PERANCANGAN SISTEM DETEKSI OBJEK BERBASIS *CONVOLUTIONAL NEURAL NETWORK* MENGGUNAKAN YOLOV4 DAN OPENCV

Gregorius Yudho Baskoro[1*)], Hadha Afrisal[2] and Aghus Sofwan[3]

[123]Program Studi Sarjana Departemen Teknik Elektro,
Universitas Diponegoro Semarang Jl. Prof. Sudharto, SH, Kampus UNDIP Tembalang, Semarang 50275, Indonesia

[*)]*Email: gregoriusyudhobaskoro@gmail.com*

Abstrak

Dalam penelitian ini yang berjudul perancangan sistem deteksi objek berbasis *convolutional neural network* menggunakan yolov4 dan opencv. Judul penelitian ini ini berawal dari riset bersama dengan dosen yang dilatarbelakangi dari bagaimana cara menerapkan kecerdasan buatan sebagai basis penglihatan komputer untuk robot untuk mendapatkan sistem yang semi otonom. Sistem deteksi objek ini secara garis besar didasari oleh convolutional neural network yang diimplementasikan menjadi model YOLO yang didukung framework Darknet. Proses implementasi ini dipilih karena framework ini mendukung kemudahan pemrosesan data dari dataset hingga menjadi model yang siap pakai. Sistem deteksi objek ini dibuat dengan framework darknet yang berbasis model YOLOv4-Tiny.

Kata kunci : Deteksi Objek, *Convolutional Neural Network, OpenCV, Darknet, YOLOv4*

ABSTRACT

In this final project entitled designing an Object Detection system based on a convolutional neural network using yolov4 and opencv. The title of this final project originates from joint research with lecturers with a background of how to apply artificial intelligence as a basis for computer vision for robots to obtain semi-autonomous systems. this Deteksi objek system is mainly based on a convolutional neural network. This implementation process was chosen because there are many frameworks that support the ease of processing data from datasets to becoming a finished model. This system is built using the darknet framework based on the YOLOv4-Tiny model.

Keywords : *Object Detection, Convolutional Neural Network, OpenCV, Darknet, YOLOv4*

1. Pendahuluan

Computer Vision adalah bidang kecerdasan buatan (AI) yang memungkinkan komputer dan sistem memperoleh informasi yang berarti dari Gambar digital, video, dan input visual lainnya dan mengambil tindakan atau membuat rekomendasi berdasarkan informasi tersebut. Jika AI memungkinkan komputer untuk berpikir, *computer vision* memungkinkan mereka untuk melihat, mengamati, dan memahami [1].

Computer Vision bekerja hampir sama dengan penglihatan manusia, kecuali manusia memiliki permulaan. Penglihatan manusia memiliki keuntungan dari konteks masa hidup untuk melatih bagaimana membedakan objek, seberapa jauh mereka, apakah mereka bergerak dan apakah ada sesuatu yang salah dalam sebuah Gambar [2].

Arthur Samuel [3] tahun 1959 mengatakan bahwa machine learningmerupakan "suatu cabang ilmu yang memberi komputer kemampuan untukbelajar tanpa diprogram secara eksplisit." *Machine Learning* adalah subbidang kecerdasan buatan (AI), yang secara luas didefinisikan sebagai kemampuan mesin untuk meniru perilaku manusia yang cerdas.

Kecerdasan buatan memanfaatkan komputer dan mesin untuk meniru kemampuan pemecahan masalah dan pengambilan keputusan dari pikiran manusia[4].

Dalam bentuknya yang paling sederhana, kecerdasan buatan adalah bidang yang menggabungkan ilmu komputer dan kumpulan data yang kuat untuk memungkinkan pemecahan masalah. Ini juga mencakup sub-bidang pembelajaran mesin dan pembelajaran mendalam, yang sering disebutkan bersamaan dengan kecerdasan buatan. Disiplin ini terdiri dari algoritme AI yang berupaya membuat sistem pakar yang membuat prediksi atau klasifikasi berdasarkan data masukan.

Deep learning pada dasarnya dibuat dari jaringan saraf buatan konvensional tetapi kinerjanya jauh lebih baik disbanding pendahulunya. Teknik *Deep Learning* yang paling baru dikembangkan telah memperoleh kinerja luar biasa yang baik di berbagai aplikasi, antara lain pemrosesan audio dan ucapan, pemrosesan data visual, dan *Natural Language Progra*mming (NLP). [5]

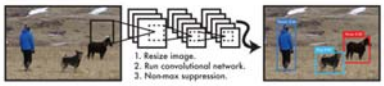
Agar sebuah model *neural network* dapat dikategorikan sebagai sebuah model *deep learning* adalah dengan memiliki 3 *layer* atau lebih. *Neural network* dengan satu

lapisan masih dapat menghasilkan prediksi yang mendekati dengan *dataset* tetapi "*Hidden Layer*" tambahan dapat membantu mengoptimalkan dan menyempurnakan akurasi. [6]

Deteksi objek adalah teknik yang membedakan objek semantik dari kelas tertentu dalam Gambar dan video digital. [7] implementasi deteksi objek dalam penelitian ini ini adalah dalam mendeteksi objek berupa barang medis.

Darknet adalah *framework* jaringan saraf *open-source* yang ditulis dalam C dan CUDA. Cepat, mudah dipasang, dan mendukug perhitungan CPU dan GPU. [8]

YOLO membingkai ulang masalah deteksi objek sebagai masalah regresi tunggal (*single regression*), langsung dari piksel ke koordinat kotak pembatas dan probabilitas kelas. Pada Gambar 1 dapat dilihat cara YOLO memprediksi objek apa yang terprediksi dan di mana objek itu berada. [9]

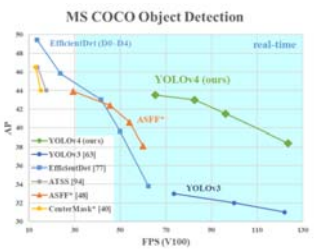


Gambar 1. Sistem Deteksi YOLO [9].

YOLO sangat sederhana, jika melihat Gambar 1, jaringan konvolusional tunggal secara bersamaan memprediksi beberapa kotak pembatas dan probabilitas kelas untuk kotak tersebut.

Deteksi dan klasifikasi objek dengan YOLOv3 mengikuti sistem YOLO9000 dengan memprediksi *bounding box* pada objek menggunakan *dimension clusters* sebagai patokan *bounding box*. Jaringan saraf buatan akan memprediksi 4 koordinat pada tiap titik *bounding box* yaitu t_x, t_y, t_w, t_h. [10]

Memahami algoritma YOLO, penting untuk menentukan apa yang diharapkan saat ini. Ini bervariasi dari sebagian besar model jaringan saraf karena menggunakan satu jaringan konvolusional yang memprediksi kotak pembatas dan probabilitas yang dihasilkan. Kotak pembatas diberi bobot oleh probabilitas dan model membuat deteksi mereka tergantung pada bobot akhir. Dengan demikian, output model end-to-end dapat langsung dimaksimalkan dan, sebagai hasilnya, Gambar dapat dihasilkan dan diproses dengan cepat. Setiap kotak pembatas bisa diwakili menggunakan empat deskriptor. [11]



Gambar 2. Perbandingan performa antara YOLOv4 dengan detektor *state-of-the-art* lainnya. [12]

Setiap titik di atas dapat digunakan untuk mewakili titik t_x, t_y, t_w, t_h. Diharapkan dengan model ini model yang ingin dilatih untuk mengenal sebuah objek dapat dilatih dengan GPU konvensional untuk melatih dan mencoba model ini sehingga bisa mendapatkan hasil model yang meyakinkan, *real-time* dan berkualitas tinggi. [12] Secara performa dan akurasi YOLOv4 juga membawa peningkatan yang lumayan baik seperti yang dapat dilihat pada grafik pada Gambar 2.

2. Perancangan Sistem

2.1. Deskripsi Model

Konsep sederhana model ini adalah bagaimana cara membuat model yang telah dilatih berdasarkan kumpulan data yang sudah diberi label, arsitektur model, loss, mAP (*mean Average Precision*), F1-Score, Precision dan Recall. Keluaran dari model ini dapat diimplementasikan pada sistem otonom robot yang berupa program yang memungkinkan robot untuk bekerja secara semi-otonom dalam mengambil dan meletakkan barang.

Pembuatan model memerlukan beberapa komponen yaitu *framework, hardware,* dan *software*. Pada komponen *framework* dalam proses pengembangan model ini, *framework* yang kami gunakan yaitu *darknet*. *Darknet* ditulis dengan bahasa C dan CUDA yang memungkinkan komputasi paralel pada GPU dan juga CPU. Komponen hardware memainkan peran penting dalam proses *training* pada model, diperlukan *hardware* dengan tingkat komputasi dan paralelisasi yang tinggi seperti GPU agar mempercepat waktu training dengan jumlah kumpulan data yang besar. Pada komponen software kami menggunakan IDE Visual Studio Code dengan *virtual environment* menggunakan anaconda. *Virtual environment* digunakan agar program *inference* dapat terisolir dari modul-modul *python* yang lain (di luar *virtual* environment) sehingga kompatibilitas antar modul dapat terjaga dengan baik.

2.1.1. Kebutuhan Fungsional

Kebutuhan fungsional merupakan Gambaran mengenai fungsi-fungsi yang dapat dilakukan oleh model ini. Kebutuhan fungsional sistem meliputi :

(1) Dapat diterapkan pada robot manipulator *mobile.*
(2) Dapat mengenal objek medis dalam hal ini alat operasi, lebih spesifiknya lagi 4 alat operasi yaitu : *curved mayo scissor, dressing scissor, gillies toothed dissector, mayo needle holder.*
(3) Menyimpan model *inference* dalam format .weights

2.1.2. Kebutuhan Perangkat Keras

Dalam proses pembuatan model, dibutuhkan perangkat keras yang memiliki kemampuan komputasi yang besar karena besarnya jumlah data yang akan diproses oleh *hardware* tersebut. Proses *training* akan semakin cepat

apabila banyak data dan kompleksitas model semakin sedikit dan sebaliknya.

Tabel 1 Kebutuhan perangkat keras

Komponen	Keterangan
CPU	AMD® Ryzen™ 7 5800X
GPU	Nvidia® RTX™ 3060Ti 8GB
RAM	TeamGroup® ElitePlus™ 16GB 3200MHz CL 22-20-20-20
Storage	SSD : Samsung® QVO™ 2TB HDD : Seagate® Barracuda™ 2TB
Motherboard	Gigabyte® B550M™ DS3H

Untuk program detektor yang dibuat untuk robot, perangkat keras yang digunakan berbeda dengan perangkat keras yang diperlukan untuk melatih sebuah model. Perangkat keras yang digunakan untuk operasional robot ini terbatas dalam segi kemampuan komputasi. Perangkat keras yang digunakan untuk detektor menggunakan konfigurasi seperti pada Tabel 2.

Tabel 2 Kebutuhan perangkat keras pada robot

Komponen	Keterangan
APU	AMD® Ryzen™ 3 3200U
RAM	12 GB
Storage	SSD 256 GB

2.1.3. Kebutuhan Perangkat Lunak

Selain *hardware*, pembuatan model juga membutuhkan banyak perangkat lunak seperti bahasa pemrograman, *virtual environment, library, framework,* dan SDK (*software development kit*).

Tabel 3. Kebutuhan perangkat Lunak

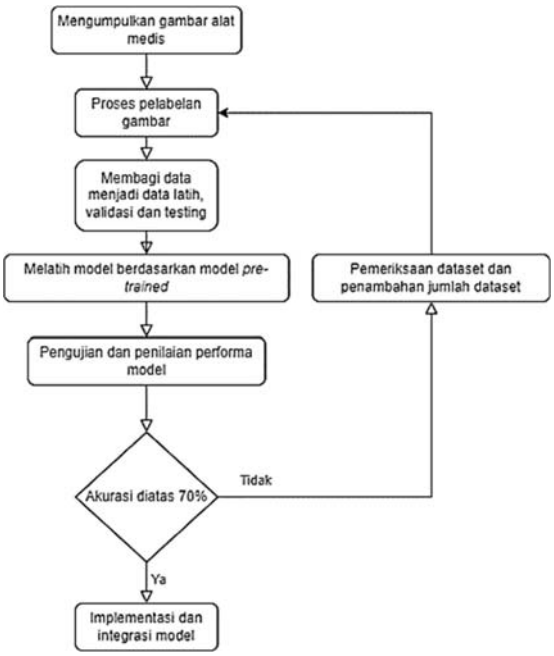
Software	Keterangan
Sistem Operasi	Windows 11
IDE	Visual Studio Code
Framework	Darknet
Bahasa Pemrograman	Python 3.6
Library dan SDK	OpenCV, PyKinect2, Wrapper For PyKinect2, Numpy, Math, Seaborn, PySerial, Kinect V2 SDK For Windows

Tabel 3 merupakan kebutuhan perangkat lunak dalam proses pembuatan model.

2.2. Perancangan Sistem

2.2.1. Alur Perancangan Sistem Deteksi Objek

Perancangan model sistem deteksi objek membutuhkan beberapa tahapan dari pengumpulan dataset hingga selesai. Keseluruhan tahapan tersebut dapat dilihat dalam diagram di bawah ini.



Gambar 3. Flowchart Perancangan Model Deteksi Objek

Perancangan model sistem deteksi objek dibagi menjadi beberapa tahapan seperti pada Gambar 3, dari tahap mengumpulkan gambar alat medis, pelabelan gambar, membagi data menjadi data latih, validasi, dan testing, melatih model berdasarkan model pre-trained, pengujian dan penilaian performa model, jika akurasi kurang dari 70% maka akan dilakukan pemeriksaan dataset dan penambahan dataset, jika akurasi lebih dari 70 % maka akan dilakukan implementasi dan integrasi model dengan sistem otonom robot medis.

Pada proses pertama yaitu pengumpulan gambar alat medis 4 kelas karena keterbatasan tempat operasional robot untuk sistem *pick and place*. Sebagai contoh kelas-kelas yang akan dipakai untuk melatih model, contoh pelabelan lengkap dengan prosesnya bisa dilihat di Gambar 4 dan Gambar 5.

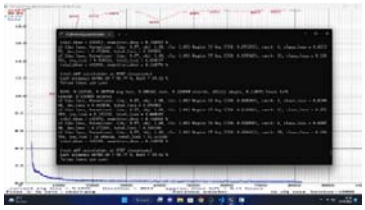
Gambar 4. Kumpulan kelas dalam satu Gambar

Gambar 5. Cuplikan Kecil Kumpulan Gambar

Proses kedua yaitu proses pelabelan gambar dapat dimulai karena sudah ditentukan metode pengumpulan dataset yang lebih baik antara kedua metode di atas. Metode pengumpulan gambar manual, seperti sudah dijelaskan di paragraf sebelumnya, diikuti dengan proses pelabelan gambar dengan *software* open-source Label IMG, proses pelabelan gambar dapat dilihat di Gambar 6.

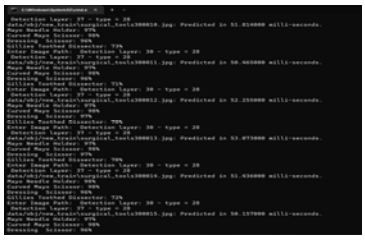
Gambar 6. Proses pelabelan Gambar dengan Label IMG

Karena proses pelabelan manual, pelabelan per gambar dan per objek, dengan bantuan *software* Label IMG maka pembagian dataset dilakukan dalam 2 fase. Fase 1 menggunakan seluruh dataset yang berlabel untuk dilatih terhadap model seperti yang terlihat pada Gambar 7.



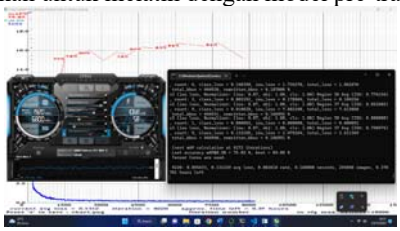
Gambar 7. Pelatihan Model dengan Dataset yang Dilabeli Secara Manual

Fase kedua bisa dilakukan jika model sudah selesai dilatih pada fase 1. Model tersebut digunakan untuk membantu membuat label pada kumpulan dataset berikutnya untuk mempermudah proses pengumpulan dataset. *Automatic-labelling* adalah nama teknik yang dilakukan pada fase kedua untuk teknik ini perlu dua langkah, langkah pertama adalah mengumpulkan gambar tanpa label sebelum diberikan label sebagai objek yang akan dideteksi, langkah kedua adalah dengan memasukkan kumpulan gambar tersebut ke *framework darknet*. Perintah dijalankan pada *command prompt* pada direktori dari *darknet*. Parameter yang bisa diatur pada saat melakukan *Auto-Labelling* adalah direktori gambar, direktori file *.weights,*besar jaringan saraf buatan, saturasi gambar, *exposure* Gambar, jumlah kelas, dan jumlah filter yang digunakan yang menghasilkan Gambar berlabel baru, *command* ini bisa dilihat luarannya pada Gambar 8.

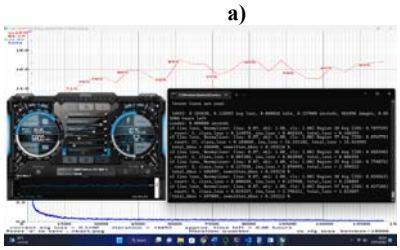


Gambar 8. Pelabelan Dataset Baru dengan Model yang Sudah Selesai Dilatih

Pada akhir pembagian data menggunakan 4249 gambar untuk data latih dan 1266 gambar untuk validasi. Fase kedua yaitu pembagian data, baik dataset dari fase pertama maupun dataset setelah melakukan *Automatic-labelling.*
Proses keempat yaitu melatih model terhadap dataset latih. Proses ini bisa dilakukan dengan dua metode, pertama menggunakan model *pre-trained* yang sudah tersedia dari *repository darknet*, kedua melatih model tanpa model *pre-trained.* Kedua metode tersebut akan menghasilkan model yang berbeda karakteristiknya yang nantinya akan dibandingkan performanya, dari perbandingan tersebut memutuskan untuk melatih dengan model pre-trained.



a)



b)

Gambar 9. Proses pelatihan model a) dengan model pre-trained b) tanpa model pretrained

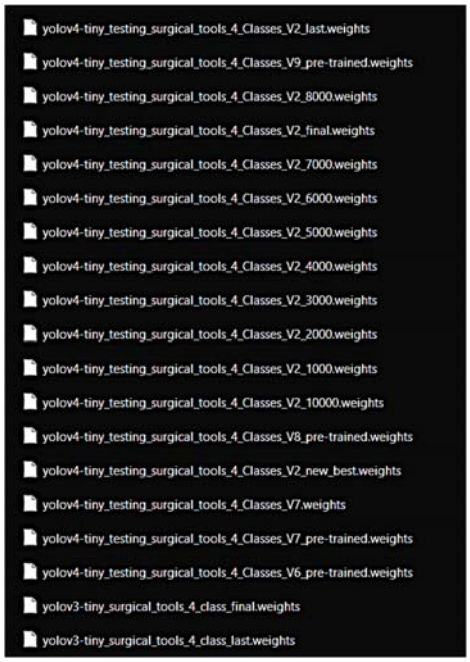
Framework darknet memiliki fungsi untuk menilai model setiap 1000 steps. Steps sendiri adalah satu iterasi pelatihan model terhadap dataset. Setiap iterasinya akan menghasilkan nilai loss. Hasil dari proses pelatihan model tersebut akan menghasilkan *chart* seperti pada Gambar 10. Pada Gambar 9 dapat dilihat *software* pemantauan penggunaan GPU yaitu MSI Afterburner. *Software* tersebut mengindikasikan adanya penggunaan GPU pada proses pelatihan model. Penggunaan GPU sebagai akselerator sudah tertanam fungsinya dalam *darknet*.

Gambar 10. *Chart* hasil dari pelatihan model terhadap dataset

Proses kelima yaitu pengujian dan penilaian performa model yang akan dibahas di subbab 2.2 dan Proses keenam yaitu implementasi dan integrasi model akan dibahas di bab berikutnya karena sudah sampai pada tahap pengujian.

2.2.2. Implementasi Model

Implementasi model dilakukan dengan menggunakan file weights pada program detektor seperti pada Gambar 11.

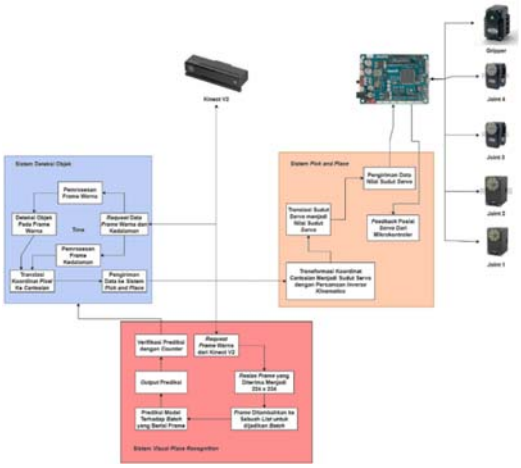


Gambar 11. Kumpulan Data Hasil Pelatihan Model

Program detektor yang digunakan untuk membaca file konfigurasi dan *weights* tertera pada lampiran B dengan judul "Kode Program Detektor pengujian". Kerja utama detektor adalah mendeteksi objek dan melacak posisi objek tersebut. Representasi secara matematis juga diperlukan untuk mendapatkan posisi objek secara akurat. Kedua hal tersebut dapat diselesaikan dengan *library* OpenCV. Langkah selanjutnya adalah mengambil nama label objek yang ingin dideteksi. Setelah membuka file label hal yang perlu dilakukan selanjutnya adalah mengambil *frame* dari kamera KinectV2. Untuk pengambilan *frame* dari kamera KinectV2 menggunakan *library* PyKinect2.

2.2.3. Diagram Blok Sistem Deteksi Objek

Diagram blok pada Gambar 12 menggambarkan keseluruhan kerja sistem baik itu sistem deteksi objek, sistem *visual place recognition*, maupun sistem *pick and place*



Gambar 12. Diagram Blok Sistem Deteksi Objek

. Untuk lebih menjelaskan sistem deteksi objek secara lebih detail maka sebelum membahas teknis deteksi langkah yang harus dilakukan adalah dengan melakukan *request* pada kamera KinectV2 untuk mengirimkan data *frame* warna dan kedalaman (RGBD). *Frame* yang diterima tidak bisa langsung diproses oleh OpenCV sebagai *frame* untuk dideteksi. Sebelum melakukan deteksi, *frame* yang diterima harus diubah format warnanya dari RGBA (Red Green Blue Alpha) ke warna RGB (Red Green Blue), membentuk ulang *frame* tersebut agar dimensinya sesuai dengan format yang didukung OpenCV yaitu bentuk *tensor*. *Frame* kedalaman yang diterima juga harus diproses ulang dengan diberikan representasi agar kedalaman bisa divisualisasikan dengan warna. Proses visualisasi *frame* kedalaman ini adalah *Color Mapping*. Proses ini merupakan proses pemetaaan nilai kedalaman menjadi nilai warna. Setelah *frame* selesai diproses maka langkah selanjutnya adalah melakukan deteksi pada *frame* tersebut. Hasil deteksi yang diperoleh dari *frame* akan mengeluarkan nilai koordinat *pixel*. Nilai koordinat *pixel* ini merupakan representasi posisi objek relatif terhadap resolusi kamera, dalam kasus penelitian ini ini kamera *KinectV2* memiliki resolusi sebesar 1920 x 1080. Untuk memperoleh koordinat kedalaman diperlukan sebuah *translator* untuk memetakan koordinat *pixel* menjadi koordinat kedalaman.

3. Pengujian dan Analisis

3.1. Integrasi Model dan Pengolahan data Sistem Deteksi

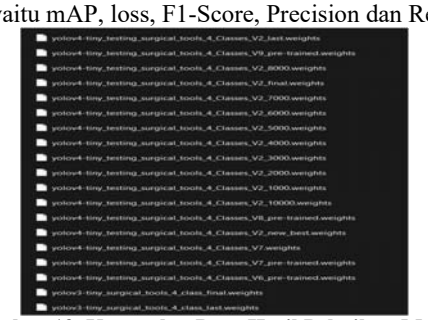
Integrasi model yang sudah selesai dilatih dilakukan dengan menggabungkan program detektor dengan sistem

inverse kinematics dan membuat sistem *pick and place* yang lengkap secara keseluruhan. Integrasi dilakukan dengan mengambil koordinat objek dengan poin x dan y relatif terhadap resolusi video. Setelah mendapatkan koordinat dari objek, koordinat tersebut diproses melalui fungsi `camera2depthpoint` pada *library* mapper yang sudah dibuat khusus untuk mentranslasikan koordinat kamera warna ke koordinat kamera kedalaman yang dikodekan dalam variabel `depth_x` dan `depth_y`. Koordinat yang sudah didapat dari translasi tersebut lalu masuk ke bagian dari kode yang akan mengekstrak kedalaman atau jarak dari koordinat kamera kedalaman.

3.1.1. Implementasi Model

Model sistem deteksi objek adalah salah satu inti dari sistem otonom robot yang harus bisa terintegrasi dengan baik dengan dua sistem yang lain yaitu, sistem pengenalan tempat secara visual dan sistem *pick and place* berbasis *inverse kinematics*. Agar model dapat diimplementasikan dan terintegrasi dengan baik diperlukan pengujian model yang baik.

Dalam pembuatan model, ada beberapa faktor yang harus diperhatikan seiring dengan proses pelatihan model. Variabel yang biasa digunakan untuk menilai sebuah model yaitu mAP, loss, F1-Score, Precision dan Recall.



Gambar 13. Kumpulan Data Hasil Pelatihan Model

3.2. Pengujian dan Penilaian Performa Model dalam Sistem Deteksi Objek

3.2.1. Pengujian Model dengan Data Validasi

Pengujian model dilakukan dengan dua metode pengujian. Metode pengujian pertama yaitu pengujian dari dataset validasi yang diuji langsung dengan *built-in script* di *framework darknet*. Pengujian model dapat dilihat pada Gambar 14.

Gambar 14. Pengujian Model dengan Data Validasi

Pengujian dengan *tools built-in* pada darknet menghasilkan nilai mAP sebesar 90,8%.

A. Pengujian dengan Kamera

Pengujian dengan kamera dilakukan dengan menggunakan program detektor dengan *wrapper* KinectV2 untuk OpenCV. Program detektor ditulis dengan bahasa pemrograman Python versi 3.6. Kebutuhan sistem untuk program detektor ini adalah sebagai berikut :

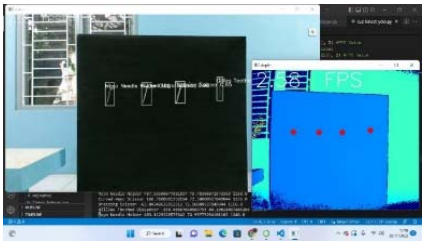
Program :Python 3.6.13, Kinect For Windows SDK V2.0_1409, Kinect For Windows Runtime V2.2_1905

Library Python : PyKinect2 V0.1.0, Numpy V1.19.5, OpenCV V4.5.5, OpenCV Contrib V4.5.5, Dynamixel SDK for Python V3.7.31, Mapper for Pykinect2 (https://github.com/KonstantinosAng/PyKinect2-Mapper-Functions)

B. Pengujian Dalam Ruangan

Pengujian model di dalam ruangan berbeda teknisnya dengan pengujian dengan data validasi. Pengujian di dalam ruangan dilakukan dengan menjalankan program detektor seperti Gambar 15 dan Gambar 16.

Gambar 15. Pengujian Deteksi Objek di Ruang Biro (EWS)



Gambar 16. Hasil Rekaman Layar Program Deteksi Objek

Pengambilan data dilakukan dengan melakukan rekaman luaran program selama 10 detik dengan *network size* 416 x 416 dan 640 x 640. Setiap besaran masukan diuji dengan objek yang diletakkan pada jarak 50 cm, 100 cm, 150 cm, dan 200 cm dari kamera. Hasil pengujian tersebut menghasilkan berapa banyak deteksi per objek di berbagai jarak selama 10 detik. Berikut adalah Tabel 4 darihasil pengujian tersebut.

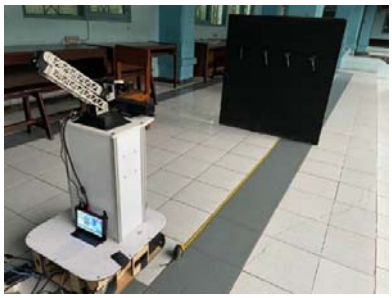
Tabel 4. Data Pengujian di Dalam Ruangan

Objek	Network Size, Jarak, dan Jumlah Deteksi							
	416 x 416				640 x 640			
	50cm	100 cm	150 cm	200 cm	50cm	100 cm	150 cm	200 cm
Curved Mayo Scissor	36	32	39	0	19	20	19	19
Gillies Toothed Dissector	37	32	0	0	19	20	19	19
Dressing Scissor	36	32	39	0	19	20	19	19
Mayo Needle Holder	36	32	39	0	18	20	19	19

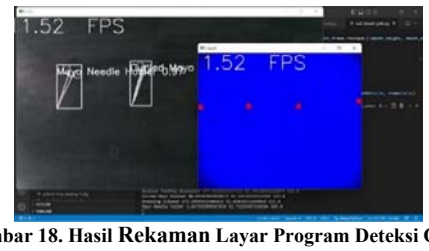
Berdasarkan hasil pengujian pada Tabel 4 model sudah bisa mendeteksi objek dengan baik, akan tetapi hanya sampai jarak tertentu saja. Jarak kerja optimal untuk *network size* 416 x 416 adalah antara 100 –150 cm dengan rata-rata ~3 deteksi/detik. Jarak kerja optimal untuk *network size* 640 x 640 adalah 50 cm - 200 cm dengan rata-rata ~1,9 deteksi/detik.

C. Pengujian Luar Ruangan

Pengujian model di luar ruangan berbeda teknisnya dengan pengujian dengan data validasi. Pengujian di luar ruangan dilakukan dengan menjalankan program detektor seperti Gambar 17 dan Gambar 18.



Gambar 17. Pengujian Deteksi Objek Di Koridor Gedung B Teknik Elektro



Gambar 18. Hasil Rekaman Layar Program Deteksi Objek

Pengambilan data dilakukan dengan melakukan rekaman luaran program selama 10 detik dengan *network size* 416 x 416 dan 640 x 640. Setiap besaran masukan diuji dengan objek yang diletakkan pada jarak 50 cm, 100 cm, 150 cm, dan 200 cm dari kamera. Hasil pengujian tersebut menghasilkan berapa banyak deteksi per objek di berbagai jarak selama 10 detik. Hasil pengujian tersebut dapat dilihat pada Tabel 5.

Tabel 5. Data Pengujian di Luar Ruangan

Objek	Network Size, Jarak, dan Jumlah Deteksi							
	416 x 416				640 x 640			
	50cm	100 cm	150 cm	200 cm	50cm	100 cm	150 cm	200 cm
Curved Mayo Scissor	30	29	34	0	18	19	19	19
Gillies Toothed Dissector	30	29	0	0	19	19	19	7
Dressing Scissor	30	29	34	2	18	19	19	19
Mayo Needle Holder	30	29	34	35	18	19	19	19

Berdasarkan hasil pengujian pada Tabel 5 model sudah bisa mendeteksi objek dengan baik, akan tetapi hanya sampai jarak tertentu saja.

D. Pengujian Akurasi Jarak Benda Terdeteksi dengan Jarak Asli

Selain pengujian terhadap model pengujian terhadap akurasi kamera juga harus dilakukan untuk memastikan kualitas data yang didapat dari deteksi objek. Pengukuran jarak didapat dari hasil triangulasi yang dilakukan dalam modul kamera Kinect V2. Pengujian akurasi jarak benda yang terdeteksi dilakukan dengan membandingkan data yang dikeluarkan oleh kamera dengan jarak yang diukur dengan pita meteran seperti yang bisa dilihat pada Gambar 19.

Gambar 19. Pengukuran Jarak Asli Antara Objek dan Kamera

Pengukuran dilakukan dengan menjalankan program dengan durasi 10 detik setiap iterasi jarak. Luaran dari program tersebut selama 5 iterasi pengukuran adalah sebagai berikut.

.

Tabel 6. Pengukuran Perbandingan Koordinat X Kamera dan Koordinat X Asli

Iterasi	Rata-Rata Koordinat X (mm)	Jarak Ukur Pita Meteran (cm)	Koordinat X Asli (cm)	Error (cm)
1	67,26	50	6	0,73
2	44,06	57	6	1,59
3	72,7	64	6	1,27
4	44,82	71	6	1,52
5	44,9	78	6	1,51
Mean Absolute Error				1.32

Tabel 7. Pengukuran Perbandingan Koordinat Y Kamera dan Koordinat Y Asli

Iterasi	Rata-Rata Koordinat Y (mm)	Jarak Ukur Pita Meteran (cm)	Koordinat Y Asli (cm)	Error (cm)
1	87,3	50	7	1,73
2	88,8	57	7	1,89
3	86,4	64	7	1,64
4	67,3	71	7	0,27
5	64,7	78	7	0,53
Mean Absolute Error				1,21

Tabel 8. Pengukuran Perbandingan Jarak Kamera dan Jarak Asli

Iterasi	Rata-Rata Jarak Kamera (cm)	Jarak Ukur Pita Meteran (cm)	Error (cm)
1	54,38	50	4,38
2	61,52	57	4,52
3	67,48	64	3,48
4	75,23	71	4,23
5	82,19	78	4,19
Mean Absolute Error			4,16

Dari luaran data program tersebut ada 4 data yang dibaca oleh program yaitu, nama objek, koordinat x, koordinat y, dan koordinat z (jarak). Dalam hal pengujian akurasi kali ini data yang digunakan adalah data koordinat x, y dan z. Koordinat x melambangkan jarak horizontal terhadap kamera, koordinat y melambangkan jarak vertikal terhadap kamera dan koordinat z melambangkan jarak objek terhadap kamera. Data koordinat x,y,z dari setiap iterasi akan di rata-rata dan diukur galatnya. Data perbandingan pengukuran dapat dilihat pada Tabel 6 sampai Tabel 8.

3.3. Keunggulan dan Batasan

Keunggulan dari algoritma yang dipakai adalah beban komputasinya yang ringan. Dengan kekuatan komputasi CPU yang dipakai yaitu AMD® Ryzen™ 3 3200U bisa menghasilkan nilai 3 deteksi/detik merupakan suatu keunggulan yang didapat dengan menggunakan model YOLOv4-Tiny.

3.3.1. Keunggulan Model

Keunggulan model YOLOv4-Tiny ini adalah beban komputasinya yang rendah. Keunggulan tersebut didapatkan dari arsitektur jaringan saraf buatan yang sederhana namun tetap mengedepankan akurasi dan kecepatan namun mengorbankan waktu pelatihan.

3.3.2. Batasan Performa

bla Batasan model YOLOv4-Tiny, walaupun dengan beban komputasinya yang ringan, adalah soal akurasi. Model yang digunakan hanya berhasil mendeteksi benda pada jarak dekat saja, sehingga model ini tidak cocok untuk digunakan untuk mendeteksi objek dari jarak yang jauh. Salah satu cara untuk mengatasi hal ini adalah untuk meningkatkan *network size* di atas 640 x 640.

Gambar 20. Percobaan Deteksi Objek pada Video

Seperti yang ditampilkan pada Gambar 20, akurasi yang hampir sempurna hanya bisa dicapai dengan meningkatkan *network size* menjadi 1024 x 1024.

4. Kesimpulan

Model untuk sistem deteksi objek sudah berhasil diimplementasikan dengan format file *weights* yang dijalankan pada *framework darknet* dan diuji dengan data validasi menghasilkan nilai mAP sebesar 90,90%, nilai *precision* sebesar 0,97, nilai *recal* sebesar 0,82, dan nilai F-1 *Score* sebesar 0,89.

Sistem deteksi objek sudah berhasil diimplementasikan dalam program gabungan sistem *visual place recognition, object detection,* dan *pick and place* menggunaan OpenCV sebagai *framework* pemrosesan model. Luaran dari program tersebut sudah dapat dijadikan sebagai *set point* sistem *pick and place* dengan MAE (*Mean Average Error*) koordinat X,Y,Z sebesar 1,32 cm, 1,21 cm, dan 4,16cm.

Data koordinat sudah dapat digunakan untuk sistem *pick and place* dengan kalibrasi luaran program sebesar 2,8 cm untuk koordinat X, 15,5 cm untuk jarak, dan 1 cm untuk

koordinat Y. Hasil kalibrasi ini berhasil menambah akurasi dan presisi dari gerakan lengan pada program *pick and place.*.

Referensi

[1] Z. Zou and Z. Shi, "Object Detection in 20 Years: A Survey," *arXiv,* no. 1905.05055v2, pp. 1-39, 2019.

[2] IBM, "IBM.com," IBM, [Online]. Available: https://www.ibm.com/id-en/topics/computer-vision. [Accessed 15 November 202].

[3] J. McCarthy and E. Feigenbaum, "In Memoriam: Arthur Samuel: Pioneer in Machine Learning," *AI Magazine,* vol. 11, no. 3, pp. 10-10, 1990.

[4] IBM Cloud Education, "IBM Cloud Learn Hub," IBM, 3 June 2020. [Online]. Available: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence. [Accessed 15 November 2022].

[5] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie and L. Farhan, "Review of deep learning: concepts, CNN," *Journal of Big Data,* vol. 8, no. 53, pp. 1-74, 2021.

[6] IBM Cloud Education, "IBM Cloud Learn Hub," IBM, 1 May 2020. [Online]. Available: https://www.ibm.com/id-en/cloud/learn/deep-learning. [Accessed 15 November 2022].

[7] S. N. Srivatsa, A. S. G, V. G and M. E. P, "Deteksi objek using Deep Learning with OpenCV and Python,," *International Research Journal of Engineering and Technology,* vol. 8, no. 1, pp. 227-230, 2021.

[8] J. Redmon, "Darknet: Open Source Neural Networks in C," Darknet, 2013-2016. [Online]. Available: http://pjreddie.com/darknet/. [Accessed 16 November 2022].

[9] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Deteksi objek," *arXiv,* pp. 779-788, 2015.

[10] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv,* pp. 1-7, 2018.

[11] D. S. V. Viraktamath, R. Byahatti and M. Yavagal, "Deteksi objek and Classification using," *International Journal of Engineering Research & Technology (IJERT),* vol. 10, no. 2, pp. 197-202, 2021.

[12] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Deteksi objek," *arXiv,* pp. 1-17, 2020.