ANALISIS PARAMETER SKALA DAN PERGESERAN UNTUK DETEKSI OBJEK PADA KERANGKA KERJA *TRAKING-LEARNING-DETECTION* (TLD)

(PARAMETER ANALYSIS OF SCALE AND SHIFT FOR OBJECT DETECTOR IN THE TRACKING-LEARNING-DETECTION (TLD) FRAMEWORK)

Putri Utami Hafgianti[1] ,Suryo Adhi Wibowo, S. T., M. T.[2] , Raditiana Patmasari, S. T., M. T.[3]
[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1]puhnasution@gmail.com, [2]suryoadhiwibowo@telkomuniversity.ac.id, [3]raditiana@telkomuniversity.ac.id

Abstrak

Learning pada TLD adalah salah satu proses yang membedakan TLD dengan metode *tracking* objek lainnya. Proses *Learning* terjadi apabila objek yang diamati dalam suatu video terjadi *out-of-view* atau terjadi oklusi, saat objek tersebut akan muncul kembali maka akan terdeteksi kembali sebagai objek yang sedang diamati karena *learning* bertugas mengestimasi kesalahan deteksi dan terdapat *training example* untuk menghindari kesalahan. Untuk merepresentasikan objek yang diamati digunakan bentuk geometrik, seperti *bounding box*.Sistem TLD diberikan input berupa *image sequences* dan diberikan nilai parameter skala dan parameter pergeseran yang sudah ditentukan. Selanjutnya dilakukan inisialisasi pada sebuah objek yang direpresentasikan dalam *bounding box* pada *frame* pertama. Setelah sistem TLD sudah selesai dijalankan, maka didapatkan output berbentuk *image sequences* yang sudah terdapat *bounding box*, dan nilai titik *bounding box*. Hasil dari tugas akhir ini direpresentasikan dalam bentuk grafik *one-pass evaluation* (OPE) yang menunjukkan hasil parameter performansi, yaitu *success plot* dan *precision plot*. Masing-masing parameter performansi juga menampilkan grafik berdasarkan sebelas *challenge problem*. Secara keseluruhan, nilai *success plot* dan *precision plot* terbaik didapat pada nilai parameter skala 0.5 dan nilai parameter pergeseran 10 yang berarti, semakin kecil nilai parameter skala dan nilai parameter pergeseran maka semakin bagus performansinya.

Kata Kunci : ***Bounding box, groundtruth,* Skala, *Tracking-learning-detection (TLD), Frame,* pergeseran, grafik one-pass evaluation (OPE).**

Abstract

Learning on TLD is one process that differentiates TLD with other object tracking method. Learning process occurs when the object observed in a video occurs out-of-view or occlusion occurs, when the object will reappear it will be detected again as the object being observed. To represent the observed objects geometric shapes are used, such as bounding boxes. This shape is measured by scale, rotation, and shift of object location. Therefore, analysis of scale and shift parameters is performed. TLD system is given input in the form of image sequences and given the value of scale parameters and shift parameters that have been determined. Next is initialized to an object that is represented in the bounding box of the first frame. After the TLD system has been completed, then the output obtained in the form of image sequences that already exist bounding box, and the point bounding box value. The result of this final project is represented in the form of one-pass evaluation graph (OPE) which shows the result of performance parameter, that is success plot and precision plot. Overall, the best plot and precision plot values are obtained at the value of the 0.5 scale parameter and the parameter value of shift 10 which means, the smaller the value of the scale parameter and the shift parameter value the better the performance.

Keywords: Bounding box, groundtruth, Scale, Tracking-learning-detection (TLD), Frame, shift, one-pass evaluation graph (OPE).

1 PENDAHULUAN

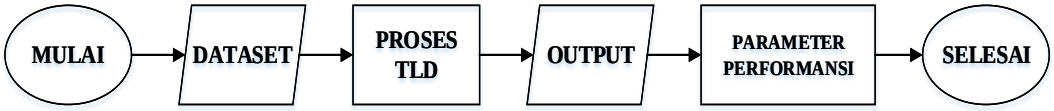
Tracking-Learning-Detection (TLD) adalah sebuah paradigma yang dirancang untuk *long-term tracking* dari sebuah objek yang tidak diketahui pada *video stream*. TLD terdiri dari 3 proses, yaitu *tracking*, *learning*, dan *detection*. *Learning* pada TLD adalah salah satu proses yang membedakan TLD dengan metode *tracking* objek lainnya. Proses *Learning* terjadi apabila objek yang diamati dalam suatu video terjadi *out-of-view* atau terjadi oklusi, saat objek tersebut akan muncul kembali maka akan terdeteksi kembali sebagai objek yang sedang diamati. Oleh karena itu, TLD dapat diaplikasikan untuk *hand writing*, *surveillance*, *robot vision*, dan lain sebagainya.

Pada penelitian ini, tracker dan detector hanya mengamati satu objek saja. Inisialisasi objek tersebut dilakukan pada frame pertama dan hasilnya direpresentasikan dalam bentuk grafik one-pass evaluation (OPE). Komponen klasifikasi tracking dan detection yang digunakan adalah bentuk geometric (bounding box). Kemudian digunakan input yang didapat dari dataset benchmark. Dataset benchmark tersebut terdapat groundtruth dan image sequences.

Penelitian ini bertujuan untuk mengetahui performansi dari TLD. Analisis performansi dari TLD ini berdasarkan nilai *success plot* dan *precision plot* yang direpresentasikan dalam bentuk grafik *one-pass evaluation* (OPE).

2 PERANCANGAN SISTEM

Pada penelitian ini, perancangan sistem bertujuan untuk menganalisis parameter skala dan pergeseran untuk mendeteksi objek pada kerangka kerja *tracking-learning-detection* (TLD). Hasil dari pengujian ini direpresentasikan dalam bentuk grafik OPE yang terdiri dari grafik OPE berdasarkan *success plot* dan *precision plot*.



Gambar 2.1 Diagram Alir Penelitian

Sebelum masuk pada proses TLD, dilakukan input nilai parameter skala dan parameter pergeseran. Pada tahap ini, terdapat beberapa skenario nilai parameter skala dan parameter pergeseran pada kerangka kerja TLD. Skenario yang diinginkan adalah seperti pada table 2.1.

Skenario pada tabel 2.1 ini bertujuan untuk menganalisis performansi kerangka kerja TLD ketika nilai parameter skala dan nilai parameter pergeseran diubah-ubah. Skenario nilai parameter skala dan pergeseran diatas dianggap sudah cukup efektif untuk dilakukan analisis, karena nilai tersebut menghasilkan perbedaan hasil nilai performansi yang cukup jelas.

Tabel 2.1 Skenario Nilai Parameter Skala dan Parameter Pergeseran

Skenario	Nilai Parameter Skala	Nilai Parameter Pergeseran
1	1	10
2	1	30
3	1	60
4	0.5	10
5	0.5	30
6	0.5	60

2.1 Tracking-Learning-Detection (TLD)

TLD dibagi menjadi beberapa proses, yaitu prasyarat, objek model, objek *tracking*, objek *detection*, integrator, dan *learning*.

2.1.1 Prasyarat

Objek direpresentasikan oleh *state*. *State* adalah salah satu dari sebuah *bounding box* yang menunjukkan bahwa objek tidak terlihat. *Bounding box* mempunyai aspek perbandingan tetap (pada inisialisasi *bounding box*) dan memiliki parameteri yaitu, lokasi dan skala dari *bounding box*. Parameter lain seperti rotasi dalam penelitian ini diabaikan. Setengah kesamaan dari dua *bounding box* akan diukur menggunakan *overlap*, yang didefinisikan seperti perbandingan antara irisan dan gabungan dari kedua *bounding box*. Persamaan antara dua patch (pi,pj) didefinisikan sebagai berikut:

$$S(pi,pj) = 0.5(NCC(pi,pj)+1) \quad (1.1)$$

Dimana NCC adalah *Normalized Correlation Coefficient*.

Kemunculan objek direpresentasikan oleh sebuah *patch* citra (P). *Patch* di *sampling* dengan objek pada *bounding box* dan kemudian di *sampling* ulang menjadi resolusi normalisasi (biasanya 15x15 pixel) tanpa memperhatikan aspek perbandingan.

Urutan dari *object state* didefinisikan sebagai *trajectory* dari sebuah objek dalam video sebaik *trajectory* yang sesuai dalam *feature space*. *Trajectory* difragmentasi seperti objek yang tidak terlihat [3].

2.1.2 Object Model

Model objek M adalah struktur data yang mewakili objek dan sekitarnya. Terdapat positif dan negatif *patch*, M = { $p1^+$, $p2^+$, ..., pm^+,$p1^-$, $p2^-$, ... , pn^- }, dimana p^+ dan p^- adalah objek dan *background patch*. Positif *patch* adalah perintah menurut waktu ketika patch ditambahkan ke dalam kelas. $P1^+$ adalah patch positif pertama yang ditambahkan ke dalam kelas, pm^+ adalah *patch positif* terakhir.

$$S^r = \frac{s^+}{s^+ + s^-} \quad (1.2)$$

Persamaan relative ini digunakan dalam NN Classifier. Batas hasil persamaan relative adalah 0 sampai 1. Hasil deteksi akan dianggap sebagai objek apabila nilai NN Classifier lebih dari 0.5. [1]

2.1.3 Object Tracking

Komponen *tracking* TLD didasarkan oleh *median flow tracker* yang kemudian dilanjutkan dengan deteksi kesalahan. *Median-flow tracker* merepresentasikan objek dengan sebuah *bounding box* dan mengestimasi pergerakan *bounding box* antara konserkutif frame. Didalamnya, *tracker* mengestimasi perpindahan dari jumlah *point* pada *bounding box* objek, mengestimasi kebenarannya. Digunakan grid 10 x 10 *point* sebagai grid minimal untuk setiap frame dan mengestimasi pergerakannya menggunakan *pyramidal Lucas Kanade Tracker*. *Lucas Kanade Tracker* menggunakan 2 level dari *pyramid*. Untuk mengatasi kesalahan *tracker* digunakan *Forward-Backward Error*.[1]

2.1.3.1 *Median-Flow Tracker*

Untuk mendukung tujuan dari *Forward-Backward error*, *median flow* digunakan untuk membangun *bounding box* yang lebih baik berdasarkan *tracker*. Pendekatan mendasar untuk mengestimasi gerakan *bounding box* menggunakan sebuah jumlah titik independen, mengukur kebenaran dan mengintegrasikan prediksi menggunakan statistik yang kuat. Menggunakan pendekatan ini, bertujuan pada *tracker* dimana tetap stabil untuk *partial occlusions* yang cepat dan efisien.

Tracker menerima bagian citra I_t, I_{t+1} dan sebuah *bounding box* b_t serta keluaran *bounding box* b_{t+1}. Satu set titik diinisialisasi di garis kotak dalam *bounding box* b_t. Titik ini di *tracking* dengan *Lucas-Kanade tracker* dari I_t sampai I_{t+1}. Kualitas dari kesalahan penempatan titik kemudian diestimasi dan setiap titik ditandai sebagai kesalahan. 50% kesalahan terburuk dikeluarkan. Prediksi sisanya diestimasi pergerakan *bounding box* menggunakan median. Sistem tracking ini disebut sebagai *Median-Flow*.

Estimasi pergerakan menggunakan median. Pergerakan *bounding box* diparameteri oleh kesalahan pada arah horizontal, vertikal, dan perubahan skala. Ketiga parameter ini diestimasi masing-masing menggunakan median. Pada Gambar 3.4 ilustrasi dari estimasi kesalahan tempat. Perubahan skala diestimasi mengikuti: setiap bagian titik, rasio antara jarak titik sekarang dengan jarak titik sebelumnya dihitung. Perubahan skala *bounding box* didefenisi sebagai rasio over median [3].

2.1.3.2 *Forward-Backward error*

Pendekatan yang digunakan untuk mendeteksi kesalahan *tracker* berdasarkan pada *forward-backward consistency asumption* yang mana tracking yang benar seharusnya menjadi direksi *time-flow* masing-masing. Pertama, *tracker* menghasilkan sebuah *trajectory* dengan melacak titik *forward* dalam waktu. Kemudian, lokasi titik dalam *frame* terakhir dari lintasan *forward* diinisialisasi menjadi sebuah lintasan yang benar. Validasi lintasan diperoleh dari *backward tracking*. Langkah selanjutnya, dua lintasan digabungkan dan jika terdapat perbedaan yang signifikan, *forward trajectory* dianggap tidak benar [3].

2.1.4 Object Detection

Tugas dari *object detection* adalah melokalisir objek dalam sebuah input citra. Definisi objek disini bervariasi. Objek bisa berbentuk *single instance* atau seluruh kelas objek [4].

2.1.4.1 *Scanning-window grid*

Sistem mengenerate semua kemungkinan skala dan pergeseran dari *bounding box* yang telah diinisialisasi dengan parameter skala, pergeseran, minimal ukuran *bounding box* mengikuti skenario yang sudah ditentukan. Bounding box yang akurat tergantung dalam aspek rasio dari *bounding box* yang diinisialisasi.

2.1.4.2 *Cascaded Classified*

Seperti jumlah dari beberapa *bounding box* yang dievaluasi adalah besar, klasifikasi dari setiap *single patch* harus sangat efisien. Struktur dari cascade classifier dibagi menjadi tiga tingkatan, yaitu *patch variance,ensemble classifier*, dan *nearest neighbor*. Masing-masing tingkat dapat melanjutkan atau menolak patch menuju tingkatan selanjutnya.

2.1.4.2.1 Patch Variance

Pada tingkat pertama ini, terjadi penolakan semua *patch* yang nilai varian *gray* nya lebih kecil 50% dari patch varian yang dipilih untuk ditracking. Tingkat ini mengeksploitasi semua nilai varian *gray* sebuah patch p yang bisa ditunjukkan sebagai $E(p^2) - E^2(p)$, dan nilai yang diharapkan E(p) bisa diukur dalam waktu konstan menggunakan integral citra. Tingkat ini mengabaikan *patch* yang lebih dari 50% dari non objek *patch*, seperti langit dan jalanan.

2.1.4.2.2 Ensemble Classifier

Ensemble classifier merupakan tingkat kedua dari *detector*. *Ensemble* terdiri dari *nn base classifier*. *Posterior* dari dasar individual *classifier* adalah rata-rata dan *ensemble classifier* dari *patch* sebagai objek jika rata-rata *posterior l*ebih dari 50%.

1) *Pixel comparisons*

Setiap *base classifier* didasarkan dalam *set pixel comparisons. Pixel comparison* dihasilkan acak atau tetap secara offline.

2) *Generating pixel comparisons*

Elemen yang vital dari *ensemble classifier* adalah independen dari *base classifier*, yaitu kasus yang dijalankan dengan menghasilkan perbedaan *pixel comparisons* untuk setiap *base classifier*.

3) *Posterior probabilities*

Setiap base *classifier* i diatur dalam sebuah distribusi dari *posterior* probabilitas.

4) Inisialisasi dan update

Pada tingkat ini, semua probabilitas dari *base posterior* diatur nol (contoh kelas negatif). Selama *run-time*, *ensemble classifier* diperbarui mengikuti aturan : contoh kelas diklasifikasi oleh *ensemble* dan jika klasifikasi tidak benar, pencocokan antara #p dan #n diperbarui.

2.1.4.2.3 Nearest Neighbor Classifier

Setelah *patch* difilter oleh filter varian dan *ensemble classifier*, sistem memberi tanda dengan beberapa *bounding box* yang belum diputuskan (≈50). Oleh karena itu, sistem bisa menggunakan model *online* dan mengklasifikasi patch menggunakan NN *classifier*. Sebuah patch diklasifikasi sebagai objek jika $S^r(p,M) > \theta_{NN}$, dimana $\theta_{NN} = 0.6$. setelah diamati terdapat kesamaan hasil yang dicapai dalam *range* (0.5 – 0.7). patch klasifikasi positif direpresentasikan sebagai respon dari objek detektor. Ketika jumlah template dalam NN *classifier* melebihi

dari threshold (dalam memory), sistem menggunakan *random forggetting* dari *template*. Sistem mengamati bahwa jumlah *template* stabil sekitar beberapa ratus, yang mudah digunakan dalam memori [1].

2.1.5 Integrator

Integrator mengkombinasi *bounding box* dari *tracker* dan *bounding box* dari *detector* dalam sebuah output *bounding box* oleh TLD. *Tracker* dan *detector* mempunyai prioritas yang identik, namun mereka menunjukkan pada dasarnya perbedaan estimasi dari *object state*. Ketika *detector* sudah melokasikan *template* yang diketahui dan *tracker* melokasikan *template* baru dan dapat dijadikan sebagai data baru untuk *detector*.

2.1.6 Komponen Learning

Tugas dari komponen *learning* adalah menginisialisasi objek *detector* dalam *frame* pertama dan mengupdate detector dalam *run-time* menggunakan kelas P dan kelas N

2.1.6.1 Inisialisasi

Dalam *frame* pertama, *learning* pertama melatih *detector* untuk menginisialisasi menggunakan contoh kelas yang dibuat mengikuti, contoh *training* positif mempersatukan inisial *bounding box*. Pertama, pilih 10 *bounding box* dalam *scanning grid* yang terdekat dengan inisial *bounding box*. Untuk setiap *bounding box*, sistem menghasilkan 20 versi dari perubahan geometris dan tambahkan mereka dengan *Gaussian noise* (5) dalam piksel.

Negatif *patch* dikumpulkan dari sekeliling inisialisasi *bounding box*, contoh negatif tidak menghasilkan contoh sintetis. Jika aplikasi membutuhkan inisialisasi yang cepat, sistem membuat *subsample* contoh *training*. Kelas *patch* training kemudian harus memperbarui objek model. Setelah diinisialisasi, objek *detector* siap untuk *run-time* dan memperbarui bagian kelas P-N

2.1.6.2 Kelas P

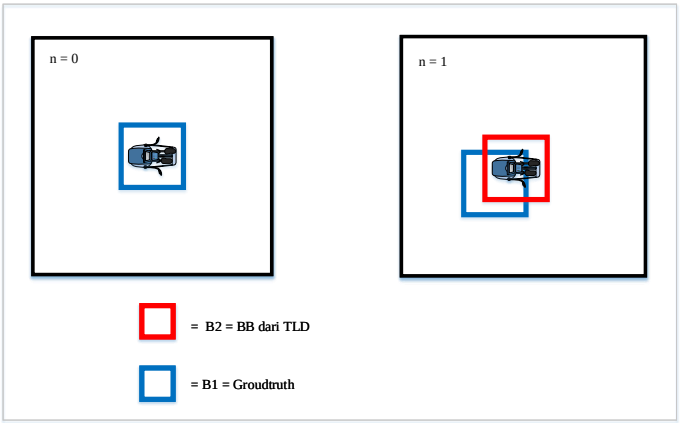
Tantangan kelas P adalah mengidentifikasi bagian kehandalan dari lintasan dan menggunakan ini untuk menghasilkan contoh kelas positif. Dalam setiap *frame*, keluaran kelas P adalah sebuah keputusan dalam data uji dari lokasi sebelumnya (kelas P memproses *online*). Jika lokasi sebelumnya benar, kelas P menghasilkan satu set contoh positif yang diupdate dari objek model dan telah *ensemble classifier*.

Sistem menghitung 10 *bounding box* dalam *scanning grid* yang terdekat dari *bounding box* sebelumnya. Untuk setiap *bounding box*, sistem menghasilkan 10 versi dari perubahan geometris dan tambahkan mereka dengan *Gaussian noise* (5) dalam piksel.

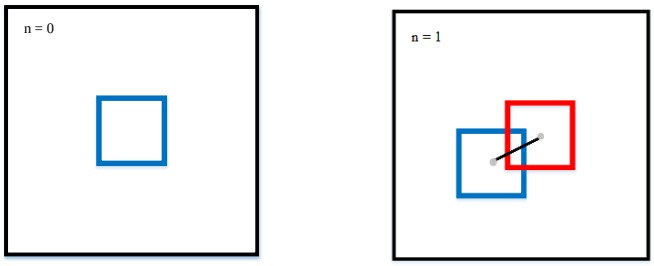
2.1.6.3 Kelas N

Kelas N menghasilkan contoh kelas *training* negatif. Asumsi utama dari kelas N adalah objek bisa terjadi oklusi pada paling banyak satu lokasi dalam citra. Oleh karena itu, jika lokasi objek telah diketahui, sekeliling dari lokasi tersebut adalah kelas negatif. Kelas N dipergunakan dalam waktu yang sama dengan kelas P, jika jalur nya benar. Kasus ini, patch yang jauh dari *bounding box* sebelumnya (*overlap* < 0.2) semua masuk dalam kelas N. Untuk *update* dari *detector* objek dan *ensemble classifier*, sistem hanya menganggap *patch* yang tidak ditolak oleh filter varian atau *ensemble classifier* [1].

2.2 Parameter Performansi



Gambar 3.8 Ilustrasi *success plot* pada *Frame* Awal dan *Frame* Selanjutnya



Gambar 3.7 Ilustrasi *Precision Plot* pada Frame Awal dan Frame Selanjutnya

Setelah output TLD sudah didapat, kemudian dilakukan proses perhitungan nilai parameter performansi. Parameter performansi yang digunakan ada dua, yaitu berdasarkan *success plot* dan *precision plot*. *Success plot* digunakan dalam evaluasi *bounding box* berdasarkan nilai *overlap* (S). Nilai *overlap* didapatkan dengan perbandingan dari irisan dan gabungan antara BB *groundtruth* (B1) dan BB TLD (B2) setiap *frame*. *Precision plot* digunakan dalam evaluasi *bounding box* berdasarkan nilai presisi (P). Presisi adalah jarak antara titik center BB *groundtruth* (x) dan titik *center* BB TLD (y). Gambar 3.6 dan 3.7 merupakan ilustrasi dari *success plot* dan *precision plot*. Grafik OPE adalah grafik yang merepresentasikan *success plot* dan *precision plot*.

$$S = \frac{B1 \cap B2}{B1 \cup B2} \; x \; 100\% \tag{1.4}$$

$$P = \sqrt{(x1 - x2)^2 + (y1 - y2)^2} \tag{1.5}$$

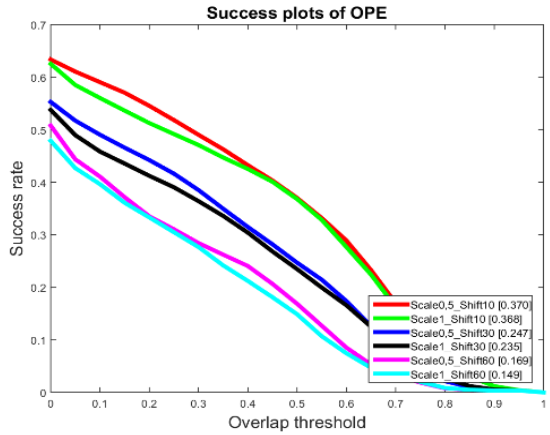
2.2.1 Pengolahan Data

Setelah didapat hasil sistem dari semua dataset pada setiap skenario, hasil sistem tersebut dihitung berdasarkan parameter performansi tersebut. Kemudian direpresentasikan dalam bentuk grafik OPE. Grafik OPE terdiri dari *success plot* dan *precision plot* pada masing-masing atribut yang ada.

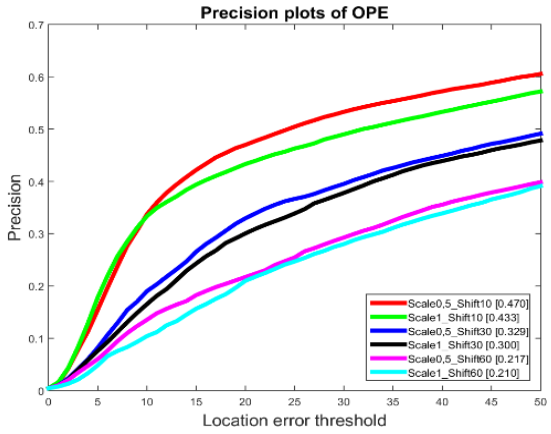
3 HASIL DAN ANALISIS

Dari 5 skenario perubahan nilai parameter skala dan pergeseran tersebut akan dilakukan analisis pengaruhnya pada parameter skala, parameter pergeseran serta atribut-atribut seperti, oklusi, deformasi dan lain sebagainya. Analisis ini ditinjau berdasarkan *success plot* dan *precision plot* yang direpresentasikan pada grafik OPE. Pada pengujian ini, diinginkan nilai *overlap threshold* = 0.5 dan nilai *location error threshold* 20 pixel.

Pada peninjauan parameter skala dan pergeseran ini, dilakukan pada 50 *dataset* dari TB-50 sebagai input. Kemudian didapatkan nilai parameter skala dan pergeseran yang terbaik.



Gambar 4.3a Grafik OPE *Success Plot*



Gambar 4.3b Grafik OPE *Precision Plot*

3.1 Pengujian Berdasarkan Parameter Skala

Berikut adalah hasil analisis dari perubahan nilai parameter skala yang telah diujikan:

3.1.1 *Success Plot*

Pada gambar 4.3a, nilai *success rate* terbesar terdapat pada nilai parameter skala 0.5 dengan nilai 0.370. kemudian dari nilai *success rate* pada grafik tersebut dapat disimpulkan bahwa nilai parameter skala 0.5 yang hasilnya lebih bagus dibandingkan dengan nilai skala 1. Oleh karena itu, semakin kecil nilai parameter skala maka semakin bagus performansi *success plot*.

3.1.2 Precision Plot

Pada gambar 4.3b, nilai *precision* terbaik terdapat pada nilai parameter skala 0.5 juga dengan nilai 0.470. kemudian dari persentase *precision* pada grafik tersebut dapat disimpulkan bahwa nilai parameter skala 0.5 yang hasilnya lebih bagus dibandingkan pada skala 1. Oleh karena itu, semakin kecil nilai parameter skala maka semakin bagus performansi *precision plot*.

3.2 Pengujian Berdasarkan Parameter Pergeseran

Berikut adalah hasil analisis dari perubahan nilai parameter pergeseran yang telah diujikan:

3.2.1 *Success Plot*

Pada gambar 4.3a, nilai *success rate* terbesar terdapat pada nilai parameter pergeseran 10 dengan nilai 0.370. kemudian dari nilai *success rate* pada grafik tersebut dapat disimpulkan bahwa nilai parameter pergeseran 10 yang hasilnya lebih bagus dibandingkan dengan nilai pergeseran 30 dan 60. Oleh karena itu, semakin kecil nilai parameter pergeseran maka semakin bagus performansi *success plot*.

3.2.2 *Precision Plot*

Pada gambar 4.3b, nilai *precision* terbaik terdapat pada nilai parameter pergeseran 10 juga dengan nilai 0.470. kemudian dari persentase *precision* pada grafik tersebut dapat disimpulkan bahwa nilai parameter pergeseran 10 yang hasilnya lebih bagus dibandingkan pada pergeseran 30 dan 60. Oleh karena itu, semakin kecil nilai parameter pergeseran maka semakin bagus performansi *precision plot*.

4 KESIMPULAN DAN SARAN

4.1 Kesimpulan

Setelah dilakukan simulasi dan analisis pada sistem, maka dapat diambil kesimpulan sebagai berikut:

1. Pengaruh parameter skala untuk deteksi objek pada TLD adalah semakin kecil nilai parameter skala maka semakin baik nilai performansi *success plot* maupun *precision plot*. Ini dikarenakan semakin kecil skala saat proses scaning grid maka akan menghasilkan kemungkinan *bounding box* yang lebih banyak.
2. Pengaruh parameter pergeseran untuk deteksi objek pada TLD adalah semakin kecil nilai parameter pergeseran maka semakin baik nilai performansi *success plot* maupun *precision plot*. Ini dikarenakan semakin pendek jarak pergeseran saat proses *scaning grid* maka akan menghasilkan kemungkinan *bounding box* yang lebih banyak.

4.2 Saran

Untuk pengembangan lebih lanjut, maka perlu diperhatikan beberapa saran dibawah ini:

1. Melakukan inisialisasi *bounding box* secara otomatis, menyerupai *groundtruth* agar pengujian lebih presisi lagi.

DAFTAR PUSTAKA

[1] Z. Kalal, K. Mikolajczyk and J. Matas, "Tracking-Learning-Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 6, Januari 2010.

[2] G. Nebehay, Robust Object Tracking Based on Tracking-Learning-Detection, Wien, 2012.

[3] Z. Kalal, Tracking Learning Detection, Surrey, 2011.

[4] M. Kolsch and M. Turk, "Fast 2D Hand Tracking with Flocks of Features and Multi-Cue Integration," in *Conference on Computer Vision and Pattern Recognition Workshop*, 2004.

[5] R. S. Chavan and S. M. Patil, "Object Tracking Based On Tracking-Learning-Detection," *International Journal of Scientific & Engineering Research,* vol. 4, no. 3, Maret 2013.

[6] H. Sajati, "Deteksi Objek Menggunakan Haar Cascade Classifier," 2015. [Online].

[7] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision.," in *International Joint Conference on Artificial Intelligence*, 1981.

[8] Y. Wu, J. Lim and M.-H. Yang, "Visual Tracker Benchmark," [Online]. Available: http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html.

[9] J. Ray, J. Jong and P. Chen, "Implementation of TLD Long-Term Tracking Framework".