BAB II

LANDASAN TEORI

2.1. Keaslian Penelitian

Untuk mendukung penelitian ini maka dibutuhkan landasan teori yang diambil dari beberapa jurnal penelitian terdahulu atau yang sudah ada dan berkaitan dengan judul penelitian dan pokok bahasan dan penelitian. Adapun tinjauan terkait dari penelitian terdahulu yang dapat dilihat pada tabel 2.1 berikut ini :

Tabel 2.1 Keaslian Penelitian

No. Literatur	Peneliti Tahun	Judul Penelitian
Literatur 01	(Vazirani & Kautsar, 2016)	Implementasi Object Tracking Untuk Mendeteksi dan Menghitung Jumlah Kendaraan Secara Otomatis Menggunakan Metode Kalman Filter dan Gaussian Mixture Model
Literatur 02	(Asri & Firmansyah, 2018)	Implementasi Objek Detection dan Tracking Menggunakan Deep Learning untuk Pengolahan Citra Digital
Literatur 03	(Prabowo & Abdullah, 2018)	Deteksi dan Perhitungan Objek Berdasarkan Warna Menggunakan Color Object Tracking
Literatur 04	(Jatmiko et al., 2019)	Sistem Tracking Objek

		Menggunakan Metode Edge Detection pada Quadcopter
Literatur 05	(Yunardi et al., 2019)	Desain dan Implementasi Visual Object Tracking Menggunakan Pan and Tilt Vision System

2.2. Tinjauan Pustaka

Dalam landasan teori ini akan diuraikan tentang hasil penelitian yang pernah dilakukan peneliti lain dan mempunyai kaitan topik dengan penelitian yang akan penulis lakukan, diantaranya:

1. Penelitian yang dilakukan oleh Havez Vazirani Al Kautsar dan Kusworo Adi (2016) dari Jurusan Fisika, Fakultas Sains dan Matematika, Universitas Diponegoro yang berjudul "Implementasi Object Tracking Untuk Mendeteksi dan Menghitung Jumlah Kendaraan Secara Otomatis Menggunakan Metode Kalman Filter Dan Gaussian Mixture Model". Dalam penelitian ini penulis mengangkat masalah yaitu proses akuisisi data volume kendaraan yang lewat masih dilakukan secara manual, sehingga proses pengumpulan data akan lama dan jumlah sumber daya manusia yang dibutuhkan juga harus banyak. Maka berdasarkan kondisi tersebut dibuatlah sistem penghitung dan deteksi kendaraan otomatis yang akurat sebagai pemantau kontrol lalu lintas dan analisa lalu lintas. Metode yang digunakan dalam mendeteksi dan menghitung jumlah kendaraan yaitu dengan metode Kalman Filter dan *Gaussian Mixture Model* (GMM). Hasil akurasi pada penelitian ini adalah pada pagi hari (pencahayaan 10.000- 25.000 *lux*)

dengan nilai *F1 Score* sebesar 0,91111, sedangkan penghitungan kendaraan yang paling tidak akurat terjadi pada malam hari (pencahayaan 0,27-1,0 *lux*) dengan *F1 Score* 0,16071.

2. Penelitian yang dilakukan oleh Jefry Sunupurwa Asri dan Gerry Firmansyah, (2018) dari universitas esa unggul yang berjudul "Implementasi Objek Detection dan Tracking Menggunakan Deep Learning Untuk Pengolahan Citra Digital". Penelitian ini bertujuan untuk mendeskripsikan peningkatan hasil pengolahan citra digital melalui *objek detection* dan *tracking* dengan memanfaatkan arsitektur deep learning. Metode yang digunakan dalam penelitian ini adalah *deep learning*, *CAMShift*, dan *Faster R-CNN*. Berdasarkan penelitian tersebut, dapat diambil kesimpulan bahwa penggabungan metode *deep learning*, *CAMShift*, dan *Faster R-CNN* dapat dilakukan, akan tetapi masih mempunyai beberapa kelemahan, yaitu pada saat proses penyimpanan data digital membutuhkan waktu proses penyimpanan yang lama dan akurasi citra gambar digital belum memuaskan.
3. Penelitian yang dilakukan oleh Dedy Agung Prabowo, Dedy Abdullah dan Ari Manik (2018) dari Program Studi Informatika, Fakultas Teknik, Universitas Muhammadiyah Bengkulu yang berjudul "Deteksi dan Perhitungan Objek Berdasarkan Warna Menggunakan Color Object Tracking". Dalam penelitian ini masalah utama yang menjadi latar belakang penelitian ini yaitu bagaimana *computer vision* dapat mengenali gambar dan memanfaatkan gambar yang ditangkapnya, sehingga dalam penelitian ini dibangun sistem yang dapat mendeteksi dan menghitung objek yang

ditangkap kamera berdasarkan warna. Berdasarkan penelitian tersebut dihasilkan bahwa *color object tracking* dapat digunakan untuk mengenali dan menghitung warna yang ditangkap dengan kamera.

4. Penelitian yang dilakukan oleh Dimas Bagus Jatmiko, Gembong Edhi Setyawan dan Hurriyatul Fitriyah (2019) dari Program Studi Teknik Komputer, Fakultas Ilmu Komputer, Universitas Brawijaya yang bejudul "Sistem Tracking Objek Menggunakan Metode Edge Detection pada Quadcopter". Dalam penelitian ini terdapat masalah yaitu bagaimana membuat sistem yang dapat mengontrol dan mengawasi objek atau target yang membutuhkan pengawasan agar dapat berjalan sebagaimana mestinya. Maka dalam penelitian ini dibuatlah sistem untuk mendeteksi dan melakukan tracking terhadap objek dengan menggunakan kamera pada *quadcopter* sehingga setiap gerakan dari objek akan diketahui langsung melalui citra kamera. Dengan menerapkan sistem pendeteksi objek berbasis pengolahan citra menggunakan metode *shape detection*. Hasil dari penelitian ini menghasilkan tingkat akurasi *tracking*/pelacakan objek yang didapatkan rata-rata keberhasilan 76,6% serta kegagalan sebesar 23,4% yang disebabkan oleh pencahayaan dalam ruangan yang tidak menentu dan benda dengan bentuk menyerupai objek yang di *tracking*.
5. Penelitian yang dilakukan oleh Riky Tri Yunardi, Ajeng W. Mardhiyah, M. Hafi Yahya dan Franky Chandra Satria Arisgraha (2019) yang berjudul "Desain dan Implementasi Visual Object Tracking Menggunakan Pan and Tilt Vision System". Dalam penelitian ini terdapat masalah yaitu bagaimana membantu petugas keamanan dalam memantau/memonitoring

ruangan yang memiliki area kerja yang luas. Maka dalam penelitian ini didesain dan dibangun sistem *visual object tracking* dengan menggunakan *Pan* and *Tilt Vision* berbasis kamera yang dapat melakukan *tracking* posisi dan mengontrol arah hadap kamera tetap dijangkauan kerja pandangan kamera. Hasil dari penelitian ini menunjukkan bahwa desain *tracking*/pelacakan objek dapat mendeteksi dan mengikuti manusia berjalan dengan jarak ideal 6 meter dan *directional angular shift* 5 derajat pada resolusi visual 360 × 240 piksel.

2.3. *Artificial Intelligence* (Kecerdasan Buatan)

Kecerdasan Buatan (AI) merupakan salah satu cabang Ilmu pengetahuan yang erat hubunganmya dengan bagaimana dalam memanfaatkan mesin untuk memecahkan suatu permasalahan yang sulit dengan cara yang lebih manusiawi (Nugraha & Winiarti, 2014). Mesin bisa bertindak seperti manusia yang dibekali pengetahuan serta kemampuan menalar yang baik. Penelitian dalam AI menyangkut pembuatan mesin untuk mengotomatisasikan tugas-tugas yang membutuhkan perilaku cerdas. Termasuk contohnya adalah pengendalian, perencanaan dan penjadwalan/pengingat, kemampuan untuk menjawab diagnosa dan pertanyaan pelanggan, serta pengenalan tulisan tangan, suara dan wajah. AI biasanya dihubungkan dengan Ilmu Komputer, akan tetapi juga terkait erat dengan bidang lain seperti Matematika, Psikologi, Pengamatan, Biologi, Filosofi, dan yang lainnya.

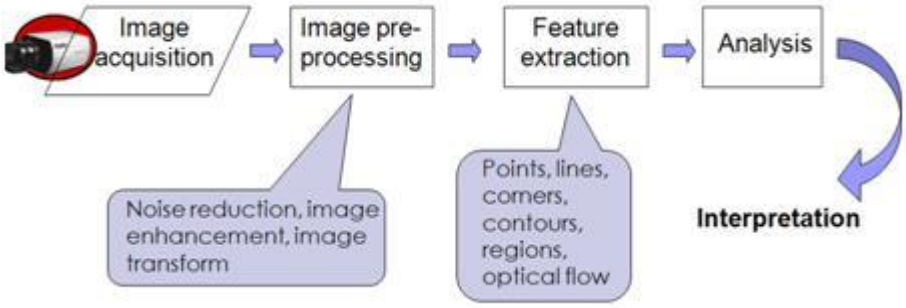
2.4. *Computer Vision*

Computer Vision merupakan cabang dari teknik kecerdasan buatan atau *artificial intelligence* yang berhubungan dengan kegiatan simulasi visualisasi manusia (Masithoh et al., 2011). Secara fungsional, *computer vision* dan penglihatan manusia adalah hal yang sama, dengan memiliki tujuan menafsirkan data spasial yaitu data yang diindeks oleh lebih dari satu dimensi. Sebagaimana layaknya mata dan otak, *computer vision* adalah suatu sistem yang mempunyai kemampuan untuk menganalisis obyek secara visual, setelah data obyek yang bersangkutan dimasukkan dalam bentuk citra.

Proses-proses dalam *computer vision* dibagi dalam 3 (tiga) aktifitas :

1. Memperoleh atau mengakuisisi citra digital, proses ini bisa disebut juga sebagai proses *image preprocessing*.
2. Melakukan teknik komputasi untuk memproses atau memodifikasi data citra berupa piksel.
3. Menganalisis dan menginterpretasi citra menggunakan hasil pemrosesan untuk tujuan tertentu, misalnya memandu robot, mengontrol peralatan, memantau manufaktur dan lain-lain

Akusisi dan pemrosesan informasi berupa *vision* dapat ditampilkan pada gambar berikut :



Gambar 2.1 Model Persepsi visual pada *computer vision*

2.5. *Machine Learning*

Menurut Ginting dan Rohmadi (2015) *Machine learning* merupakan bagian dari kecerdasan buatan (*Artificial Intelligence*). *Machine learning* adalah salah satu disiplin ilmu dari *computer science* yang mempelajari tentang bagaimana membuat komputer atau mesin agar memiliki suatu kecerdasan. Agar mempunyai suatu kecerdasan, komputer atau mesin harus dapat belajar. Atau dengan kata lain *machine learning* adalah bidang keilmuan yang mempelajari tentang bagaimana pembelajaran komputer atau mesin agar menjadi cerdas. *Machine learning* berhubungan erat dengan *computational statistics* yang berfokus pada suatu prediksi atau pembuatan keputusan berdasarkan penggunaan komputer. Beberapa implementasi dari *machine learning* yaitu seperti *text analysis, image processing, fincance, search and recommendation engine, speech understanding.*

2.6. *Deep Learning*

Menurut Zulfa dan Winarko (2017) *Deep Learning* merupakan area baru dalam penelitian *Machine Learning*, yang memanfaatkan jaringan syaraf tiruan untuk implementasi/penerapan permasalahan dengan dataset/sekumpulan data

yang besar. *Deep Learning* adalah tentang bagaimana mempelajari beberapa tingkat representasi/gambaran dan abstraksi/gambaran umum yang dibuat untuk membantu dalam memahami data seperti suara, gambar, dan teks.

2.7. Citra

2.7.1. Definisi Citra

Citra atau *image* adalah keluaran/*output* dari suatu sistem perekaman data yang dapat bersifat optik berupa foto/gambar, yang bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital sehingga dapat disimpan langsung pada suatu pita magnetik. Citra dapat diklasifikasikan menjadi 4 kelas berdasarkan presisi yang dipakai untuk menyatakan titik-titik koordinat dan untuk menyatakan nilai/tingkat keabuan (*gray scale*) atau warna suatu citra, yaitu:

1. Citra kontinu-kontinu
2. Citra kontinu-diskrit
3. Citra diskrit-kontinu
4. Citra diskrit-diskrit

Label pertama menyatakan presisi dari titik-titik koordinat pada bidang citra, sedangkan label kedua menyatakan tingkat keabuan atau warna. Kontinu dinyatakan dengan presisi angka tak terhingga sedangkan diskrit dinyatakan dengan presisi angka tertentu.

2.7.2. Definisi Citra Digital

Citra digital adalah sebuah matriks dimana indeks baris dan kolom menyatakan suatu titik pada suatu citra dan elemen matriksnya yang disebut

sebagai elemen gambar atau piksel menyatakan tingkat nilai derakat keabuan pada titik tersebut. Proses perubahan citra analog menjadi citra digital disebut dengan proses Digitasi. Digitasi adalah proses pengubahan gembar atau teks dari benda yang dapat dilihat ke dalam bentuk data elektronik yang dapat di simpan dan di proses untuk keperluan lainnya.

Untuk mengubah citra yang bersifat kontinu menjadi citra digital, diperlukan proses pembuatan kisi-kisi arah horizontal dan vertikal, sehingga diperoleh gambar dalam bentuk array dua dimensi. Setiap elemen *array* tersebut dikenal sebagai elemen gambar atau piksel (pixel) yang merupakan singkatan dari *picture element*. Tingkat keabuan setiap piksel dinyatakan dengan suatu *integer*.

2.7.3. Tipe Citra Digital

Citra digital dapat dikategorikan mejadi beberapa jenis, diantara adalah sebagai berikut:

1. Citra Biner

   Citra biner adalah citra yang hanya punya satu channel warna dan hanya memiliki dua kemungkinan nilai, yaitu 0 dan 1. Piksel yang bernilai 0 melambangkan warna putih dan piksel yang bernilai 1 melambangkan warna hitam pada saat menampilkan citra. Untuk mengubah citra berwarna menjadi citra biner, citra berwarna diubah ke bentuk citra keabuan lalu tentukan nilai threshold (batas), jika nilai pada piksel sama dengan atau melebihi nilai threshold, maka nilai piksel tersebut dikonversi menjadi 1, dan jika sebaliknya, dikonversi menjadi 0, seperti yang dinyatakan pada rumus berikut ini :

$$y' = \begin{cases} 1, & y \geq threshold \\ 0, & otherwise \end{cases}$$

2. Citra Keabuan

   Citra keabuan (greyscale) adalah citra yang disetiap pikselnya mengandung satu layer dimana nilai intensitasnya berada pada nilai 0 (hitam) sampai 255 (putih). Untuk mengubah citra berwarna menjadi citra grayscale, ada beberapa rumus yang dapat kita gunakan, salah satunya adalah sebagai berikut :

$$y = \frac{1}{3}(R + G + B)$$

3. Citra Warna

   Citra warna (RGB) adalah citra digital yang memiliki informasi warna pada setiap pikselnya. Sistem pewarnaan citra warna ada beberapa macam seperti RGB, CMYK, HSV, dll. RGB adalah model warna yang terdiri dari merah, hijau, dan biru yang digabungkan dalam membentuk suatu susunan warna yang luas. Untuk monitor komputer, nilai rentangnya paling kecil = 0 dan paling besar = 255.

2.8. *Image Processing*

Menurut Ahmad (2014) *image processing* atau pengolahan citra adalah proses dalam mengamati dan menganalisa suatu objek tanpa harus berhubungan langsung dengan objek yang diamati. Proses dan analisanya melibatkan persepsi visual melalui data *input* dan data *output* yang diperoleh berupa citra dari objek yang diamati. Teknik *image processing* meliputi penajaman citra, penonjolan fitur

tertentu pada suatu citra, kompresi citra dan koreksi citra yang tidak fokus atau *blur*.

Sebuah gambar dapat disebut sebagai citra digital jika gambar yang dihasilkan berasal dari proses sebuah komputer, kamera, scanner atau perangkat elektronik lainnya. Tujuan pengolahan citra antara lain dapat memperbaiki kualitas gambar dilihat dari aspek radiometrik (peningkatan kontras, transformasi warna, restorasi citra) dan dari aspek geometrik (rotasi, translasi, skala, transformasi geometrik). Pengolahan citra banyak digunakan dalam dunia fotografi (misalnya mengubah intensitas cahaya sebuah foto), dunia perfilman (animasi), dunia kedokteran (untuk membuat analisa medis), dan dunia game.

2.9. Segmentasi

Segmentasi adalah sebuah metode untuk membagi sebuah gambar menjadi beberapa sub gambar yang disebut sebagai area atau region. Terdapat 2 pendekatan yang dapat digunakan dalam membagi beberapa gambar tersebut menjadi beberapa daerah tertentu yaitu *Region-based* dan *Boundary estimation* dengan menggunakan *edge detection*.

Pada pendekatan *Region-based*, semua piksel yang berhubungan dengan sebuah objek dikelompokkan bersama dan diberikan *flag* yang menandakan bahwa semua piksel tersebuit merupakan satu area. Dua prinsip yang penting dalam pendekatan ini adalah *value similiarity* dan *spatial proximity*. Dua piksel dapat dikelompokkan menjadi satu apabila memiliki karakteristik intensitas yang sama atau jika keduanya memiliki jarak yang saling berdekatan. Sedangkan pada pendekatan *boundary estimation* menggunakan *edge detection*, segmentasi

dilakukan dengan menemukan piksel- piksel yang terletak pada sebuah batas area. Piksel/*edge* tersebut dapat diperoleh dengan melihat piksel-piksel yang berdekatan (*neighbouring pixels*).

2.10. *Object Tracking*

Gede dkk., (2015) menyatakan bahwa *Object Tracking* merupakan suatu metode yang digunakan dalam menemukan sebuah objek tertentu pada sebuah citra yang terdapat pula objek-objek lain didalamnya yang ikut terekam bersama-sama dengan objek yang dijadikan pusat perhatian.

Menurut Yilmaz dkk., (2006) Object Tracking biasanya digunakan dalam beberapa hal antara lain sebagai berikut:

a. Pemantauan secara otomatis, hal ini biasanya digunakan dalam mengawasi/memonitor kegiatan atau kejadian mencurigakan atau peristiwa yang tidak biasa.
b. Pengenalan objek bedasarkan gerak, contohnya dalam mengidentifikasi manusia berdasarkan gaya berjalan, deteksi objek otomatis dan lain-lain.
c. *Video Indexing*, dapat dilakukan dengan cara mengambil video dari database multimedia.
d. Interaksi manusia dengan komputer, contohnya dalam pengenalan gerakan manusia, pengenalan bola mata manusia untuk input ke dalam komputer dan lain sebagainya.
e. Pemantauan lalu lintas, yaitu mengumpulkan data statistik dari lalu lintas.
f. Navigasi kendaraan, membentuk rute perjalanan berbasis video dan kemampuan menghindari objek.

Beberapa perbandingan Algoritma untuk *tracking* objek menurut (Kanjee, 2020) :

1. Tracktor ++ dinilai cukup akurat, namun terdapat satu kelemahan besar yang dimiliki Tracktor yaitu tidak dapat digunakan untuk pelacakan secara *realtime*. Hasil menunjukkan eksekusi rata-rata sebesar 3 FPS.
2. TrackR-CNN bagus karena menyediakan segmentasi sebagai bonus. Tetapi sama halnya seperti Tracktor ++, sulit digunakan untuk pelacakan *realtime*, dengan eksekusi rata-rata 1,6 FPS.
3. JDE menampilkan kinerja rata-rata 12 FPS yang menjadikannya layak. Namun perlu diperhatikan bahwa ukuran input untuk model tersebut adalah 1088x608, oleh karena itu, jika JDE digunakan untuk full HD akan memiliki FPS. Meski demikian, JDE memiliki akurasi yang tinggi dan menjadi pilihan yang baik.
4. *DeepSORT* adalah yang tercepat dari ketiga algoritma diatas, berkat kesederhanaannya. *DeepSORT* menghasilkan rata-rata 16 FPS dan tetap mempertahankan tingkat keakuratan yang baik, yang menjadikannya pilihan yang tepat untuk melakukan deteksi dan *tracking* beberapa objek.

2.11. *Object Detection*

Menurut Jalled dan Voronkov (2016), *Object detection* dapat menentukan keberadaan sebuah objek dan ruang lingkupnya serta lokasi pada sebuah gambar. Hal ini dapat dilakukan sebagai pengenalan objek dua kelas, dimana satu kelas mewakili kelas objek dan kelas lain mewakili kelas non-objek. *Object Detection* dapat dibagi lagi menjadi *soft detection* dan *hard detection*. Dimana *Soft detection*

hanya mendeteksi adanya objek saja sedangkan *hard detection* dapat mendeteksi adanya objek serta lokasi.

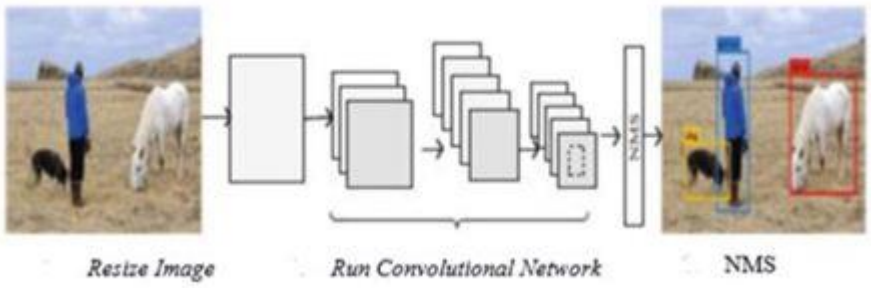
2.12. Pengenalan Pola (*Pattern Recognition*)

Pengenalan pola yaitu mengelompokkan data secara numerik dan simbolik (termasuk citra) secara otomatis oleh komputer (Rafiah et al., 2015). Tujuan pengelompokkan adalah untuk mengenali suatu objek di dalam citra. Manusia bisa mengenali objek yang dilihatnya karena otak manusia telah belajar mengklasifikasi objek-objek yang ada di alam sehingga mampu membedakan objek satu dengan objek lainnya. Kemampuan sistem visual manusia inilah yang dicoba ditiru oleh komputer. Komputer menerima *input* berupa citra objek yang akan diidentifikasi, kemudian memproses citra tersebut dan memberikan *output* berupa deskripsi objek di dalam sebuah citra.

2.13. *You Only Look Once* (YOLO)

Dalam Jupiyandi et al., (2019), Unsky menyatakan bahwa *You Only Look Once* (Yolo) merupakan sebuah algoritma yang dikembangkan untuk melakukan deteksi terhadap sebuah objek secara *real-time*. Sistem pendeteksian dilakukan dengan menggunakan *repurpose classifier* atau *localizer*. Sebuah model diterapkan pada sebuah citra di beberapa lokasi dan skala. Daerah dengan citra yang dengan score yang paling tinggi akan dianggap sebagai sebuah pendeteksian. Cara kerja *You Only Look Once* (YOLO) adalah dengan mengamati seluruh gambar dalam sekali waktu, lalu melewati jaringan syaraf sekali waktu pula lalu langsung mendeteksi objek yang ada (Wasril dkk, 2019).

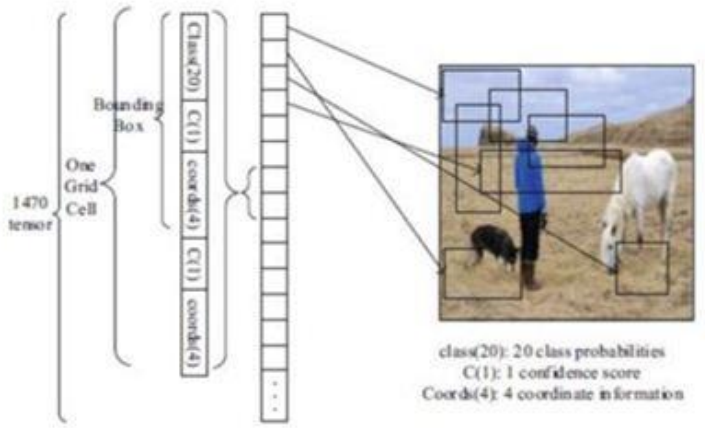
Pertama-tama, gambar input diubah menjadi 448 x 448. Kemudian menjalankan *single convolutional network* pada gambar, *bounding box* serta skor keyakinannya akan diperoleh. Pada akhirnya, NMS (*non-max suppression*) digunakan untuk mengthreshold deteksi yang dihasilkan oleh kepercayaan model, probabilitas kelas final dan koordinat *bounding box* yang diperoleh seperti yang ditunjukkan pada bagian paling kanan dari gambar 2.2 berikut:



Gambar 2.2 Proses Deteksi YOLO

YOLO membagi gambar input menjadi 7 x 7 grid. Setiap sel grid sesuai dengan 2 kotak pembatas. Setiap sel grid memprediksi satu set probabilitas kelas (termasuk 20 kelas) terlepas dari jumlah kotak. Dan setiap *bounding box* terdiri dari 5 prediksi: x, y, w, h, dan keyakinan. Koordinat (x, y) mewakili pusat kotak relatif terhadap batas – batas sel grid. (w, h) mewakili lebar dan tinggi kotak relatif terhadap keseluruhan gambar. Dengan demikian, setiap grid memiliki prediksi (4 +1) x 2 + 20 = 30 dan seluruh gambar memiliki 7 x 7 x 30 = 1470 prediksi. YOLOs CNN mencakup 25 *convolutional layers*, 4 *pooling layers*, 1 *dropout layer*, 1 *fully connected layer* dan 1 lapisan deteksi. Dimensi dari lapisan yang terhubung sepenuhnya adalah 1470, yang menunjukkan posisi dan keyakinan kotak prediksi yang terikat dan probabilitas kelasnya. Untuk CNN yang diinisialisasi secara acak, output dari lapisan yang terhubung sepenuhnya juga

acak pada awal proses pelatihan, yaitu ada kesenjangan besar antara kotak prediksi dan kotak *ground truth* di awal. Gambar yang ditunjukkan pada gambar 2.3, mengungkapkan pemetaan antara lapisan yang terhubung penuh dan kotak prediksi pada gambar. Diperlukan banyak iterasi untuk mendekati kotak ground truth, menghasilkan kecepatan konvergensi yang lambat. Disisi lain, YOLO mengusulkan kotak batas yang jauh lebih sedikit, hanya 7 x 7 x 2 = 98 per gambar, jadi kesalahan lokalikasinya besar.



Gambar 2.3 Pemetaan antara layer yang terhubung penuh dan kotak prediksi YOLO

2.14. *DeepSORT*

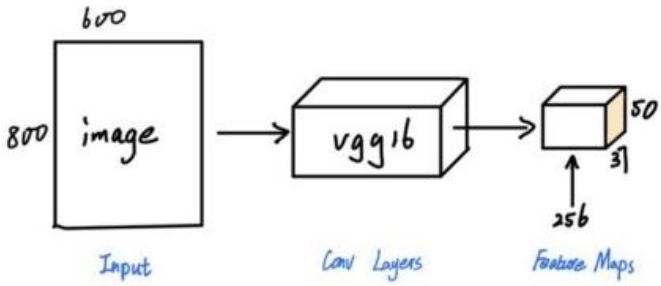
Menurut Wojke dkk., (2018) *Deep SORT* merupakan pengembangan dari kinerja *Simple Online and Realtime Tracking* (SORT) dalam hal melacak objek melalui periode *occlusion* (keadaan macet) yang lebih lama, dan secara efektif mengurangi jumlah pergantian identitas. Sebagian besar kesulitan proses komputasi dilakukan dalam tahap *pre-training offline* di mana *deep association metric* (metrik asosiasi yang mendalam) lebih dipelajari pada dataset identifikasi

ulang objek yang berskala besar. Pada aplikasi online, *measurement-to-track associations* (asosiasi pengukuran-ke-trek) menggunakan kueri tetangga terdekat dalam ruang tampilan visual.

Terdapat 4 komponen penting dalam SORT di antaranya adalah sebagai berikut:

1. Deteksi

Deteksi saja tidak cukup untuk melakukan pelacakan, namun kualitas pendeteksian memiliki pengaruh yang signifikan untuk kinerja *tracking*.



Gambar 2.4 Ilustrasi Deteksi

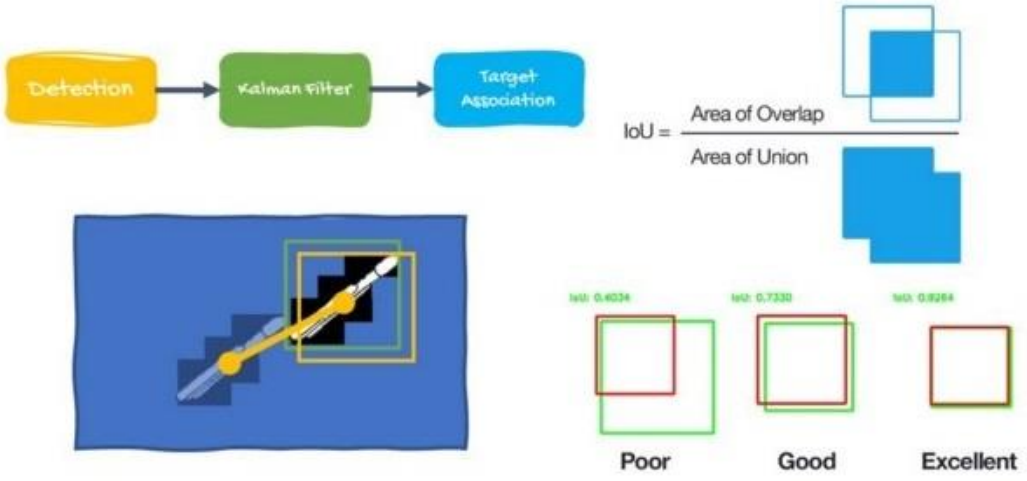
2. Estimasi/perkiraan

Selanjutnya, menyebarkan deteksi dari frame saat ini ke frame berikutnya menggunakan model kecepatan konstan linier. Saat deteksi dihubungkan kepada target, *bounding box* yang telah terdeteksi digunakan untuk memperbarui status target di mana komponen kecepatan diselesaikan secara optimal melalui kerangka kerja Kalman filter. Namun jika tidak ada target yang terdeteksi, maka statusnya hanya diprediksi tanpa memeriksa menggunakan model kecepatan linier.

3. Asosiasi Target

Dalam menetapkan deteksi ke target yang ada, setiap geometri *bounding box* target diperkirakan dengan memprediksi lokasi barunya dalam *frame* terbaru. Nilai matriks penugasan kemudian dihitung sebagai jarak *intersection-over-union*

(IOU) antara setiap deteksi dan semua *bounding box* yang telah diprediksi pada target yang ada.



Gambar 2.5 Ilustrasi Asosiasi Target

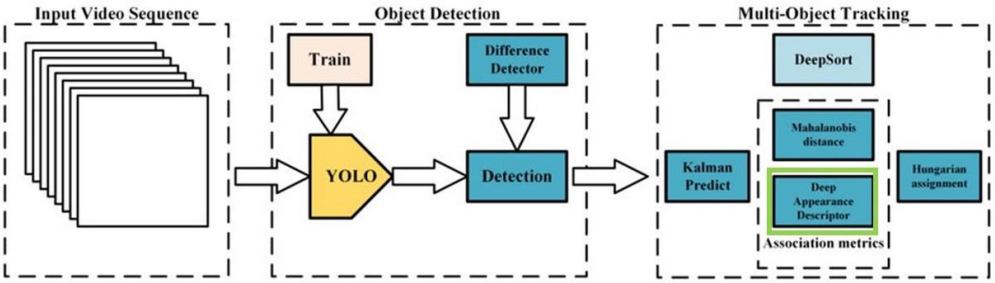
Tugas diselesaikan secara optimal menggunakan algoritma Hungaria. Sehingga bekerja sangat baik ketika satu target menghalangi yang lain.

4. Melacak Siklus Hidup Identitas/*Track Identities Life Cycle*

Ketika objek masuk dan keluar dari gambar, identitas yang unik perlu dibuat atau dihilangkan. Untuk membuat pelacak, setiap deteksi apa pun yang *overlap* kurang dari IOUmin untuk menandakan keberadaan objek yang tidak terlacak. Pelacak diinisialisasikan menggunakan geometri *bounding box* dengan kecepatan diatur ke nilai nol. Karena kecepatan tidak dapat teramati pada titik ini, maka kovariansi komponen kecepatan diinisialisasi dengan nilai yang besar, yang menandakan ketidakpastian. Selain itu, pelacak baru kemudian menjalani masa percobaan di mana target perlu dihubungkan dengan deteksi untuk mengumpulkan cukup bukti yang berfungsi untuk mencegah pelacakan positif yang palsu.

Tracks dihentikan bila tidak terdeteksi frame TLost, hal ini dapat kita tentukan sendiri berapa jumlah frame yang seharusnya untuk TLost. Jika sebuah objek muncul kembali, pelacakan secara mutlak akan dilanjutkan dengan nomor identitas yang baru.

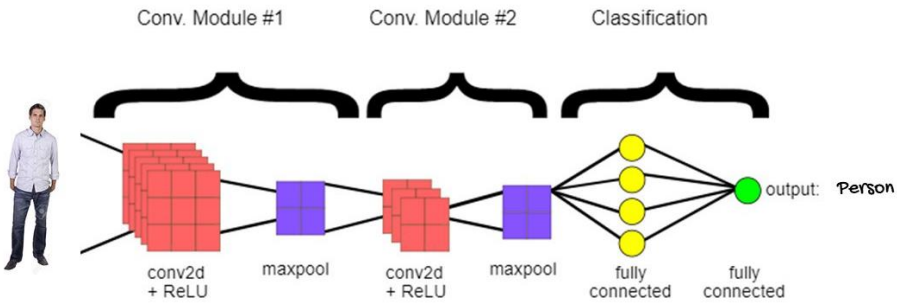
Objek Detector melakukan deteksi, Kalman filter melacaknya dan memberi *track* yang hilang, algoritma Hungarian menghubungkan deteksi dengan objek yang terlacak.



Gambar 2.6 Diagram *DeepSORT*

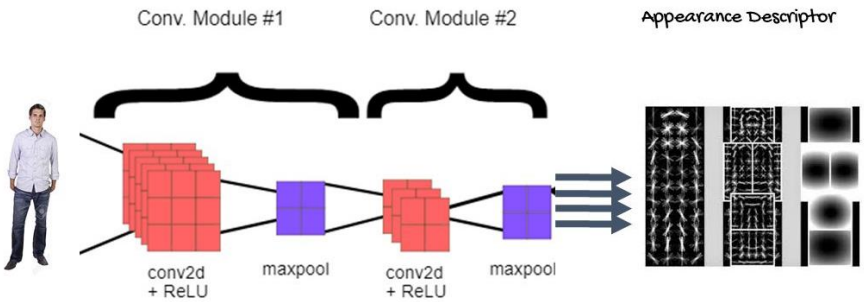
SORT memiliki kinerja yang baik secara keseluruhan dalam hal ketepatan dan akurasi *tracking*, dan terlepas dari efektivitasnya Kalman filter, SORT mengembalikan jumlah *identity switches* yang relatif tinggi namun memiliki kekurangan dalam hal melacak melalui oklusi dan sudut pandang yang berbeda, dll. Untuk meningkatkan hal tersebut, penulis DeepSORT memperkenalkan metrik jarak lain berdasarkan "*appearance*/tampilan" objek.

Appearance menampilkan Vektor



Gambar 2.7 Ilustrasi *Appearance Descriptor* (1)

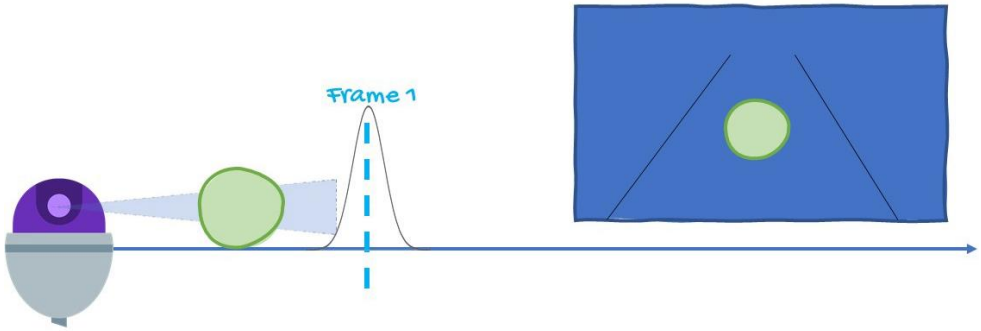
Jadi pengklasifikasi dibangun berdasarkan dataset yang telah di *training* hingga mencapai tingkat akurasi yang cukup baik. Kemudian kita mengambil jaringan ini dan mengupas lapisan klasifikasi terakhir untuk meninggalkan lapisan yang padat sehingga menghasilkan vektor fitur tunggal, menunggu untuk diklasifikasikan. Fitur vektor ini dikenal sebagai *appearance descriptor*/deskriptor tampilan.



Gambar 2.8 Ilustrasi *Appearance Descriptor* (2)

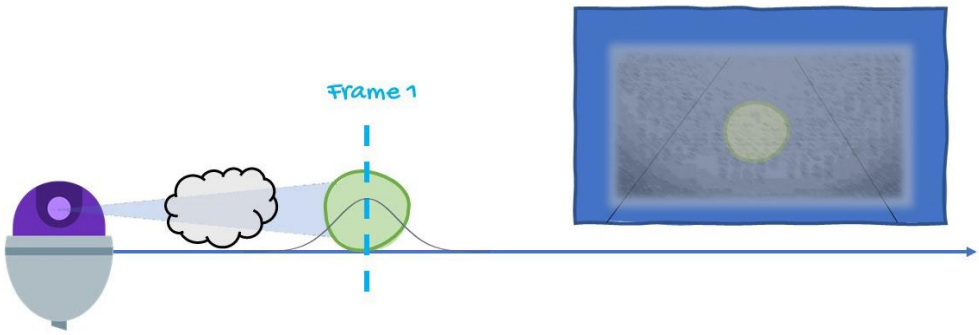
Sekarang cara kerjanya adalah setelah deskriptor *appearance*/tampilan diperoleh, gunakan kueri tetangga yang terdekat dalam *appearance*/tampilan visual untuk menetapkan asosiasi pengukuran-ke-trek. *Measurement-to-track association* atau MTA adalah proses untuk menentukan hubungan antara pengukuran dengan *track* yang ada. Jadi sekarang gunakan jarak *Mahalanobis* untuk *Measurement-to-track association* (MTA).

Kalman Filter



Gambar 2.9 Ilustrasi Kalman Filter (1)

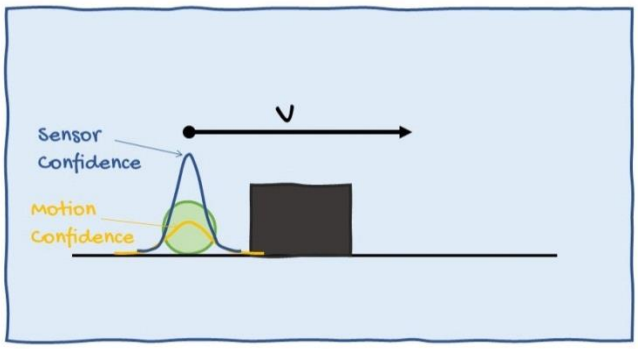
Ketika melacak bola yang menggelinding dalam 1 dimensi. Kita dapat dengan mudah mendeteksinya dalam setiap frame. Deteksi tersebut adalah sinyal input yang dapat diandalkan selama ada garis pandang yang jelas ke bola, dengan noise yang sangat rendah. Sekarang selama proses deteksi, simulasikan kondisi mendung menggunakan mesin kabut yang digunakan untuk terakhir. Kita masih dapat melihat bola tersebut namun sekarang sensor penglihatan memiliki noise di dalamnya, sehingga mengurangi tingkat kepercayaan di mana posisi bola itu berada.



Gambar 2.10 Ilustrasi Kalman Filter (2)

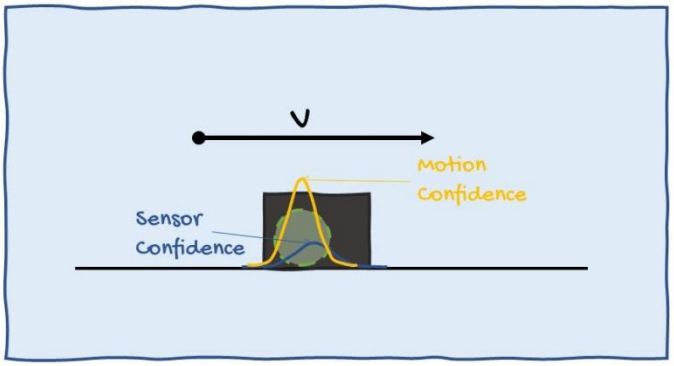
Bola bergerak di belakang kotak sehingga bola tersebut tidak dapat terlihat. Dan disinilah Kalman Filter bekerja, Kalman Filter dapat melacak sesuatu yang

tidak dapat kita lihat. Dengan asumsi model kecepatan konstan dan distribusi *gaussian*. Kita dapat memperkirakan letak bola berdasarkan model *motion*-nya. Ketika bola dapat dilihat, kita dapat lebih mengandalkan data sensor dan sehingga memberi lebih banyak beban kerja padanya. Saat sebagian tertutup, kita dapat menempatkan beban atau mengandalkan data pengukuran sensor dan *motion*.



Gambar 2.11 Ilustrasi Kalman Filter (3)

Dan jika sepenuhnya tertutup, maka akan banyak mengalihkan beban pada data *motion*. Dan bagian terbaik dari Kalman filter yaitu rekursif, artinya di mana kita dapat melakukan pembacaan untuk memprediksi keadaan saat ini, lalu menggunakannya untuk pengukuran dan memperbarui prediksi tersebut. Filter Kalman merupakan komponen yang sangat penting dalam DeepSORT.



Gambar 2.12 Ilustrasi Kalman Filter (4)

2.15. *Python*

Python merupakan salah satu contoh bahasa pemrograman tingkat tinggi. Dimana Python didesain dengan sangat mudah agar dapat mudah di baca dan dipahami, karena sama seperti bahasa pemograman yang lainnya yaitu dengan menggunakan kata dalam bahasa inggris. Bahasa tingkat tinggi juga mudah diubah portabel untuk disesuaikan dengan mesin yang menjalankannya. Hal ini berbeda dengan bahasa mesin yang hanya dapat digunakan untuk mesin tersebut. Dengan berbagai kelebihan ini, maka banyak aplikasi ditulis menggunakan bahasa tingkat tinggi.

Python dibuat dan dikembangkan lebih lanjut oleh Guido Van Rossum, yaitu seorang programmer yang berasal dari Negara Belanda. Dibuat dan dikembangkan di kota Amsterdam, Belanda pada tahun 1990. Pada tahun 1995 Python dikembangkan lagi agar lebih kompatibel oleh Guido Van Rossum. Kemudian di awal tahun 2000 versi Python dikembangkan dan diperbaharui lagi sehingga kini Bahasa Pemrograman Python telah mencapai Versi 3.

Python bisa dijalankan dan ditulis untuk membangun aplikasi di beragam sistem operasi seperti :

1. Linux/Unix
2. Microsoft Windows
3. Mac OS
4. Android
5. Java Virtual Machine
6. Symbian OS

7. Amiga
8. Palm
9. OS/2

2.16. *Darknet*

Menurut Alfarisi (2020) Darknet adalah *framework neural network* yang bersifat *open source* yang ditulis dalam Bahasa C dan CUDA. Darknet memiliki kecepatan yang sangat tinggi, proses penginstallan yang mudah serta mendukung komputasi dengan CPU ataupun GPU. Darknet memiliki beberapa fitur yang telah dikembangkan untuk proses *machine learning* diantaranya ada *object detection* dan *classification*. Penginstallan Darknet merupakan sesuatu yang harus dilakukan untuk mencoba arsitektur dan algoritma *detection* yang sedang dikembangkan, seperti :

1. *ImageNet Classification*

   Proses pengklasifikasian *image* dengan model yang popular seperti *ResNet* dan *ResNeXt*.

2. *Tiny Darknet*

   Klasifikasi gambar dengan proses dan dataset yang lebih kecil.

3. YOLO (*You Only Look Once*)

   YOLO adalah pendeteksian objek secara *real time* dengan proses kecepatan sekitar 40–90 fps jika digunakan pada GPU TITAN X.

YOLO awalnya diimplementasikan dengan bahasa C dengan *framework* yang diberi nama Darknet, kemudian dikembangkan lagi oleh Trinh Hoang Trieu

untuk diimplementasikan dengan bahasa Python dan semua platfrom dan diberi nama Darkflow. Kedua framework ini dapat menjalankan arsitektur dari YOLO.

Darknet dan Darkflow memiliki *toolkit* dan *library* yang sama, meskipun keduanya menggunakan bahasa pemrograman yang berbeda. Akan tetapi ada satu hal yang membuat darknet ini lebih baik daripada darkflow, yaitu pada kecepatan dalam video *processing*. Dalam situs resmi YOLO, dijelaskan bahwa dalam menggunakan Darkflow, waktu proses untuk video adalah sekitar 292 detik dengan frame rate 14.7 fps sedangkan dengan menggunakan darknet, waktu proses lebih efektif sekitar 111 detik dengan frame rate 38.7 fps. Maka dari itu hal tersebut mengindikasikan bahwa Darknet lebih efektif digunakan jika dibandingkan dengan darkflow terlebih jika menggunakan GPU dalam prosesnya dan menggunakan spesifikasi hardware yang terbatas.

2.17. *Precision, Recall dan F1 Score*

Menurut Setiawan (2020) *precision* adalah perbandingan antara True Positive (TP) dengan banyaknya data yang diprediksi positif. Atau dapat dirumuskan seperti :

$$precision = \frac{TP}{TP + FP}$$

Sedangkan Recall adalah perbandingan antara Benar Positif (TP) dengan banyaknya data yang sebenarnya positif. Atau dapat dirumuskan seperti ini:

$$Recall = \frac{TP}{TP + FN}$$

Perbedaan di antara keduanya adalah *precision* memiliki variabel False Positive (FP) sedangkan *recall* memiliki variabel False Negative (FN).

Dari persamaannya, dapat disimpulkan semakin kecil False Positive (FP), membuat precision semakin besar. Sedangkan untuk recall, semakin kecil False Negative (FN) membuat recall semakin besar.

Kemudian, F1-Score adalah *harmonic mean* dari *precision* dan *recall*. Atau dapat dituliskan rumus seperti ini:

$$F1 - Score = \frac{1}{2}\left(\frac{1}{precision} + \frac{1}{recall}\right)$$

Nilai terbaik F1-Score adalah 1.0 dan nilai terburuknya adalah 0. Secara representasi, jika F1-Score punya skor yang baik mengindikasikan bahwa model klasifikasi punya *precision* dan *recall* yang baik.