makalah_erwin_unsri.docx

WORD COUNT	3258	TIME SUBMITTED	18-JAN-2018 12:55PM
		PAPER ID	34457152

PELACAKAN OBJEK BERGERAK PADA VIDEO MENGGUNAKAN OPTIMASI ANNEALED GAUSSIAN PARTICLE SWARM OPTIMIZATION

Erwin[1], Diding Nuriska[2]

[1] Sistem Komputer, [2]Teknik Informatika, Fakultas Ilmu Komputer, Universitas Sriwijaya
, Indonesia
e-mail: [1]erwin@unsri.ac.id, [2]didingnuriska@ymail.com

Abstrak

Pelacakan objek pada video merupakan salah satu area permasalahan yang penting pada bidang *Computer Vision* dan memiliki tingkat kesulitan yang cukup tinggi karena harus dilakukan secara *realtime* dan faktor-faktor lain yang mempengaruhinya, seperti kualitas video, perubahan bentuk objek dan lain-lain, karena itu dibutuhkan metode yang dapat melakukan pelacakan objek yang memiliki akurasi yang baik. Pada penelitian ini digunakan metode optimasi *Annealed Gaussian Particle Swarm Optimization* (AGPSO) untuk melakukan pelacakan objek yang bergerak pada video. AGPSO merupakan varian dari metode *Particle Swarm Optimization* (PSO) yang menggunakan sekumpulan partikel yang bekerja secara iteratif untuk mencari solusi optimal. Sekumpulan partikel akan mencari objek pada setiap urutan frame di video. Pada setiap pencarian, partikel akan membentuk histogram menggunakan ruang warna *hue*. Fungsi fitness *Bhattacharyya Coefficient* dan toleransi kesalahan. Dari hasil penelitian dengan 12 data video diketahui bahwa PSO dapat melakukan pelacakan objek pada video dengan akurasi yang baik, yaitu dengan tingkat keberhasilan sebesar 93,4%.

Kata kunci : *Pelacakan Objek, Particle Swarm Optimization, AGPSO.*

I. Pendahuluan

Pelacakan objek pada video merupakan suatu bagian yang sangat penting dalam *Computer Vision* (CV), hal ini dikarenakan pelacakan objek berkaitan dengan banyak aplikasi dalam CV diantaranya adalah *automated video surveillance*, *augmented reality*, *intelligence transportation system*, *smart room* dan lain-lain. Pelacakan objek digunakan untuk melacak posisi objek yang bergerak pada setiap selang waktu pada video. Hasil dari pelacakan objek adalah posisi objek pada video pada waktu tertentu.

Pelacakan objek merupakan suatu area permasalahan yang cukup sulit karena bekerja secara *realtime*, sehingga dibutuhkan teknik yang mampu bekerja dengan akurasi yang baik. Akurasi dari proses pelacakan objek juga dipengaruhi oleh beberapa faktor, diantaranya adalah faktor pencahayaan, kualitas video, perubahan objek setiap selang waktu dan lain-lain. Beberapa penelitian tentang pelacakan objek antara lain adalah *Condensation Algorithm* [1] yang menggunakan teknik stokastik *factored sampling* untuk memprediksi posisi objek secara dinamis, dimana sekumpulan partikel yang memiliki bobot akan disebar dan dievaluasi dengan fungsi distribusi probabilitas yang telah ditentukan sebelumnya. Teknik ini cukup banyak digunakan dan dikembangkan oleh para peneliti karena memiliki akurasi yang baik dalam pelacakan objek. [2] menggunakan sekumpulan histogram berbobot yang dihitung menggunakan *circular region* untuk merepresentasikan objek. Teknik ini menggunakan *mean shift procedure* untuk mengestimasi posisi objek dengan menghitung nilai kesamaan histogram objek yang menggunakan teknik *Bhattacharyya Coefficient*. Teknik ini dikenal sebagai teknik *mean shift* yang memiliki akurasi yang cukup baik namun memiliki kelemahan dalam melacak objek yang bergerak cepat.

II. Annealed Gaussian Particle Swarm Optimization

A. Particle Swarm Optimzation

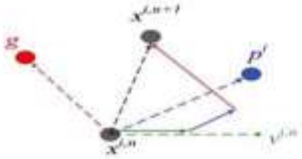
Particle Swarm) adalah salah satu algoritma pencarian dengan teknik acak yang mengadaptasi tingkah laku sosial dari sekumpulan burung yang bekerja sama dalam mencari makanan [3]. Algoritma ini pertama kali ditemukan oleh Kennedy dan Eberhart pada tahun 1995. PSO menggunakan sekumpulan partikel sejumlah N dibangkitkan secara acak. Masing-masing partikel memiliki kemampuan untuk mengolah informasi dan bekerja sama satu sama lain untuk menyelesaikan masalah pada dimensi ruang tertentu.

Menurut [4] *Swarm Intelligence* adalah suatu unit pengolah informasi yang memiliki kemampuan menyelesaikan masalah dengan berinteraksi satu sama lainnya. Unit pengolah informasi ini bisa dimodelkan dengan sekumpulan burung, serangga, elemen larik, dan lain-lain. PSO adalah salah satu contoh dari *swarm intelligence*, dimana partikel adalah unit pengolah

informasi dan memiliki kemampuan dalam menyelesaikan masalah dalam dimensi ruang masalah tertentu. Setiap partikel memilki posisi dan kecepatan dalam mencari penyelesaian masalah. Sekumpulan partikel tadi akan bergerak pada dimensi ruang masalah dengan kecepatan tertentu untuk mencari solusi terbaik. Perpindahan posisi dari suatu partikel ke posisi lainnya dipengaruhi oleh aspek kognetif (kecerdasan dirinya) dan aspek sosial. Aspek kognetif ini adalah posisi terbaik yang dihasilkan partikel dalam penyelesaian masalah yang dilambangkan dengan p*Best*, sedangkan aspek sosial adalah sebuah solusi yang paling *optimal* yang dihasilkan oleh partikel-partikel yang dilambangkan dengan g*Best*.

Untuk mengetahui posisi suatu objek, detection dan tracking saling terkait erat satu sama lain karena detection diperlukan pada fase awal sedangkan tracking digunakan pada fase selanjutnya dengan hasil yang cukup efektif[5].

Perpindahan posisi dan perubahan kecepatan dari partikel disimbolkan dengan vektor,



Gambar 1. Permodelan perpindahan posisi partikel dengan vektor [3]

Rumus untuk perpindahan dan perubahan kecepatan dari partikel adalah:

$$v^{i,n+i} = w^n v^{i,n} + \varphi_1 u_1 \left(pBest^i - x^{i,n}\right) + \varphi_2 u_2 \left(gBest - x^{i,n}\right) \quad (1)$$

$$x^{i,n+1} = x^{i,n} + v^{i,n+1} \quad (2)$$

Keterangan:

i	: indeks partikel
n	: indeks iterasi / pengulangan
x	: posisi partikel
v	: kecepatan partikel
p*Best*	: solusi terbaik dari partikel
g*Best*	: solusi global terbaik
w^n	: bobot inersia
u_1, u_2	: bilangan *random* antara (0-1)
φ_1	: faktor kognetif (kecerdasan individu)
φ_2	: faktor sosial (kecerdasan sosial)

Menurut [6] untuk merubah bobot inersia digunakan persamaan berikut:

$$w^n = (w^n - (w_{max} - w_{min})/T) \quad (3)$$

Keterangan:

w_{max} , w_{min}	: ditetapkan sebelumnya oleh user,
T	: iterasi maksimum.

Fungsi *fitness* adalah sebuah fungsi yang digunakan untuk mengevaluasi solusi yang dihasilkan dari partikel-partikel pada setiap iterasi. Fungsi *fitness* bergantung pada *domain* masalah yang akan diselesaikan dengan PSO ini.

Proses perubahan g*Best* dan p*Best* dilakukan setelah mengevaluasi solusi-solusi yang dihasilkan partikel dengan fungsi tertentu.

Rumus perubahan g*Best* (partikel dengan fungsi fitnes terbaik secara global) dan p*Best* (nilai terbaik yang dihasilkan oleh setiap partikel selama iterasi) adalah:

$$pBest^i = \begin{cases} x^{i,n} & (pBest^i \\ pBest^i & else \end{cases} \quad (4)$$

$$gBest = arg\ max\ f\ (pBest^i) \quad (5)$$

B. Annealed Gaussian Particle Swam Optimization

Menurut [3] teknik konvensional PSO masih memiliki kelemahan, diantaranya lambatnya konvergensi pada dimensi ruang pencarian yang tinggi dan banyaknya parameter yang harus diatur. *Annealed Gaussian* PSO (AGPSO) adalah varian dari PSO yang dikenalkan oleh [3]. AGPSO menggunakan *noise* yang dihasilkan dari distribusi *gaussian* untuk menghindari terjebaknya partikel pada *local optimal*. AGPSO hanya memiliki 1 parameter untuk diatur, yaitu *annealing constant* yang dilambangkan dengan c.

$$v^{i,n+1} = |r1\,(p^i - x^{i,n}) + |r2(g - x^{i,n}) + \eta \quad (6)$$

$$x^{i,n+1} = x^{i,n} + v^{i,n+1} \quad (7)$$

Keterangan:

r1, r2	: bilangan acak dari distribusi *gaussian* N(0,1)
η	: *noise* yang dihasilkan dari distribusi *gaussian*.

η digunakan untuk menghindari terjebaknya algoritma dari *local optimal*, dimana kovarian matriksnya berganti dengan persamaan berikut:

$$\sum_{\eta}^{n} = \sum e^{-cn} \quad (2.11) \quad (8)$$

Keterangan:

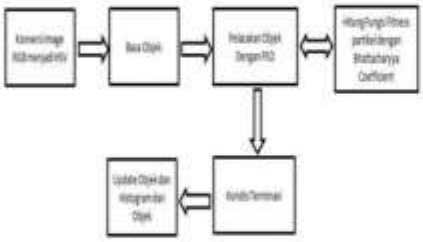
$\sum$	: kovarian matriks yang telah ditentukan sebelumnya
e	: epsilon
c	: *annealing constant*
n	: iterasi ke-n

III. AGPSO Pada Pelacakan Objek Pada Video

A. Gambaran Umum

Perangkat lunak pelacakan objek yang bergerak pada video menggunakan PSO merupakan perangkat lunak yang mampu melacak keberadaan objek pada video setiap selang waktu video. Objek pada pelacakan video harus dimodelkan agar dapat memudahkan dalam melaksanakan pelacakan objek pada video. Permodelan objek pada perangkat lunak ini menggunakan permodelan histogram dengan warna *hue*. Perangkat lunak ini menggunakan algoritma PSO untuk melacak keberadaan posisi objek pada *frame* berikutnya di video dengan menggunakan metode *Bhattacharyya Coefficient* untuk menghitung fungsi *fitness* dari masing-masing partikel pada PSO. Bhattacharyya Coefficient merupakan metode untuk

membandingkan nilai kedekatan antara 2 buah histogram, semakin rendah nilai kedekatan maka akan semakin baik begitu juga sebaliknya. Secara umum gambaran d penerapan metode-metode diatas pada perangkat lunak ini dapat dilihat pada gambar dibawah ini.



Gambar 2. Gambaran umum perangkat lunak pelacakan objek yang bergerak pada video dengan PSO pada waktu t

Proses pelacakan pada pelacakan objek pada perangkat lunak ini dilakukan setiap selang waktu pada video. Pada setiap waktu t citra akan dikonversi dari model warna RGB menjadi model warna HSV, selanjutnya partikel akan disebar disekitar posisi partikel terbaik. Proses penyebaran ini akan difilter menggunakan nilai maksimal dari pergerakan partikel. Nilai maksimal menggunakan posisi partikel terbaik sebagai titik (0, 0). Proses selanjutnya adalah melakukan pelacakan objek hingga kondisi terminasi terpenuhi. Posisi dan kecepatan dari masing-masing partikel akan diubah dan fungsi *fitness* dari partikel juga akan dihitung menggunakan metode *Bhattacharyya Coefficent* pada setiap iterasi. Proses pelacakan akan berhenti jika kondisi terminasi telah tercapai atau telah mencapai batas iterasi. Hasil dari pelacakan objek adalah posisi objek pada video yang digambarkan dengan kotak. Histogram dari partikel terbaik juga akan diubah.

B. Toleransi Kesalahan

Pelacakan objek merupakan permasalahan yang bersifat dinamis, artinya pelacakan objek akan dilakukan pada setiap *frame* dari video. Objek dan *background* pada video dapat mengalami perubahan pada setiap urutan *frame* pada video, baik perubahan bentuk atau perubahan intensitas warna, sehingga diperlukan sebuah metode yang mampu untuk menangani perubahan tersebut. Penggunaan metode *Bhattacharyya Coefficient* dan perubahan histogram [7] untuk menghitung fungsi fitness pada pelacakan objek menunjukkan hasil yang baik.

Peneliti mengusulkan suatu teknik untuk menghitung toleransi kesalahan dari partikel terhadap target. Perhitungan toleransi kesalahan terdiri dari 2 jenis perhitungan, yaitu, perhitungan toleransi kesalahan jumlah piksel dan perhitungan toleransi kesalahan perbedaan intensitas warna hue antara objek dan partikel. Histogram objek dan partikel akan diurutkan dari menurun seperti yang terlihat pada gambar dibawah ini;

Intensitas	Jumlah Piksel
0	10
1	9
2	17
3	62
.	.
.	.
.	.
360	

Pengurutan

Intensitas	Jumlah Piksel
3	62
2	17
1	10
0	9
.	.
.	.
.	.
360	

Gambar 3. Pengurutan histogram menurun

- Toleransi Kesalahan Jumlah Piksel

Perhitungan ini digunakan untuk menghitung toleransi kesalahan jumlah piksel dari objek dan partikel. Histogram dari objek dan partikel yang telah diurutkan akan dihitung toleransi kesalahan jumlah piksel-nya menggunakan 2 parameter, yaitu jumlah perbandingan yang disimbolkan dengan S dan toleransi perbedaan piksel yang dilambangkan dengan T_{pix}.

Algoritma untuk menghitung toleransi kesalahan jumlah piksel adalah sebagai berikut:

Deklarasi :
i:int; C_{pix}:int; T_{pix}:int; S:int; value:float
Inisialisasi:
i=0; C_{pix} =0; T_{pix}; S; value;
Repeat
value =[pikselObjek[i]-pikselPartikel[i]]
if (value >= T_{pix}) **then**
C_{pix} ++
i++
Until i < S
return C_{pix}

- Toleransi Kesalahan Intensitas Warna *Hue*

Perhitungan ini digunakan untuk menghitung toleransi kesalahan intensitas warna *hue* dari objek dan partikel. Histogram dari objek dan partikel yang telah diurutkan akan dihitung toleransi kesalahan intensitas warna hue-nya menggunakan 2 parameter, yaitu jumlah perbandingan yang disimbolkan dengan S dan toleransi intensitas warna *hue* yang dilambangkan dengan T_{range}. Algoritma untuk menghitung toleransi kesalahan intensitas warna *hue* adalah sebagai berikut:

Deklarasi :
i:int; C_{range}:int; T_{range}:int; S:int; value:float
Inisialisasi:
i=0; C_{range}=0; T_{range}; S; value;
Repeat
value = [pikselObjek[i]-pikselPartikel[i]]
if (value >= T_{range}) **then**
C_{range} ++
i++
Until i < S
return C_{range}

Untuk menghitung total toleransi kesalahan akan menggunakan 2 parameter yaitu faktor penguat toleransi kesalahan jumlah piksel yang dilambangkan dengan F_{pix} dan faktor penguat toleransi kesalahan intensitas warna *hue* yang dilambangkan dengan F_{range}. Faktor penguat digunakan untuk mempertegas apakah toleransi kesalahan jumlah piksel atau toleransi kesalahan piksel yang memiliki pengaruh besar dalam perhitungan toleransi kesalahan.

Rumus yang digunakan untuk menghitung total toleransi kesalahan adalah sebagai berikut:

$$T = \begin{cases} 1 & if\ Cpix \geq s\ and\ Crange \geq s \\ \dfrac{(Cpix * Fpix) + (Crange * Frange)}{(s * Fpix) + (s * Frange)}, & else \end{cases} \quad (9)$$

Keterangan:
T : Total Toleransi Kesalahan
C_{pix} : Total toleransi kesalahan jumlah piksel
C_{range} : Total toleransi kesalahan intensitas warna hue
F_{pix} : Faktor penguat toleransi kesalahan jumlah piksel
F_{range} : Faktor penguat toleransi kesalahan intensitas warna *hue*
S : Total perbandingan

C. Pengaturan Parameter

Berdasarkan hasil penelitan dan *studi literature* yang dilakukan oleh peneliti penentuan nilai parameter digunakan pada perangkat lunak ini ditunjukkan pada tabel dibawah ini;

Tabel 1. Penentuan nilai parameter

Parameter	Nilai
totalParticle	150
maxXhunt	35
maxYHunt	35
iterasi	7
c	0.3
Treshold	0.2 - 0.3
S	15
FPix	1
FRange	2
TPix	10
TRange	10

D. Fungsi Fitnes

Bhattacharyya Coefficient [2] merupakan sebuah metode yang digunakan untuk mencari nilai kedekatan antara 2 buah histogram H1 dan H2, semakin rendah jarak yang dihasilkan maka semakin tinggi nilai kedekatan antara 2 buah histogram tersebut. Rumus yang digunakan yaitu,

$$BC(H1, H2) = \sum_{i=1}^{n} \sqrt{H_{1i} H_{2i}} \quad (10)$$

$$D(H1, H2) = \sqrt{1 - BC(H_1, H_2)} \quad (11)$$

Keterangan:
BC = *Bhattacharya Coefficient*
H1 = Histogram 1
H2 = Histogram 2
D = jarak (nilai kedekatan)

Penggunaan toleransi kesalahan pada penelitian ini dapat diterapkan pada saat perhitungan fungsi fitness dari partikel, sehingga fungsi fitness dari partikel dapat dihitung dengan menggunakan metode *Bhattacharyya Coefficient* dan total toleransi kesalahan;

$$P_{fit}^{i} = \begin{cases} D, & if\ T \geq 1 \\ D * T, & else \end{cases} \quad (12)$$

Keterangan:
P_{fit}^{i} = Nilai fitness partikel ke-i
D = *Bhattacharyya Coefficient*
T = Total toleransi kesalahan

Penambahan proses perhitungan toleransi kesalahan pada proses perhitungan fungsi fitness partikel dapat mempengaruhi nilai *fitness* yang dihasilkan dari partikel, semakin kecil fungsi fitness yang dihasilkan maka semakin baik, begitu juga sebaliknya

E. Algoritma

Untuk melakukan pelacakan objek pada video menggunakan AGPSO akan melalui tahapan sebagai berikut:

1. Inisialisasi objek secara manual
2. Untuk setiap selang waktu *file* video, lakukan
3. Konversi dari citra RGB ke citra HSV.
4. Sebar partikel disekitar objek (partikel terbaik)
5. Untuk setiap iterasi, lakukan
6. Untuk setiap partikel
   - 6.1 *Update* posisi dan kecepatan dari masing-masing partikel
   - 6.2 *Update* histogram partikel
   - 6.3 Hitung fungsi *fitness* partikel menggunakan *Bhattacharyya Coefficient* dan Toleransi Kesalahan
7. Urutkan partikel dengan partikel terbaik pada *index* ke-0
8. Jika telah memenuhi kondisi terminasi maka update partikel global terbaik, tampilkan hasil pelacakan objek, lalu stop dan ulangi ke langkah 2
9. Jika belum memenuhi kondisi terminasi ulangi ke langkah 5

IV. HASIL PERCOBAAN

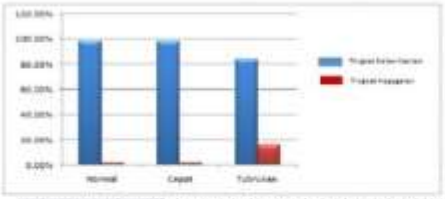
A. Hasil Percobaan pada Pelacakan Objek pada Video

Berdasarkan hasil pengujian terhadap data video dengan 3 kategori dapat dikatakan perangkat lunak ini berhasil dalam melakukan pelacakan objek dengan tingkat akurasi yang baik yaitu dengan rata-rata tingkat keberhasilan sebesar 93.4 % dan tingkat kegagalan sebesar 6.6 %. Tabel dibawah ini menunjukkan tingkat keberhasilan dan tingkat kegagalan dari proses pelacakan objek yang bergerak pada video menggunakan algoritma PSO pada masing-masing kategori.

Tabel 2 Tabel tingkat keberhasilan dan kegagalan pelacakan objek yang bergerak pada video

Kategori	Tingkat Keberhasilan
Pergerakan objek pada video berlangsung normal	98.2 %
Pergerakan objek pada video berlangsung cepat	98 %
Objek mengalami tubrukan pada video	84 %
Total	93.4 %

Perbandingan tingkat keberhasilan dan kegagalan pada proses pelacakan objek yang bergerak pada video menggunakan algoritma PSO pada masing-masing kategori juga dapat dilihat pada gambar grafik dibawah ini.



Gambar 4. Grafik tingkat keberhasilan dan kegagalan pelacakan objek yang bergerak pada video

Dari gambar grafik diatas terlihat bahwa pada pengujian pada data video dengan kategori pergerakan objek pada video berlangsung normal perangkat lunak ini mampu mencapai nilai akurasi yang sangat baik yaitu dengan tingkat keberhasilan sebesar 98.2 % dan tingkat kegagalan sebesar 1.8 %. Pada data video dengan kategori pergerakan objek pada video berlangsung cepat perangkat lunak ini juga mampu menghasilkan akurasi yang sangat baik, yaitu dengan dengan tingkat keberhasilan sebesar 98 % dan tingkat kegagalan sebesar 2 %, sedangkan pada kategori objek mengalami tubrukan pada video perangkat lunak ini menghasilkan akurasi yang cukup baik, yaitu dengan dengan tingkat keberhasilan sebesar 84 % dan tingkat kegagalan sebesar 16 %. Beberapa contoh hasil pelacakan objek pada video dengan AGPSO dapat dilihat pada gambar dibawah ini

Gambar 5 Hasil Pelacakan Objek dengan AGPSO

B. Analisis Hasil

Berdasarkan pengujian yang telah dilakukan, peneliti berpendapat bahwa ada beberapa faktor yang mempengaruhi tingkat keberhasilan dalam proses pelacakan objek yang bergerak pada video pada perangkat lunak ini, yaitu;

1. Penentuan objek oleh *user*
2. Perbedaan warna antara objek dengan *background* pada video dan dengan objek lainnya yang ada pada video.
3. Kecepatan pergerakan objek pada video.
4. Penentuan nilai parameter

1. Penentuan objek oleh *user*

Penentuan *region* objek oleh *user* pada video memiliki pengaruh yang penting pada proses pelacakan objek ini. Penentuan objek yang tepat dapat mengurangi *noise* pada objek sehingga pelacakan objek dapat menghasilkan akurasi yang baik dan mencapai tingkat *konvergensi* dengan cepat. Ukuran dari objek juga mempengaruhi proses pelacakan objek pada perangkat lunak ini. Ukuran objek yang besar dapat memperberat proses komputasi sehingga dapat menyebabkan algoritma ini lambat dalam mencapai tingkat konvergensi.

2. Perbedaan warna antara objek dengan *background* pada video dan dengan objek lainnya yang ada pada video.

Penggunaan warna *hue* sebagai histogram yang digunakan pada proses perhitungan fungsi *fitness* dari masing-masing partikel memiliki beberapa kelemahan dalam hal *ambiguitas* warna, yang dapat menyebabkan kegagalan dalam proses pelacakan objek. Perangkat lunak ini dapat gagal menentukan objek yang sebenarnya pada keadaan objek yang memiliki tingkat kesamaan warna yang tinggi dengan *background* pada video ataupun objek yang lainnya yang ada pada video. Hal ini terbukti pada salah satu pengujian pada data video dengan kategori objek mengalami tubrukan dengan nama *file* occ-3-twoperson-outdoor.mov dan occ-4-twoperson-outdoor.mov yang telah dijelasakan pada subbab sebelumnya. Pada *file* occ-3-twoperson-outdoor.mov perangkat lunak ini gagal dalam menentukan objek yang sebenarnya dan berpindah ke objek lainnya yang memiliki tingkat kesamaan warna yang tinggi. Pada *file* occ-4-twoperson-outdoor.mov perangkat lunak ini gagal dalam menentukan objek hingga durasi video habis.

3. Kecepatan pergerakan objek pada video.

Kecepatan pergerakan objek pada video merupakan faktor kesulitan tersendiri pada permasalahan pelacakan objek. Objek yang bergerak cepat akan sulit dilacak, hal ini dapat diatasi dengan memperbesar area penyebaran partikel pada saat proses pelacakan dengan cara pemberian nilai parameter batas maksimum yang besar, namun hal ini juga dapat menyebabkan kegagalan pada proses pelacakan objek atau gagal dalam menentukan objek yang sebenarnya yang harus dilacak.

4. Penentuan nilai parameter

Penentuan nilai parameter seperti maxXHunt, maxYHunt, iterasi, jumlah partikel, *treshold*, dan c juga mempengaruhi proses pelacakan objek pada perangkat lunak ini. Jumlah partikel dan iterasi yang besar akan menghasilkan akurasi yang lebih baik namun akan menyebabkan tingkat komputasi yang semakin besar dan proses pelacakan objek akan berjalan lambat. Nilai maxXHunt, maxYHunt dan c yang besar akan memperbesar area penyebaran partikel namun dapat namun hal ini juga dapat menyebabkan kegagalan pada proses pelacakan objek atau gagal dalam menentukan objek yang sebenarnya yang harus dilacak. Penentuan nilai parameter pada perhitungan toleransi kesalahan juga mempengaruhi proses pelacakan objek, terutama pada proses menentukan posisi objek yang sebenarnya.

Pada proses pengujian dilakukan perbandingan perhitungan fungsi fitness dengan menggunakan toleransi kesalahan dan tanpa menggunakan toleransi kesalahan. Berdasarkan hasil pengujian yang dilakukan, didapatkan hasil pengujian sebagai berikut:

Tabel 3. Perbandingan penggunaan toleransi kesalahan pada proses perhitungan fungsi fitness

Fungsi Fitness	Tingkat Keberhasilan
Dengan Toleransi Kesalahan	93.4 %
Tanpa Toleransi Kesalahan	90 %

Dari hasil pengujian diketahui bahwa penggunaan teknik toleransi kesalahan menghasilkan akurasi yang lebih baik. Berdasarkan hasil pengujian, proses perhitungan fungsi fitness tanpa menggunakan toleransi kesalahan memiliki hasil pelacakan objek yang baik pada objek yang bergerak normal tetapi cenderung gagal dalam menemukan objek pada kategori objek yang bergerak cepat pada video seperti yang terlihat pada gambar dibawah ini.

Gambar 6. Kegagalan menemukan objek pada proses perhitungan fungsi fitness tanpa toleransi kesalahan

V. KESIMPULAN DAN SARAN

Kesimpulan yang didapat setelah melalui penelitian ini adalah:

1. Algoritma AGPSO dan penggunaan *Bhattacharyya Coefficient* dan Toleransi Kesalahan untuk menghitung fungsi *fitness* mampu melakukan pelacakan objek yang bergerak pada video dengan akurasi dan kecepatan yang baik, yaitu dengan tingkat keberhasilan sebesar 93.4%.

2. Penggunaaan model dan fungsi *fitness* pada pelacakan objek yang bergerak pada video menggunakan Algoritma AGPSO dan penggunaan *Bhattacharyya Coefficient* dan Toleransi Kesalahan untuk menghitung fungsi *fitness* memiliki pengaruh yang besar, semakin baik model dan fungsi *fitness* maka hasil dari pelacakan objek akan semakin baik.

3. Penggunaan toleransi kesalahan dalam perhitungan fungsi *fitness* meningkatkan akurasi dari hasil pelacakan objek.

Saran penulis untuk pengembangan lebih lanjut dari penelitian ini diantaranya sebagai berikut:

1. Penambahan Algoritma tertentu untuk digunakan pada proses penentuan objek secara otomatis.
2. Penambahan jumlah objek yang lebih dilacak lebih dari satu.
3. Penelitian lebih lanjut terhadap perhitungan toleransi kesalahan untuk penggunaan umum atau pada bidang pelacakan objek pada video dengan Algoritma PSO.
4. Menggunakan perangkat lunak ini sebagai modul untuk pengembangan perangkat lunak lainnya, seperti pengenalan wajah pada video, pengenalan aksi, *video surveillance* dan lain-lain.

Daftar Pustaka :

[1] Isard, Michael. and Blake, Andrew. 1998. *CONDENSATION—Conditional Density Propagation for Visual Tracking*. 1998. International Journal of Computer Vision 3 (1), 5–28 (1998)

[2] Kailath, Thomas. 1967. *The Divergence and Bhattacharyya Distance Measures in Signal Selection*. IEEE Transactions on Communication and Technology Volume 15.

[3] Zhang, Xiaoqin, Hu, Weiming. Maybank, Steve. Li, Xi. Zhu, Ming g. 2008. *Sequential Particle Swarm Optimization for Visual Tracking*. Conference on Computer Vision and Pattern Recognition. IEEE. June 23-28. 2008.

[4] Kennedy, James and Eberhart, R.C. 1995. *Particle swarm optimization.In Proceedings of IEEE International Conference on Neural Networks, IEEE Press*. 1995. 1942–1948.

[5] Bimbo, D and Dini, F. 2011. *Particle Filter-Based Visual Tracking With a First Order Dynamic Model and Uncertainty Adaptation*.Computer Vision and Image Understanding. Vol. 115, No. 6. Hal 771–786

[6] Zhang, Xiaoqin. Hu,Weiming. Maybank, Steve. 2009. *A Smarter Particle Filter. Proceeding of ACCV Volume 2*. 2009. 236-246.

[7] Zheng, Yuhua. and Meng, Yan. 2008. *Swarm Intelligence Based Dynamic Object Tracking*. 2008. IEEE Congress on Evolutionary Computation (CEC 2008).

makalah_erwin_unsri.docx

ORIGINALITY REPORT

8%

SIMILARITY INDEX

PRIMARY SOURCES

1	hal.archives-ouvertes.fr Internet	44 words — 1%
2	mpra.ub.uni-muenchen.de Internet	19 words — 1%
3	Stefan Roth. "On the Spatial Statistics of Optical Flow", International Journal of Computer Vision, 04/26/2007 Crossref	16 words — < 1%
4	www.stevens.edu Internet	15 words — < 1%
5	Gao, Zhen, and Dan Zhang. "Workspace Representation and Optimization of a Novel Parallel Mechanism with Three-Degrees-of-Freedom", Sustainability, 2011. Crossref	15 words — < 1%

6	www.undana.ac.id Internet	14 words — < 1%
7	airccse.org Internet	12 words — < 1%
8	eprints.upnjatim.ac.id Internet	12 words — < 1%
9	ejournal-s1.undip.ac.id Internet	11 words — < 1%
10	140.113.87.112 Internet	10 words — < 1%
11	towwer4.blogspot.com Internet	10 words — < 1%
12	id.scribd.com Internet	9 words — < 1%
13	si.fst.uinjkt.ac.id Internet	9 words — < 1%
14	perahujagad.blogspot.com Internet	9 words — < 1%

15	9w2pns.blogspot.com Internet	9 words — < 1%
16	repository.unand.ac.id Internet	9 words — < 1%
17	media.neliti.com Internet	9 words — < 1%
18	nlpr-web.ia.ac.cn Internet	9 words — < 1%
19	Siti Nurlaila. "PELATIHAN EFIKASI DIRI UNTUK MENURUNKAN KECEMASAN PADA SISWA-SISW I YANG AKAN MENGHADAPI UJIAN AKHIR NASIO IAL", GUIDENA: Jurnal Ilmu Pendidikan, Psikologi, Bimb ngan dan Konseling, 2011 Crossref	8 words — < 1%
20	ikadwisaputra.blogspot.com Internet	8 words — < 1%
21	repository.its.ac.id Internet	8 words — < 1%
22	www.cs.uoi.gr Internet	%

8 words — < 1

EXCLUDE QUOTES OFF

EXCLUDE MATCHES OFF

EXCLUDE BIBLIOGRAPHY ON