ISSN. 1412-0100 VOL 16, NO 1, APRIL 2015

Moving Object Tracking using CAMSHIFT and SURF Algorithm

Ronsen Purba[1], Irpan Adiputra Pardosi[2], Feredy Lestari Pandia[3], Yudi Pratama Hasibuan[4]
STMIK Mikroskil, Jl. Thamrin No. 112, 124, 140, Telp. (061) 4573767, Fax. (061) 4567789
[1,2,3,4]Jurusan Teknik Informatika STMIK Mikroskil
[1]ronsen@mikroskil.ac.id, [2]irpan@mikroskil.ac.id, [3]fredypandia4@gmail.com,
[4]yudi.pratamahsb129@gmail.com

Abstrak

Penjejakan objek bergerak (moving object tracking) sebagai sebuah permasalahan yang berperan penting dalam bidang computer vision dan secara luas dapat diterapkan dalam banyak aplikasi dunia nyata seperti pengawasan otomatis, estimasi posisi tubuh manusia, navigasi kendaraan, pemantauan lalu lintas, dan robot vision. Moving object tracking membutuhkan metode yang memiliki akurasi dan ketahanan yang baik terhadap perubahan yang terjadi pada objek. Penelitian ini membangun sebuah aplikasi menggunakan bahasa visual basic untuk membandingkan Algoritma Camshift (Continuosly Adaptive Mean-Shift) dan Algoritma SURF (Speeded Up Robust Feature). Aplikasi dapat melakukan penjejakan dengan menggunakan kedua metode sekaligus. Pengujian dilakukan dengan lima kondisi pergerakan objek yang berbeda pada tiga warna latar belakang berbeda untuk membandingkan waktu komputasi dan akurasi dari kedua metode. Hasil pengujian menunjukkan bahwa akurasi Camshift lebih baik dibanding Surf, sementara untuk waktu komputasi Surf mengungguli Camshift.

Kata kunci— *Penjejakan objek bergerak, computer vision, algoritma camshift dan surf*

Abstract

Moving object tracking has an important role in computer vision and widely applied in many real lif application such as otomatic controlling, human pose estimation, vehicle navigation, traffic controlling, and robot vision. This task needs methods that have a high accuracy and good robustness due to object position changes. The objective of this research is to compare Camshift (Continuosly Adaptive Mean-Shift) dan Algoritma SURF (Speeded Up Robust Feature) algorithm in terms of accuracy and execution time. In conducting the experiment, we develop an application using visual basic and run the application by moving the object and using the camera of the laptop capturing the movement of the object. The experiments are conducted by comparing five different movement with three different background color. The result shows that SURF is faster than Camshift has, but in term of accuracy Camshift surpasses SURF

Keywords— *moving object tracking, computer vision, camshift and surf algorithm*

1. PENDAHULUAN

Penjejakan objek bergerak atau *object tracking* merupakan sebuah cara yang digunakan untuk mendeteksi perubahan-perubahan yang terjadi pada objek dari satu *frame* ke *frame* berikutnya di dalam sebuah video [1 – 3, 5]. Pemanfaatan penjejakan objek bergerak berperan penting dalam bidang *computer vision* dan secara luas dapat diterapkan dalam banyak aplikasi seperti pengawasan otomatis, *human pose estimation*, navigasi kendaraan, pemantauan lalu lintas, dan *robot vision*. Penjejakan objek bergerak sangat kompleks sehubungan dengan perubahan pergerakan seperti bentuk yang tidak kaku (fleksibel), perubahan cahaya, perubahan akibat transformasi, dan perubahan sudut pandang [3, 5]. Penjejakan objek bergerak memerlukan metode yang memiliki akurasi dan ketahanan yang baik terhadap perubahan yang terjadi pada objek. Ada beberapa faktor yang dapat mempengaruhi hasil dari

penjejakan objek bergerak dalam video menggunakan webcam dimana sebagian objek ditutupi oleh objek lain, sebagian objek keluar dari *frame* video, dan latar belakang objek yang memiliki warna dan ciri yang sama.

Dalam literatur dan aplikasi nyata penjejakan objek bergerak dapat dilakukan menggunakan metode seperti: Mean-Shift, CAMSHIFT (*Continuosly Adaptive Mean-Shift*), SIFT (*Scale Invariant Feature Transform*) dan SURF (*Speeded Up Robust Feature*). Metode *mean-shift* tidak memiliki ketergantungan pada jenis objek yang diamati, baik yang bersifat kaku maupun fleksibel. Penggunaan metode *mean-shift* memerlukan perbaikan karena tidak dapat melakukan adaptasi terhadap distribusi probabilitas warna yang selalu berubah tiap pergantian *frame* dari video. Metode *CAMSHIFT* dikenal sebagai perbaikan terhadap metode *mean-shift* [2, 3, 5].

Metode SURF merupakan metode yang mampu mendeteksi fitur-fitur pada penjejakan objek bergerak dengan memanfaatkan kecepatan komputasi tapis kotak atau *box filter* dengan menggunakan citra integral. Hasil deteksi dan deskripsi fitur–fitur lokal dapat dipergunakan dalam penjejakan objek bergerak. Kelebihan dari metode SURF adalah kecepatan dalam mendeteksi fitur-fitur objek pada citra yang lebih cepat dibandingkan dengan metode SIFT dan memiliki ketahanan (*robustness)* terhadap transformasi rotasi (*rotation*), perubahan skala, pencahayaan, dan perubahan sudut pandang [2].

Penelitian ini membandingkan kinerja dari algoritma *CAMSHIFT* dengan SURF terkait dengan akurasi dan waktu komputasi. Struktur dari artikel disusun dengan organisasi metode penelitian, hasil dan pembahasan, kesimpulan dan diakhir dengan saran (penelitian lanjutan).

2. METODE PENELITIAN

2.1 Cara Kerja Penjejakan Objek Bergerak atau *Object Tracking*

Penjejakan objek bergerak atau *object tracking/ motion tracking* adalah proses mencari lokasi dari objek yang akan diamati dari suatu data video untuk setiap satu satuan *frame* secara terus menerus dalam durasi waktu tertentu. Aplikasi ini menerapkan teknik pengolahan citra digital. Terdapat dua pendekatan yang umum digunakan, yakni: berdasarkan gerakan objek dan berdasarkan pengenalan objek. Berbagai metode dikembangkan untuk menyelesaikan permasalahan penjejakan objek bergerak khususnya yang berbasiskan pola gerakan objek dan pengenalan objek antara lain menggunakan Kalman *filter, extended* Kalman *filter,* ataupun *particle filter, SIFT, SURF,* dan *CAMSHIFT* [2, 4, 5].

2.1.1 *Continuously Adaptive Mean Shift* (CAMSHIFT)

CAMSHIFT merupakan singkatan dari *Continuously Adaptive Mean Shift*, yang merupakan pengembangan dari algoritma *Mean Shift* yang secara terus menerus (kontinu) melakukan adaptasi atau penyesuaian terhadap distribusi probabilitas warna yang selalu berubah tiap pergantian *frame* dari *video sequence* [3 – 5].

Langkah-langkah dari algoritma *Mean Shift* adalah sebagai berikut:

1. Tentukan ukuran *search window*
2. Tentukan lokasi awal *search window*
3. Hitung daerah *mean* dalam *search window*
4. Posisikan *search window* ke tengah daerah *mean* seperti dihitung pada langkah 3
5. Ulangi langkah 3 dan 4 hingga konvergen (atau hingga pergeseran daerah *mean* kurang dari *threshold* / batas yang ditentukan).

Sedangkan langkah-langkah dari algoritma *CAMSHIFT* adalah sebagai berikut:

1. Tentukan ukuran awal *search window*.
2. Tentukan lokasi awal dari *search window*.
3. Tentukan daerah kalkulasi (*calculation region*) pada bagian tengah *search window* dengan ukuran lebih besar dari *search window*.

4. Konversi *frame* citra video ke dalam sistem warna HSV (*Hue*, *Saturation*, *Value*), dan dilakukan *color histogram lookup* dalam *calculation region* yang akan menghasilkan citra distribusi probabilitas warna gambar.
5. Lakukan algoritma *Mean Shift* seperti di atas (satu atau banyak iterasi) dengan input berupa ukuran dan lokasi *search window* serta citra distribusi probabilitas warna, simpan *zeroth moment*.
6. Set nilai x, y, z, dan *head roll* yang diperoleh dari langkah 5.
7. Gunakan nilai x, y untuk menentukan titik tengah *search window*, ($2\sqrt{area}$) untuk set ukuran *search window*.

   Untuk citra distribusi probabilitas warna gambar, daerah *mean* (*centroid*) di dalam *search window* dapat dihitung sebagai berikut:

   Cari *Zeroth Moment*:

$$M_{00} = \sum_x \sum_y I(x,y) \tag{1}$$

   Cari *First Moment* untuk x dan y:

$$M_{10} = \sum_x \sum_y xI(x,y) \text{ dan } M_{01} = \sum_x \sum_y yI(x,y) \tag{2}$$

   sehingga lokasi mean dalam *search window* (*Centroid*) adalah :

$$x_c = \frac{M_{10}}{M_{00}} \qquad y_c = \frac{M_{01}}{M_{00}} \tag{3}$$

   dimana $I(x,y)$ adalah nilai warna piksel di posisi (x,y) pada *search window*. Orientasi objek diperoleh dengan melakukan perhitungan *second moments* dengan persamaan:

$$M_{20} = \sum_x \sum_y x^2 I(x,y) \text{ dan } M_{02} = \sum_x \sum_y y^2 I(x,y) \tag{4}$$

   dimana orientasi objek adalah :

$$\theta = \frac{\arctan\left(\frac{2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\frac{M_{20}}{M_{00}} - x_c^2\right) - \left(\frac{M_{02}}{M_{00}} - y_c^2\right)}\right)}{2} \tag{5}$$

   Jika:

$$a = \left(\frac{M_{20}}{M_{00}} - x_c^2\right) \quad b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right) \quad c = \left(\frac{M_{02}}{M_{00}} - y_c^2\right)$$

   maka panjang l dan lebar w dari distribusi *centroid* adalah:

$$l = \frac{\sqrt{(a+c) + \sqrt{b^2 + (a-c)^2}}}{2}$$
$$w = \frac{\sqrt{(a+c) - \sqrt{b^2 + (a-c)^2}}}{2} \tag{6}$$

   Persamaan tersebut menghasilkan nilai x, y, rotasi objek, panjang dan lebar (area atau nilai z)
8. Ulangi langkah 3 untuk setiap pergantian *frame* citra video

2.1.2 *Speeded Up Robust Feature* (SURF)

Algoritma SURF merupakan pengembangan dari algoritma SIFT dimana SURF memanfaatkan kecepatan komputasi tapis kotak dengan menggunakan citra integral. Citra integral adalah sebuah citra matriks 3×3 yang nilai tiap pikselnya merupakan akumulasi dari nilai piksel atas dan kirinya. Algoritma ini dipilih karena beberapa faktor sebagai berikut [2, 5]:

a. Dapat mendeteksi fitur-fitur dengan cepat (diklaim lebih cepat dari SIFT) dengan respresentasi citra integral. Dapat mendeskripsikan fitur-fitur yang terdeteksi secara unik.
b. Memiliki ketahanan (*Invariant*) terhadap transformasi citra seperti rotasi citra, perubahan skala, perubahan pencahayaan, dan perubahan sudut pandang yang relatif kecil.
c. Tahan terhadap gangguan (*noise*) dengan intensitas tertentu

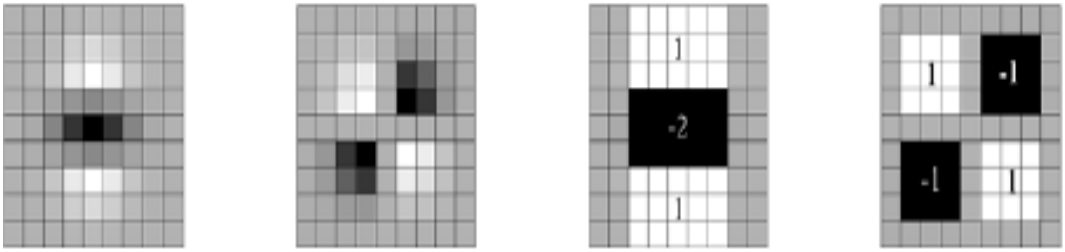
SURF bekerja dengan tiga tahapan utama, yaitu:

1. Konversi Citra RGB ke *Grayscale*
   Tahap awal dari algoritma SURF ini adalah mempersiapkan citra masukan dengan format *grayscale* $32 - bit$. Setiap fitur akan sangat efektif dihitung dengan menggunakan citra integral yang juga disebut sebagai *summed area tables*. Citra integral I adalah representasi tengah (*intermediate*) untuk citra dan terdiri dari jumlah nilai keabuan dari citra N dengan tinggi $= y$ dan lebar $= x$. Perumusannya adalah sebagai berikut [2]:

$$I(x,y) = \sum_{x=0}^{x} \sum_{y=0}^{y} N(x',y') \quad (7)$$

2. Mendeteksi Titik-Titik Fitur
   Dari fitur *Haar-like* maka hanya dipilih fitur gelembung (*blob-like feature*) untuk mendeteksi titik-titik fitur. Adapun pertimbangan pemilihan fitur gelembung adalah untuk mengurangi jumlah fitur yang terdeteksi dan lebih mempercepat waktu komputasi. Langkah ini terdiri dari:

   a. Pembentukan Piramid Citra
      Untuk membentuk piramid citra digunakan *box filter* sebagai aproksimasi dari turunan parsial kedua dari *Gaussian* sebagai berikut:



Gambar 1. Turunan Parsial kedua dari *Gaussian* (*discretized* dan *Cropped*) [5]

   b. Mencari ekstrema dari determinan matriks *Hessian*
      Citra dengan skala yang lebih kecil dibentuk dari konvolusi citra I dengan *up-scalling box filter* sebelumnya. Pada tahap pendeteksian fitur digunakan matriks *Hessian* karena memiliki performa yang baik dalam kecepatan waktu komputasi dan akurasi. Matriks *Hessian* di titik $x = (x, y)$ dari citra *I* dengan skala σ didefinisikan sebagai berikut:

$$D = (A+B+C+D)-(A+B)-(A+C)+A \quad (8)$$

      Agar fitur yang terdeteksi tahan terhadap penskalaan maka dicari ekstrema dari matriks *Hessian* dengan perumusan sebagai berikut:

$$det(H_{approx} = D_{(xx)}D_{yy} - (0{,}9D_{xy})^2 \quad (9)$$

      Kemudian ekstrema dari determinan matriks *Hessian* diinterpolasikan pada skala ruang dengan metode yang diajukan oleh Brown. Metode ini akan diterapkan pada setiap calon fitur untuk mencari lokasi extrema setelah diinterpolasi. $3D$ *quadratic* menggunakan ekspansi

Taylor terhadap fungsi *scale-space*, $D(x,y,\sigma)$, yang digeser sedemikian rupa sehingga titik originnya digunakan sebagai titik uji:

$$D(x) = D + \frac{\partial D^T}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{dx^2}x \quad (10)$$

dimana, *D* dan turunannya dihitung pada titik uji dan $x = (x,y,\sigma)^T$ adalah simpangan dari titik uji. Sedangkan lokasi ekstremum dapat dihitung sebagai berikut:

$$\hat{x} = \frac{\partial^2 D^{-1}}{\partial x^2} \times \frac{\partial D}{\partial x} \quad (11)$$

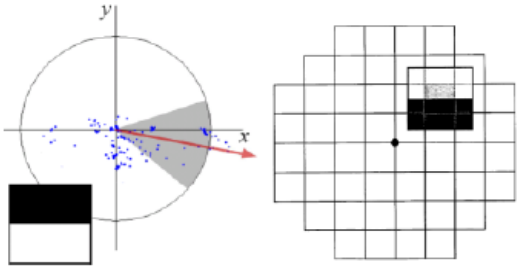
Jika nilai simpangan $\hat{x}$ lebih besar dari $0{,}5$ dalam dimensi manapun, maka ekstremum terletak pada titik lain di dekat titik tersebut. Untuk mencari lokasi ekstremum tersebut, maka titik uji dipindahkan ke titik yang memiliki simpangan lebih dari $0{,}5$ tersebut dan dihitung kembali simpangannya pada titik itu, dan hasilnya dijumlahkan dengan titik uji.

3. Pendeskripsian Fitur

Pendeskripsian titik-titik fitur menjadi deskriptor vektor dilakukan agar titik-titik fitur memiliki ketahanan terhadap rotasi, kekontrasan, dan perubahan sudut pandang. Langkah ini terdiri dari:

a. Pemberian orientasi

Agar tahan terhadap rotasi, maka setiap fitur yang terdeteksi akan diberikan orientasi. Pertama akan dihitung respon *Haar-wavelet* terhadap sumbu$-x$ dan sumbu$-y$ dengan titik-titik di lingkungan tetangganya pada radius $6s$ di sekitar titik fitur, dengan s adalah skala pada titik fitur yang terdeteksi. Untuk itu akan digunakan perhitungan citra integral untuk pentapisan yang cepat, seperti Gambar 2 berikut.



Gambar 2. Pemberian Orientasi [5]

b. Ekstraksi Komponen Deskriptor

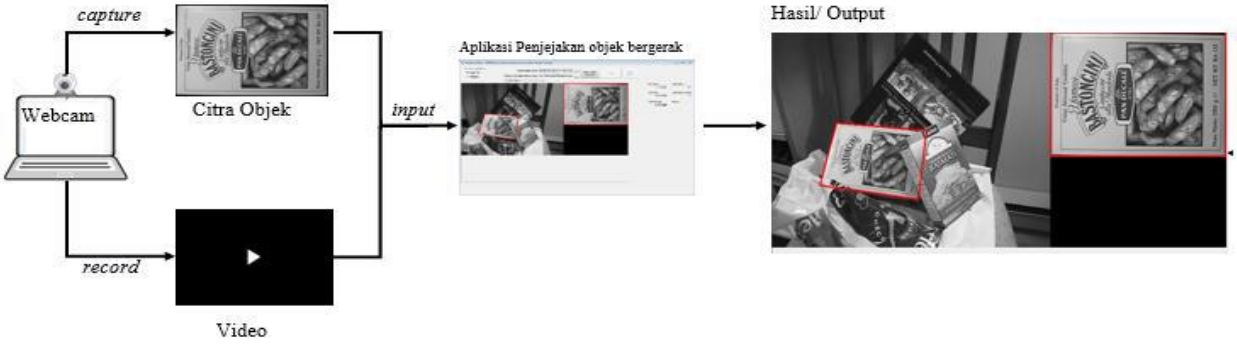
Untuk mengekstraksi deskriptor, langkah pertama yang diambil adalah membuat daerah bujur sangkar yang berpusat di sekitar titik fitur, dan orientasinya mengarah ke orientasi yang sudah ditentukan sebelumnya. Ukuran dari jendela bujur sangkar tersebut adalah $20s$. Untuk alasan kesederhanaan, respon *Haar-wavelet* pada arah horisontal akan disebut dengan dx dan respon *Haar-wavelet* arah vertikal disebut dengan dy [4]. Yang dimaksud "vertikal" dan "horisontal" dalam hal ini didefenisikan sesuai dengan orientasi dari titik fitur yang bersangkutan. Untuk meningkatkan ketahanannya terhadap deformasi geometrik dan kesalahan lokalisasi, maka respon dx dan dy akan dibobot dengan sebuah *Gaussian* $(\sigma = 3{,}3s)$ yang berpusat pada titik fitur. Setelah itu, citra target pada memori dan citra dari kamera diambil fitur-fiturnya dan diperoleh deskripsi vektornya.

c. Pencocokan *Keypoints*

Setiap vektor fitur pada citra kamera dicocokkan dengan vektor-vektor fitur dari citra yang berasal dari memori. Calon fitur yang paling cocok ditentukan dengan melihat *nearest neighbor* pada citra dalam memori. *Nearest neighbors* didefinisikan sebagai fitur yang memiliki jarak *Euclidean* yang paling kecil antara vektor deskriptornya.

2.2 Alur Penelitian

Proses penjejakan objek bergerak menggunakan Algoritma Camshift dan Algoritma SURF dapat dilihat pada gambar di bawah ini.

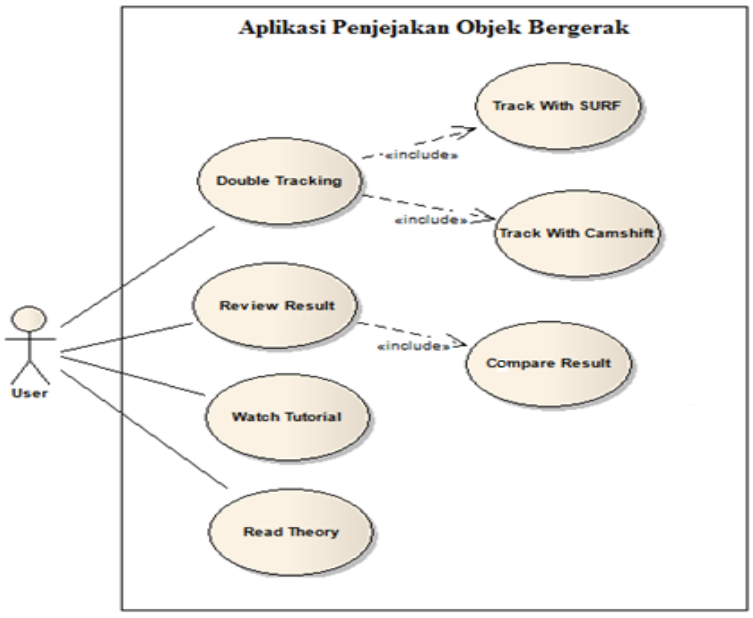


Gambar 3. Skema penjejakan terhadap objek bergerak

Dalam penelitian ini dibutuhkan sebuah webcam yang sudah menjadi bagian yang menyatu dalam laptop yang digunakan. Berikut ini adalah tahapan dalam proses penjejakan objek bergerak menggunakan Algoritma Camshift dan Algoritma SURF :

1. Pengambilan *capture* objek yaitu untuk proses pengambilan objek yang akan dijejaki.
2. Citra input yang telah ditangkap kemudian ditelusuri menggunakan Algoritma Camshift dan Algoritma SURF *frame* per *frame* didalam video secara *realtime*.
3. Berdasarkan hasil perhitungan ekstraksi fitur dari kedua algoritma, maka diperoleh hasil berupa informasi waktu kecepatan komputasi dan akurasi *frame* per *frame*.

Selama eksperimen objek akan terus digerakkan di depan kamera dengan kecepatan normal. Pergerakan objek dapat berupa translasi, rotasi, skala dengan menjauh dan mendekatkan objek ke kamera, serta *skewing*. Selama dalam pengamatan aplikasi akan mencatat keberadaan objek dan menyimpan hasilnya ke dalam record. Aplikasi juga dilengkapi dengan tutorial dan tampilan teori penjejakan objek bergerak. Pemodelan sistem yang digunakan dalam perangkat lunak ini dapat digambarkan menggunakan diagram *use case*. Diagram *use case* ini menggambarkan interaksi pengguna dengan sistem, seperti terlihat pada gambar 4 di bawah ini:



Gambar 4. Use Case Aplikasi penjejakan objek bergerak

3. HASIL DAN PEMBAHASAN

3.1 Hasil

Aplikasi dirancang menggunakan bahasa pemrograman visual basic dengan menggunakan perangkat lunak Microsoft visual studio 2010 dan library EmguCV v.2.4.0.1717. Proses pengujian meliputi kondisi latar belakang dengan warna putih keabuan, warna bervariasi dan memiliki ciri yang sama dengan objek yang akan dideteksi. Dalam setiap kondisi latar belakang akan dilakukan transformasi terhadap objek seperti: rotasi, translasi, *scalling*, dan *skewing.* Hasil pengujian dirangkum dalam Tabel 1 sampai Tabel 3 di bawah ini.

Tabel 1. Rangkuman hasil pengujian dengan latar belakang putih keabuan

No	CAMSHIFT		SURF	
	Waktu Komputasi (*sec/frame*)	Akurasi(%)	Waktu Komputasi (*sec/frame*)	Akurasi(%)
1	0.2065438	14.29%	0.2762483	100%
2	0.1972995	100%	0.360945	100%
3	0.2190414	100%	0.411814	100%
4	0.1823843	42.86%	0.3308249	71.43%
5	0.2155172	71.43%	0.3543236	100%

Dari Tabel 1, diperoleh:
Algoritma CAMSHIFT:
Waktu rata-rata = 0.20415724 *second/frame*
Rata-rata Akurasi = 65.72%
Algoritma SURF:
Waktu rata-rata = 0.34683116 *second/frame*
Rata-rata Akurasi = 94.286%

Tabel 2. Rangkuman hasil pengujian dengan latar belakang bervariasi

No	CAMSHIFT		SURF	
	Waktu Komputasi (*sec/frame*)	Akurasi(%)	Waktu Komputasi (*sec/frame*)	Akurasi(%)
1	0.3508387	100%	0.5045667	100%
2	0.3528156	100%	0.4800642	85.71%
3	0.4771396	100%	0.6706186	100%
4	0.4302189	100%	0.6231287	*85.71%*
5	0.4745334	100%	0.669319	100%

Dari Tabel 2, diperoleh:
Algoritma CAMSHIFT:
Waktu rata-rata = 0.41710924 *second/frame,*
Rata-rata Akurasi = 100%
Algoritma SURF: Waktu rata-rata =
0.58953944 *second/frame,*
Rata-rata Akurasi = 94.286%

Tabel 3 Rangkuman hasil pengujian dengan latar belakang mirip objek

No	CAMSHIFT		SURF	
	Waktu Komputasi (*sec/frame*)	Akurasi(%)	Waktu Komputasi (*sec/frame*)	Akurasi(%)
1	1.0647772	*100%*	1.733436	100%
2	1.0754724	85.71%	1.7305798	28.57%
3	1.0995911	85.71%	1.7895821	100%
4	0.885032	100%	1.7639863	100%

Dari Tabel 3, diperoleh:
Algoritma CAMSHIFT:
Waktu rata-rata = 1.0312181 *second/frame,*
Rata-rata Akurasi = 92.855%
Algoritma SURF:
Waktu rata-rata = 1.75439605 *second/frame,*
Rata-rata Akurasi = 82%

3.2 Pembahasan

Dari hasil pengujian yang telah dilakukan, maka dapat diketahui bahwa:

1. Pada kondisi latar belakang objek berwarna putih keabuan algoritma SURF memiliki akurasi yang lebih baik dibanding CAMSHIFT, yakni 94.28% dibanding dengan 65,716%. Akan tetapi pada kondisi latar belakang dengan warna bervariasi CAMSHIFT mengungguli SURF, yakni 100% dibanding dengan 94.28%. Sementara untuk latar belakang memiliki ciri yang sama dengan objek yang dideteksi, CAMSHIFT memiliki akurasi yang lebih baik dibanding dengan SURF, yaitu 92,85% dan 82%
2. Secara umum, tingkat akurasi total algoritma CAMSHIFT lebih tinggi dari pada SURF, yaitu 91.4286 % dibanding dengan 89.52381% dari total 105 *frame* yang diuji
3. Untuk waktu komputasi CAMSHIFT lebih baik dibanding dengan SURF, yaitu 0.51718758 *second/frame* dibanding dengan 0.823198433 *second/frame.*

4. KESIMPULAN

Berdasarkan pengujian yang telah dilakukan, maka dapat ditarik kesimpulan bahwa:pada kondisi latar belakang objek berwarna putih keabuan, algoritma SURF memiliki akurasi yang lebih baik dibanding dengan CAMSHIFT, sementara pada kondisi latar belakang warna bervariasi, algoritma CAMSHIFT memiliki akurasi yang lebih baik dibanding dengan SURF, Kemudian pada kondisi latar belakang memiliki ciri yang sama dengan objek, algoritma CAMSHIFT memiliki akurasi yang lebih baik dibanding dengan SURF. Secara keseluruhan tingkat akurasi algoritma CAMSHIFT lebih tinggi dari pada SURF, sementara untuk waktu komputasi CAMSHIFT lebih baik dibanding dengan SURF.

5. SARAN

Objek yang diamati dalam penelitian ini masih terbatas pada satu bentuk. Perlu dilakukan penelitian lanjutan untuk bentuk objek berbeda. Selanjutnya lingkungan pengujian dapat diperkaya dengan latar belakang yang lebih banyak dan intensitas cahaya berbeda serta kecepatan gerakan yang bervariasi.

DAFTAR PUSTAKA

[1] Bay, H., Tuytelaars, T., dan Van, G. L., 2006, *SURF: Speeded Up Robust Features. Proceedings of the ninth European Conference on Computer Vision*, www.**vision**.ee.ethz.ch/~**surf**/eccv06.pdf

[2] Chen, X., Wu, H., Li, X., Luo, X., dan Qiu, T., 2012, *Real-time Visual Object Tracking via CamShift-Based Robust Framework*. International Journal of Fuzzy Systems, Vol. 14, June 2012, No. 2, http://connection.ebscohost.com/c/articles/88950919/real-time-visual-object-tracking-via-camshift-based-robust-framework

[3] Lienhart, R., dan Jochen, M., (2002), *An extended set of haar-like features for rapid object detection*, In: IEEE ICIP 2002, Vol.1, 900-903, www.lienhart.de/Prof. Dr.../ICIP2002.pdf

[4] Liu, T., dan Wang, Z., *Adaptive Double Kalman Filter and Mean Shift for Robust Fast Object Tracking*, International Journal of Advancements in Computing Technology, March 2013, Vol. 5 Issue 6, http://connection.ebscohost.com/c/articles/98929798/adaptive-double-kalman-filter-mean-shift-robust-fast-object-tracking

[5] Wang, Z., dan Yang, F., 2012, *Object Tracking Algorithm Based on Camshift and Grey Prediction Model in Occlusions*, The 2nd International Conference on Computer Application and System Modeling (2012), http://www.atlantis-press.com/php/download_paper.php?id=2718