan dengan mengubah nilai longitude (rotasi) dan latitude (tilt). Nilai rotasi yang dapat disimulasikan berkisar antara 0,…, b / t < 180°. Nilai b = 72° dan *k* adalah nilai integer sedangkan t adalah nilai tilt yang digunakan pada proses perulangan. Apabila nilai rotasi sama dengan 90° maka tidak perlu dilakukan simulasi karena pada posisi tersebut gambar sama dengan gambar yang normal. Nilai tilt yang digunakan dalam simulasi berkisar antara 1, a, $a^2$, …, $a^n$ dimana a = $\sqrt{2}$ karena nilai tersebut dinilai menghasilkan nilai optimal dalam segi akurasi. Nilai n berdasarkan batas minimal dan maksmal nilai tilt (min = 1 dan max = $4\sqrt{2}$) sebanyak 5.

*1.*

embuat *Scale space pyramid* dan menghitung (*DoG*)

Proses ini merupakan proses pendeteksian interest point atau yang nantinya disebut sebagai keypoint. Pendeteksian dilakukan dengan mencari nilai extrema pada matriks gambar. Nilai extrema ini dapat berupa nilai tertinggi atau nilai terendah dilihat dari nilai tetangga sekitarnya. Namun sebelum mencari nilai extrema tersebut, gambar yang akan dicari keypoint-nya terlebih dahulu kita buat pyramid gambar dari gambar tersebut. Dalam sebuah pyramid, gambar yang diolah memiliki beberapa oktaf gambar yang setiap oktafnya merupakan hasil down sample atau down resize sebesar setengah dari ukuran oktaf sebelumnya.

Dalam tugas akhir ini, jumlah oktaf yang digunakan sebanyak tiga oktaf. Penentuan jumlah oktaf yang digunakan sebenarnya bebas ditentukan jumlahnya. Semakin banyak oktaf yang digunakan maka akan menghasilkan ciri yang banyak namun ciri-ciri tersebut sangat sensitif dengan adanya noise sehingga penggunaan oktaf yang terlalu banyak juga tidak dianjurkan. Setiap oktaf memiliki beberapa skala yang berbeda. Maksud dari skala ini adalah setiap oktaf memiliki gambar yang dikonvolusi dengan skala atau tingkat blur yang berbeda. Perbedaan tingkat blur ditentukan dengan faktor *k* dikalikan dengan σ (sigma), dimana $k = 2^{1/s}$ dan σ = 1.6. Nilai *k* akan naik seiring dengan naiknya interval dari skala gambar. Dalam satu oktaf dianjurkan memiliki jumlah skala sebesar *S* = *s* + 3. Lowe menyarankan untuk menggunakan jumlah *s* sebanyak 3. Hal itu dikarenakan dalam percobaan yang dilakukan oleh Lowe mengindikasikan bahwa *s* sebanyak 3 memiliki hasil yang optimum sehingga dalam satu oktaf akan terdapat 6 buah gambar yang telah dikonvolusi dengan skala yang berbeda satu sama lainnya.

Setelah didapatkan pyramid gambar, maka proses selanjutnya adalah menghitung nilai Difference of Gaussian-nya. Nilai Difference of Gaussian atau DoG didapatkan dengan cara mengurangi gambar pada skala atas dengan gambar yang berada dibawahnya. Misal gambar pada interval 1 dikurangi dengan gambar pada interval dst.

2.

encari nilai *extrema* ( *Extrema Detection* )

Setelah semua interval dihitung nilai DoG nya maka langkah selanjutnya adalah mencari nilai extrema pada setiap oktaf dengan membandingkan tiap piksel dengan tetangga sekitarnya yang berjumlah 8 dan tetangga yang berada pada skala atas dan skala bawahnya (lihat gambar 2.4.). Apabila nilai piksel tersebut merupakan nilai yang tertinggi atau terendah maka piksel tersebut dijadikan sebagai keypoint.

*3.*

liminasi *keypoint*

Pada proses ini, *keypoint* yang telah didapat akan difilter dengan melakukan pengecekan nilai setiap keypoint. *Keypoint* yang memiliki *low contrast* dan cenderung berada pada tepian akan dieliminasi karena *keypoint* tersebut cenderung tidak stabil sehingga harus dieliminasi. Proses eliminasi pertama akan dihilangkan *keypoint* yang memiliki *low contrast* dengan menggunakan *Quadratic Taylor* dari DoG dengan mensubstitusikan persamaan (2.4) ke persamaan (2.3). Nilai |D(*xhat*)| yang kurang dari 0.03 akan dieliminasi.

. Selanjutnya adalah mengeliminasi keypoint yang berada pada tepian dengan menggunakan *Hessian matrix* dengan menghitung nilai *principal curvature*-nya dengan

menggunakan persamaan (2.6) dan persamaan (2.7). Apabila nilai rasio principal curvature berada dibawah threshold, maka keypoint tersebut akan dieliminasi.

*4.*

itung nilai *Magnitude* dan *Orientation*

Setelah didapatkan *keypoint* yang stabil, maka lagkah selanjutnya adalah menghitung *magnitude* dan *orientation* dari gambar. Perhitungan menggunakan persamaan (2.8) untuk menghitung magnitude dan persamaan (2.9) untuk menghitung *orientation* dari gambar. *Keypoint* yang sudah didapat pada proses selanjutnya akan dihitung nilai *magnitude* dan *orientation*-nya dengan melakukan pembobotan disekitar keypoint dengan maksud semakin jauh letak nilai dari pusat keypoint maka nilai tersebut akan semakin kecil. Pembobotan dilakukan dengan melakukan konvolusi Gaussian filter dengan mask 9x9 dengan posisi keypoint sebagai pusatnya. Mask 9x9 digunakan karena sigma yang digunakan pada saat pembobotan sebesar 1.5 sehingga apabila mengikuti *3-sigma rule* akan didapatkan mask sebesar 9x9.

Setelah itu bobot yang dihitung tersebut akan diambil sebagai *magnitude* dan *orientation* dari keypoint yang sebagai pusatnya. Nilai bobot tersebut akan dimasukkan kedalam sebuah histogram yang dibentuk dengan mengkuantisasi orientasi pada skala 360 derajat sehingga tercipta 36 bin atau batang histogram. Arah orientasi yang dominan akan didapatkan dengan melihat bin yang memiliki nilai tertinggi pada histogram. Apabila sebuah histogram memiliki nilai yang lebih dari 0.8 itu mengindikasikan adanya *multiple orientation* sehingga keypoint pada posisi tersebut akan diduplikasi pada posisi yang sama namun berbeda pada arah orientasinya.

5.

itung vektor ciri ( *keypoint descriptor* )

Pembentukan descriptor pada ASIFT pada dasarnya sama dengan cara kerja *orientation assignment* yang membedakan antara keduanya adalah ukuran masking dan jumlah bin yang digunakan. Pada tahap ini ukuran masking yang digunakan lebih besar yaitu 16x16 yang nantinya akan dibagi menjadi ukuran yg lebih kecil yaitu 4x4 sehingga akan didapatkan 4 sub mask 4x4. Jumlah bin yang digunakan pada tahap ini juga lebih sedikit yaitu 8 buah bin. 8 bin ini merepresentasikan 8 arah orientation atau gradient arah dari setiap piksel dalam mask 4x4 tersebut. Dengan demikian jumlah vektor ciri yang akan dimiliki oleh setiap keypoint sama dengan 4x4x8 = 128 vektor ciri. Setelah didapatkan vektor ciri, nilai vektor tersebut akan dinormalisasi berdasarkan panjang unit vektornya . Hal ini digunakan untuk mengurangi perubahan illuminasi yang tidak terlalu besar. Hasil normalisasi yang nilainya lebih dari 0.2 akan diubah nilainya menjadi 0.2. Ini dimaksudkan untuk menghindari adanya gradient nilai yang terlalu besar. Nilai ini kemudian akan dinormalisasi kembali berdasarkan panjang unit vektornya.

### 3.3 Feature Matching

Proses Matching pada tugas akhir ini menggunakan sebuah algoritma variansi dari *k-nearest neighbor* (kNN) yang disebut dengan *Best Bin First* (BBF). Jika dalam kNN nilai kedekatan antar ciri dihitung dengan Euclidean distance, maka pada BBF nilai kedekatan dihitung dari jumlah selisih tiap bin antara dua ciri. Setiap keypoint pada gambar pertama akan dihitung nilai kedekatannya dengan setiap keypoint pada gambar kedua, kemudian akan dipilih nilai terkecil (*best*) dan nilai terkecil kedua (*second_best*). Setelah itu kita bandingkan nilai best dan second_best dikalikan sebuah threshold. Apabila nilai best lebih kecil, maka keypoint antar dua gambar yang dibandingkan dianggap match. Nilai threshold yang digunakan berkisar antara 0.1 – 1 dan dilakukan pengujian untuk menentukan nilai threshold yang paling optimal.

Setelah didapatkan keypoint yang match, proses selanjunya adalah membuat *bounding box* pada area sekitar keypoint. Pembuatan *bounding box* tidak bisa dilakukan secara sembarang karena letak keypoint yang biasanya tersebar. Oleh karena itu, *bounding box* akan dibuat apabila memenuhi sebuah kondisi dimana dalam radius *bounding box* harus terdapat minimal 3 atau lebih keypoint sehingga *bounding box* lebih bisa akurat penempatannya.

### 3.4 *Tracking*

Proses *tracking* dimulai tergantung ada atau tidaknya *bounding box* hasil pada proses matching sebelumnya. Jika tidak ada maka, sistem tidak akan menjalankan *tracking*, apabila ada maka sistem akan menjalankan *tracking* dimulai dari dengan melakukan konvolusi nilai yang berada dalam *bounding box* dengan Gaussian filter. Gaussian filter tersebut dapat disebut juga sebagai parzen windows yang digunakan untuk melakukan smoothing terhadap data agar dalam perhitungan *density* nantinya tidak terjadi salah penentuan objek. Setelah gambar di-smoothing gambar akan dihitung nilai *density*-nya. Nilai *density* ini lah yang nantinya akan dicari kemiripan dengan nilai *density* pada frame berikutnya. Pencarian *similarity* ini akan terus berlanjut hingga tercapainya salah satu kondisi dimana kondisinya adalah nilai *similarity* tidak melebihi 0.2 dan jumlah iterasi tidak melebihi iterasi yang telah ditentukan sebelumnya. Nilai 0.2 diambil karena dalam beberapa sumber nilai tersebut dapat mencapai *density* yang tepat secara optimal. Setelah proses selesai maka selesai pula proses *tracking*.

IV. PENGUJIAN

## 4.1 Pengujian Sistem

Pengujian sistem dilakukan terkait dengan proses observasi dari parameter-parameter yang dirancang untuk mengetahui tingkat akurasi dari metode yang digunakan pada penelitian ini. Parameter yang akan dilakukan observasi adalah matching objek berdasarkan *threshold* dan radius *bounding box*, untuk dicari *threshold* dan radius berapakah yang paling sesuai untuk *bounding box*.

Pengujian pada sistem kali ini akan dibagi menjadi dua tahap yaitu tahap *matching* dan tahap *tracking*.Tahap *matching* dilakukan untuk menentukan parameter threshold dan radius yang optimal sehingga pada saat pengujian *tracking* didapatkan hasil yang optimal pula. Sedangkan pada tahap *tracking*, akan dilakukan pengujian untuk melihat hasil dari implementasi metode ASIFT-Mean Shift dan membandingkannya dengan metode Full ASIFT sehingga dapat terdapat perbandingan nilai performansi.Implementasi Sistem

**1) Matching**

Pada pengujian *matching*, data yang digunakan adalah kumpulan foto dari sebuah objek dengan variansi sudut pandang yang berbeda. Variansi dari sudut pandang tersebut berkisar antara 0-180 dengan kelipatan 30 (0 – 30 … 180). Sehingga akan didapatkan 7 buah gambar setiap objeknya dengan sudut yang berbeda. Objek yang digunakan ada 3 buah yaitu, wajah, box, dan botol dimana ketiga benda tersebut selain diambil dari berbagai sudut juga diambil dengan lingkungan yang terkontrol (*controlled environment*) dan tidak terkontrol (*uncontrolled environment*). Lingkungan terkontrol diambil di dalam sebuah ruangan dengan pencahayaan yang diatur sedangkan lingkungan tidak terkontrol akan diambil di luar ruangan dengan pencahayaan alami. Setiap gambar akan dilakukan ektraksi ciri menggunakan metode ASIFT dan kemudian akan dicocokan dengan menggunakan BBF.

Proses pencocokan ciri akan dilakukan dengan variansi threshold yang berkisar antara 0 – 1 pada setiap gambar. Selain parameter threshold, parameter radius juga akan diuji dengan variansi 6% - 10% dari ukuran gambar. Sebuah objek dikatakan *match* apabila minimal ada 3 buah *keypoint* yang berada pada radius *bounding box* dan minimal 50% bagian benda berada didalam *bounding box* tersebut atau minimal 50% luas *bounding box* terisi oleh objek Apabila bagian benda yang berada didalam *bounding box* kurang dari 50% atau tidak pada objek yang seharusnya maka dikatakan *mismatch*.

Hasil dari pengujian *matching* (skenario 1 dan skenario 2) ini berupa nilai threshold dan radius yang dianggap optimal berdasarkan pengujian yang telah dilakukan sebelumnya. Hasil dari pengujian ini akan diterapkan pada pengujian selanjutnya yaitu pengujian *tracking*.

**2) Tracking**

Pengujian *tracking* menggunakan video sebagai datanya. Data yang digunakan berupa video yang beresolusi 640x480 piksel dengan durasi berkisar antara 8-10 detik dan setiap detik terdiri dari *frame* yang berkisar 50 *frame*. Data tersebut akan dilakukan proses *matching* terlebih dahulu dengan menggunakan nilai parameter-parameter yang dianggap menghasilkan hasil yang optimal. Nilai parameter-parameter tersebut didapat pada tahap sebelumnya yaitu pengujian *matching*. Selanjutnya hasil *matching* yang berupa *bounding box* akan dilakukan *tracking* menggunakan metode *mean shift*. Objek akan digerakan membentuk sesuatu seperti huruf atau sebuah garis lurus. Objek yang digunakan adalah objek yang dapat dikategorikan sebagai senjata seperti *cutter*, gunting, pisau, pisau lipat, dan pistol. Lingkungan yang digunakan pun sama yaitu lingkungan terkontrol (*controlled environment*) dan tidak terkontrol (*uncontrolled environment*).

Objek akan digerakan dengan variansi sudut dan membentuk alfabet 'L'. *Tracking* akan dianggap benar apabila jalur *bounding box* sama atau menyerupai bentuk yang telah dibuat tersebut. *Tracking* akan dianggap salah apabila gerakkan *bounding box* tidak menyerupai bentuk atau tidak ada *bounding box* yang tercipta dengan kata lain objek tidak terdeteksi. Pada jalur hasil *tracking* ditetapkan toleransi pergerakan yang dikarenakan oleh posisi *bounding box* yang dapat berpindah posisi namun masih berada pada objek. Nilai toleransi yang ditentukan adalah 20 dari tepian *bounding box*. Apabila objek bergerak ke bawah atau ke atas maka, hanya nilai sumbu Y yang diberikan toleransi. Apabila objek bergerak ke kanan atau ke kiri maka, hanya nilai sumbu X yang diberikan toleransi.

### 4.2 Tujuan Pengujian

Tujuan dilakukannya pengujian ini adalah :

1. Menentukan nilai threshold yang paling sesuai dan optimal untuk melakukan *matching* dengan benar.
2. Menetukan radius yang optimal dalam penentuaan letak *bounding box* sehingga didapatkan objek yang optimal dan menjadi inputan yang baik untuk tracking.

#### 4.2.1 Analisis Skenario Pengujian 1

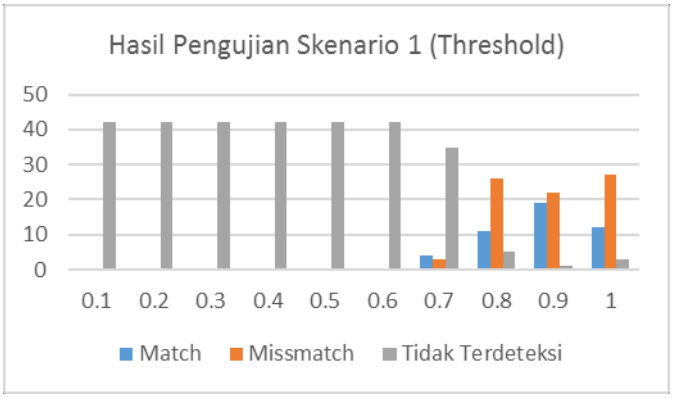
Pada pengujian skenario 1 akan dilakukan pengujian matching dengan tujuan untuk mengetahui parameter threshold yang menghasilkan hasil ang optimal, gambar query (Gambar yang dijadikan acuan matching) akan dicocokan dengan gambar data (Gambar yang akan dicari kecocokannya). Gambar data merupakan gambar dari sebuah objek yang telah diambil dengan variansi sudut pandang yang berbeda. Interval variansi sudut berkisar antara $0^o$ – $180^o$ dengan penambahan sudut berkelipatan $30^o$ sehingga

didapatkan 7 buah gambar dari satu objek. Objek yang digunakan ada 3 buah dan diambil dari kondisi lingkungan yang berbeda yaitu lingkungan terkontrol dan lingkungan tidak terkontrol. Setiap gambar data akan dicocokan dengan gambar query dengan threshold yang berbeda-beda dengan kisaran nilai 0-1. Hasil pengujian dapat dilihat pada tabel 4.1.

Tabel 5.1. *Hasil pengujian* threshold

| Threshold | Tidak Terdeteksi | Missmatch | Match |
|---|---|---|---|
| 0.1 | 42 | 0 | 0 |
| 0.2 | 42 | 0 | 0 |
| 0.3 | 42 | 0 | 0 |
| 0.4 | 42 | 0 | 0 |
| 0.5 | 42 | 0 | 0 |
| 0.6 | 42 | 0 | 0 |
| 0.7 | 35 | 3 | 4 |
| 0.8 | 5 | 26 | 11 |
| 0.9 | 1 | 22 | 19 |
| 1 | 3 | 27 | 12 |

Tabel 4.1 diatas menampilkan data hasil pengujian matching gambar dengan berbagai variansi threshold. Berdasarkan tabel diatas terlihat bahwa ada 3 kondisi yang muncul pada saat pengujian matching yaitu tidak terdeteksi, *missmatch*, dan *match*. Kondisi tidak terdeteksi terjadi apabila tidak ada bounding box yang muncul pada saat melakukan matching, kondisi missmatch terjadi apabila bounding box yang muncul tidak tepat pada objek yang seharusnya atau bounding box tidak mencakup 50% dari benda sehingga dikategorikan missmatch, sedangkan kondisi match terjadi apabila bounding box yang muncul sesuai dengan benda yang seharusnya atau bounding box mencakup 50% luas objek. Perlu ditambahkan pula, kodisi match dapat juga tercapai apabila lebih dari 50% luas bounding box terisi atau tercakup oleh benda yang seharusnya. Hal itu dapat terjadi apabila benda yang dituju terlalu besar ukurannya atau ukuran bounding box yang terlalu kecil. Berdasarkan data pada tabel 4.1 dapat dijadikan sebuah grafik yang menunjukan threshold mana yang menghasilkan performa yang optimal. Hasil tersebut dapat dilihat pada gambar 4.3.




Gambar 5.2. Grafik Hasil Pengujian 1

Dapat dilihat bahwa secara keseluruhan jumlah objek dengan variansi sudut pandang yang berbeda serta dalam kondisi lungkangan yang berbeda, threshold 0.9 menghasilkan nilai match yang paling tinggi dari threshold yang lainnya dan pada grafik ditunjukkan dengan trand naik pada yang puncak tertingginya berada pada threshold dengan nilai 0.9. Selain itu, kita juga harus memperhatikan bahwa pada garis *missmatch*, terlihat trend turun pada threshold 0.9 yang berbeda dengan threshold lainnya yang menunjukkan trend naik pada garis *missmatch*. Grafik tersebut mengindikasikan bahwa threshold 0.9 merupakan nilai threshold yang paling optimal secara keseluruhan. Sehingga didapatkan bahwa threshold dengan nilai 0.9 merupakan threshold yang optimal.

### 4.2.2 Analisis Skenario Pengujian 2

Pada pengujian skenario 2 akan diuji radius *bounding box* untuk matching yang optimal. Pada pengujian sebelumnya radius yang digunakan adalah nilai terbesar yang akan diujikan yaitu 10% dari ukuran gambar data. Pada skenario 2, langkah -langkah pengujian yang akan dilakukan hampir sama dengan pengujian 1 yang membedakan hanya parameter yang diujikan. Parameter radius akan divariansikan persentasenya dengan kisaran 6% - 10% dari ukuran gambar data. Threshold yang akan digunakan pada pengujian ini adalah 0.9 karena berdasarkan pengujian sebelumnya threshold dengan nilai 0.9 menghasilkan nilai *match* yang optimal.Hasil pengujian pada skenario 2 dapat dilihat pada tabel 4.2.

Tabel 5.2. Hasil Pengujian Radius

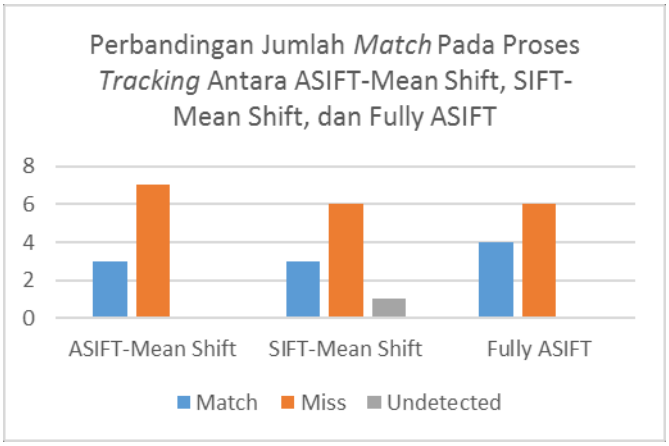
| Radius | Missmatch | Match |
|---|---|---|
| 6 | 26 | 16 |
| 7 | 26 | 16 |
| 8 | 25 | 17 |
| 9 | 25 | 17 |
| 10 | 22 | 20 |

Tabel 4.2 menampilkan hasil pengujian radius dengan dua buah kondisi yaitu missmatch dan match. Pada pengujian kali ini akan dicari nilai radius yang menghasilkan nilai match yang paling tinggi. Berdasarkan tabel diatas terlihat bahwa semakin besar radius yng digunakan maka akan semakin besar pula nilai match yang dihasilkan. Ini dikarenakan jumlah ciri yang tercakup dalam radius semakin banyak dan radius dalam gambar yang tercakup juga semakin besar sehingga jumlah nilai match akan semakin banyak. Dari tabel 4.2 juga dapat kita analisa bahwa kerapatan atau kedekatan posisi antar ciri juga mempengaruhi radius dan juga hasil pencocokan. Apabila kerapatan antar ciri pada objek yang dituju kecil maka radius yang digunakan sebaiknya dengan nilai radius yang kecil sebaliknya kerapatan ciri yang renggang akan membuat radius yang digunakan sebaiknya yang besar. Namun perlu diketahui bahwa kerapatan ciri yang renggang atau tersebar pada gambar, mengindikasikan bahwa

kemungkinan objek yang dituju tidak ada pada gambar data yang kita cocokan dengan kata lain kemungkinan terjadinya missmatch juga besar.

### 4.2.3 Analisis Skenario Pengujian 3

Pada pengujian skenario 3 dilakukan pengujian *tracking* terhadap metode ASIFT-Mean shift. Pada pengujian kali ini, objek akan digerakan dengan berbagai variansi gerakan untuk dapat menguji metode yang digunakan. Video yang digunakan berdurasi sekitar 8 – 10 detik dengan jumlah frame per detik sekitar 50 frame. Hasil tracking dari metode ASIFT-Mean shift tersebut akan dibandingkan dengan hasil pengujian metode lain dengan menggunakan data yang sama. Gambar 4.4 menunjukan perbandingan jumlah objek yang *match* atau memiliki jalur yang sesuai dengan yang telah ditentukan pada proses *tracking*.




Gambar 5.3. Grafik Perbandingan Jumlah *Match* Pada Proses *Tracking* Antara ASIFT-Mean Shift, SIFT-Mean Shift dan Fully ASIFT.

Berdasarkan grafik diatas dapat kita lihat bahwa metode SIFT-Mean Shift dan ASIFT-Mean Shift mempunyai jumlah *match* yang sama, namun pada metode SIFT-Mean Shift terdapat objek yang tidak terdeteksi. Hal ini dikarenakan pada metode SIFT-Mean Shift menghasilkan ciri yang cocok lebih sedikit dari batasan jumlah ciri yang cocok yang telah ditentukan sebelumnya sehingga objek tersebut dianggap tidak terdeteksi oleh sistem. Berbeda dengan metode SIFT-Mean Shift, metode ASIFT-Mean Shift dapat mendeteksi semua objek pada data yang diuji. Hal ini dikarenakan metode ASIFT-Mean Shift dapat mendeteksi ciri yang cocok lebih banyak daripada metode SIFT-Mean Shift. Banyaknya ciri yang terdeteksi disebabkan pada metode ASIFT-Mean Shift gambar disimulasi berdasarkan sudut pandang sehingga ciri cocok didapatkan jauh lebih banyak dibandingkan dengan SIFT-Mean Shift.

Dilihat dari jumlah banyaknya *match* pada proses tracking, metode Fully ASIFT memiliki jumlah match terbanyak. Hal itu dikarenakan proses *tracking* pada metode Fully ASIFT dilakukan dengan *matching* secara terus menerus terhadap setiap frame video sehingga hasilnya lebih baik dibandingkan dengan kedua metode lainnya. Pada metode ASIFT-Mean Shift dan SIFT-Mean Shift, proses *tracking* dijalankan dengan menggunakan metode Mean Shift yang berbasiskan warna sehingga jalur *tracking* dapat saja berubah-ubah sesuai dengan keadaan lingkungan objek.

Waktu yang dibutuhkan untuk menjalankan tracking juga menjadi salah satu yang dijadikan acuan dalam tracking karena metode dengan waktu proses yang lama tidak dapat diimplementasikan dengan baik. Tabel 4.3 menunjukan jumlah waktu yang dibutuhkan oleh setiap metode dalam menjalankan proses *tracking*.

Tabel 5.3. Perbandingan Jumlah Waktu Pada Proses Tracking

| Metode | Objek | Indoor Waktu (detik) | Outdoor Waktu (detik) |
|---|---|---|---|
| ASIFT-Mean Shift | Cutter | 335 | 1518 |
| | Gunting | 377 | 1764 |
| | Pisau | 298 | 1206 |
| | PisauLipat | 372 | 1355 |
| | Pistol | 255 | 1494 |
| SIFT-Mean Shift | Cutter | 323 | 201 |
| | Gunting | Undetected | 370 |
| | Pisau | 289 | 133 |
| | PisauLipat | 243 | 361 |
| | Pistol | 304 | 330 |
| Fully ASIFT | CutterBesar | 1623 | 15037 |
| | GuntingKecil | 3365 | 28058 |
| | PisauBesar | 2034 | 16507 |
| | PisauLipat | 2508 | 33496 |
| | PistolBesar | 6105 | 7219 |

Dari tabel 4.3 terlihat jelas bahwa metode Fully ASIFT memerlukan waktu proses yang terbanyak dibandingkan dengan dua metode lainnya. Hal ini dikarenakan pada metode Fully ASIFT proses *tracking* dilakukan dengan mencocokan gambar query dengan

setiap frame yang ada pada video atau dengan kata lain proses *matching* dilakukan secara berulang-ulang sehingga waktu komputasi sistem menjadi sangat lama. Sedangkan pada Metode ASIFT-Mean Shift atau SIFT-Mean Shift, proses *matching* dilakukan hanya sekali untuk mendeteksi keberadaan benda lalu proses *tracking*-nya dilakukan oleh metode Mean Shift yang waktu pemrosesannya lebih sederhana daripada FullyASIFT.

Berdasarkan hasil *tracking* yang telah terdapat pada gambar 4.4, dapat dihitung nilai akurasi masing-masing metode dengan perhitungan sederhana. Nilai akurasi masing-masing metode dapat dilihat pada tabel 4.4.

Tabel 5.4. Perbandingan Akurasi Metode

| Metode | Jumlah Match | Jumlah Missmatch | Akurasi |
|---|---|---|---|
| ASIFT-Mean Shift | 3 | 7 | 30% |
| SIFT-Mean Shift | 3 | 7 | 30% |
| Fully ASIFT | 4 | 6 | 40% |

Dapat dilihat pada tabel 4.4, bahwa akurasi metode ASIFT-Mean Shift dan SIFT-Mean Shift berjumlah sama yaitu sebesar 30%. Namun, perlu diingat bahwa pada metode SIFT-Mean Shift terdapat sebuah data atau objek yang tidak muncul *bounding box*-nya sehingga dikategorikan sebagai *undetected*. Hal itu terjadi karena jumlah ciri cocok yang dihasilkan terlalu sedikit untuk dianggap objek pada data tersebut terdeteksi. Berdasarkan tabel 4.4, metode Fully ASIFT memiliki akurasi yang paling tinggi dibandingkan metode yang lain yaitu sebesar 40%. Hal itu dikarenakan proses matching yang dilakukan secara terus menerus pada setiap frame video sehingga objek dapat terdeteksi secara baik.

## 5. Kesimpulan dan Saran

### 5.1 Kesimpulan

Kesimpulan yang didapatkan dalam pengerjaan tugas akhir ini adalah :

1. Semakin besar nilai parameter threshold yang digunakan maka akan semakin banyak pula hasil match pada proses *matching*. Pada tugas akhir ini, nilai yang optimal dengan menghasilkan hasil match yang tinggi adalah nilai threshold sama dengan 0.9
2. Penggunaan nilai parameter radius bergantung pada kerapatan ciri yang cocok pada gambar. Semakin kecil jarak antar ciri yang cocok, maka akan semakin kecil nilai radius yang digunakan. Berdasarkan pengujian didapatkan nilai radius yang optimal adalah 10% dari ukuran gambar.
3. Berdasarkan hasil pengujian pada skenario 3, dapat disimpulkan bahwa metode ASIFT-Mean Shift dapat mengatasi permasalahan perubahan *point of view* yang terjadi pada object tracking dimana nilai akurasi yang dihasilkan sama dengan 30% dan semua objek dapat dideteksi.

### 5.2 Saran

1. Dalam melakukan matching, sebaiknya tambahkan atribut-atribut lain pada ciri seperti keadaan sekitar ciri baik tetangga, warna, ataupun orientasi tetangganya
2. Gunakan radius yang adaptif terhadap ukuran objek agar pencarian nilai density pada mean shift tidak terlalu jauh sehingga menghasilkan tracking yang lebih efisien.
3. Pada proses pencarian gambar hasil simulasi *point of view* yang tepat dengan frame video pada saat pencocokan ciri, sebaiknya menggunakan sebuah nilai batasan atau threshold dimana apabila nilai kecocokan sebuah gambar hasil simulasi *point of view* pada saat pencarian gambar yang cocok kurang dari nilai batasan tersebut maka gambar tersebut langsung dianggap sebagai gambar yang cocok sehingga waktu komputasi dapat berkurang.
4. Sebaiknya gunakan bahasa pemrograman C atau C++ karena penggunaan bahasa pemrograman Matlab sangat jauh lebih lama daripada C atau C++.

## REFERENCES


[1] Comaniciu, Dorin, *et al.* (2000)."Real-Time Tracking of Non-Rigid Objects using Mean Shift". IEEE CVPR

[2] Dekaer, Moeammar. "Definisi Pengolahan Citra Digital" dalam http://moeammardekaer.blogspot.com/2013/11/pengolahan-citra-digital-menggunakan.html diakses pada Senin, 4 November 2013

[3] Hermawati, Fajar Astuti. (2013). *Pengolahan Citra Digital*. Yogyakarta : ANDI

[4] Lowe, David G.(2004). "Distinctive Image Features form Scale-Invariant Keypoints". Canada: Computer Science Department, University of British Columbia.

[5] Mathew, Namitha *et al*. (2013), "Certain Approaches of Real Time Object Tracking in Video Sequences on Embedded Linux Platform", International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075.

[6] Morel , Jean-Michel, dan Goushen Yu, (2009) "ASIFT: A New Framework for Fully Affine Invariant Image Comparison", SIAM Journal on Imaging Sciences, PP: 438-496.

[7] Oji, Reza, (2012) "An Automatic Algorithm for Object Recognition and Detection Based on ASIFT Keypoints", Signal & Image Processing: An International Journal (SIPIJ) Vol. 3 No. 5.

[8] Sari, Marlindia Ike, (2011) "Desain Segmentasi dan Pengenalan Karakter pada Plat Nomor Kendaraan", Prosiding Konferensi Nasional ICT-M Politeknik Telkom (KNIP).

[9] Yu, Goushen, dan Jean-Michel Morel. "ASIFT: An Algorithm for Fully Affine Invariant Comparison " dalam http://www.ipol.im/pub/art/2011/my-asift/ diakses pada Senin, 4 November 2013

[10] Zhou, Huiyou, *et al.* (2009) "Object Tracking Using SIFT Features and Mean Shift", Elsevier Computer Vision and Image Understanding 113, PP: 345-352