



UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

MODI DI FALLIMENTO DELLE
TELECAMERE RGB ED EFFETTI NELLE
APPLICAZIONI DI GUIDA AUTONOMA

ON FAILURES OF RGB CAMERAS AND
THEIR EFFECTS IN AUTONOMOUS
DRIVING APPLICATIONS

FRANCESCO SECCI

Relatore: *Andrea Ceccarelli*

Anno Accademico 2019-2020

Francesco Secci: *Modi di Fallimento delle telecamere RGB ed effetti nelle applicazioni di guida autonoma*, Corso di Laurea in Informatica, © Anno Accademico 2019-2020

INDICE

1	Introduzione	11
2	Dependability dei Sistemi Critici	15
2.1	Attributi	17
2.2	Minacce	18
2.2.1	Guasto	19
2.2.2	Errore	19
2.2.3	Fallimento	19
2.3	Mezzi	20
2.3.1	Fault Prevention	20
2.3.2	Fault Tolerance	20
2.3.3	Fault Removal	21
2.3.4	Fault Forecasting	21
3	Elementi principali di Guida Autonoma	23
3.1	Livelli di Automazione della Guida - SAE	24
3.1.1	Livelli 0 - 2	25
3.1.1.1	Autopilot di Tesla	26
3.1.2	Livello 3	27
3.1.3	Livello 4	27
3.1.4	Livello 5	28
3.2	Visione Artificiale	29
3.2.1	LIDAR - Light Detection And Ranging	30
3.2.2	RADAR - RAdio Detection And Ranging	31
3.2.3	Videocamera	31
3.2.4	GPS - Global Positioning System	31
3.3	Reti Neurali	32
3.3.1	Classificatori	33
3.3.2	Object Recognition	33
4	FMECA di una Videocamera in ambito Automotive	35
4.1	Valutazione del Rischio	37
4.1.1	Metodo RPN	37
4.1.2	Analisi della Criticità	37
4.2	FMECA della Videocamera	39
4.2.1	Componenti della Videocamera	40
4.2.2	Modi di Fallimento	47
4.2.2.1	Obiettivo	48

4.2.2.2	Corpo Macchina	59
4.2.2.3	Filtro di Bayer	63
4.2.2.4	Sensore	64
4.2.2.5	ISP - Image Signal Processor	71
4.2.2.6	Sommario	77
5	Simulatore CARLA e Iniezione dei Modi di Fallimento	79
5.1	Simulatore CARLA	80
5.1.1	Motore - CARLA	81
5.1.2	Ambiente - CARLA	81
5.1.3	Sensori - CARLA	83
5.2	Campagna Sperimentale	84
5.2.1	CARLA e Learning By Cheating Benchmarking	84
5.2.1.1	CoRL2017	86
5.2.2	Modi di Fallimento Iniettati	87
5.3	Risultati	107
5.3.1	Success Rate e Numero di Collisioni	107
5.3.2	Andamento Run rispetto al Meteo	115
5.3.3	Tempi di Simulazione	119
5.3.4	Approfondimento sul Fallimento DeadPixel	124
5.3.5	Riepilogo	131
6	Stato dell'Arte	135
7	Conclusioni	141
Appendice		145
A	Codice Python per simulazione Fallimenti	147
B	Tabella FMECA Videocamera	169

ELENCO DELLE FIGURE

Figura 1	Fallimento e Ripristino di un Servizio	16
Figura 2	L'albero della Dependability e della Security	17
Figura 3	La Catena Guasto-Errore-Fallimento	18
Figura 4	I 6 Livelli di Automazione SAE [1]	25
Figura 5	Olli, la navetta a guida autonoma della Sacramento State University [2]	28
Figura 6	Le tre principali tecnologie utilizzate per la guida autonoma [3]	29
Figura 7	I Sensori presenti nelle auto a guida autonoma	30
Figura 8	Schema training e impiego della Rete Neurale su un veicolo	34
Figura 9	Matrice di Criticità	39
Figura 10	Schema dei componenti analizzati nella Videocamera	40
Figura 11	Filtro di Bayer	42
Figura 12	Immagine originale utilizzata per introduzione del fallimento [4]	43
Figura 13	Simulazione della non Demosaicizzazione dell'Immagine	43
Figura 14	Matrice dei pixel ricoperti di Filtri di Bayer [5]	44
Figura 15	Legenda dei componenti della Videocamera	46
Figura 16	Livelli di Severity e Detectability	47
Figura 17	Immagine esempio del fallimento BRIGHT LINES [6]	67
Figura 18	Immagine esempio del fallimento BANDING (scarsa qualità sensore)	69
Figura 19	Immagine esempio del fallimento BANDING (poca profondità colore, toni insufficienti) [7]	70
Figura 20	Immagine esempio del fallimento NO NOISE REDUCTION [8]	72
Figura 21	Immagine esempio del fallimento NO CHROMATIC ABERRATION CORRECTION [9]	75

4 Elenco delle figure

Figura 22	Alcune tipologie di meteo disponibili nel simulatore CARLA [10]	83
Figura 23	Tre tipologie di sensori forniti dal simulatore CARLA [10]	83
Figura 24	Fallimento Blurred nella simulazione	89
Figura 25	Configurazioni del fallimento White nella simulazione	90
Figura 26	Fallimento Black nella simulazione	91
Figura 27	Fallimento Broken Lens - configurazione 1 nella simulazione	91
Figura 28	Fallimento Broken Lens - configurazione 2 nella simulazione	92
Figura 29	Fallimento Dirty Int. - Dirty Ext. configurazione 1 nella simulazione	92
Figura 30	Fallimento Dirty Int. - Dirty Ext. configurazione 2 nella simulazione	93
Figura 31	Fallimento Rain nella simulazione	93
Figura 32	Fallimento Condensation nella simulazione	94
Figura 33	Fallimento Ice - configurazione 1 nella simulazione	95
Figura 34	Fallimento Ice - configurazione 2 nella simulazione	95
Figura 35	Fallimento No Bayer Filter nella simulazione	96
Figura 36	Fallimento Dead Pixel - configurazione 1 nella simulazione	96
Figura 37	Fallimento Dead Pixel - configurazione 2 nella simulazione	97
Figura 38	Fallimento Dead Pixel - configurazione 3 nella simulazione	97
Figura 39	Fallimento Dead Pixel - configurazione 4 nella simulazione	98
Figura 40	Fallimento Dead Pixel - configurazione 5 nella simulazione	98
Figura 41	Fallimento Dead Pixel - configurazione 6 nella simulazione	99
Figura 42	Fallimento Dead Pixel - configurazione 7 nella simulazione	99
Figura 43	Fallimento Dead Pixel - configurazione 8 nella simulazione	100

Figura 44	Fallimento Dead Pixel - configurazione 9 nella simulazione 100
Figura 45	Fallimento Dead Pixel - configurazione 10 nella simulazione 101
Figura 46	Fallimento Banding nella simulazione 101
Figura 47	Fallimento No Demosaicing nella simulazione 102
Figura 48	Fallimento No Demosaicing senza ridimensionamento del frame 102
Figura 49	Fallimento No Noise Reduction - configurazione 1 nella simulazione 103
Figura 50	Fallimento No Noise Reduction - configurazione 2 nella simulazione 103
Figura 51	Fallimento No Sharpness Correction nella simulazione 104
Figura 52	Fallimento No Chromatic Aberration Correction - configurazione 1 nella simulazione 104
Figura 53	Fallimento No Chromatic Aberration Correction - configurazione 2 nella simulazione 105
Figura 54	Success Rate per ogni Fallimento per ogni Scenario 108
Figura 55	Success Rate in ordine crescente per ogni Fallimento per FullTown02 110
Figura 56	Grafico Radar per Nr. Collisioni per ogni Fallimento per ogni Scenario 112
Figura 57	Numero di Collisioni per ogni Fallimento in FullTown02 113
Figura 58	Grafico Numero di Collisioni per ogni Fallimento per ogni Meteo 116
Figura 59	Relazione fra Tempo di Simulazione CARLA e Success Rate 120
Figura 60	Relazione fra Numero di Collisioni e Tempo di simulazione CARLA 121
Figura 61	Tempi di Simulazione CARLA e in ore reali del Fallimento No Chromatic Aberration Correction 123
Figura 62	Tempi di Simulazione del Fallimento DeadPixel nelle sue configurazioni - FullTown02 125
Figura 63	Tempi di Simulazione in relazione con i Success Rate - FullTown02 127
Figura 64	Nr. Collisioni per ogni Meteo - Scenari uniti 129
Figura 65	Esempio Fotocamera Gated [11] 136
Figura 66	FMECA della Videocamera 170

ELENCO DELLE TABELLE

Tabella 1	Tabella riassuntiva con breve descrizione dei Fallimenti analizzati	78
Tabella 2	Tabella riassuntiva per i dettagli sull'implementazione dei Fallimenti analizzati	106
Tabella 3	Media del numero di Collisioni per Meteo	117
Tabella 4	Percentuali Collisioni per Meteo GoldenRun	117
Tabella 5	Percentuali minime e massime di Collisioni per Meteo	118
Tabella 6	Tempi Medi run suddivisi per Meteo	130
Tabella 7	Tempi Medi run suddivisi per Meteo dopo la revisione	131
Tabella 8	Tempi Medi pre e post revisione	131

"I don't get the little ship thing. You can't show up at Mars in something the size of a rowboat. What if there are Martians? It would be so embarrassing."

— Elon Musk

1

INTRODUZIONE

La Guida Autonoma ha avuto una crescente attenzione negli ultimi anni con un aumento della domanda, nonché degli investimenti [12]. L'obiettivo di un sistema di guida autonomo è guidare da solo senza alcun intervento umano: il veicolo rileva l'ambiente circostante, individua la sua posizione e opera per raggiungere in sicurezza la destinazione specificata.

Le tecnologie fondamentali in questo campo riguardano i sensori, gli algoritmi di fusione dei dati e di inferenza ed in particolare le applicazioni di intelligenza artificiale e di machine learning. Questi sono coinvolti nella maggior parte dei compiti essenziali svolti dal veicolo come: la rappresentazione dell'ambiente, la comprensione dello scenario, la segmentazione semantica, la fusione dei dati, la localizzazione, il rilevamento e il riconoscimento degli oggetti [13]. Tra i sensori nell'ambito guida autonoma la telecamera RGB è la più utilizzata e insostituibile [13]. Nonostante le telecamere presentino degli svantaggi come la forte sensibilità all'illuminazione esterna e il campo visivo limitato, i sistemi di riconoscimento visivo sono tra le applicazioni più solide ed affidabili in questo ambito [14]. Le telecamere dei veicoli sono già sfruttate in molte applicazioni come il riconoscimento dei segnali stradali, il rilevamento delle corsie, il rilevamento degli ostacoli, ecc. [13], [12], [15]. Ulteriori studi vengono effettuati riguardo alle precedenze: ad esempio, agli incroci la conoscenza della posizione di pedoni e ciclisti, nonché degli altri veicoli, può consentire all'auto di prendere sofisticate decisioni di precedenza [15].

Quando le immagini catturate dalla videocamera risultano degradate possono verificarsi incidenti mortali. Gli agenti addestrati delle applicazioni di IA/ML responsabili dell'elaborazione degli input possono fare affidamento su dati errati e di conseguenza portare a decisioni non sicure, talvolta fatali.

Molti lavori hanno esplorato come rendere gli agenti addestrati robusti alle immagini di input artificialmente fabbricate o manipolate [16], [17], [18] ed altri come proteggere una telecamera da attacchi diretti che potrebbero compromettere il corretto funzionamento della stessa [19], [20]. Tuttavia, pochi o nessun lavoro si è concentrato sull'attenta identificazione di una serie realistica e completa di alterazioni accidentali di immagini da una videocamera RGB guasta. Le opere esistenti considerano un insieme credibile di alterazioni dell'immagine, ma senza un'analisi sistematica di quali sono i possibili malfunzionamenti di una telecamera, senza un modello di guasti condiviso e senza librerie software per la riproduzione di tali fallimenti.

I fallimenti delle telecamere e i loro effetti sul sistema globale dovrebbero essere sistematicamente studiati per comprendere appieno le conseguenti minacce alla sicurezza e le possibili mitigazioni dovrebbero essere identificate con attenzione. Tutto questo andrebbe a vantaggio degli ingegneri del sistema e degli ingegneri del software, sia per quanto riguarda l'architettura sia per la valutazione della robustezza delle applicazioni di IA/ML basate su immagini utilizzate.

Quanto fatto in questo lavoro di tesi è definito come segue:

- creiamo e discutiamo un modello di guasti per telecamere di veicoli nel dominio della guida autonoma, analizzando i diversi fallimenti, le loro cause e i loro effetti sul sistema,
- esaminiamo le mitigazioni esistenti a livello di componente,
- riproduciamo gli effetti dei guasti sull'immagine via software,
- confermiamo gli effetti e discutiamo dei rischi associati alla sicurezza utilizzando come riferimento un'applicazione per la guida autonoma.

Otteniamo i suddetti dati rispettivamente attraverso:

- la definizione di una FMECA (Failure Mode, Effects and Criticality Analysis, [21]) sui componenti di una telecamera RGB, supponendo che la telecamera sia collocata su di un veicolo (insieme ad un'attenta revisione della letteratura),

- lo sviluppo di una libreria di fallimenti in codice Python, resa pubblicamente disponibile in [22], per alterare le immagini secondo il modello dei guasti,
- l'iniezione di fallimenti nella telecamera frontale di un veicolo attraverso l'uso di un simulatore di guida autonoma (CARLA [10]), dove è in esecuzione l'agente addestrato da [23].

L'ultimo punto consente di discutere il comportamento dell'agente addestrato, in termini di success rate, numero di collisioni, distanza dall'obiettivo, ecc. Inoltre, viene dimostrato come anche i fallimenti della telecamera che alterano leggermente l'immagine possono ingannare l'agente addestrato e in qualche modo dipendere dalle condizioni meteorologiche.

Il lavoro di tesi è organizzato come segue:

- inizialmente è stata data la definizione di Dependability dei Sistemi Critici e sono stati trattati gli elementi che la compongono (Capitolo 2),
- sono stati presi in considerazione gli Elementi principali di Guida Autonoma elencandoli e descrivendone il funzionamento (Capitolo 3),
- è stato descritto cosa si intende per FMECA ed in particolare quali risultati ha prodotto questa tipologia di analisi applicata su una telecamera in ambito guida autonoma (Capitolo 4),
- è stato descritto il simulatore di guida autonoma utilizzato (CARLA) e ed il modo in cui sono stati creati e iniettati i diversi fallimenti evidenziati dall'analisi condotta sul dispositivo telecamera (Capitolo 5),
- è stata data una descrizione dello Stato dell'Arte in questo settore e di come nessun lavoro di ricerca considera un modello completo di guasti di una telecamera per veicoli, compresa un'analisi degli effetti e dei rischi per la sicurezza, nonché la creazione di una libreria software per la replicazione dei fallimenti (Capitolo 6),
- infine, le Conclusioni, nelle quali si evidenzia come l'identificazione di un modello di guasti attraverso l'applicazione di una FMECA e la conseguente replicazione dei fallimenti attraverso una libreria

in codice Python, produca un vantaggio nel comprendere quali siano i fallimenti più rilevanti per il sistema, dal punto di vista della criticità (Capitolo 7).

2

DEPENDABILITY DEI SISTEMI CRITICI

In questo capitolo verrà trattato l'argomento della Dependability.

Oggigiorno, i sistemi che ci circondano sono in gran parte considerati sistemi critici. Con sistema critico si intende un generico sistema che, in caso di fallimento nel suo funzionamento, può provocare:

- morte o gravi rischi per le persone,
- perdita o grave danneggiamento di mezzi e materiali,
- gravi danni ambientali [24].

Per il funzionamento corretto di questi sistemi è di fondamentale importanza la capacità nell'analizzare gli aspetti quantitativi e qualitativi [25] che li riguardano: questi possono essere relativi alla performance e all'efficienza nonché alla sicurezza, alla disponibilità e all'affidabilità, che garantiscono che il sistema operi nel modo più sicuro possibile.

Prima di definire la Dependability, vanno introdotti i seguenti concetti: Servizio, Utente e Funzione. Con Servizio si intende il comportamento del sistema stesso. L'Utente, invece, è un altro sistema che interagisce col precedente, attraverso un'interfaccia. La Funzione è quello che ci si aspetta dal sistema. Il servizio è corretto se realizza la funzione del sistema, altrimenti è definito non corretto (Fig. 1).

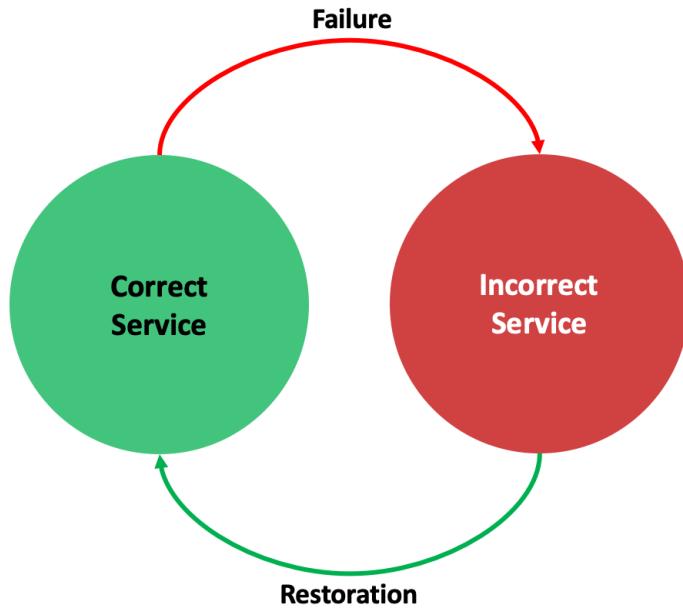


Figura 1.: Fallimento e Ripristino di un Servizio

Definizione 1 (Dependability) *La Dependability è la capacità di un sistema di fornire un servizio su cui è possibile fare affidamento in modo giustificato [25].*

La Dependability può essere suddivisa in 3 elementi: Attributi, Minacce e Mezzi (Fig. 2). Gli Attributi si possono considerare quando si vuole valutare l'affidabilità di un sistema. Con Minacce si intendono tutte quelle cose che possono influire negativamente sull'affidabilità del sistema. I Mezzi sono considerati come le azioni e/o i modi per aumentare l'affidabilità del sistema.

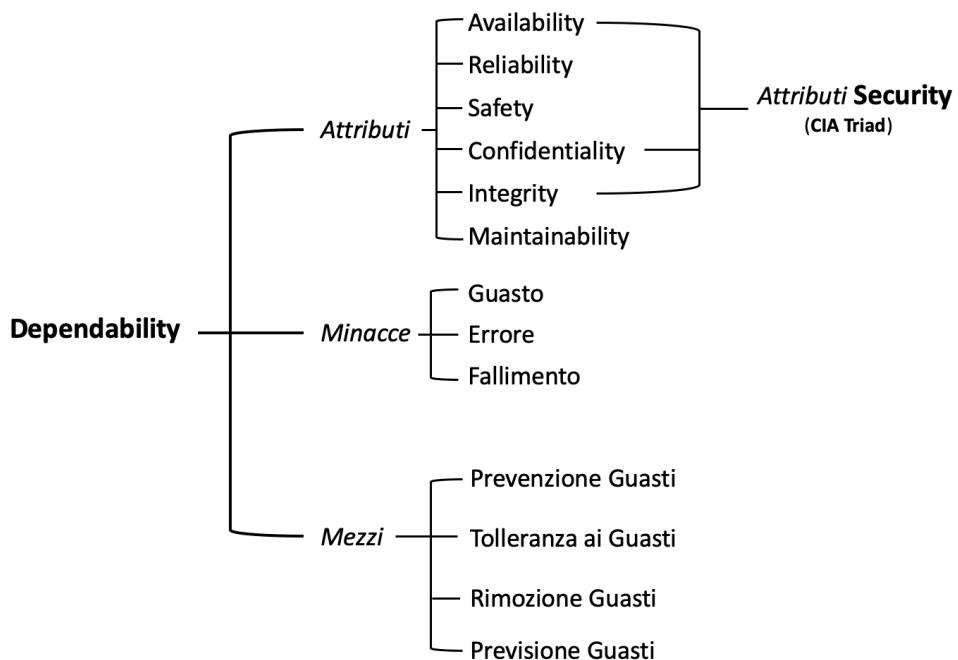


Figura 2.: L'albero della Dependability e della Security

2.1 ATTRIBUTI

Dependability è un concetto di integrazione che comprende i seguenti attributi [26]:

- **Availability** → prontezza del sistema nell'erogare un servizio corretto (si può definire come $\frac{MTTF}{MTTF+MTTR}$, dove MTTF è il tempo medio al fallimento, mentre MTTR è il tempo medio per riparare il sistema dopo il suo fallimento),
- **Reliability** → capacità di un sistema di fornire un servizio corretto in modo continuo (può essere rappresentata dal MTTF, ovvero il tempo medio al fallimento, e dal MTBF, ossia il tempo medio fra fallimenti),
- **Safety** → assenza di conseguenze catastrofiche per utente/i e ambiente,
- **Confidentiality** → assenza di diffusione non autorizzata di informazioni,
- **Integrity** → assenza di alterazioni improprie del sistema,

- **Maintainability** → capacità del sistema di subire modifiche e riparazioni (può essere rappresentata dal **MTTR**, cioè il tempo medio al ripristino).

In Fig. 2 è presente anche la voce **Security**. Questa è composta dagli attributi di Confidentiality, Integrity e Availability (**CIA Triad**), quindi può essere vista come la contemporanea esistenza di Availability solo per utenti autorizzati, Confidentiality e Integrity (dove per "improprie alterazioni" si intende "non autorizzate") [26].

I sistemi dovrebbero proteggere se stessi e i loro dati da interferenze esterne. La Security, oggi, è essenziale, in quanto la maggior parte dei sistemi è collegata in rete in modo tale da consentire l'accesso tramite Internet.

Le specifiche di Dependability e Security di un sistema devono includere i requisiti per gli attributi in termini di frequenza accettabile e gravità dei fallimenti del servizio, per specifiche classi di guasti e per un certo ambiente di utilizzo. Uno o più attributi potrebbero non essere richiesti per un determinato sistema [26].

2.2 MINACCE

Le Minacce sono elementi che possono influire su un sistema e causare un calo della Dependability. Come osservabile in Fig. 3 le minacce sono rappresentate dai Guasti, che possono trasformarsi in Errori, che possono diventare a loro volta Fallimenti, una volta arrivati all'interfaccia di servizio.



Figura 3.: La Catena Guasto-Errore-Fallimento

2.2.1 *Guasto*

Si definisce guasto la causa accertata o ipotizzata di un errore, derivante da malfunzionamenti di componenti, interferenze ambientali di natura fisica, sbagli dell'operatore o da una progettazione fallace [25].

Un guasto può rimanere dormiente per un certo periodo, fino alla sua attivazione. Questa porta ad un errore, che è la parte dello stato del sistema che può causare un successivo fallimento [27].

2.2.2 *Errore*

Un errore è la parte dello stato del sistema che può causare un susseguente fallimento [25]. Come detto, l'errore deriva dall'attivazione di un guasto. Un errore può agire nello stesso modo di un guasto, creando ulteriori condizioni di errore, propagandosi all'interno dei confini del sistema (system boundaries), senza causare mai un fallimento osservabile. Quando questo si propaga al di fuori dei confini del sistema, allora si verifica un fallimento. Infatti, un fallimento è fondamentalmente il momento in cui si può affermare che un servizio non riesce più a soddisfare le sue specifiche. Dato che l'output di un servizio può essere considerato l'input per un altro, il fallimento può propagarsi in un altro servizio come guasto. In questo modo la catena Guasto-Errore-Fallimento avrà di nuovo inizio.

2.2.3 *Fallimento*

Un fallimento del sistema è un evento che occorre quando un errore raggiunge l'interfaccia di servizio, alterando il servizio stesso. Quando un sistema viola la sua specifica di servizio si dice che è avvenuto un fallimento. Questo può quindi essere inteso come transizione da un servizio corretto ad un servizio non corretto, mentre la transizione inversa può essere definita ripristino [25], [26](Fig. 1).

I modi di fallimento caratterizzano un servizio non corretto da quattro punti di vista:

- il dominio dei fallimenti,
- la possibilità di rilevare i fallimenti,

- la consistenza dei fallimenti,
- le conseguenze dei fallimenti [25].

2.3 MEZZI

Nel corso degli ultimi cinquant'anni sono stati sviluppati molti mezzi per raggiungere e mantenere gli attributi di Dependability. Tali mezzi possono essere raggruppati in quattro categorie principali:

- Fault Prevention,
- Fault Tolerance,
- Fault Removal,
- Fault Forecasting.

Le tecniche di Fault Prevention e di Fault Tolerance garantiscono il *conseguimento* della dependability: assicurare al sistema la capacità di fornire un servizio sempre fedele alle specifiche.

Le tecniche di Fault Removal e di Fault Forecasting rappresentano, invece, la *validazione* della dependability: essere ragionevolmente confidenti nella capacità del sistema di fornire un servizio secondo le specifiche.

2.3.1 *Fault Prevention*

In questi ricadono i mezzi per prevenire il verificarsi o l'introduzione di guasti all'interno del sistema. Ciò può essere realizzato mediante l'uso di: programmazione strutturata e modulare, linguaggi fortemente tipati, appropriata formazione del personale, materiali specifici (guasti di origine fisica), firewall ed altri dispositivi (guasti da attacchi esterni), ecc. [25],[27].

2.3.2 *Fault Tolerance*

La Fault Tolerance mira a preservare l'erogazione di un servizio corretto in presenza di **guasti attivi** nel sistema. Viene solitamente implementata mediante rilevazione degli errori e conseguente recupero dello stato del sistema [25]. Esistono due tecniche di rilevazione degli errori: **concurrent**

error detection e **preemptive error detection**. La prima viene effettuata durante l'erogazione del servizio, mentre l'altra viene effettuata quando l'erogazione è sospesa e controlla la presenza di errori latenti e guasti dormienti.

Il recupero del sistema, invece, consiste in **error handling** e **fault handling**. Il primo elimina gli errori dallo stato del sistema e il secondo impedisce che i guasti localizzati vengano nuovamente attivati [25].

2.3.3 *Fault Removal*

Questa si occupa di ridurre il numero e la gravità dei guasti. Questi mezzi vengono usati sia in fase di sviluppo che durante l'uso del sistema [27]. La rimozione durante lo sviluppo richiede una verifica in modo che i guasti possano essere rilevati e rimossi prima che un sistema venga messo in produzione.

Una volta che i sistemi sono sul mercato, è necessario un modo per registrare i guasti e rimuoverli tramite un ciclo di manutenzione.

La rimozione dei guasti è uno degli obiettivi del processo di **Verifica e Validazione (V&V)** [25].

2.3.4 *Fault Forecasting*

La Fault Forecasting è condotta effettuando una valutazione del comportamento del sistema rispetto all'occorrenza e all'attivazione dei guasti.

La valutazione ha due aspetti:

- **qualitativa** o ordinale, che mira a identificare, classificare e ordinare le modalità di errore o le combinazioni di eventi (guasti dei componenti o condizioni ambientali) che porterebbero a guasti del sistema,
- **quantitativa** o probabilistica, che mira a valutare in termini di probabilità la misura in cui alcuni degli attributi sono soddisfatti [26].

I metodi per la valutazione qualitativa e quantitativa sono specifici: ad esempio la **FMEA** (Failure Modes and Effects Analysis) per una *valutazione* [26].

*zione qualitativa; le Catene di Markov (**MC**) e le Reti di Petri Stocastiche (**SPN**) per una valutazione quantitativa.*

Inoltre, ci sono metodi che possono essere utilizzati per eseguire entrambe le forme di valutazione: Reliability Block Diagram (**RBD**), Fault-Tree Analysis (**FTA**) [26].

3

ELEMENTI PRINCIPALI DI GUIDA AUTONOMA

Nell'ultimo decennio, il settore dell'Automotive è stato in continua evoluzione. In particolar modo, le grandi aziende automobilistiche e molti gruppi di ricerca si sono mossi verso l'automazione della guida ed il miglioramento delle infrastrutture già esistenti. I motivi di questo interessamento del settore verso la guida autonoma sono vari:

- riduzione del numero di incidenti, anche fino al 90%,
- riduzione dei consumi,
- riduzione delle emissioni di CO₂,
- nonché la riduzione di tutte le varie problematiche della sicurezza stradale.

Inoltre, soprattutto riguardo all'inquinamento dei veicoli, negli ultimi anni, hanno sempre preso più piede i motori ibridi e quelli totalmente elettrici.

Oggi non si può ancora parlare di guida autonoma, bensì di guida assistita. Infatti, i problemi e le difficoltà che il mondo della guida autonoma porta con sé sono molti ed eterogenei fra loro. Il dilemma che sorge quando si parla di guida autonoma è legato per lo più alla capacità decisionale del veicolo: uno studio del MIT [28] ha cercato di sondare il pensiero pubblico con un quiz online, denominato *The Moral Machine*, nel quale si chiedeva agli utenti di prendere una serie di decisioni etiche riguardo ad incidenti automobilistici simulati, simili al cosiddetto *Problema del Carrello Ferroviario* [29]. Da questo studio si è evidenziato come ci siano delle preferenze globali coerenti, ad esempio:

- risparmiare gli esseri umani piuttosto che gli animali,

- risparmiare quante più persone possibile,
- e, infine, come venga favorita la giovane età rispetto a quella più adulta.

Per i veicoli autonomi o quasi autonomi, nel 2014, SAE International, un ente di standardizzazione automobilistica, ha pubblicato un nuovo standard internazionale, J3016 (“*Levels of Driving Automation*”).

Nei prossimi paragrafi verranno trattati i seguenti argomenti:

- i Livelli di Automazione della Guida - SAE (Paragrafo 3.1),
- la Visione Artificiale, da intendersi come l’insieme dei sensori e dispositivi che compongono il sistema (Paragrafo 3.2),
- le Reti Neurali, ovvero modelli computazionali con la funzione di elaborazione dati e decisione dell’azione corretta da eseguire (Paragrafo 3.3).

3.1 LIVELLI DI AUTOMAZIONE DELLA GUIDA - SAE

In questo sistema di classificazione, all’aumentare del livello di automazione, la responsabilità del conducente passa dalla guida alle attività di supervisione. I livelli proposti in questo standard vanno dal Livello 0, nel quale non si ha nessuna automazione, al Livello 5, nel quale si considera un veicolo che può guidare in modo completamente autonomo ovunque ed in ogni condizione, come mostrato in Fig. 4.



Figura 4.: I 6 Livelli di Automazione SAE [1]

3.1.1 *Livelli 0 - 2*

Un veicolo con automazione di livello 0 è un veicolo privo di funzioni di assistenza alla guida. Molti veicoli moderni hanno una singola funzione di assistenza al conducente, ad esempio il controllo automatico della velocità, quindi verrebbero definiti come veicoli di livello 1.

Una combinazione di due o più sistemi avanzati di assistenza alla guida (ADAS - Advanced Driver-Assistance Systems), in cui questi aiutano a controllare la frenata, l'accelerazione e/o lo sterzo, è definita di livello 2. Esistono molti nomi e acronimi diversi utilizzati dalle diverse case automobilistiche per descrivere gli ADAS, come:

- **Adaptive Cruise Control** → il guidatore può lasciare il pedale dell'acceleratore e il mezzo viaggerà ad una velocità costante percepindo il veicolo che precede e rallentando per mantenere la distanza impostata,

- **Lane-Keeping Assist** —> controlla lo sterzo del veicolo per impedire al conducente di uscire involontariamente dalla propria corsia autostradale,
- **Automatic Emergency Braking** —> il veicolo rileva un incidente imminente e applica i freni per prevenire o limitare la gravità della collisione.

A questi livelli, il conducente deve mantenere il controllo del veicolo in ogni momento.

Oggi, uno dei più avanzati sistemi di assistenza alla guida è sicuramente rappresentato dall'**Autopilot di Tesla**.

3.1.1.1 *Autopilot di Tesla*

L'Autopilot di Tesla è un'ADAS presente in ogni modello della casa americana. Questo offre diverse funzionalità, fra cui: il mantenimento di corsia, il controllo adattivo della velocità, il parcheggio automatico e la possibilità, soltanto azionando una freccia direzionale, di far cambiare in modo autonomo corsia autostradale alla vettura [30].

Uno degli obiettivi del marchio americano, forse il più importante, annunciato da Elon Musk, è quello di arrivare, in un futuro non troppo lontano, ad avere una vera e propria guida autonoma in tutto e per tutto, con la quale saremo soltanto passeggeri.

Con il primo Autopilot, offerto per la prima volta su una Tesla Model S, nell'Ottobre del 2014, le capacità autonome dell'auto erano molto limitate, ad esempio a funzionalità che permettevano il parcheggio del veicolo senza interazione dell'utente. Ultimamente, quello che il veicolo riesce a fare è di gran lunga superiore alla prima versione rilasciata anni fa. Infatti, negli ultimi anni, dal 2016, la casa americana ha cominciato a parlare della cosiddetta Full Self-Driving (FSD). Questo pacchetto è già disponibile sulle auto attualmente in vendita ed è considerato un'optional selezionabile in fase di ordinazione. Rispetto al precedente Autopilot, questo si presenta come un qualcosa di molto più avanzato, non fermandosi alla possibilità di sterzare, accelerare e frenare automaticamente, ma andando oltre:

- gestione automatica dello svincolo autostradale, inclusi incroci e sorpassi di auto più lente;
- cambi di corsia automatici durante la guida in autostrada;
- parcheggio parallelo e perpendicolare (**Autopark**);
- parcheggio e recupero automatico del veicolo (**Summon**);
- riconoscimento e risposta ai semafori e ai segnali di stop;
- guida automatica sulle strade cittadine;
- possibilità di far uscire l'auto dal parcheggio per venire a prenderti (**Summon Avanzato**).

Naturalmente, questi ultimi sviluppi si vanno a collocare in un Livello SAE diverso, in quanto l'auto riesce a gestire situazioni anche fuori dai tratti autostradali in modo completamente autonomo. Anche se queste capacità potrebbero rientrare nel Livello 4, per motivi soprattutto legali, si può dire che queste sono limitate al Livello 3 (**Assistenza condizionata**, Fig. 4).

3.1.2 *Livello 3*

Da questo livello si può cominciare a parlare di guida semi-autonoma. Infatti, il veicolo è in grado di gestire diverse situazioni, anche al di fuori del contesto autostradale. Dato che l'auto si prenderà la responsabilità soltanto per alcuni momenti di un viaggio, ad un certo punto, il sistema avviserà il conducente che deve riprendere il controllo del mezzo. Questo avviso, se va a buon fine, porterà il conducente di nuovo a controllare il veicolo. Qualora gli avvisi non fossero uditi, percepiti dal conducente, molto probabilmente il sistema provvederà ad arrestare il veicolo nel modo più sicuro possibile.

Al momento non esistono in commercio veicoli che hanno questo livello di guida autonoma.

3.1.3 *Livello 4*

Con questo livello si entra in quella che può essere definita guida autonoma. I veicoli dotati di questa tecnologia sono in grado di procedere

per molti chilometri e superare una varietà di ostacoli, ad esempio caselli autostradali.

Tuttavia, il funzionamento di questi veicoli sarà in qualche modo limitato:

- ad una determinata area geografica,
- da condizioni meteo avverse,
- da una velocità massima e così via.

Un esempio di questi veicoli si può ritrovare nelle navette recentemente impiegate dalla Sacramento State University nel proprio campus (Fig. 5).



Figura 5.: Olli, la navetta a guida autonoma della Sacramento State University
[2]

3.1.4 *Livello 5*

Il funzionamento senza pilota e senza restrizioni definisce i veicoli di livello 5. In questi veicoli il conducente molto probabilmente non esisterà più. Infatti, saranno capaci di procedere in ogni condizione e luogo senza che un umano prenda parte all'azione di guida. Questo livello è molto

utilizzato in produzioni cinematografiche, soprattutto fantascientifiche, in quanto forse mai riusciremo ad avere una tecnologia simile.

Altro aspetto da considerare è quello che molti esperti del settore credono che il livello 4, veicoli ad alta capacità, sia veramente l'obiettivo da raggiungere e che potrà trasformare le città e di conseguenza il mondo [31].

3.2 VISIONE ARTIFICIALE

Uno dei principali problemi da affrontare con le auto a guida autonoma è la loro percezione dell'ambiente. Infatti, non sono dotati di un occhio o più occhi umani, per prendere le loro decisioni, bensì di occhi artificiali. In questi veicoli si parla di visione artificiale avanzata e le principali tecnologie (Fig. 6) utilizzate in questo ambito comprendono [31]:

- Videocamere,
- Radar,
- Lidar,
- GPS.



Figura 6.: Le tre principali tecnologie utilizzate per la guida autonoma [3]

Tutte insieme, queste tecnologie, mirano a creare un modello del mondo reale (3D) partendo da immagini bidimensionali (2D), nel modo più fedele possibile alla realtà. Lo scopo della visione artificiale, infatti, è proprio quello di riprodurre la vista umana [32]. In questo ambito, "vedere" non è inteso come la sola acquisizione del fotogramma, ma significa interpretarne il suo contenuto, in modo da poter prendere decisioni autonomamente.

Un'idea di come queste tecnologie vengono posizionate su un veicolo si ha in Fig. 7.

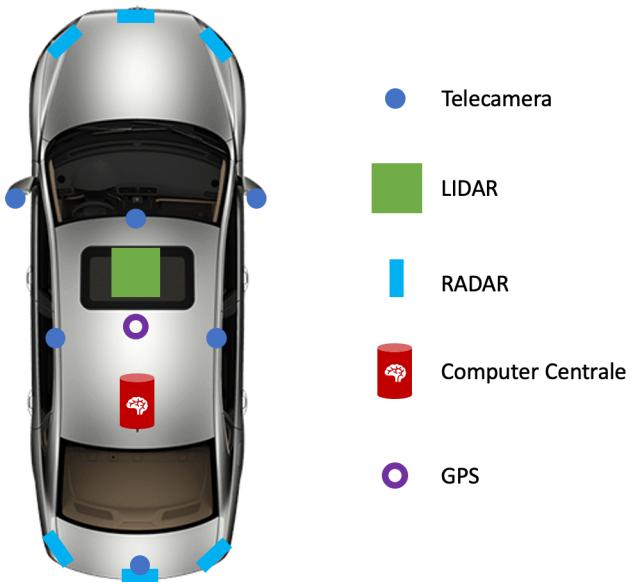


Figura 7.: I Sensori presenti nelle auto a guida autonoma

Il componente analizzato in questo elaborato è la Telecamera. Sui veicoli di oggi sono presenti più dispositivi di questo genere ed il loro obiettivo è catturare i fotogrammi e renderli disponibili come input per software impiegati nella scelta del comportamento da tenere.

3.2.1 LIDAR - Light Detection And Ranging

È una tecnologia di telerilevamento. I suoi usi sono molto vari soprattutto perché il suo obiettivo principale è quello di raccogliere informazioni 3D e usare la luce sotto forma di un laser pulsato per misurare diverse distanze. Il suo ruolo è quello di raccogliere informazioni cinematiche sul veicolo e informazioni fisiche sull'ambiente circostante. Il sensore ottico LIDAR è installato sul tettuccio del veicolo autonomo. È composto da un laser, un filtro per obiettivo, un ricevitore, un regolatore di potenza, uno specchio rotante e un processore integrato [31].

Diversamente dal Radar, che al posto della luce utilizza onde radio,

la distanza dell'oggetto è determinata misurando il tempo trascorso fra l'emissione dell'impulso e la ricezione del segnale retrodiffuso. L'unità di elaborazione deve essere collocata in una posizione abbastanza rialzata da terra; inoltre, sono necessarie misure di sicurezza per proteggere l'unità da urti o vibrazioni derivanti da incidenti o dalla navigazione su terreni accidentati, che potrebbero causare guasti al sistema.

3.2.2 RADAR - RAdio Detection And Ranging

È un sistema che utilizza onde radio per il rilevamento e la determinazione della posizione ed eventualmente della velocità di oggetti fissi o mobili, come aerei, navi, veicoli, formazioni atmosferiche o il suolo. Le onde radio vengono trasmesse nell'ambiente per raccogliere informazioni sugli ostacoli intorno al veicolo e aumentare la consapevolezza del posizionamento degli altri soggetti davanti e dietro. Questo sensore tiene d'occhio le altre auto e indica all'auto autonoma di accelerare o rallentare a seconda del comportamento degli altri conducenti [31].

È utile anche per posteggiare il veicolo ed in questo caso si sente parlare di sensori anteriori e posteriori.

3.2.3 Videocamera

Le telecamere sono necessarie nel sistema di trasporto intelligente affinché si riconoscano gli ostacoli rispetto alla posizione e alla velocità del veicolo in considerazione. Le immagini bidimensionali derivanti da una singola fotocamera o le mappe 3D risultanti dall'utilizzo della doppia fotocamera, potrebbero individuare stereoscopicamente lo spazio disponibile per i movimenti autonomi del veicolo. Queste immagini o mappe vengono utilizzate per estrarre informazioni quantitative dalle scene e per tracciare gli obiettivi del veicolo. Le immagini sono segmentate in un certo numero di pixel. Ogni pixel viene elaborato e memorizzato, il che richiede un'elevata velocità di calcolo ed un elevato spazio di archiviazione [31].

3.2.4 GPS - Global Positioning System

Il Sistema di Posizionamento Globale, attraverso una rete dedicata di satelliti artificiali in orbita, fornisce a un terminale mobile o ad un ricevitore

GPS informazioni sulle sue coordinate geografiche e sul suo orario in quasi tutte le condizioni atmosferiche ed ovunque non vi siano ostacoli che potrebbero impedirne l'invio e la ricezione dei segnali (edifici molto alti, gallerie, ecc.). A causa dei disturbi del segnale e altre interferenze dell'atmosfera, la posizione stimata utilizzando un GPS potrebbe essere sbagliata di diversi metri. La localizzazione avviene tramite la trasmissione di un segnale radio da parte di ciascun satellite e l'elaborazione dei segnali ricevuti da parte del ricevitore [31].

Per far procedere autonomamente il veicolo, il GPS, con l'aiuto di altri sensori, crea mappe precise della carreggiata e guida nella direzione esatta. Questo rilevamento potrebbe anche permettere di vedere in tempo reale altri veicoli sulla stessa strada e mostrare la posizione attuale.

3.3 RETI NEURALI

Le Reti Neurali sono oggi ampiamente accettate come soluzioni dominanti per la visione artificiale, per il riconoscimento vocale e per l'elaborazione del linguaggio naturale. La cosa su cui ci siamo concentrati in questo elaborato è proprio la visione artificiale che un veicolo a guida autonoma deve avere per poter garantire la sicurezza di passeggeri e ambiente esterno. Il problema principale di questi sistemi è il riconoscimento di quello che hanno intorno e la conseguente decisione che prendono per far muovere il veicolo. Infatti, come accaduto, ad esempio durante l'utilizzo del cosiddetto Autopilot di Tesla, il veicolo può non riconoscere una certa situazione di pericolo od un ostacolo e quindi minare la sicurezza, non solo degli individui nell'abitacolo, ma anche di quelli presenti all'esterno, che essi siano altri pedoni o altri mezzi nelle vicinanze.

Per quanto riguarda le prestazioni di queste Reti Neurali, sono stati registrati punteggi di accuratezza spesso corrispondenti a quelli di un individuo umano, su benchmark chiave. Ben diversi, invece, sono stati i punteggi osservati quando si considerano i casi peggiori. Con casi peggiori si vuole intendere quei casi definiti *Adversarial Examples* [33] o input perturbati, anche solo leggermente, che dal punto di vista umano risultano sempre comprensibili, mentre da quello della macchina, cambiano totalmente di significato. Naturalmente, se queste Reti vengono utilizzate su veicoli che devono muoversi in un ambiente nel quale si trova un'eterogeneità di soggetti e comportamenti, si parla di contesti

nei quali affidabilità, sicurezza e tutti gli altri attributi che definiscono la **dependability** di un sistema sono al primo posto in ordine di importanza.

3.3.1 *Classificatori*

Molto importante, per la guida autonoma, è il processo di classificazione: esso si può definire come individuazione, dato un insieme di oggetti (nel nostro caso immagini o fotogrammi), di particolari caratteristiche che permettono al sistema di prendere una decisione sulla base dei dati estrapolati. Nel nostro contesto si parlerà di classificatori basati sulle Reti Neurali. Su di essi sono state osservate diverse vulnerabilità prendendo in considerazione immagini di input **trasformate spazialmente** [33]: con questo termine si intende rappresentare tutte le possibili trasformazioni che un'immagine può subire, ad esempio ruotandola, tagliandola, scalandola ecc. Queste sono trasformazioni naturali e sono utilizzate in vari contesti:

- verificare il livello di completezza di un classificatore,
- allenare una rete neurale,
- ricreare scenari o situazioni con cui il veicolo su cui è installata una determinata rete può avere a che fare.

Con il termine "trasformata spazialmente" non ci si deve immaginare una modifica profonda del fotogramma in questione, ma anzi, basta anche solo una piccola rotazione, di meno di un grado, per far sì che il classificatore confonda, ad esempio, un revolver con una trappola per topi [33], il che, pensando ad un utilizzo anche diverso da quello che se ne fa su un veicolo a guida autonoma, può far sorgere molte preoccupazioni e domande (si pensi all'ambito aeroportuale).

3.3.2 *Object Recognition*

In generale, ci si riconduce al problema dell'**Object Recognition** o riconoscimento artificiale degli oggetti. Questo richiede che ci sia una nozione di somiglianza visiva, difficile da far apprendere ad un sistema: significherebbe catturare e insegnare ad una macchina la nozione di percezione umana [33].

Tuttavia, oggi, le **DNN** (Deep Neural Network) hanno raggiunto prestazioni all'avanguardia, talvolta competitive con la percezione umana.

Una delle sfide principali nella guida autonoma è il cambiamento ambientale che un qualsiasi sistema autonomo può incontrare, ovvero: le differenti distanze e angolazioni con cui viene percepito l'ambiente circostante dai sistemi di visione artificiale variano ogni volta che il veicolo si muove nello spazio [34].

Purtroppo, nel contesto della guida autonoma, anche un piccolo errore di valutazione nella navigazione potrebbe causare incidenti altamente devastanti.

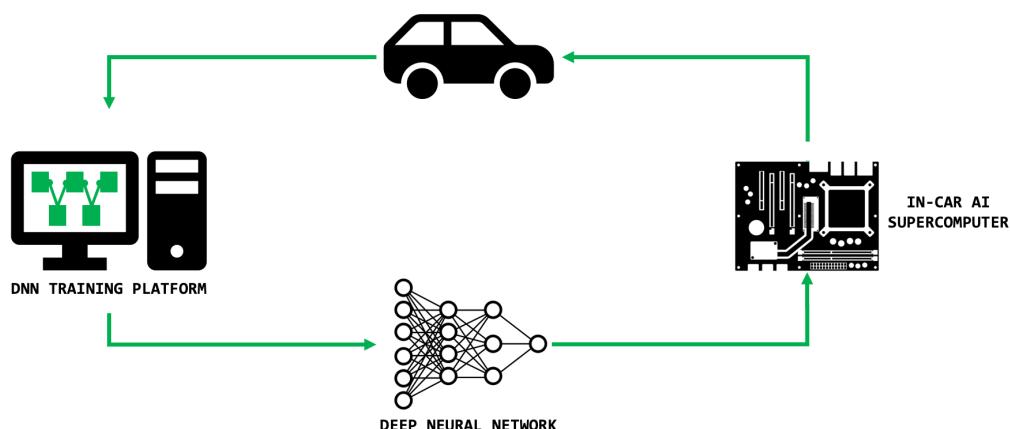


Figura 8.: Schema training e impiego della Rete Neurale su un veicolo

Diversi gruppi di ricerca hanno sviluppato un modello preliminare di analisi del rischio per veicoli autonomi. In tali studi sono stati considerati dei sottosistemi di veicoli autonomi e/o alcuni componenti dell'infrastruttura di trasporto; tuttavia, condizioni meteorologiche, altri utenti della strada (guidatori non autonomi, ciclisti, pedoni, ecc.) e condizioni della superficie stradale non sono state incluse [31].

4

FMECA DI UNA VIDEOCAMERA IN AMBITO AUTOMOTIVE

La **FMECA** (Failure Mode, Effects, and Criticality Analysis - Analisi dei modi, degli effetti e della criticità dei guasti) è un'estensione della FMEA (Failure Mode and Effects Analysis), in aggiunta alla quale include un'analisi di criticità usata per valutare, mediante opportuni diagrammi, la gravità delle conseguenze di un guasto correlata con la probabilità del suo verificarsi [21].

FMEA e FMECA sono metodologie progettate per identificare potenziali modalità di fallimento per un prodotto o processo, per valutare il rischio associato a tali modalità, per classificare i problemi in termini di importanza e identificare e attuare azioni correttive per affrontare per primi quelli più gravi [35].

Questa tipologia di analisi è stata scelta in base a quello che il lavoro di Tesi richiedeva di evidenziare. Infatti, insieme ad un'attenta revisione dello stato dell'arte, la FMECA ha permesso di estrapolare quali fossero i fallimenti che possono colpire una telecamera utilizzata in ambito guida autonoma e, in quasi la totalità dei casi, evidenziarne le possibili mitigazioni atte a correggere o prevenire un dato fallimento.

La procedura di analisi di base per FMEA o FMECA comprende diversi passaggi:

- formare un team,
- stabilire le regole di base,
- raccogliere e rivedere le informazioni pertinenti,
- identificare gli articoli o i processi da analizzare,

- identificare le funzioni, i guasti, gli effetti, le cause e i controlli per ciascun articolo o processo da analizzare,
- valutare il rischio associato ai problemi identificati dall'analisi,
- dare priorità e assegnare azioni correttive,
- eseguire azioni correttive e rivalutare il rischio [36], [37].

Il presupposto sottostante all'applicazione delle tecniche FMEA e FMECA è il principio secondo il quale il rischio, definito come una combinazione tra la probabilità di occorrenza di un fallimento e la severità di quest'ultimo, sia correlato non soltanto alla probabilità che si verifichi il fallimento, ma anche alla gravità delle sue conseguenze e alla possibilità di intercettarlo prima che accada (**rilevabilità**).

Ci sono diverse domande base che vengono utilizzate nella fase di analisi, fra cui:

- Qual è il sistema a cui tale analisi viene applicata?
- Come ogni componente può fallire?
- Quali meccanismi potrebbero produrre queste modalità di fallimento?
- Quali potrebbero essere gli effetti se si verificassero i fallimenti?
- Il fallimento occorso è nella direzione sicura o non sicura?
- Come viene rilevato l'errore?
- Quali mitigazioni sono previste nel progetto per contrastare l'errore?

Gli argomenti trattati nel Capitolo sono i seguenti:

- la Valutazione del Rischio (Paragrafo 4.1),
- la FMECA della Videocamera (Paragrafo 4.2).

4.1 VALUTAZIONE DEL RISCHIO

Una tipica FMECA incorpora alcuni metodi per valutare il rischio associato ai potenziali problemi identificati attraverso l'analisi. I due metodi più comuni sono:

- Metodo RPN (Risk Priority Number),
- Analisi della Criticità (Criticality Analysis).

4.1.1 *Metodo RPN*

Per valutare il rischio con tale metodo, il team di analisi deve:

- valutare la **gravità** (G) di ogni effetto del fallimento,
- valutare la **probabilità di occorrenza** (O) per ogni causa di fallimento,
- valutare la **probabilità di rilevare** (R) il problema prima che raggiunga l'utente o il cliente finale,
- calcolare l'RPN come moltiplicazione dei tre valori sopracitati:

$$RPN = G \cdot O \cdot R.$$

L'RPN può quindi essere utilizzato per confrontare problemi all'interno dell'analisi e dare la priorità a quelli che necessitano di azioni correttive di più rispetto ad altri [36].

4.1.2 *Analisi della Criticità*

Esistono due tipi di analisi della criticità: quantitativa e qualitativa.

Per utilizzare il metodo di analisi della criticità **quantitativa**, il team deve:

- (A) definire l'affidabilità per ciascun componente, in un determinato tempo operativo,
- (B) identificare la porzione di componenti inaffidabili che possono essere causa di ognuno dei potenziali fallimenti (sottoforma di rapporto),

- (C) valutare la probabilità di perdita (o gravità) che deriverà da ciascuna modalità di fallimento individuata,
- calcolare la criticità per ogni potenziale modalità di fallimento con il prodotto dei tre fattori (Mode Criticality):

$$\text{Criticità della Modalità} = A \cdot B \cdot C,$$

- calcolare la criticità per ciascun componente del sistema, attraverso la somma delle criticità per ogni modalità di fallimento identificata per il componente stesso:

$$\text{Component Criticality} = \text{SOMMA delle Criticità delle Modalità}.$$

Per utilizzare il metodo di analisi della criticità **qualitativa**, valutando il rischio e dando priorità alle azioni correttive, il team di analisi deve considerare:

- la gravità dei potenziali effetti del fallimento,
- la probabilità di occorrenza per ogni potenziale modalità di fallimento,
- di confrontare le modalità di fallimento tramite una **Matrice di Criticità**, che ne identifica la gravità, decrescente da sinistra verso destra, e la probabilità che occorra, decrescente dall'alto verso il basso (Fig. 9). In questa, ogni voce rappresenta il rischio che si ha per un determinato fallimento.

		Gravità			
		Catastrofica	Critica	Marginale	Minore
Probabilità di Occorrenza	Frequente	Alto	Alto	Serio	Medio
	Probabile	Alto	Alto	Serio	Medio
	Occasionale	Alto	Serio	Medio	Basso
	Remoto	Serio	Medio	Medio	Basso
	Improbabile	Medio	Medio	Medio	Basso

Figura 9.: Matrice di Criticità

FMEA e FMECA sono strumenti utilizzati per molti scopi diversi. Possono contribuire a migliorare i progetti di prodotti e processi, con conseguente maggiore affidabilità, qualità, sicurezza e soddisfazione del cliente, diminuendo anche i costi. Possono essere utilizzati per stabilire e ottimizzare i piani di manutenzione per i sistemi riparabili e/o contribuire ai piani di controllo e ad altre procedure di garanzia della qualità. Forniscono una base di conoscenza delle modalità di fallimento e informazioni sulle azioni correttive che possono essere utilizzate come risorsa per future attività di risoluzione dei problemi e come strumento di formazione.

4.2 FMECA DELLA VIDEOCAMERA

Come precedentemente spiegato, la FMECA è un tipo di analisi volto ad individuare possibili criticità in un processo od in un sistema, in modo da poterle eliminare o renderle innocue. Nel nostro caso, l'analisi è stata svolta su una Videocamera che può essere utilizzata in ambito Self-Driving. Il suo utilizzo in questo settore fa sì che essa debba essere ad un livello di affidabilità molto alto, in quanto è uno degli occhi, oltre agli altri sensori, che il sistema auto utilizza per muoversi in modo autonomo in un ambiente sconosciuto e mutevole nel tempo. Infatti, se si pensa a quanti fattori possono cambiare in un tragitto di pochi km, ci si rende conto che l'affidabilità di sensori di ogni genere è cruciale per la sicurezza del passeggero nonché degli altri utenti della strada.

Nell'analisi condotta si è pensato ad ogni fattore, interno ed esterno

al sistema, che potesse influenzare il funzionamento di alcuni o tutti i componenti facenti parte del dispositivo.

4.2.1 Componenti della Videocamera

Nel sistema Videocamera sono stati individuati 5 componenti (Fig. 10): l'Obiettivo, il Corpo Macchina, il Filtro di Bayer, il Sensore di immagini e l'ISP (Image Signal Processor).



Figura 10.: Schema dei componenti analizzati nella Videocamera

Di seguito verrà spiegato ognuno dei termini precedentemente elencati.

OBIETTIVO Gli obiettivi fotografici sono dispositivi ottici in grado di raccogliere e riprodurre un'immagine. L'obiettivo è il componente che ha maggior impatto sulla qualità delle fotografie. L'obiettivo fotografico può essere composto da una o più lenti e/o da catadiottri come sistemi di specchi concavi e convessi, spesso abbinati anche essi a diottri [38].

Il fattore fondamentale che distingue un obiettivo da un altro è in primis la **lunghezza focale**, che permette di suddividerli in macro categorie. Un secondo fattore che caratterizza gli obiettivo è la **luminosità**, inoltre possono essere o non essere **stabilizzati**. Un terzo fattore di distinzione tra gli obiettivi sono quelli definiti macro. L'ultimo fattore che può essere preso in considerazione è la **messa a fuoco**: questa può essere sia manuale che automatica [38]. Nell'obiettivo risiede anche un minimo di elettronica, necessaria per il motore della messa a fuoco (quando automatica) e per lo zoom.

CORPO MACCHINA Il Corpo Macchina è il contenitore di tutta l'elettronica della macchina fotografica. In questo, come mostrato nella figura precedente, si può vedere come risiedono gli altri 3 componenti principali individuati nell'analisi: Filtro di Bayer, Sensore di immagine e ISP.

Nelle normali fotocamere il corpo macchina ha la funzione di far impongere al meglio il dispositivo, nonché di proteggere tutto quello che non deve essere esposto ad ogni tipo di contatto con l'esterno. Dunque, l'involucro, protegge il sensore dalla luce e da altri fattori di rischio.

FILTRO DI BAYER Il Filtro di Bayer (o Bayer pattern) è uno schema per la disposizione degli elementi sensibili ai diversi colori al di sopra di sensori usati per l'acquisizione di immagini digitali. Infatti, i fotodiodi impiegati in un sensore di immagine sono daltonici per natura: possono solo registrare sfumature di grigio. Per ottenere il colore nell'immagine, sono coperti con diversi filtri di colore: rosso, verde e blu (RGB) secondo il modello designato dal Filtro di Bayer. Questo prende il nome da Bryce Bayer, ricercatore della Kodak che lo propose per primo [39].

La sua caratteristica è quella di raggruppare i sensori per i tre colori fondamentali necessari per la sintesi additiva (RGB, rosso, verde e blu) in celle di due fotositi per due (Fig. 11). Ogni cella contiene due elementi verdi, uno rosso e uno blu [39].

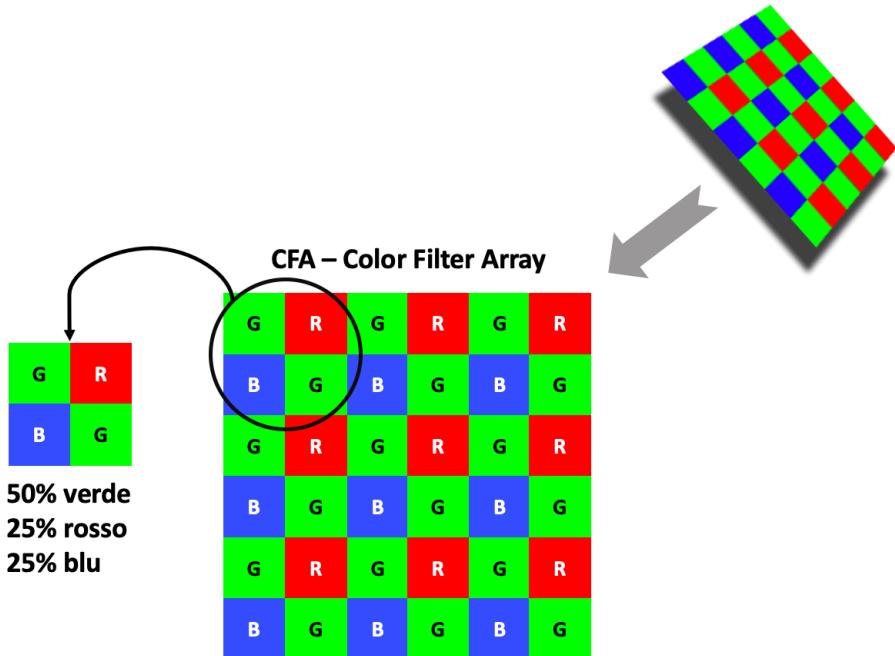


Figura 11.: Filtro di Bayer

L'output non elaborato con Filtro di Bayer viene indicato come Immagine di Bayer. Poiché ciascun pixel viene filtrato per registrare solo uno dei tre colori, i dati di ciascun pixel non possono specificare completamente ciascuno dei valori rosso, verde e blu. Per ottenere un'immagine a colori, è possibile utilizzare vari algoritmi di demosaicing o demosaicizzazione per interpolare un insieme di valori rossi, verdi e blu completi per ciascun pixel. Questi algoritmi utilizzano i pixel circostanti per stimare i valori per un particolare pixel. Un'Immagine di Bayer si presenta come mostrato in Fig. 13.



Figura 12.: Immagine originale utilizzata per introduzione del fallimento [4]

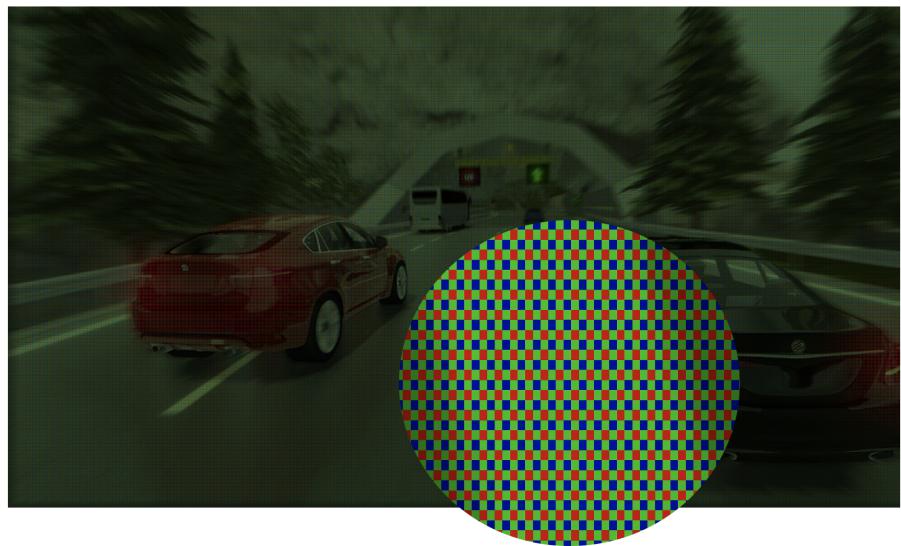


Figura 13.: Simulazione della non Demosaicizzazione dell'Immagine

In Fig. 13 si può osservare come effettivamente il pattern di Bayer risulti sia nel dettaglio che nell'immagine estesa, mostrando una prevalenza di verde. La simulazione della non Demosaicizzazione è stata possibile attraverso il codice in A.2.

SENSORE DI IMMAGINE Per Sensore d'Immagine si intende il trasduttore che converte l'immagine ottica in una sua rappresentazione o codifica elettrica, si tratta essenzialmente di un chip di silicio in grado di catturare e misurare la luce, ovvero la quantità di fotoni che lo raggiungono. Il sensore ha generalmente una forma rettangolare e dimensioni tipicamente contenute, variabili da costruttore a costruttore e da modello a modello.

La superficie del sensore è formata da milioni di minuscoli ricevitori disposti secondo una griglia regolare. Questi ricevitori chiamati anche **photosites** (o fotosito) sono, in effetti, i microsensori che effettuano la conversione da fotoni in elettroni (Fig. 14). Ogni singolo ricevitore sarà in grado di fornire in uscita una carica elettrica proporzionale alla quantità di fotoni che lo hanno colpito. La carica rilevata verrà poi convertita da un apposito circuito di conversione analogico-digitale in un valore numerico. Ognuno dei valori ottenuti dai photosites costituirà un pixel dell'immagine ottenuta.

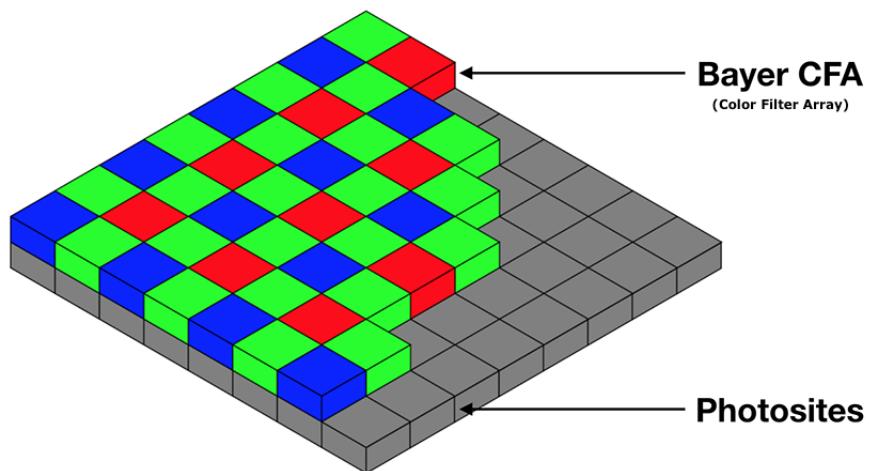


Figura 14.: Matrice dei pixel ricoperti di Filtri di Bayer [5]

Attualmente ci sono due tipi di sensori d'immagine in commercio: **CCD** (Charge Coupled Device) e **CMOS** (Complementary Metal Oxide Semiconductor). Negli ultimi anni l'avvento dei sensori CMOS ha rivoluzionato notevolmente il mercato a favore di questi ultimi soppiantando i sensori CCD in molte applicazioni. Entrambi si basano sul concetto di

convertire la carica di ogni photosites in un formato digitale mediante un **ADC** (Analog to Digital Converter), ma differiscono per come le informazioni vengono processate. Infatti, per i sensori CCD l'informazione sulla carica viene prelevata dai photosites riga per riga e memorizzata in un registro il cui contenuto viene passato ad un amplificatore e successivamente all'ADC. Dopo che la riga è stata interamente processata viene eliminata dal registro d'uscita e viene caricata la riga successiva che subisce lo stesso trattamento. Invece, nei sensori CMOS grazie alla tecnologia che permette di ridurre costi e dimensioni, insieme al photosites è stata integrata anche una serie di transistor che realizzano una amplificazione e conversione della carica in tensione. Attraverso una struttura matriciale è possibile selezionare singolarmente ogni photosites, attraverso l'indirizzo di riga e colonna, dopodiché inviare la tensione ad un ADC che esegue la conversione finale [40].

ISP Per ISP (Image Signal Processor) si intende un tipo di media processor o processore di segnale digitale (DSP) specializzato, utilizzato per l'elaborazione delle immagini, in macchine fotografiche digitali o altri dispositivi [41]. Le sue funzioni sono molteplici e fondamentali per il risultato finale dopo una prima acquisizione del fotogramma, come:

- Bayer filtering,
- Demosaicing,
- Correzione del Sensore d'Immagine,
- Riduzione del rumore,
- Correzione Nitidezza dell'immagine,
- Ridimensionamento dell'immagine,
- Correzione distorsione lente,
- Correzione dell'Aberrazione Cromatica,
- Compressione immagine e codifica JPEG,
- Compressione Video,
- Elaborazione/Compressione/Codifica audio e altro ancora [41].

Nell'analisi effettuata è stata creata una tabella (Appendice B), nella quale vengono riassunte tutte le informazioni utili dal punto di vista della criticità di ogni componente individuato. Ad ognuno di questi componenti è stato assegnato un colore, come mostrato nella legenda in Fig. 15. Questo permette una maggiore leggibilità della stessa tabella nonché una migliore individuabilità del componente.



Figura 15.: Legenda dei componenti della Videocamera

La tabella è composta dalle seguenti colonne:

- **ID** → numero identificativo del fallimento,
- **Component** → il componente preso in esame fra quelli elencati sopra,
- **Macro-Failure Mode** → la denominazione del fallimento analizzato,
- **Failure Mode** → la descrizione specifica del fallimento analizzato,
- **EFFECTS OF FAILURE** → questa si divide in **Local** e **Global** (la prima fa riferimento soltanto agli effetti locali sulla Videocamera, mentre la seconda anche su quelli globali, ad esempio pensando alle varie fasi post-acquisizione che avranno a che fare con il risultato di un fallimento),
- **Existing mitigations** → l'elenco e una breve spiegazione di tutte le possibili mitigazioni, se esistono, che possono far fronte al fallimento osservato,
- **Detectability/easiness to implement** → il punteggio assegnato al fallimento in base a quanti e quali mitigazioni sono state trovate ed alla facilità di implementazione di esse (Fig. 16),

- **Severity** → direttamente collegata alla Detectability, in quanto, anche se la mitigazione esiste, potrebbe essere talmente difficile da utilizzare o onerosa dal punto di vista computazionale e economico da non poter essere utilizzata (Fig. 16).

Per quanto riguarda la Detectability si possono assegnare 3 livelli: Low per indicare la massima facilità nell'identificare una mitigazione ed implementarla, fino ad arrivare a High con cui si indica la difficoltà massima nel trovare mitigazioni e di conseguenza implementarle.

La Severity, invece, con uno o più dei 4 valori assegnabili, è usata per indicare quanto sia importante introdurre tempestivamente la mitigazione trovata, se esistente, per far sì che il sistema non fallisca.



Figura 16.: Livelli di Severity e Detectability

In realtà, anche se non mostrata in Appendice B, è presente anche un'altra colonna (**Notes**), nella quale sono state inserite delle note su quanto scritto nelle varie colonne, per ogni fallimento. Queste saranno prese in considerazione in 4.2.2, per discutere su tutti i *Modi di Fallimento* trovati nell'analisi condotta. Il lavoro di analisi svolto può essere scaricato da [42].

4.2.2 *Modi di Fallimento*

In questo paragrafo verranno descritti tutti i modi di fallimento che sono stati presi in considerazione durante l'analisi effettuata. Per semplicità e migliore comprensione verranno suddivisi in base al componente specifico soggetto al fallimento preso in questione. L'ordine con il quale i componenti verranno trattati sarà lo stesso utilizzato nella tabella FMECA (Appendice B): Obiettivo, Corpo Macchina, Filtro di Bayer, Sensore

e ISP. Alcuni dei fallimenti presentati sono stati considerati per più di un componente, dunque, scorrendo la seguente lista, è possibile che dei fallimenti compaiano come verificabili per più di uno degli elementi indicati sopra.

4.2.2.1 *Obiettivo*

Il primo componente preso in esame è l'Obiettivo che viene contraddistinto dal colore  all'interno della tabella FMECA. Di seguito i fallimenti trovati per tale componente.

BLURRED

In questo si prende in considerazione il fatto che l'immagine non sia a fuoco. Questo fallimento del dispositivo può essere molto grave e può essere causato da diversi fattori. È di fondamentale importanza, soprattutto in quest'ambito, che le immagini catturate siano di buona qualità e quindi a fuoco. L'elaboratore centrale deve prendere delle decisioni, riguardo la marcia del veicolo, in base al contenuto informativo delle immagini che gli vengono presentate. Dunque, tanto più affidabile sarà un componente e tanto più questo potrà mantenere al sicuro gli utenti all'interno ed all'esterno dell'abitacolo.

Per quanto riguarda le mitigazioni trovate durante l'analisi, sono stati trovati due articoli ([43] e [44]) nei quali si tenta di eliminare il più possibile l'effetto di sfocatura da immagini di input corrotte. I risultati che vengono mostrati sono promettenti e sembrano poter risolvere diverse situazioni nelle quali si può presentare questa necessità. Naturalmente, in casi più estremi, come terreni molto dissestati, le vibrazioni potrebbero dover richiedere diverse accortezze aggiuntive.

Non si è voluto assegnare un valore di Severity basso ma una via di mezzo, in quanto dall'elaborazione deve risultare un'immagine comprensibile all'interprete, pena la corretta scelta dell'azione da eseguire. Naturalmente queste mitigazioni non possono essere svolte dall'Obiettivo, come componente, ma dal sistema videocamera che si propone di elaborare l'immagine dopo l'acquisizione.

BLACK - WHITE

Questi tipi di fallimento rappresentano il limite massimo che si può raggiungere con la rottura di componenti fondamentali (considerando una granularità meno grossolana) dell'obiettivo, quali ad esempio l'otturatore o il diaframma. In questo caso, infatti, se l'otturatore presenta un malfunzionamento che non permette l'entrata di una giusta quantità di luce attraverso l'obiettivo della videocamera, i fotogrammi risultanti potrebbero essere completamente neri oppure completamente bianchi se ne entrasse troppa. Mentre l'otturatore controlla per quanto tempo il sensore resta esposto alla luce, il diaframma ha la funzione di aprirsi e chiudersi in base a quanta luce vuole far entrare. Naturalmente i due microcomponenti del componente Obiettivo sono entrambi essenziali e devono funzionare in modo perfetto perché l'immagine catturata sia di una qualità accettabile per l'elaboratore centrale.

Purtroppo con fallimenti di questo genere le mitigazioni non esistono, in quanto è impossibile recuperare informazioni necessarie all'interprete per decidere come far muovere il veicolo nell'ambiente circostante. A livello di componente, quindi, non può essere trovata una mitigazione, ma si deve ragionare a livello di sistema. Una soluzione potrebbe essere quella di progettare un controllore di fotogrammi che, dopo l'acquisizione, decide se l'immagine contiene informazioni utili oppure no. Altra soluzione potrebbe essere quella di adottare il metodo di ridondanza (più videocamere) per aumentare l'affidabilità del sistema stesso.

Per quanto riguarda Detectability e Severity, si è preferito dare una sorta di range, in quanto è poco probabile che tutte le telecamere (ce ne saranno alcune, non soltanto una sul veicolo) si guastino nel medesimo modo contemporaneamente.

BROKEN VR

Questo fallimento riguarda una funzionalità presente su ormai ogni macchina fotografica o videocamera in commercio, ovvero lo stabilizzatore di immagine. Come esposto per il primo fallimento (BLURRED), l'immagine che viene acquisita deve presentare un certa qualità: un'immagine chiara, che non abbia sfocature di alcun genere e che permetta di distinguere l'ambiente circostante e gli attori esterni, in movimento e non, in ogni situazione. Anche in questo caso ci possiamo ricondurre agli articoli (che non sono gli unici sull'argomento) riportati per il fallimento

BLURRED, ovvero [43] e [44], in quanto fondamentalmente si tenta di riparare allo stesso errore. Anche qui, la mitigazione non può essere trovata a livello di componente, dato che l'Obiettivo non può svolgere tutti i controlli che possono essere fatti dopo l'acquisizione del fotogramma da controllori interni che provvedono al rilevamento e correzione (quando possibile) di fallimenti di questo tipo. Inoltre, seppur di metodi per effettuare le elaborazioni di Deblurring dell'immagine ce ne siano in abbondanza, i meccanismi utilizzati per correggere questi fallimenti devono essere applicabili in un tempo computazionale adeguato. Infatti, mentre il computer centrale decide sul da farsi, l'auto risulta in marcia e in attesa di informazioni fondamentali, dunque la Severity espressa nella tabella FMECA non può essere che critica.

BROKEN LENS

Qui si intende la vera e propria rottura di una o più lenti dell'Obiettivo. L'immagine che verrà acquisita presenterà una o più linee, più o meno visibili, che andranno a modificare la qualità e la correttezza delle informazioni contenute nel fotogramma. In questo caso, quindi, se l'immagine non avrà altri problemi nelle successive fasi, l'elaboratore centrale, incaricato di prendere decisioni sulla base di quello che "vede" in ogni fotogramma, si troverà ad estrarre delle informazioni non corrette e conseguentemente a prendere delle decisioni sbagliate. Per quanto riguarda le mitigazioni, per questo fallimento l'unica possibile è quella della sostituzione dell'intero obiettivo oppure delle singole lenti all'interno del gruppo di lenti specifiche.

Questo particolare fallimento può essere dovuto a diverse cose: detriti di vario tipo che si abbattono sulla telecamera durante la marcia o da ferri, maltempo, in particolare grandine (chicchi di una certa dimensione e durezza, tali da far infrangere la lente) ed anche intenzionalità da parte di un soggetto esterno (atti di vandalismo). Per quanto riguarda la Detectability, si è assegnato Low, in quanto la rottura di un pezzo così fondamentale è facile sia da rilevare che da mitigare. Per la Severity si è voluto assegnare a questo fallimento un valore minimo, in quanto con la riparazione (se è solo la rottura di uno o più vetri all'interno dell'obiettivo) si risolve il problema.

DIRTY INT. - DIRTY EXT.

Questi due fallimenti possono essere discussi insieme, in quanto riguardano entrambi i detriti, di vario genere e dimensione, che si possono depositare sulla lente, all'interno od all'esterno dell'obiettivo. La differenza più significativa fra i due è che lo sporco esterno può essere rimosso, mentre quello interno richiede più tempo e, a volte, personale specializzato. Se si pensa che l'obiettivo montato su un'auto a guida autonoma non potrà essere uguale a quello che si trova su una semplice macchina fotografica, ci si accorge come anche il solo separarlo dal corpo macchina, operazione banale sulle macchine fotografiche in commercio, non risulterebbe un'operazione tanto semplice.

Dunque, l'eventuale polvere all'interno dell'obiettivo deve essere rimossa da centri di assistenza specializzati poiché lo smontaggio dell'obiettivo comporta problemi che possono precludere il buon funzionamento dello stesso. Non esiste per questo particolare microcomponente dell'obiettivo un modo automatizzato per la pulitura. Una soluzione può essere descritta dall'articolo [45] nel quale si cerca di rimuovere artefatti dovuti a lenti sporche. Altro lavoro che prende questa situazione come problema da risolvere è [46]. La soluzione proposta da questi studi è comunque da considerare a livello di sistema, in quanto l'elaborazione per correggere l'immagine non avviene nello stesso componente Obiettivo, ma dopo la fase di acquisizione. Oltre a questo si può dire che, qualora lo sporco sull'obiettivo fosse molto esteso e non si limitasse a soli piccoli punti sulla lente, simili a gocce d'acqua, anche l'utilizzo di queste soluzioni espresse negli elaborati presentati sarebbe arduo e forse non possibile.

Per quanto riguarda lo sporco esteriore dell'obiettivo, invece, più in particolare dell'ultima lente a contatto con l'ambiente esterno, esistono soluzioni come quella proposta dalla società Ficosa Corporation: questa utilizza acqua con una pompa lavavetri per la pulizia degli obiettivi delle videocamere montate sul veicolo [47].

FLARE

Il cosiddetto flare è un fenomeno dovuto al riflesso del sole o altre sorgenti luminose sulle lenti interne all'obiettivo. Questo causa delle macchie, di solito più di una e concentrate su una retta immaginaria, di

vario colore, che possono coprire dettagli nei fotogrammi dove questo fenomeno si presenta. Le fasi successive di elaborazione, seppur l'immagine venga catturata correttamente, saranno influenzate dalla presenza di queste macchie. Data la costruzione degli attuali obiettivi, con una serie di lenti distanziate di poco fra di loro, e tutte con un compito specifico, è difficile eliminare del tutto questo fenomeno, in quanto la costruzione dell'obiettivo stesso ne provoca la manifestazione.

Ci sono vari articoli in cui si propone di rilevare e poi rimuovere anche automaticamente queste tipologie di macchie dovute a sorgenti luminose di vario genere. In particolare si cerca di rimuoverle in [48]. In questo, in una delle ultime immagini presentate, viene mostrato come un "flare" di colore verde venga interpretato come un semaforo, mentre non era altro che una macchia luminosa, di colore verde, derivante da un'altra sorgente luminosa. Naturalmente l'articolo non si propone di eliminare tutti i possibili fenomeni di flare che si possono verificare, ma con quelli più gestibili la soluzione è valida.

RAIN

Con questo fallimento si intende il caso in cui sulla lente della videocamera sono presenti diverse piccole macchie, per lo più di colore bianco, dovute al depositarsi di gocce d'acqua sulla lente esterna. Per eliminare questo tipo di fallimento esistono diverse soluzioni, ma se si pensa ad una videocamera utilizzata durante la marcia e mentre è in corso una precipitazione, ci si accorge di quanto può essere difficile farlo in un tempo computazionalmente accettabile. Differente può essere considerare l'eliminazione di queste macchie durante la sosta del veicolo e quando non siamo in presenza di precipitazioni. Data però la variabilità che il veicolo può trovare nell'ambiente circostante, non possiamo soltanto considerare il caso più semplice.

I modi per rilevare ed eliminare le macchie di questo tipo (ad esempio quando si guarda un vetro su cui è appena piovuto) sono vari: si va dall'apparato meccanico che cerca di togliere le macchie in questione con delle specie di tergicristalli per la lente, arrivando alla soluzione software nella quale si cerca di eliminare tali macchie in fase di elaborazione. Per quanto riguarda la soluzione software, in [46] si cerca di fare proprio questo. Naturalmente, si parla sia di sporco (fallimento DIRTY INT. - DIR-

TY EXT.) che di gocce d'acqua, in quanto le due cose possono assumere uguali forme di disturbo. Se invece non si vuole intervenire via software, una soluzione potrebbe essere quella di utilizzare prodotti idrorepellenti da applicare (in modo manuale o automatico) direttamente sulla lente (ad esempio "Clarifii: Water Repellent and Anti Fog" [49]). Anche in questo caso la mitigazione può essere considerata sia a livello di sistema che del solo componente: se si ricorre ad un qualsiasi strumento software per la rimozione post-acquisizione allora si parla di mitigazione a livello di sistema, mentre se si utilizzano prodotti come quello indicato sopra, si agisce con una mitigazione a livello del solo componente.

La pioggia è una condizione che l'auto a guida autonoma deve affrontare. Infatti, almeno che le auto a guida autonoma non si facciano viaggiare nel sottosuolo o in posti nel mondo dove c'è una forte scarsità di precipitazioni, la pioggia sarà sicuramente uno dei problemi da tenere in considerazione, dato che il veicolo esegue comandi derivanti dall'analisi delle immagini acquisite e da tutti gli altri dati provenienti dal resto dei sensori utilizzati.

I punteggi di Detectability e Severity sono stati assegnati tenendo conto di un algoritmo che sia in grado di fare quanto detto sopra e soprattutto di farlo in tempi brevi, come richiede un veicolo a guida autonoma. Naturalmente se si considera il solo utilizzo di prodotti idrorepellenti, le due metriche scendono ai minimi valori, in quanto si ha una soluzione veloce e facile da impiegare.

CONDENSATION

Quando la temperatura dell'aria esterna si riduce bruscamente, la condensa può apparire sulle lenti dell'obiettivo di una telecamera. La condensa, o umidità, provoca il degrado delle immagini e, se penetra all'interno del dispositivo, anche il degrado delle parti elettroniche. Questo fenomeno fa sì che l'immagine venga acquisita nel modo corretto, ma ogni fotogramma potrebbe presentare delle pesanti modifiche dovute ad aloni sulle lenti. La condensa, infatti, può formarsi anche all'interno delle lenti dell'obiettivo. Questo può provocare degli annebbiamimenti nei fotogrammi o determinare la completa inutilità dell'immagine acquisita. Il fenomeno, nei casi più fortunati, potrebbe attenuarsi da solo, ma è un fallimento a cui si deve prestare molta attenzione. Una soluzione comune, proposta

inizialmente per telecamere di sorveglianza, potrebbe essere quella illustrata in [50]: in questo documento si propone una soluzione al formarsi della condensa. La mitigazione è comune a tutto il dispositivo telecamera (corpo e obiettivo). Questa prevede l'utilizzo di un contenitore secondario per la telecamera col quale si favorisce lo scorrere dell'aria attraverso l'insenatura che viene a formarsi fra i due involucri. Inoltre, in questo brevetto, si propone anche una componente riscaldante per la telecamera, in modo che la temperatura rimanga sempre la stessa e non si abbia lo sbalzo termico che favorisce il formarsi della condensa. Sullo stesso principio si basa un'altra soluzione al problema della condensa, sempre proposto in ambito sorveglianza, che prende sempre in considerazione la possibilità di avere un doppio involucro in modo da creare uno spazio d'aria vuoto fra il contenitore primario e quello secondario ([51]). Un'altra soluzione può essere trovata nel brevetto [52] in quanto pensato proprio per un uso su veicolo.

Le soluzioni trovate, anche se riguardanti telecamere usate per un'altra funzione, sono un punto di partenza per sviluppare progetti che possano essere impiegati nel campo dell'automotive. In particolare, più che la doppia campana considerata per le telecamere di sorveglianza, è da tenere presente il riferimento ad un componente che permette il riscaldamento del dispositivo (per temperature basse) per mantenere sempre lo stesso grado di umidità locale. Va detto, però, che a questo dispositivo di riscaldamento andrebbe affiancato anche uno che svolga la funzione opposta, in condizioni di caldo eccessivo (discusso nel seguente punto).

HEAT

Questo tipo di fallimento riguarda il calore del quale un componente, in questo caso l'obiettivo, può risentire nella sua vita operativa. Nei casi estremi potrebbe portare ad un'evaporazione dei liquidi lubrificanti delle parti mobili. Se si pensa ad esempio a parti mobili che si possono trovare su una videocamera, in particolare sull'obiettivo, l'impossibilità di usarle normalmente potrebbe precludere l'uso dello zoom (se presente) e di altri componenti che hanno lo scopo di rendere l'immagine il più chiara possibile (strumenti per la messa a fuoco ecc.). Le fasi successive all'acquisizione, dunque, potrebbero dover processare immagini non chiare, con la possibilità di interpretarle erroneamente.

In questo caso, una possibile mitigazione è quella di utilizzare garnizioni e altri componenti del corpo macchina soggetti a deformazione di vario genere, a prova di calore (temperature estreme). Se non sono utilizzati componenti di questo genere si può incorrere in una rottura non sempre riparabile. Una soluzione si ritrova nel campo della videosorveglianza, in cui l'azienda Axis Communications, leader nel settore, fabbrica telecamere a prova di temperature desertiche come la AXIS Q60-C PTZ, dotata di raffreddamento attivo integrato, che permette una maggiore affidabilità in ambienti ad alta temperatura ($[-20\text{ }^{\circ}\text{C}, 75\text{ }^{\circ}\text{C}]$; soddisfano lo standard militare MIL-STD-810G).

Per questo fallimento si parla più di come deve essere progettata esternamente la videocamera, andando a considerare tutti i materiali impiegabili nella costruzione di essa. Infatti, se qualcosa fallisce all'esterno, permette ad agenti esterni di entrare e rovinare la componentistica interna, nei vari modi anche elencati in questa analisi.

SAND

In questo fallimento si considerano i problemi che la sabbia potrebbe causare nella componentistica. Infatti, per causa di questa, ci può essere una possibile corrosione dei componenti esterni dell'obiettivo (percentuale di sale nella sabbia), con conseguente introduzione di agenti esterni all'interno del dispositivo. La telecamera potrebbe non acquisire immagini esatte in quanto la sabbia, specialmente nell'obiettivo, causerebbe, ad esempio, il blocco dell'otturatore e di altri componenti che hanno lo scopo di rendere l'immagine il più chiara possibile (strumenti per la messa a fuoco, zoom ecc.). Per questo problema non c'è alcuna cosa che, una volta danneggiata una giunzione o un componente isolante, permetta di risolvere la situazione aspirandola, sciogliendola o quant'altro. L'unica soluzione è progettare l'involucro della telecamera in modo tale che la sabbia, anche se depositata su questo per un lungo tempo, non ne danneggi la robustezza e non entri. La mitigazione deve essere considerata per tutto l'apparecchio, quindi a livello di sistema, dato che gli stessi problemi che si riscontrano per l'obiettivo si ritrovano nel corpo macchina.

Non è stato considerato il problema di quando la sabbia rimane sulla lente, corrompendo l'immagine. Questo è stato fatto volutamente in

quanto si presume che la sabbia, se il veicolo è in marcia, non si accumuli sulla lente per via della sua forma e progettazione. Inoltre, il fallimento dell'immagine "sporca" è già stato analizzato.

ICE

Per questo fallimento sono state dedicate (per il componente obiettivo) due righe della tabella FMECA. Infatti, non si è soltanto considerato cosa il ghiaccio può fare ai materiali esterni dell'obiettivo (**A**), ma anche la possibilità che si formi uno strato di ghiaccio sulla lente esterna che non permetta l'acquisizione di immagini chiare (**B**), ovvero che non contengono informazioni.

Per la prima casistica (**A**), alcuni materiali potrebbero risentire delle basse temperature tanto da rompersi in modo anomalo (temperature di molto inferiori allo zero). I materiali di cui si parla sono da intendere come quelli utilizzati per far sì che il dispositivo sia isolato in maniera perfetta dall'ambiente esterno. Qualora si riuscissero a trovare materiali che rispondono a queste necessità di robustezza, un problema aggiuntivo potrebbe essere dato da quello che viene definito "caos nella fisica dell'elettronica": si parla solitamente di temperature al di sotto dei -50 °C. Per questo problema, la mitigazione che si propone sarà comune a tutto l'apparecchio nella sua completezza (corpo macchina + obiettivo). A questo fallimento è stato dato il massimo grado di pericolosità congiunto (fra Detectability e Severity come sopra indicati). Questo perché se il veicolo deve essere in grado di guidare da solo in ogni situazione, come vuole il massimo grado SAE 5, lo dovrà fare anche con una temperatura esterna estrema (-70 °C a Oymyakon in Siberia e fino a -98 °C in Antartide). Quindi, se i materiali con cui si costruisce tale dispositivo non sono adeguati a queste temperature, l'auto potrebbe non poter più svolgere tutte le azioni in modo autonomo (intervento del conducente).

Per quanto riguarda la seconda casistica (**B**), invece, contro la coltre di ghiaccio che potrebbe venire a formarsi sulla lente esterna, una soluzione possibile potrebbe dover essere ancora inventata, senza considerare il fatto che esistono sul mercato prodotti a base alcolica che permettono una risoluzione del problema in tempi brevi. Si è pensato però che una soluzione potrebbe derivare da meccanismi o strumenti già inventati e utilizzati in tutto il mondo: è l'esempio del lunotto posteriore termico.

Si tratta di un semplice finestrino dotato di resistenze elettriche, inserite o applicate, con la funzione di evitare fenomeni di condensa interna. Questo potrebbe essere preso in considerazione come punto di partenza per lo sviluppo di materiali riscaldabili alla stessa maniera, ma, possibilmente, non attraversati dalle resistenze. Questa soluzione un po' futuristica è necessaria in quanto la telecamera non si può permettere di processare immagini con linee orizzontali fisse, dato che potrebbero provocare un comportamento indesiderato del veicolo durante la marcia. Il punteggio di Detectability-Severity assegnato è stato più basso. Infatti, lo sviluppo nel campo automobilistico è all'avanguardia e costruire lenti che, come il lunotto posteriore, possano scaldarsi, qualora ce ne sia il bisogno, non sembra impossibile (Abrams Talbert, Milford B. Moore e Nelson Roy hanno pensato ad un riscaldatore dell'obiettivo nel 1948 [53]).

WATER

Il nome di questo fallimento dice già abbastanza su quello che abbiamo pensato possa influenzare in modo negativo il funzionamento del dispositivo telecamera. Infatti, qualora l'acqua penetrasse all'interno dell'obiettivo potrebbe bloccare molte componenti azionate elettricamente e probabilmente non acquisire più immagini o acquisirle completamente senza significato (contenuto). Oggi sul mercato esistono telecamere di ogni tipo che permettono di andare a diversi metri sott'acqua, senza che succeda alcunché all'obiettivo ed al corpo macchina. La tecnologia per ovviare a questo fallimento è esistente e possibile da utilizzare su un veicolo a guida autonoma. I livelli di Detectability e Severity assegnati a questo fallimento, considerato sia sull'Obiettivo che sul Corpo Macchina, sono gli stessi, dato che portano alla stessa conclusione, ovvero un fallimento dell'elettronica interna. I valori assegnati sono meno critici di altri, in quanto, anche se il fallimento porterebbe all'inutilizzabilità della telecamera, le tecnologie per mitigarlo sono esistenti e presenti sul mercato da molto tempo.

BRACKISH/SALT WATER

Con questo fallimento si vuole intendere quello che viene comunemente chiamato salmastro. Il fenomeno è comune nelle zone costiere e mette a dura prova la durabilità di molti materiali se non trattati con prodotti

adeguati e/o non manutenuti nel tempo. Grazie al suo potere corrosivo, dovuto alla consistente percentuale di sale che porta con sé, potrebbe danneggiare esternamente l'obiettivo dopodiché permettere ad agenti esterni di entrare nei circuiti interni dalla breccia creata. Sicuramente, nel peggior scenario immaginabile, questo fenomeno potrebbe influenzare l'acquisizione del fotogramma (immagine non fedele alla realtà) e far sì che il dispositivo non sia più in grado di catturare immagini. Per questo fallimento, l'unico modo di prevenirlo è costruire il corpo esterno ed anche l'obiettivo (componentistica esterna in generale) con materiali e tecnologie che non favoriscono la corrosione. A questo proposito, il brevetto [54], propone un dispositivo e un metodo per prevenire incrostazioni e corrosione delle superfici esposte di una struttura a contatto con acqua marina. Questo sembra un punto di partenza da prendere in considerazione. Anche per questo fallimento le mitigazioni andranno a riguardare tutto l'apparecchio, non un componente in particolare (mitigazione a livello di sistema).

La soluzione proposta può essere presa in considerazione anche se necessita di un processo di adeguamento all'ambito in analisi (Automotive). Per quanto riguarda i punteggi di Detectability e Severity, questi sono rispettivamente High e Catastrophic, ovvero il massimo della criticità che può essere assegnata ad un componente. Infatti, se si considerano i danni che il salmastro può fare all'esterno, e conseguentemente all'interno, questo è uno dei peggiori fallimenti che possono accadere e per il quale una soluzione vera e propria non esiste.

WIND

In questo, si vanno a considerare quelle parti di componente che presentano o che possono presentare intercapedini che, con la forza del vento, in marcia e/o in sosta, potrebbero portare a danneggiamenti esterni (tanto quanto basta) con conseguente infiltrazione di agenti vari all'interno del dispositivo. L'acquisizione delle immagini quindi potrebbe essere errata: possibilità di immagini non a fuoco o che presentano artefatti vari. La soluzione è rappresentata dalla costruzione dell'obiettivo (e della componentistica esterna in generale) in modo che questa faccia il meno possibili resistenza all'aria. I punteggi di Detectability e di Severity assegnati non sono fra i più critici, in quanto la progettazione di telecamere o componentistica esterna a bassa resistenza all'aria sono già sul mercato e

si hanno tecniche avanzate (gallerie del vento) per trovare la soluzione perfetta.

4.2.2.2 *Corpo Macchina*

Il secondo componente preso in esame è il Corpo Macchina che viene contrassegnato col colore  all'interno della tabella FMECA. Di seguito i fallimenti trovati per tale componente.

CONDENSATION

Come già descritto per il componente Obiettivo, la condensa può portare ad un degrado dell'elettronica interna. La telecamera, infatti, potrebbe non essere più in grado di svolgere il suo lavoro nel modo corretto e nei casi peggiori arrivare al totale blocco dell'apparecchio. Pensando ad un veicolo a guida autonoma, sicuramente le videocamere montate su esso saranno più di una. Questo può portare ad un blocco parziale (nei casi peggiori totale) del sistema auto. Infatti, le videocamere non funzionanti, faranno sì che la parte o le parti di ambiente esterno sorvegliate dalle telecamere affette da questo fallimento non verranno elaborate, andando a creare una sorta di punto cieco per il sistema.

Una soluzione comune, proposta inizialmente per telecamere di sorveglianza, potrebbe essere quella illustrata in [50], come detto in precedenza: in questo documento si propone una soluzione al formarsi della condensa. Questa è comune a tutto il dispositivo (Obiettivo e Corpo Macchina): prevede l'utilizzo di un contenitore secondario per la telecamera col quale si favorisce lo scorrere dell'aria attraverso l'insenatura che viene a formarsi fra i due involucri. Quando la temperatura dell'aria esterna si riduce bruscamente, la condensa può apparire sulle lenti dell'obiettivo. L'umidità provoca il degrado delle immagini e, se penetra all'interno del dispositivo, anche il degrado delle parti elettroniche. Inoltre, in questo brevetto si propone anche una componente riscaldante per la telecamera, in modo che la temperatura rimanga sempre la stessa e non si abbia lo sbalzo termico che favorisce il formarsi della condensa.

Sulla stesso principio si basa un'altra soluzione, sempre proposta in

ambito sorveglianza, che allo stesso modo prende in considerazione la possibilità di avere un doppio involucro per creare uno spazio d'aria vuoto fra il contenitore primario e quello secondario ([51]).

Le soluzioni trovate, anche se riguardanti telecamere usate per un'altra funzione, sono un punto di partenza per sviluppare progetti che possono essere impiegati nel campo automotive. In particolare, più che la doppia campana considerata per le telecamere di sorveglianza, è da tenere presente il riferimento ad una componente che permette il riscaldamento del dispositivo (per temperature basse) per mantenere sempre lo stesso grado di umidità locale. Va detto, però, che a questo dispositivo di riscaldamento andrebbe affiancato anche uno che svolge la funzione opposta per condizioni di caldo eccessivo.

HEAT

Come anticipato per il componente Obiettivo, questo fallimento pre-suppone che ci sia una condizione di caldo estremo all'esterno dell'apparecchio, tanto da far danneggiare in modo irreparabile guarnizioni ed altre parti della videocamera. Con il danneggiamento di queste, infatti, si potrebbero aprire brecce, anche di piccole dimensioni, che favorirebbero l'entrata di agenti esterni fra cui, ad esempio, anche l'acqua. Come in precedenza, questo fallimento trova soluzione nei materiali utilizzati per la costruzione di tutto il dispositivo. Inoltre, anche meccanismi di raffreddamento automatici sono da tener presente come base di progettazione per l'ambito in cui questa analisi è stata svolta (Automotive).

Nominata precedentemente, una soluzione si trova nel campo della videosorveglianza con l'azienda Axis Communications, leader nel settore, che fabbrica telecamere a prova di temperature desertiche come la AXIS Q60-C PTZ, dotata di raffreddamento attivo integrato, che permette una maggiore affidabilità in ambienti ad alta temperatura ($[-20\text{ C}^{\circ}, 75\text{ C}^{\circ}]$; soddisfano lo standard militare MIL-STD-810G).

SAND

Nello stesso modo del componente Obiettivo, si riscontra una possibile corrosione delle guarnizioni dovuta alla percentuale di sale che si

trova nella sabbia. Il danneggiamento di queste, come detto, può portare al degrado di parti fondamentali che hanno l'obiettivo di mantenere fuori gli agenti esterni, dunque garantire l'isolamento del dispositivo. Lo scenario peggiore che può presentarsi è la rottura di questi pezzi fondamentali per l'isolamento, con una conseguente entrata, anche duratura nel tempo, di agenti esterni, deleteri per la circuiteria interna.

Per quanto riguarda il problema della sabbia, non c'è alcuna cosa che, una volta danneggiata una giunzione o un componente isolante, permetta di risolvere la situazione aspirandola, sciogliendola o quant'altro. L'unico modo è progettare l'involucro della telecamera in modo tale che la sabbia, anche se depositata su questo per un lungo tempo, non ne danneggi la robustezza e non entri. La mitigazione deve essere considerata per tutto l'apparecchio, quindi a livello di sistema, dato che i problemi che si trovano sono gli stessi anche per il componente Obiettivo.

ICE

Come descritto precedentemente per l'Obiettivo, il ghiaccio può causare la deformazione e conseguente rottura di parti esterne della videocamera. I materiali utilizzati per l'isolamento esterno del dispositivo devono essere tra i più resistenti per quanto riguarda le temperature estreme (sia estremamente basse che, come detto, molto alte). Inoltre, se da una parte questi materiali possono esistere, dall'altra esistono evidenze per cui, superata una certa temperatura, il freddo può provocare il "caos nella fisica dell'elettronica" (si parla di temperature inferiori ai -50 °C). Anche in questo caso, la mitigazione che si propone sarà comune a tutto l'apparecchio nella sua completezza (mitigazione a livello di sistema).

A questo fallimento è stato dato il massimo grado di pericolosità congiunto (fra Detectability e Severity). Questo perché se il veicolo deve essere in grado di guidare da solo in ogni situazione, come vuole il massimo grado SAE 5, lo dovrà fare anche con una temperatura esterna estrema (-70 °C a Ojmjakon in Siberia e fino a -98 °C in Antartide). Quindi, se i materiali con cui si costruisce tale dispositivo non sono adeguati a queste temperature, l'auto potrebbe non poter più svolgere tutte le azioni in modo autonomo (intervento del conducente, ovvero non più guida autonoma vera e propria).

WATER

Nello stesso modo in cui l'acqua può danneggiare la minore circuiteria interna del componente Obiettivo, anche per il Corpo Macchina si ha lo stesso problema. L'entrata dell'acqua all'interno del corpo causerebbe sicuramente un blocco totale del dispositivo. Per blocco totale si intende l'impossibilità di acquisire le immagini, di elaborarle e quindi di dare informazioni in input all'elaboratore centrale (in attesa di decidere quale azione intraprendere). Come anticipato per il componente Obiettivo, ci sono tantissime tipologie di videocamere sul mercato che permettono di immergersi a diversi metri sott'acqua senza che questa possa entrare all'interno del dispositivo. Oltre a queste soluzioni, si possono trovare migliaia di involucri nei quali poter avvolgere la propria telecamera non impermeabile.

I livelli di Detectability e di Severity assegnati non sono critici, in quanto, anche se il fallimento porterebbe all'inutilizzabilità della telecamera, le tecnologie per mitigarlo sono esistenti e presenti sul mercato da molto tempo.

BRACKISH/SALT WATER

Come già scritto per il componente Obiettivo, il salmastro è un fattore di deterioramento importante per le parti isolanti esterne. Questo problema, nel tempo, può causare una rottura di microcomponenti fondamentali, lasciando agli agenti esterni, di qualsiasi genere, strada libera per l'interno del dispositivo. Per non avere questo problema, quindi non incorrere in questo fallimento, le soluzioni proposte per l'Obiettivo sono raccomandabili anche per il Corpo Macchina. A questo proposito, il brevetto [54], precedentemente nominato, propone un dispositivo e un metodo per prevenire incrostazioni e corrosione delle superfici esposte di una struttura a contatto con acqua marina. Questo sembra un punto di partenza da prendere in considerazione. Per questo fallimento le mitigazioni andranno a riguardare tutto l'apparecchio, non un componente in particolare (a livello di sistema).

Per quanto riguarda i punteggi di Detectability e Severity, questi sono critici, in quanto se si considerano i danni che può fare all'esterno, e conseguentemente all'interno, è uno dei peggiori fallimenti che pos-

sono accadere e per il quale un rimedio o soluzione vera e propria non c'è.

ELECTRICAL OVERLOAD

Con questo fallimento il dispositivo non acquisisce più le immagini come dovrebbe o non le acquisisce affatto: l'aumento eccessivo e pericoloso della temperatura nei conduttori potrebbe determinare delle rotture ed altre conseguenze nel corpo macchina. Il dispositivo con questo problema generico di carattere elettrico potrebbe smettere di funzionare o ritrovarsi in uno stato, successivo all'acquisizione, nel quale non è possibile processare le immagini. Esistono due tipi di protezione per le unità elettriche che si possono considerare:

- protezione dei cavi elettrici che alimentano i circuiti da un sovraccarico superiore alla loro capacità di carico;
- protezione da sovraccarico dei singoli apparecchi e apparecchiature elettriche collegate a un circuito di alimentazione.

Nel nostro caso, la prima soluzione è da tenere presente, andando ad isolare la circuiteria presente (anche col controllo dell'amperaggio). Qui la mitigazione deve essere considerata a livello di sistema, in quanto l'energia elettrica è fondamentale per il funzionamento di ogni componente, e dunque in ognuno si ritrovano circuiti soggetti a questo fallimento.

Esistono oggi molti modi per evitare il sovraccarico di corrente elettrica, ma questo fallimento, come mostrano i punteggi di Detectability e Severity assegnati, non deve essere tralasciato, in quanto causerebbe un danno irreparabile di tutta la telecamera ed una conseguente mancanza di dati fondamentali per prendere decisioni di marcia.

4.2.2.3 *Filtro di Bayer*

Il terzo componente preso in esame è il Filtro di Bayer, fondamentale per la colorazione dell'immagine, contrassegnato col colore  all'interno della tabella FMECA. Di seguito i fallimenti trovati per tale componente.

NO BAYER FILTER

Con questo fallimento, più che il malfunzionamento vero e proprio del componente denominato Filtro di Bayer, si vuole evidenziare il fatto che, senza la presenza di questo, per un qualsiasi motivo, le immagini acquisite risulterebbero con i colori sbagliati. Colori sbagliati, in questo caso, sta a significare immagini visualizzate come scala di grigi. Le fasi successive all'acquisizione quindi si troverebbero ad elaborare delle immagini totalmente sbagliate dal punto di vista cromatico.

Un problema che può derivare dalla prolungata, erronea, esposizione ad una fonte di luce, potrebbe però decolorare alcune parti del filtro causando macchie sul fotogramma acquisito. Il problema deriverebbe, quindi, soprattutto dall'esposizione ai raggi UV. Tuttavia, in una videocamera utilizzata su un veicolo a guida autonoma, un filtro per questi raggi sarebbe applicato sicuramente. Se così non fosse, però, questa rimarrebbe la soluzione prediletta per questo problema.

4.2.2.4 *Sensore*

Il quarto componente preso in esame è il Sensore d'Immagine che viene contrassegnato col colore  all'interno della tabella FMECA. Di seguito i fallimenti trovati per tale componente.

SPOTS

Questo fallimento inherente il componente Sensore di Immagini si verifica quando piccole particelle di polvere (o altro tipo di materiale) si depositano su di esso. Questo deposito fa sì che sulle immagini acquisite correttamente siano presenti delle piccole macchie visibili soprattutto su colori chiari. Tali ombreggiature sono per lo più di forma circolare e sono molto comuni nella fotografia amatoriale, in particolare quando si fa uso di più obiettivi. Infatti, durante il cambio di essi, l'entrata di agenti esterni, anche di dimensione impercettibile, è molto probabile e possono andare a depositarsi sul componente, in quel momento non protetto.

La soluzione a questo problema deriva dal mondo della fotografia digita-

le. Infatti, su molte marche produttrici di DSLR o Mirrorless è installato un sistema chiamato **Dust reduction system**. Questo serve per rimuovere la polvere dal sensore di immagine. La polvere può essere generata da parti mobili interne o può essere mossa da correnti d'aria all'interno della telecamera. Alcuni sistemi puliscono il sensore vibrando a una frequenza molto elevata, tra 100 hertz e 50 kilohertz. Se questo non avesse gli effetti aspettati, allora sarebbe necessario procedere manualmente alla pulitura del sensore. Inoltre, esistono molti programmi di fotografia come Adobe Lightroom e Adobe Photoshop che hanno la funzionalità di "Rimozione macchie", che permette di centrare una macchia e sostituirla con una texture il più simile possibile selezionata all'interno della stessa foto.

Per una soluzione automatizzata via software, sono stati trovati articoli e brevetti che vogliono risolvere tale problema: [55] e [56].

Dunque, se si utilizza un approccio (quando possibile) manuale per la pulizia superficiale del sensore, si può parlare di mitigazione a livello di componente, mentre quando si vanno ad utilizzare strumenti software per rimuovere imperfezioni dall'immagine (in questo caso ombreggiature per lo più circolari) si parla di mitigazione a livello di sistema, in quanto si prevede un'elaborazione interna post-acquisizione.

Nel caso di un veicolo a guida autonoma la soluzione migliore sarebbe quella di rendere il più automatizzato possibile il processo di acquisizione e sanitizzazione delle immagini (fotogrammi). Un soggetto umano, più incline ad errori dovuti a disattenzione o stanchezza, sarebbe sicuramente più impreciso e lento di un computer. Naturalmente il primo approccio da utilizzare sarebbe quello di provvedere, una volta rilevata la macchia (con un detector come quello illustrato nell'articolo [56]), a pulire il sensore di immagine facendo uso del Dust reduction system. Se questo non bastasse si potrebbe procedere ad una rimozione via software. Se anche dopo questo step le macchie fossero sempre visibili, sarebbe opportuno rivolgersi ad un'assistenza specializzata, in quanto si pensa che il dispositivo montato su un auto a guida autonoma non sarebbe facilmente gestibile come una normale macchina fotografica.

I livelli di Detectability e Severity sono espressi come range e non presentano valori eccessivamente critici dato che il problema ha più soluzioni, già esistenti e di facile utilizzo.

DEAD PIXEL

Con questo fallimento le immagini sono acquisite correttamente, ma presentano uno o più difetti di dimensione di un pixel. Viene denominato DEAD PIXEL, in quanto si fa riferimento ad uno stesso guasto che però può essere notato sui monitor, quando appunto un pixel smette di funzionare nel modo corretto e nella sua locazione presenta sempre il colore nero. Il singolo pixel non funzionante molto probabilmente non preclude la buona interpretazione delle immagini catturate, ma se fossero molti di più potrebbe esserci un degrado nelle prestazioni di elaborazione e comprensione dei fotogrammi da parte del computer centrale.

Esistono vari modi per verificare che uno o più pixel si siano danneggiati. Uno di questi, nel campo della fotografia digitale è accendere lo schermo LCD per il Live View e controllare se si vede il difetto anche dopo l'acquisizione, in fase di presentazione. Altri modi, più applicabili all'ambito Automotive, prevedono un rilevamento automatico di tali malfunzionamenti e provano a correggere l'errore sul fotogramma acquisito, via software. Un articolo che si propone di fare questo è [57]: in esso ci si occupa della detection del problema. Un'altra possibile soluzione è data da [58] nel quale si propone un modo per, una volta rilevato il cosiddetto pixel nero, correggere il fotogramma andando a sostituire il suo segnale con il segnale del pixel immediatamente precedente nell'array. Quest'ultimo sembra essere una soluzione adeguata alle esigenze, in quanto come fallimento sembra essere molto raro, dunque i pixel da processare sarebbero pochi. Inoltre, se si pensa al fatto che il pixel per definizione si trova sempre nello stesso posto, questa correzione può essere fatta senza utilizzare sempre la detection. Per rilevare, invece, i nuovi pixel neri che possono fallire nel tempo, si può pensare ad un controllo periodico, che confronti la situazione attuale con una precedente: salvarsi la locazione dei pixel falliti e aggiornarla se, dopo un controllo, ne vengono trovati altri.

In questo caso si è preferito dare un livello di Detectability non critico, in quanto, una volta trovato il modo di rilevare il problema, questo rimane sempre fissato in un punto. Infatti, se si considera il fallimento SPOTS, le molecole di polvere possono continuare a muoversi una volta arrivate sul sensore, mentre il pixel nero rimane dove viene rilevato.

BRIGHT LINES

Questo fallimento al giorno d'oggi è abbastanza raro. Con questo, le immagini acquisite potrebbero presentare delle linee più o meno luminose verticali e/o orizzontali. La causa di questi difetti, anche molto evidenti, è dovuta all'utilizzo delle tecnologie LIDAR (discusse nel paragrafo 3.2.1), in quanto questi fanno uso di una tecnologia laser che per la sua intensità luminosa (non visibile all'occhio umano) può causare gravi danni ai sensori d'immagine (più sensibili dell'occhio umano). In Fig. 17 si può vedere il danno molto visibile.

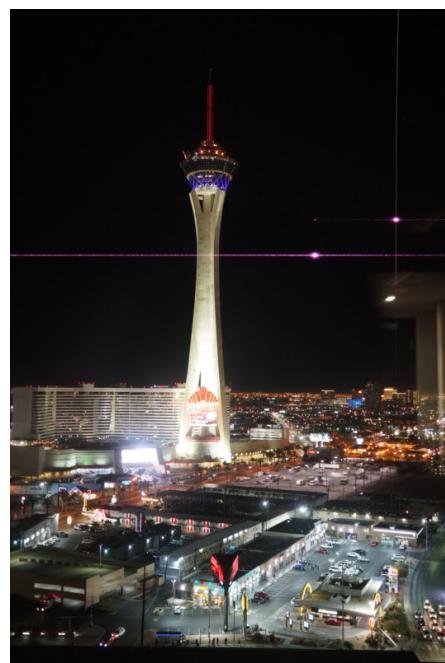


Figura 17.: Immagine esempio del fallimento BRIGHT LINES [6]

Si trova la dimostrazione di questo danneggiamento del sensore in vari articoli online: in particolare quello di un fotografo che ha provato il danno sulla propria fotocamera [59] ripreso anche in [60].

Non esistono mitigazioni per questo genere di fallimenti e il fatto che questi laser non siano visibili all'occhio umano fa sì che non ci sia modo di accorgersi quando evitare di catturare un'immagine che sicuramente danneggerà il nostro sensore. Le fotocamere digitali normalmente includono filtri che impediscono alla luce infrarossa di raggiungere i sensori

CMOS o CCD che altrimenti ne risentirebbero a determinate lunghezze d'onda. La vulnerabilità del sensore ai danni dei raggi infrarossi dipende dal design dei filtri utilizzati per proteggerlo. Le telecamere impiegate su auto autonome presumibilmente utilizzano filtri incorporati direttamente nei chip per bloccare la lunghezza d'onda lidar di 900 nanometri ampiamente usata, ma la loro vulnerabilità ai lidar di lunghezze d'onda più lunghe è sconosciuta.

Il problema in questione è molto serio, ma è anche facile da riscontrare a livello software. Infatti, sull'immagine catturata, come illustrato nell'articolo precedentemente nominato, le linee che si vengono a creare si accendono quando nell'ambiente esterno si ha poca luce, mentre rimangono nere in ambienti con una luminosità medio-alta. Come detto, il problema è serio se si considera che la tecnologia LIDAR è una delle maggiori tecnologie utilizzate dalle case automobilistiche nel campo della guida autonoma (insieme a Radar, Videocamere e Ultrasuoni per manovre con oggetti ravvicinati e a bassa velocità). Se si pensa ad un futuro fatto di auto a guida autonoma, che fanno tutte uso della suddetta tecnologia insieme a delle videocamere che possono danneggiarsi in modo irreparabile soltanto puntando uno di questi apparecchi, non è di certo dei più rosei. La soluzione primaria a questo problema, apparentemente irrisolvibile, è quella di montare sugli obiettivi (o direttamente sui sensori) delle videocamere dei filtri che schermino a tal punto da far risultare immune il dispositivo ai raggi emessi dai LIDAR (come l'occhio umano).

La Detectability assegnata non è critica, in quanto di filtri ne esistono già molti. Riguardo alla Severity, invece, i valori assegnati sono racchiusi in un range dato che, come detto precedentemente, prove con diverse lunghezze d'onda dei laser e studi su questo argomento ancora mancano. Dunque, se da una parte la "soluzione" può già esistere, non si sa fino a che punto protegga questi dispositivi.

BANDING

In questo fallimento risultano visibili tante righe orizzontali parallele in secondo piano nell'immagine acquisita. Le linee sono più visibili guardando i colori più scuri, ma anche sui più chiari possono essere percepiti (Fig. 18).

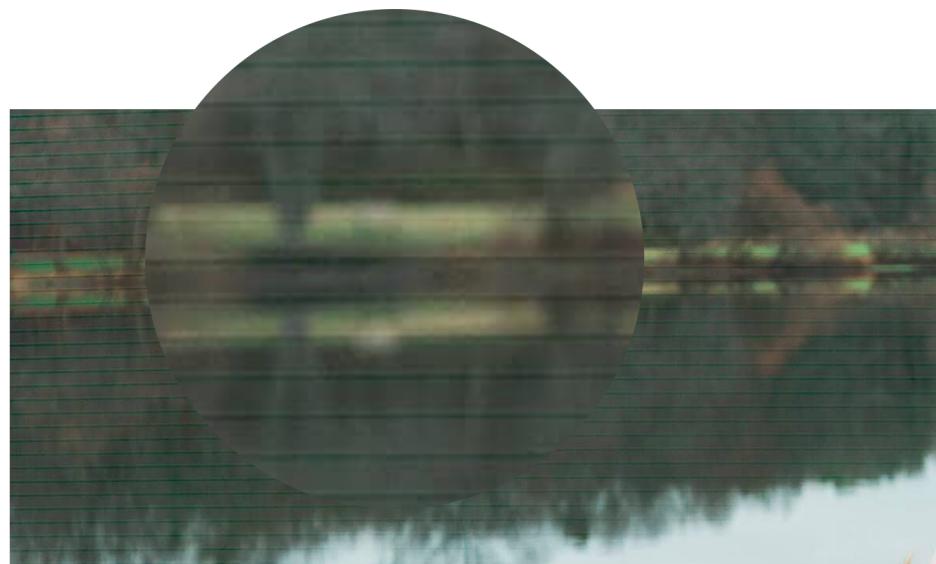


Figura 18.: Immagine esempio del fallimento BANDING (scarsa qualità sensore)

Il banding è un fenomeno non visto di buon occhio dai fotografi professionisti ed anche dai non professionisti. In particolare, questo tipo di banding, è soprattutto dovuto alla scarsa qualità del sensore.

Un'altro tipo di banding è quello che si può notare su immagini dove il colore unito di sfondo degrada, dal punto di vista dei toni, in colori più chiari (Fig. 19).

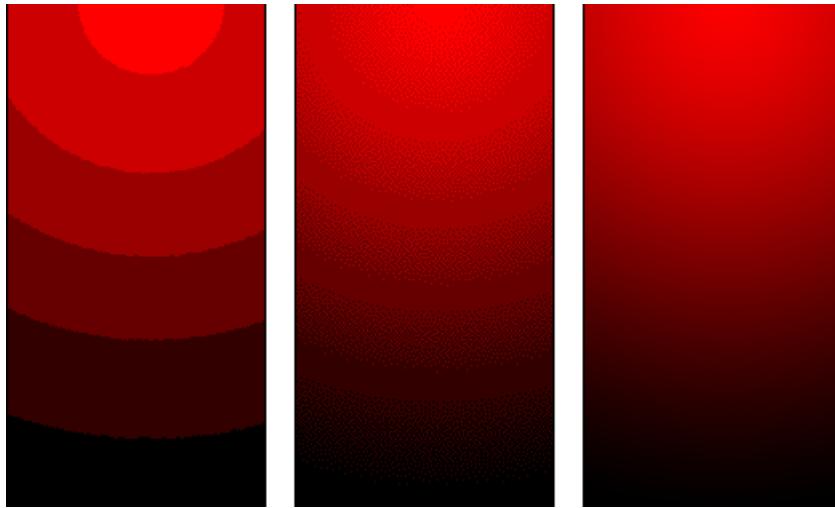


Figura 19.: Immagine esempio del fallimento BANDING (poca profondità colore, toni insufficienti) [7]

In Fig. 19 si vede come, passando via via da 8 bit a 8 bit dithered (il **dithering**, nella elaborazione numerica di segnali, è una forma di rumore con una opportuna distribuzione, che viene volontariamente aggiunto [61]), arrivando a 24 bit nell'ultima immagine (partendo da sinistra), l'effetto banding sia sempre meno visibile.

Questo fenomeno si crea, come detto, soprattutto in condizioni nelle quali si passa da un colore chiaro ad uno scuro. In questo scalare della tonalità si viene a formare il banding. Il problema si viene a creare quando non sono disponibili toni sufficienti per ricreare una gradazione uniforme, motivo per cui è più comune nelle immagini a 8 bit e con immagini pesantemente compresse. Le soluzioni vengono presentate in [62], nel quale vengono dati diversi consigli: primo fra tutti è quello di convertire l'immagine in una a 16 bit; il secondo è quello di non comprimere troppo il file. Infine, nei casi più gravi, si richiede un trattamento più attento, andando ad utilizzare un po' di rumore (dithering).

La mitigazione è poco probabile che sia soltanto a livello di componente, dato che il fotogramma necessiterà di varie elaborazioni che in alcuni casi non bastano a sanitizzare l'immagine completamente.

Naturalmente, le soluzioni proposte non devono produrre uno sforzo computazionale sul sistema molto elevato. Infatti, deve essere possibile

svolgere tali elaborazioni, insieme a tutte le altre, tenendo conto che il veicolo può muoversi anche a velocità elevate. Tanto più la velocità di marcia è elevata e tanto più veloci devono essere le fasi di elaborazione.

4.2.2.5 ISP - Image Signal Processor

Il quinto e ultimo componente preso in esame è l'ISP a cui viene assegnato il colore  all'interno della tabella FMECA. Di seguito i fallimenti trovati per tale componente.

NO DEMOSAICING - INCOMPLETE DEMOSAICING

No Demosaicing si riferisce al fatto che il fallimento prende in considerazione il caso in cui il fotogramma venga acquisito in formato grezzo o, nel gergo, in formato RAW. Questo significa che il processo di demosaicing, discusso in [63], non è ancora stato eseguito e quindi l'immagine si presenta con ogni pixel che contiene un valore blu, rosso oppure verde. In questo caso l'Array di Bayer non è ancora stato interpretato e l'immagine risulta più pixelata rispetto al normale, con prevalenza della tonalità verdastra (dovuto alla percentuale di verde, doppia rispetto a quella di rosso e blu, sul Filtro di Bayer).

Questa fase di elaborazione è la prima e la più importante. L'algoritmo di Demosaicing e l'architettura utilizzata per farlo operare devono avere un'affidabilità massima, pena la mancata corretta interpretazione della scena. L'algoritmo descritto nell'articolo [64] si propone come una valida alternativa a molti altri, svolgendo un buon lavoro anche con immagini problematiche dal punto di vista della ricostruzione dei colori. Interessante anche il brevetto [65], nel quale si fa riferimento ad un motore di Demosaicing. Questo può essere configurato per generare, mediante una tecnica adattativa selettiva per colore, un'immagine policromatica basata sui valori di pixel monocromatici catturati e su quelli stimati.

È fondamentale che questa fase funzioni al meglio. Per questo motivo, i punteggi di Detectability e Severity, non possono essere poco critici. Infatti, i metodi per fare in modo che questo processo abbia buon fine ci sono, ma sono difficili da implementare e comunque sfruttano anche

l'affidabilità della componentistica ausiliaria interna.

NO NOISE REDUCTION - INCOMPLETE NOISE REDUCTION

Con questo particolare fallimento il dispositivo cattura l'immagine, ma nelle fasi di elaborazione, in questo caso quella della riduzione del rumore, si ha un errore che determina la non corretta sanitizzazione del fotogramma (totale o parziale). Questo procedimento, qualora l'algoritmo si interrompesse durante l'esecuzione, potrebbe portare al blocco del dispositivo. Infatti, se l'immagine arrivasse in questa fase e non riuscisse ad andare oltre, le fasi successive non avrebbero alcun input su cui lavorare. Alla fine, dunque, il computer centrale non avrebbe informazioni per poter prendere le decisioni sulla marcia del veicolo e per mantenere, quindi, la sicurezza dei passeggeri e degli altri utenti della strada.

In determinate condizioni, l'immagine catturata da un qualsiasi dispositivo, può presentare del rumore. Questo, ad esempio, è ben visibile quando si cerca di scattare una foto in condizioni di scarsa luminosità. La scarsa luminosità ci porta ad alzare i tempi di esposizione, ad alzare il valore ISO e non solo. Il valore ISO indica la sensibilità alla luce del sensore della fotocamera digitale [8]. Alzarlo di molto spesso provoca il cosiddetto rumore (Fig. 20).

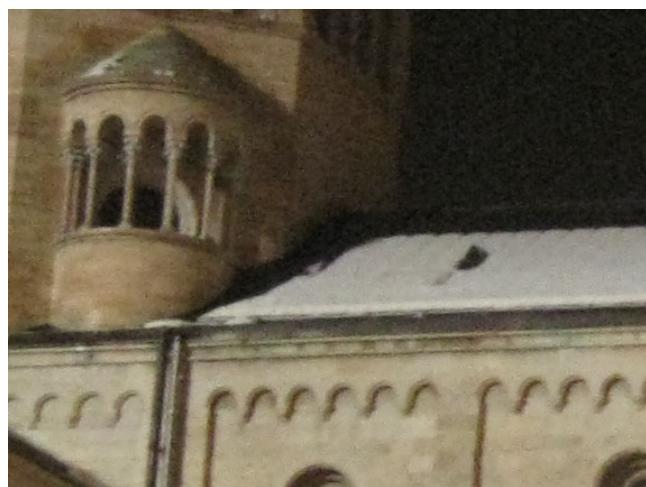


Figura 20.: Immagine esempio del fallimento NO NOISE REDUCTION [8]

Il risultato è un'immagine non ben definita che presenta del rumore. Que-

sto può essere mal interpretato dall'elaboratore centrale, il quale potrebbe percepire oggetti o situazioni non veritieri. Di algoritmi che effettuano questo processo di sanitizzazione dell'immagine ne esistono una grande varietà, con diverse qualità. La soluzione principale per risolvere questo problema è sempre l'affidabilità del processore, il quale non deve saltare questa fase, né deve effettuarla parzialmente. Per quanto riguarda la riduzione del rumore, l'articolo [66] sembra essere perfetto per il nostro caso: infatti, si tratta di un algoritmo adatto per l'ISP, nel quale non deve essere ricostruito il modello di rumore ad ogni immagine, ma una sola volta. Pensando a quante operazioni deve fare il processore, è raccomandabile risparmiare quanto più tempo computazionale possibile se ce n'è la possibilità.

NO SHARPNESS - INCOMPLETE SHARPNESS

Per No Sharpness si intende il fatto che le immagini catturate, durante le fasi di elaborazione, incorrono in un errore nella fase di correzione della nitidezza (in pratica la capacità di un apparecchio fotografico di identificare e definire il limite di separazione tra due aree contigue che abbiamo diversa luminosità e/o colore [67]). La nitidezza è da intendere come la chiarezza dei bordi e dei confini degli elementi che compongono un'immagine [67]. Nel nostro caso, dato che si parla di riconoscimento di tutto quello che sta intorno al veicolo, si potrebbe pensare a correggerla, aumentandone il valore. Infatti, come detto in precedenza, più i bordi di ogni soggetto sono definiti, più sarà possibile, per il computer centrale, comprenderne le forme.

Esistono vari articoli scientifici e/o studi che forniscono metodi ed appari per effettuare al meglio questo processo di aumento della nitidezza dell'immagine. L'importante, anche in questo caso, è che il processore non deve fallire nell'eseguire questa azione. Dunque, l'affidabilità di quest'ultimo deve essere il principale fattore di preoccupazione in fase di progettazione e costruzione.

I livelli di Detectability e Severity non possono essere poco critici, in quanto, pur esistendo una grande varietà di algoritmi per produrre i risultati desiderati, va considerata anche l'affidabilità del processore che deve eseguirli. Quindi, l'algoritmo può essere perfetto (e ne esistono di ben progettati), ma se non consideriamo l'affidabilità del processore

nel processo di sanitizzazione dell'immagine, questo non ci può portare molto lontano. Inoltre, qualora questo processo faccia sì che la qualità dell'immagine degradi, ci si potrebbe ricondurre al fallimento BLURRED (dove l'immagine non è affatto chiara).

NO LENS DISTORTION CORRECTION - INCOMPLETE LENS DISTORTION CORRECTION

Questo fallimento potrebbe essere un problema, ma solo su veicoli dove si utilizzano lenti grandangolari, che tendono a deformare l'immagine. Infatti, con queste lenti, l'immagine catturata appare come mappata attorno ad una sfera che risulta più sporgente verso l'osservatore, proprio al centro del fotogramma. Se la conversione a proporzioni normali (simmetriche naturali) non va a buon fine, l'immagine potrebbe bloccarsi in questa fase ed il sistema ritrovarsi in uno stato di stallo. Qualora l'immagine fosse processata, ma non completamente, le fasi successive potrebbero dover elaborare fotogrammi con proporzioni e forme degli oggetti circostanti falsate.

Per la mitigazione di tale problema dobbiamo sempre fare affidamento sulla bontà del processore che si utilizza. Infatti, di algoritmi e metodi per la riduzione della distorsione delle lenti o la loro correzione via software (dopo una misurazione dei loro coefficienti di distorsione) ce ne sono moltissimi, come i seguenti presi in considerazione: [68], [69] e anche [70].

Per quanto riguarda la Detectability e la Severity è stato considerato un valore di range non critico per la prima metrica, mentre un unico valore, anch'esso non critico, per la seconda. Questo per la varietà di metodi esistenti per fronteggiare il problema della distorsione delle lenti. Inoltre, si è considerato anche il fatto che in molti metodi viene effettuata una misurazione sulla lente che si vuole impiegare e, con i coefficienti di distorsione derivanti da tale misurazione, si procede con l'eliminazione via software, riducendo al minimo la possibilità di errore e i tempi di elaborazione.

NO CHROMATIC ABERRATION CORRECTION - INCOMPLETE CHROMATIC ABERRATION CORRECTION

Con questo fallimento nell'ISP, si vuole intendere la mancata elaborazione (totale o in parte) del fotogramma acquisito per la rimozione dell'aberrazione cromatica. In ottica l'aberrazione cromatica assiale (cioè presente anche lungo l'asse ottico) è un difetto nella formazione dell'immagine dovuta al diverso valore di rifrazione delle diverse lunghezze d'onda che compongono la luce che passa attraverso il mezzo ottico. Questo si traduce in immagini che presentano ai bordi dei soggetti aloni colorati. È un difetto del quale, in diversa misura, sono affetti tutti i sistemi ottici a lenti [71]. I fotogrammi presentano, come detto, delle "frange" di vario colore (viola per lo più) e una sorta di sfocatura generale (Fig. 21).



Figura 21.: Immagine esempio del fallimento NO CHROMATIC ABERRATION CORRECTION [9]

Anche in questo caso le fasi successive a questa procedura potrebbero non elaborare il fotogramma corrente perché bloccato nella fase di rimozione dell'aberrazione. Il sistema può andare in blocco (l'elaboratore centrale non elaborerà nessuna delle immagini acquisite). Se però l'immagine venisse elaborata in modo erroneo, le fasi successive potrebbero ricevere in input fotogrammi parzialmente corretti o del tutto "non sanificati", che potrebbero far incorrere in errore il computer centrale in sede di decisione.

Questo è un fallimento che può portare dei cambiamenti di colore, fondamentali per quanto riguarda la segnaletica stradale. Dunque, sempre tenendo conto dell'affidabilità del processore, esistono molti metodi e algoritmi che, in modo automatico, risolvono il problema dell'aberrazione cromatica (chiamata anche distorsione cromatica). Uno di questi è discus-

so nell'articolo [72], che mostra in che modo può essere eliminato questo disturbo dall'immagine processata. In questo si fa riferimento ad una singola immagine, ma può essere un buon punto di partenza per far sì che ci sia una correzione continua e automatica di ogni fotogramma che viene acquisito dalla telecamera. Altro metodo interessante è descritto nello studio [73], nel quale non solo si corregge l'aberrazione cromatica, ma si considera anche un detector che la rilevi.

I punteggi di Detectability e Severity assegnati sono gli stessi dei precedenti, in quanto, considerando che l'azione può essere svolta in modo corretto se l'algoritmo è valido e soprattutto se il processore è affidabile, i modi per implementare una correzione di questo genere sono vari e ne esistono molti a disposizione, pronti per lo sviluppo.

NO ACTION

Questo fallimento rappresenta il caso in cui l'ISP non risponde e quindi l'elaborazione delle immagini acquisite non avviene, come non avviene il seguente trasferimento dell'immagine "arricchita" al computer centrale per la fase decisionale. Il fotogramma acquisito rimane in formato grezzo, senza elaborazioni di alcun tipo. Anche qui, l'affidabilità del processore è il primo aspetto da tenere in considerazione e forse anche l'unico, in quanto, come detto, se il processore non è affidabile e tende a sbagliare o a non rispondere, il sistema non può contarcisi. L'affidabilità di un componente così importante si basa su molti aspetti. La cosa più importante è tener presente che, essendo impiegato su un veicolo a guida autonoma per l'elaborazione di immagini che servono per la visione artificiale del sistema, la sua progettazione deve essere eseguita facendo riferimento ai più alti standard di sicurezza e affidabilità. Una possibile mitigazione, che riguarda il sistema nella sua interezza, può essere quella di sottoporre il processore a dei test periodici (periodi brevi) per valutarne il corretto funzionamento. Oltre a questo potrebbe essere necessario aumentare l'affidabilità di un componente così fondamentale considerando l'utilizzo di più processori che svolgono le stesse funzioni: se non vengono rilevati problemi possono anche funzionare tutti con l'obiettivo della divisione del carico di lavoro; nel momento in cui venga rilevato un qualsiasi errore in uno o più di questi, i rimanenti potrebbero farsi carico del lavoro che non può essere svolto da quelli falliti. Così, anche se ne rimanesse uno, ma questo fosse in grado di elaborare tutta la mole di dati che arriva

con l'acquisizione dei fotogrammi, il sistema potrebbe continuare la sua marcia, anziché doversi fermare per un errore che non permette più al computer centrale di prendere decisioni.

Per quanto riguarda Detectability e Severity sono stati assegnati dei punteggi più critici rispetto a tutti gli altri fallimenti. Questo in quanto, qui, si presuppone di avere un pezzo fondamentale non funzionante. Negli altri casi discussi c'era sempre una fase successiva, mentre con questo non si arriva più in là dell'acquisizione. Dunque, questa è una condizione molto più critica rispetto alle precedenti analizzate.

La soluzione a questo tipo di fallimento non può essere trovata, ma la presenza di una quantità infinita di standard e regole che accertino l'affidabilità di tale fondamentale elemento, fa pensare che sia certamente possibile progettare e creare una componentistica ad alta affidabilità. Inoltre, con lo sviluppo tecnologico si può confidare in un'evoluzione nella maggior parte dei campi toccati dal problema della guida autonoma.

4.2.2.6 *Sommario*

In generale, tutti i fallimenti trovano soluzione in articoli scientifici, brevetti e/o prodotti già presenti sul mercato. Per alcuni, la soluzione non è stata descritta, in quanto la vita del componente stesso dipende dalle tecniche di realizzazione e progettazione adottate, nonché dall'affidabilità che si riesce ad ottenere per tutti i componenti e sottocomponenti chiave. Per i fallimenti evidenziati nel componente ISP, ad esempio, si ha quasi sempre l'opzione del post-processing, ovvero la possibilità di correggere l'artefatto in un tempo successivo. Naturalmente il fattore chiave in questo ambito è proprio la tempestività. Pensare di far attendere il sistema decisionale per provare a correggere ulteriormente i frame elaborati non sarebbe sensato, in quanto ci si aspetta che il veicolo prenda le decisioni del caso in completa autonomia e senza quindi l'intervento umano. I fallimenti analizzati in questo lavoro sono molti e la spiegazione di alcuni di essi può essere complessa. La Tabella 1 vuole essere di aiuto per riassumere quali fallimenti sono stati trovati, per quali componenti questi possono rappresentare una criticità e darne una breve descrizione al fine di comprenderne il potenziale pericolo che rappresentano per il sistema.

Fallimenti	Componente assoggettato	Descrizione breve
Black	OBBIETTIVO	L'immagine nera è una conseguenza della rottura di un componente fondamentale dell'obiettivo, il diaframma. Il diaframma dosa la luce che attraversa le lenti dell'obiettivo e permette quindi di sfruttare tutta o soltanto in parte la loro superficie.
White		Simile a quanto detto per il fallimento Black, ma qui entra troppa luce che rende il frame completamente bianco.
Blurred		Un'immagine non a fuoco si presenta con bordi non definiti ed il soggetto non chiaro. Si possono ottenere immagini non a fuoco per vari motivi: rottura dell'auto-focus, mano non ferma, rottura generica, terreno accidentato (telecamera veicolo), ecc.
Broken Lens		In questo caso si vuole intendere la rottura di una o più lenti presenti all'interno dell'obiettivo.
Broken VR		Il risultato di una rottura dello stabilizzatore può essere un'immagine mossa. Si può intendere come un'immagine non a fuoco.
Dirty Internal		Con detriti interni si intendono tutti quei detriti che possono annidarsi (in modo molto raro data la composizione del componente) fra le lenti interne all'obiettivo o esterne.
Dirty External		Il flare dell'obiettivo si riferisce a un fenomeno in cui la luce viene diffusa in un sistema di lenti, spesso con una luce intensa, producendo un artefatto di vari colori all'interno dell'immagine.
Flare		L'immagine può presentare macchie di varia misura e forma, che possono influire sul normale funzionamento dell'interprete.
Rain		
Condensation		
Heat	OBBIETTIVO - CORPO MACCHINA	La condensa può formarsi all'interno delle lenti dell'obiettivo e nel corpo macchina, diventando una minaccia per il sistema.
Sand		Nell'obiettivo e nel corpo macchina esistono parti mobili che fanno uso di sostanze lubrificanti e materiali atti a preservare l'isolamento dall'esterno. Il calore estremo può rappresentare un pericolo.
Ice		Obiettivo e corpo macchina se non ben isolati potrebbe incorrere in problemi gravi di blocco di parti mobili o infiltrazioni di granelli di sabbia nella ghiera dello zoom o della messa a fuoco.
Water		Molti materiali esposti a temperature molto al di sotto dello zero possono rompersi in modo anomalo e far sì che ghiaccio (e quindi l'acqua) ed altri agenti esterni entrino all'interno del corpo macchina (pericolo per la parte elettronica). Un altro problema è rappresentato dalla formazione di uno strato di ghiaccio sulle lenti dell'obiettivo.
Brackish/Salt-water		L'acqua fa sì che l'elettronica smetta di funzionare, con l'impossibilità di acquisire immagini.
Wind		L'obiettivo può essere danneggiato negli stessi modi del corpo macchina. Infatti, se i materiali adoperati non sono adeguati a fronteggiare tale fenomeno, potrebbero rompersi (corrosione).
Electrical overload		Se obiettivo e corpo macchina non vengono progettati in modo da fare meno resistenza possibile all'aria si può verificare un danno nei punti in cui ci sono aperture o spiragli che permettono al vento di passarci attraverso e, se abbastanza forte, provocare danni: rottura di pezzi, spostamento di lenti, ecc.
No Bayer Filter	FILTRO DI BAYER	Si definisce sovraccarico la condizione in cui si trova un sistema complesso o un singolo componente che ha raggiunto e superato i limiti di carico previsti per il suo corretto funzionamento o utilizzo.
Spots		Se questo componente non è presente o non funzionante, l'immagine presenterà dei colori non corretti.
Banding		Le immagini acquisite presentano piccole ombreggiature di forma per lo più circolare (molecole di polvere).
Bright lines		Fenomeno che si presenta per un malfunzionamento del sensore o per una scarsa qualità di esso. Nel frame acquisito si notano linee verticali e orizzontali disposte a griglia, soprattutto sui colori più chiari.
Dead Pixel		Fenomeno che si presenta ogniqualvolta la fotocamera o videocamera viene puntata verso un laser o una fonte luminosa con una potenza tale da danneggiare irreparabilmente il sensore d'immagine.
No Action	SENZORE	In questo caso si vuole intendere la presenza di uno o più difetti (piccoli quadratini definiti pixel del sensore) che in seguito ad una rottura non ricevono tanta luce quanta dovrebbero, restando neri.
No Chromatic Aberration Correction		L'Image Signal Processor risulta bloccato e non riceve né elabora alcunché. Sistema bloccato.
No Demosaicing		Le immagini con forte aberrazione cromatica presentano elementi di sfocatura in zone di dettaglio e dei contorni non ben definiti che assumono per lo più una colorazione viola.
No Lens Distortion Correction		Il dispositivo acquisisce un'immagine in formato RAW (non ancora sottoposto a demosaicing), dunque contiene solo un valore rosso, verde o blu in ogni pixel.
No Noise Reduction		La distorsione di cui si parla si può notare in obiettivi grandangolari (effetto anche ricercato in fotografia).
No Sharpness Correction		L'effetto viene definito fisheye (occhio di pesce) per la forma anormale che l'immagine presenta.
	ISP	L'immagine acquisita non viene ripulita dal rumore dovuto a poca luminosità o scarsa qualità del dispositivo.
		La fase di correzione o applicazione della nitidezza adeguata non viene eseguita.

Tabella 1.: Tabella riassuntiva con breve descrizione dei Fallimenti analizzati

5

SIMULATORE CARLA E INIEZIONE DEI MODI DI FALLIMENTO

In questo capitolo verrà illustrato, partendo dalla descrizione del simulatore di guida autonoma CARLA, come i modi di fallimento presentati nel capitolo precedente vengono iniettati per creare un discostamento dal normale funzionamento del simulatore stesso. Per normale funzionamento si intende il modo in cui il simulatore ottiene il massimo punteggio, in termini di successi, durante le diverse run che vengono eseguite. L'obiettivo, con l'iniezione dei fallimenti, è quello di rompere l'affidabilità garantita dalla rete neurale sottostante in un ambiente incondizionato: modificando quello che la Videocamera montata sul veicolo "vede". Ogni fotogramma acquisito è stato alterato ed ha effettuato lo stesso percorso dei fotogrammi "puri" fino all'interprete incaricato di prendere la decisione del caso: svolta a destra, svolta a sinistra, fermati, riparti, frena e così via.

La ricerca sulla guida autonoma urbana è ostacolata dai costi di infrastruttura e dalle difficoltà logistiche dei sistemi di addestramento e di prova nel mondo fisico. La strumentazione e il funzionamento di una sola auto robotica richiedono fondi e manodopera esorbitanti e un singolo veicolo è lungi dall'essere sufficiente per raccogliere i dati necessari che coprono la molitudine di corner cases che devono essere elaborati sia per l'addestramento che per la convalida. La formazione e la validazione di modelli di controllo sensomotorio per la guida autonoma urbana nel mondo fisico è fuori dalla portata della maggior parte dei gruppi di ricerca [10]. Dunque, l'alternativa è quella di addestrare e validare le strategie di guida nelle simulazioni. La simulazione è necessaria anche per la verifica del sistema, poiché alcuni scenari sono troppo pericolosi per essere messi in scena nel mondo fisico: si pensi al classico bambino che corre sulla strada davanti alla macchina. La simulazione è stata utilizzata per l'addestramento dei modelli di guida sin dai primi giorni della

ricerca in ambito guida autonoma [74]. Più recentemente, i simulatori sono stati utilizzati per valutare nuovi approcci alla guida autonoma. Con le simulazioni si abbassano i costi e soprattutto i rischi che test nel mondo fisico reale potrebbero far sorgere. I rischi non sono da intendersi soltanto per chi, in quel momento, si trovasse sull'auto autonoma, ma anche e soprattutto per altri individui che si troverebbero nello stesso ambiente, magari inconsapevoli che l'auto che li sta superando non è guidata da un umano ma da un computer.

Nei prossimi paragrafi verranno trattati i seguenti argomenti:

- il Simulatore di Guida Autonoma CARLA (Paragrafo 5.1),
- la discussione della Campagna Sperimentale effettuata con la presentazione dei fallimenti iniettati (Paragrafo 5.2),
- i Risultati ottenuti (Paragrafo 5.3).

5.1 SIMULATORE CARLA

CARLA è un simulatore open-source per la ricerca in ambito guida autonoma. Questo è stato sviluppato da zero per supportare lo sviluppo, la formazione e la validazione di sistemi di guida autonoma urbani. Oltre al codice e ai protocolli open-source, CARLA fornisce risorse digitali aperte (layout urbani, edifici, veicoli) che sono state create e possono essere utilizzate liberamente. La piattaforma di simulazione supporta specifiche flessibili delle suite di sensori e delle condizioni ambientali. Viene utilizzato per studiare le prestazioni di tre approcci alla guida autonoma: modular pipeline, imitation learning e reinforcement learning. Gli approcci vengono valutati in scenari controllati di crescente difficoltà e le loro prestazioni vengono esaminate tramite metriche fornite dal simulatore stesso.

Il controllo sensomotorio in ambienti tridimensionali rimane una grande sfida nell'apprendimento automatico e nella robotica. Lo sviluppo di veicoli terrestri autonomi è un'istanziazione a lungo studiata di questo problema [74].

La piattaforma di simulazione supporta la configurazione flessibile di suite di sensori e fornisce segnali che possono essere utilizzati per addestrare strategie di guida, come coordinate GPS, velocità, accelerazione, e

dati dettagliati su collisioni e altre infrazioni.

5.1.1 *Motore - CARLA*

CARLA è implementato come livello open source su **Unreal Engine 4** (UE4) [75], consentendo future estensioni da parte della comunità. CARLA simula un mondo dinamico e fornisce una semplice interfaccia tra il mondo e un agente che interagisce con il mondo stesso. Per supportare questa funzionalità, CARLA è progettato come un sistema client-server, in cui il server esegue la simulazione ed il rendering della scena. L'API client è implementata in Python ed è responsabile dell'interazione tra l'agente autonomo e il server tramite socket. Il client invia comandi e meta-comandi al server e riceve in cambio le letture dei sensori. I comandi controllano il veicolo e includono: sterzo, accelerazione e frenata. I meta-comandi controllano il comportamento del server e vengono utilizzati per ripristinare la simulazione, modificare le proprietà dell'ambiente e modificare la suite di sensori. Le proprietà ambientali comprendono condizioni meteorologiche, illuminazione e densità di automobili e pedoni. Quando il server viene ripristinato, l'agente viene reinizializzato in una nuova posizione specificata dal client [10].

5.1.2 *Ambiente - CARLA*

L'ambiente realizzato per CARLA è composto da modelli 3D di oggetti statici e dinamici. Per statici si possono considerare gli edifici, la vegetazione, i segnali stradali (che cambiano nel tempo come i semafori oppure semplici cartelli) e le infrastrutture. Per dinamici si intendono invece tutti quegli oggetti che partecipano all'esplorazione dell'ambiente, dunque le altre auto, i veicoli a due ruote e i pedoni. Come specificati in [10] questi modelli 3D sono stati creati cercando di conciliare qualità visiva e velocità di rendering. Tutti i modelli 3D condividono una scala comune e le loro dimensioni riflettono quelle degli oggetti reali. La collezione di oggetti creati comprende:

- 40 edifici diversi,
- 16 modelli di veicoli,
- 50 modelli di pedoni.

Gli ambienti urbani disponibili con CARLA sono stati creati osservando una precisa scaletta nella loro realizzazione:

- tracciare strade e marciapiedi,
- posizionare manualmente case, vegetazione, terreno e infrastrutture,
- specificare le posizioni in cui possono apparire oggetti dinamici.

Le due città che sono state create offrono diverse possibilità: nella prima, denominata **Town1** si ha un totale di quasi 3km di strada percorribile, mentre nella seconda, denominata **Town2** il chilometraggio disponibile si abbassa a quasi la metà della prima. Oltre al veicolo prescelto per la simulazione devono essere considerati anche tutti gli altri che possono muoversi liberamente nell'ambiente: i cosiddetti **non-player vehicles**. Per questi è stato implementato un controller di base che ne regola il comportamento: seguire la corsia, rispettare i semafori, rispettare i limiti di velocità e il processo decisionale agli incroci. Veicoli e pedoni possono rilevarsi ed evitarsi l'un l'altro. Controller più avanzati potranno essere integrati in futuro [10].

I pedoni percorrono le strade secondo una mappa di navigazione specifica della città. I pedoni camminano lungo i marciapiedi e gli incroci segnalati, ma possono anche attraversare le strade in qualsiasi punto. Questi vagano per la città secondo questa mappa, evitandosi l'un l'altro e cercando di evitare i veicoli. Se un'auto si scontra con un pedone, questo viene eliminato dalla simulazione ed uno nuovo viene generato in una posizione diversa dopo un breve intervallo di tempo.

Inoltre, è possibile specificare una vasta gamma di condizioni ambientali, compreso il tempo e l'ora del giorno. Alcune di tali condizioni ambientali sono illustrate nella Fig. 22.



Figura 22.: Alcune tipologie di meteo disponibili nel simulatore CARLA [10]

5.1.3 *Sensori - CARLA*

CARLA consente una configurazione flessibile della suite di sensori dell'agente. I sensori sono limitati alle fotocamere RGB ed agli pseudo-sensori che forniscono ground-truth depth e ground-truth semantic segmentation (Fig. 23). Nella figura sottostante, a partire da sinistra, è illustrata la visione attraverso i tre sensori precedentemente nominati.

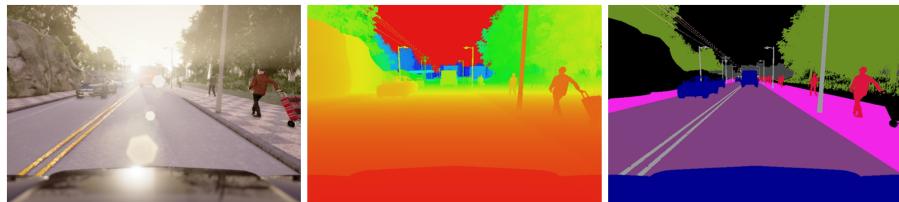


Figura 23.: Tre tipologie di sensori forniti dal simulatore CARLA [10]

Il numero di telecamere ed il loro tipo e posizione possono essere specificati dal client. I parametri della fotocamera includono la posizione 3D, l'orientamento 3D rispetto al sistema di coordinate dell'auto, al campo visivo e alla profondità di campo. Lo pseudo-sensore di semantic segmentation fornisce 12 classi semantiche:

- strada,

- segnaletica di corsia (linee),
- segnali stradali,
- marciapiede,
- recinzioni,
- pali,
- muri,
- edifici,
- vegetazione,
- veicoli,
- pedoni e altro.

5.2 CAMPAGNA Sperimentale

Di seguito verrà presentato il lavoro sperimentale eseguito sul simulatore CARLA. In 5.2.1 sarà descritto l'approccio utilizzato per andare a valutare ogni scenario creato. Dopodiché verranno presentati e discussi i risultati ottenuti, utilizzando dei grafici per una migliore comprensione.

5.2.1 CARLA e Learning By Cheating Benchmarking

La guida urbana basata sulla visione artificiale è difficile. Il sistema autonomo deve imparare a percepire il mondo e ad agire in esso. In [23] viene mostrato come il precedente problema di apprendimento può essere semplificato scomponendolo in due fasi:

- in una viene formato un agente che ha accesso a informazioni privilegiate (cheater agent o **agente privilegiato**),
- nell'altra l'agente privilegiato agisce come un insegnante che forma un **agente non privilegiato** (sensorimotor agent) basato esclusivamente sulla visione artificiale.

Il primo agente può osservare direttamente il layout dell'ambiente e le posizioni degli altri partecipanti al traffico, ovvero sapere tutto su tutti in ogni momento. Il secondo agente nominato, invece, non imbroglia e prende tutte le decisioni sulla base di quello che vede e percepisce. Questo viene addestrato per imitare l'agente privilegiato. Dunque, il secondo utilizza solo input visivi da sensori legittimi (una singola fotocamera rivolta in avanti), senza alcuna informazione privilegiata.

Nonostante i notevoli progressi, l'apprendimento della guida urbana basata sulla visione per imitazione rimane difficile e recenti studi hanno dimostrato che non produce politiche di guida soddisfacenti. Dunque, perché viene usato proprio questo in [23]? Se l'Imitation Learning diretto (dalle traiettorie di esperti alla guida basata sulla visione) è difficile, perché la divisione del processo di apprendimento in due fasi, in cui entrambe eseguono Imitation Learning, è migliore? La domanda trova presto risposta. L'efficacia della decomposizione in fasi è controtuitiva. L'apprendimento sensomotorio diretto unisce due compiti difficili: imparare a vedere ed imparare ad agire. In [23] si ha che per l'agente privilegiato, nella prima fase, la percezione viene risolta fornendo l'accesso diretto allo stato dell'ambiente e l'agente può quindi concentrarsi sull'*apprendimento dell'azione*. Mentre, nella seconda fase, l'agente privilegiato agisce come "insegnante" e fornisce una ricca supervisione allo "studente" sensomotorio, la cui responsabilità primaria è *imparare a vedere*. I vantaggi che sono stati riscontrati in [23] sono:

- l'agente privilegiato opera su una rappresentazione intermedia compatta dell'ambiente (vista dall'alto) e può quindi apprendere più velocemente e generalizzare meglio,
- l'agente privilegiato addestrato può fornire una supervisione molto più forte rispetto alle traiettorie estrapolate dalla guida di esperti (può essere interrogato da qualsiasi stato dell'ambiente, non solo dagli stati visitati nelle traiettorie originali),
- l'agente privilegiato prodotto nella prima fase è una "white box", nel senso che il suo stato interno può essere esaminato a piacimento (se l'agente privilegiato viene addestrato tramite Imitation Learning condizionale, può fornire un'azione per ogni possibile comando nella seconda fase, in qualsiasi stato dell'ambiente).

Riguardo all'ultimo punto, si può dire quindi che, tutti i rami condizionali dell'agente privilegiato possono addestrare tutti i rami dell'agente sensomotorio in parallelo, ovvero in ogni stato visitato durante l'allenamento, lo "studente" sensomotorio può in effetti chiedere all'"insegnante" privilegiato domande del tipo "Cosa faresti se dovessi girare a sinistra qui?", "Cosa faresti se dovessi girare a destra qui?" ecc. Questo viene definito come data augmentation e un segnale di apprendimento ad alta capacità [23].

5.2.1.1 CoRL2017

Il benchmark utilizzato nel lavoro di tesi è chiamato CoRL2017, anche definito come benchmark CARLA originale. Ad ogni frame, gli agenti ricevono un'immagine RGB, una velocità e un comando di alto livello per calcolare lo sterzo, l'acceleratore e il freno, al fine di raggiungere gli obiettivi specificati. Gli agenti vengono valutati in un contesto di guida urbana, con incroci e semafori. Il benchmark CoRL2017 è costituito da quattro condizioni di guida, ciascuna con 25 rotte di navigazione predefinite. Le quattro condizioni di guida sono: guida dritta, guida con una svolta, navigazione completa con più svolte e le stesse rotte ma con traffico. Questi scenari sono chiamati:

- FullTown02 → navigazione completa con più svolte in Town02,
- StraightTown02 → guida dritta in Town02,
- TurnTown02 → guida con una svolta in Town02 (5.1.2).

Nell'approccio sperimentale scelto, ogni run eseguita conta 30 pedoni e 50 veicoli che vanno ad arricchire e complicare l'ambiente in cui il veicolo a guida autonoma, del quale si sta testando la robustezza del sistema di visione, deve muoversi verso l'obiettivo.

Una prova su una determinata rotta è considerata riuscita se l'agente raggiunge l'obiettivo entro un certo limite di tempo. Il limite di tempo corrisponde alla quantità di tempo necessaria per completare il percorso ad una velocità di crociera di 10 km/h. Nel benchmark CoRL2017, le collisioni e le violazioni dei semafori rossi non vengono considerati fallimenti. Una modifica apportata a questo benchmark è stata quella di

introdurre il fallimento della run in corso qualora il veicolo fosse andato a collidere con altri veicoli o pedoni, oppure con oggetti appartenenti all'ambiente esterno. Questo cambiamento nel codice, nei casi in cui i fallimenti introdotti producevano una sorta di cecità dell'apparecchio di cattura, ha fatto risparmiare, come vedremo in 5.2.2, molto tempo di simulazione, invalidando subito la run.

Nella campagna sperimentale condotta su questo benchmark sono state considerate 3 condizioni meteorologiche sulle 15 offerte dal simulatore CARLA [76]. Queste sono:

- 1 - ClearNoon → Mezzogiorno Sereno,
- 4 - WetCloudyNoon → Mezzogiorno Nuvoloso Bagnato,
- 13 - HardRainSunset → Pioggia Forte al Tramonto.

Inoltre, la versione del simulatore CARLA utilizzata è la 0.9.6, non paragonabile con le precedenti ($\leq 0.9.5$), in quanto ha subito una revisione significativa aggiornando il motore di rendering e la logica pedonale [23].

5.2.2 *Modi di Fallimento Iniettati*

La campagna sperimentale è stata svolta in due fasi:

- la prima, eseguendo una Golden Run, che produce dei dati "puri", ovvero senza introdurre alcun tipo di modifica nei fotogrammi catturati dalla videocamera del simulatore,
- la seconda, che si suddivide in più parti, nella quale si eseguono lo stesso numero di run della prima fase, ma introducendo delle modifiche più o meno consistenti ad ogni fotogramma acquisito, in modo da simulare il fallimento che si vuole ottenere.

I codici utilizzati per alterare le immagini sono riportati nell'Appendice A e scaricabili da [22].

Ogni simulazione che è stata eseguita sul simulatore presenta le stesse caratteristiche, quali: numero di pedoni, numero di veicoli, le tre tipologie

di meteo sopra descritte e lo stesso numero di run avviate (50). Il numero 50 è stato scelto in quanto abbastanza grande da poter mostrare un'evidenza statistica nei dati che si ottengono e non è estremamente elevato, in modo da consentire l'esecuzione di tutte le run della campagna in un tempo ragionevole.

I modi di fallimento iniettati non si fermano soltanto a quelli che sono i modi di fallimento evidenziati nella FMECA (Appendice B), ma considerano anche diverse intesità o configurazioni che ogni fallimento può o meno assumere. Ad esempio, come sarà illustrato più avanti, quando si è considerato il fallimento WHITE, non è stato tenuto conto soltanto di cosa succede quando la telecamera registra fotogrammi tutti bianchi (quindi privi di informazione), ma anche cosa succede quando la luminosità del fotogramma viene settata a diversi livelli di intensità, fino ad arrivare al fotogramma totalmente bianco. Naturalmente, tutti i dati raccolti su un fallimento e le sue possibili altre configurazioni saranno analizzati insieme oltre che separatamente.

Golden Run

La Golden Run, come detto, rappresenta la fase sperimentale nella quale sono state eseguite 50 run sul simulatore, nelle quali non si è influenzato in alcun modo quello che la telecamera del veicolo a guida autonoma percepiva. Questo è stato fatto perché serviva un insieme di dati non corrotti da modifiche effettuate sui fotogrammi acquisiti. I dati raccolti in questa fase saranno considerati in un confronto successivo, dove avremo: un "prima dell'iniezione del fallimento" e un "dopo l'iniezione del fallimento". Questo ci permetterà di capire quanto bene sia stato allenato l'agente ed inoltre qual è il limite per cui questo fallisce. Oltre a questo, si potranno evidenziare quali (e con quale intensità o configurazioni) sono i fallimenti che causano un success rate più basso.

Blurred

Il fallimento Blurred consiste nello sfocare l'immagine acquisita per far sì che venga simulato un guasto all'interno dell'obiettivo. Il guasto, come descritto in 4.2.2.1, presuppone che ci sia un problema nella messa a fuoco.

Il codice Python utilizzato per simulare questo comportamento da parte dell'obiettivo è A.9. Non è un codice molto complesso. Quello che fa è andare a leggere l'immagine e aggiungere a questa una sfocatura che può avere varie intensità. I valori possono essere cambiati a piacimento, ma per la campagna sperimentale si è deciso che questi fossero uguali a 12. Il risultato del codice sopra descritto, applicato al primo fotogramma della simulazione, è in Fig. 24.

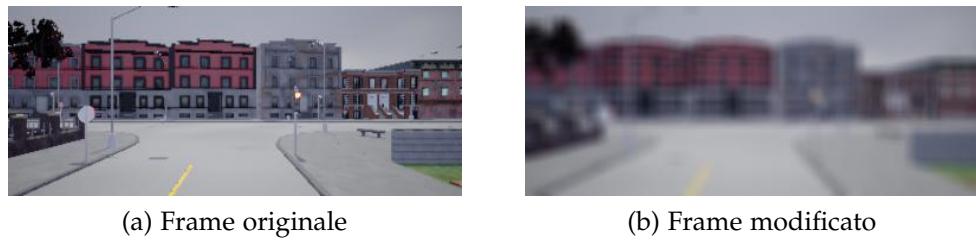


Figura 24.: Fallimento Blurred nella simulazione

Brightness - White

Questo tipo di fallimento è stato esteso, in quanto inizialmente si è preso in considerazione soltanto il fotogramma totalmente bianco. Il problema che si considera nella descrizione del fallimento White (4.2.2.1) fa riferimento all'immagine che viene acquisita bianca per via di un errore interno dell'obiettivo (diaframma). Dunque, nella campagna sperimentale, sono state aggiunte delle configurazioni che partissero da una luminosità aumentata di poco ad una che arrivasse al cosiddetto fotogramma bianco.

Il codice utilizzato per simularlo lo sviluppo di questo fallimento nelle varie configurazioni mostrate in Fig. 25 è A.10 per le due configurazioni di luminosità aumentata e A.13 per la trasformazione del fotogramma corretto in uno totalmente bianco.

In Fig. 25 si può notare come partendo da un fotogramma privo di modifiche (Fig. 25a) si passa ai fotogrammi con luminosità aumentata (Fig. 25b e Fig. 25c) per arrivare a quello del frame totalmente bianco (Fig. 25d).

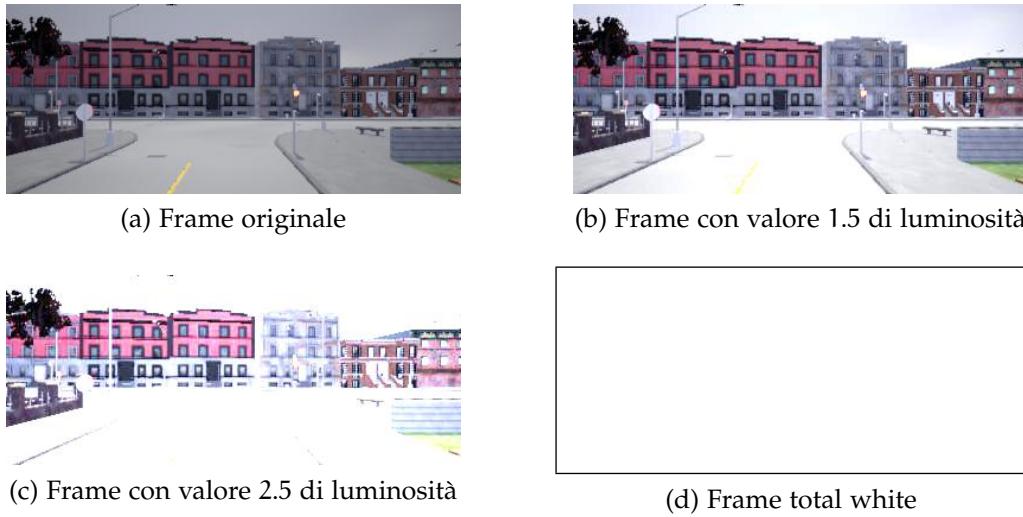


Figura 25.: Configurazioni del fallimento White nella simulazione

Black

Questo fallimento è del tutto uguale al precedente, in quanto, pur non considerando lo stesso numero di configurazioni, il codice che si utilizza per simularlo è lo stesso (A.13). Infatti, l'unica differenza fra il codice del fotogramma totalmente bianco e quello del fotogramma totalmente nero è il colore che si dà ai pixel nella scansione dell'immagine. Nel precedente fallimento i pixel venivano tutti settati a bianco, mentre in questo vengono tutti impostati a nero.

Come detto per White, anche in questo si presuppone che il guasto sia nei meccanismi dell'obiettivo che permettono di catturare più o meno luce a seconda delle condizioni. In 4.2.2.1 è descritto in modo più approfondito di cosa si tratta. Nella Fig. 26 è illustrata la pesante modifica che il fotogramma subisce ogni volta che viene catturato. Questo frame verrà poi elaborato dopo l'acquisizione per far sì che il computer centrale dell'auto possa prendere decisioni riguardo la marcia del veicolo.

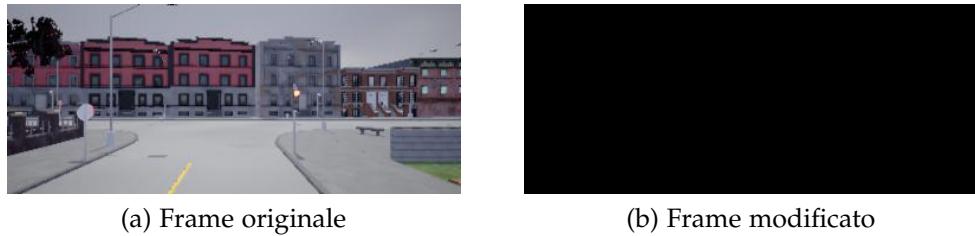


Figura 26.: Fallimento Black nella simulazione

Broken Lens

Per questo fallimento si è preso in considerazione il fatto in cui la lente esterna venga colpita da un corpo estraneo e finisca per rompersi. Sono state considerate 2 configurazioni, entrambe rappresentanti la segnatura, più o meno calcata, di una superficie scalabile come il vetro. Il codice utilizzato per entrambe le configurazioni è A.3, in cui si sovrappone al fotogramma originale un'immagine rappresentante, in questo caso, un vetro danneggiato. Nel codice, oltre ad impostare l'immagine da sovrapporre, si può anche definire con che valore l'una deve prevalere sull'altra. Tramite il codice **Image.blend(img,img2,0.5)** si definiscono le due immagini da combinare ed il valore di prevalenza o trasparenza che ci deve essere fra queste.

I risultati della prima e della seconda configurazione sono osservabili in Fig. 27 ed in Fig. 28.

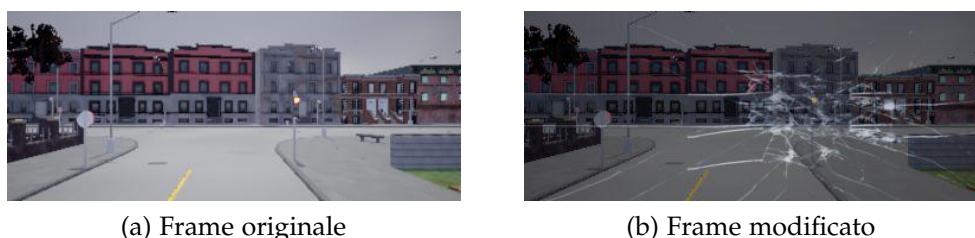


Figura 27.: Fallimento Broken Lens - configurazione 1 nella simulazione



Figura 28.: Fallimento Broken Lens - configurazione 2 nella simulazione

Dirty Int. - Dirty Ext.

Questi due fallimenti, seppur con nomi diversi, possono essere simulati insieme, dato che il risultato finale nel frame modificato sarebbe pressoché lo stesso. Infatti, come descritto in 4.2.2.1, si prende in considerazione il caso in cui ci siano detriti o particelle di qualsiasi genere che causano degli artefatti sul fotogramma acquisito. Non c'è distinzione fra interno ed esterno in quanto il risultato sul frame, come detto, risulterebbe uguale. La distinzione è da tener presente quando, in sede di FMECA, se ne descrivono le mitigazioni possibili.

Anche per questo fallimento il codice utilizzato è A.3, dato che si vanno a posizionare immagini che simulano dei residui o aloni su delle superfici trasparenti, come appunto può essere considerata quella della lente esterna. Le configurazioni sono 2 e si possono osservare in Fig. 29 e in Fig. 30.

Per la prima configurazione presentata, si è voluto lasciare il fotogramma scurito dalla sovrapposizione dell'immagine scelta per il fallimento in questione, senza ritoccare la luminosità di questo. Per quanto riguarda la seconda, le due immagini unite hanno generato un frame che presenta una luminosità adeguata alla simulazione, quindi né troppo scura né troppo chiara.



Figura 29.: Fallimento Dirty Int. - Dirty Ext. configurazione 1 nella simulazione



Figura 30.: Fallimento Dirty Int. - Dirty Ext. configurazione 2 nella simulazione

Rain

In questo fallimento si simula la pioggia che può depositarsi sulla lente esterna dell'obiettivo, andando a confondere l'elaboratore centrale in fase di decisione. Il fallimento è descritto più approfonditamente in 4.2.2.1. Per questo si è considerato una sola configurazione, in quanto considerarne di più sarebbe stato molto difficile, sia per l'assenza di una grande varietà di immagini da sovrapporre al frame corretto sia per il fatto che il risultato finale avrebbe sempre lo stesso aspetto. In più, alcuni scenari, non possono essere simulati efficacemente in CARLA: fari che illuminano le gocce di pioggia sulla lente quando si incontra un altro veicolo di notte; la possibilità di inserire insegne luminose la cui luminosità può essere amplificata dalla gocce sulla lente; gli stessi semafori che, nella realtà di oggi, a volte, risultano molto più luminosi di altri ed anche in condizioni normali possono dare noia all'occhio umano ecc.

In Fig. 31 viene rappresentato questo fallimento, sempre utilizzando il codice A.3.



Figura 31.: Fallimento Rain nella simulazione

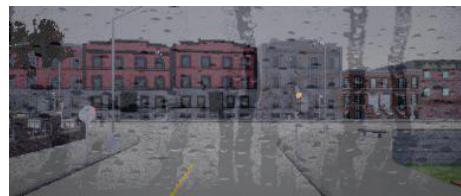
Condensation

Questo fallimento ha le stesse caratteristiche del precedente, in quanto si parla sempre di residui di acqua sulla lente esterna. In questo caso la

causa è definita condensa e in 4.2.2.1 viene meglio approfondita. Anche qui si è considerata una sola configurazione, sempre per le motivazioni precedentemente espresse per il fallimento Rain. Il codice utilizzato per simulare questa situazione è sempre A.3, con il quale si sovrappone un'immagine rappresentante della condensa depositatasi su una superficie trasparente al frame originale privo di artefatti. Il risultato è visibile in Fig. 32.



(a) Frame originale



(b) Frame modificato

Figura 32.: Fallimento Condensation nella simulazione

Ice

Il fallimento Ice viene rappresentato sovrapponendo, mediante il codice A.3, l'immagine di ghiaccio secco attaccato ad una superficie trasparente, al fotogramma originale. Il fallimento viene meglio descritto in 4.2.2.1, anche se è ben chiaro, guardando le figure sottostanti, di che tipo di scenario si tratta.

Per questo si sono considerate 2 configurazioni, in quanto per questa casistica le immagini da sovrapporre sono più numerose di altre tipologie di fallimenti. Nella prima configurazione è stata scelta un'immagine che provasse ad impedire una corretta lettura di segnali stradali o segnaletica in generale. Infatti, in Fig. 33 si vede come l'immagine di disturbo sopra il frame originale sia scostata verso destra. Naturalmente, va precisato che, se prendessimo in considerazione tutte le casistiche, dovremmo testare anche il caso in cui l'immagine copra la parte sinistra del frame, in quanto, in paesi come Regno Unito o Giappone, la guida è a destra. Tuttavia, questo può essere preso in considerazione per un futuro sviluppo di questo elaborato.



Figura 33.: Fallimento Ice - configurazione 1 nella simulazione

Per la seconda configurazione si è invece optato per un’immagine che fosse posta al centro del frame. Si può osservare questo in Fig. 34.

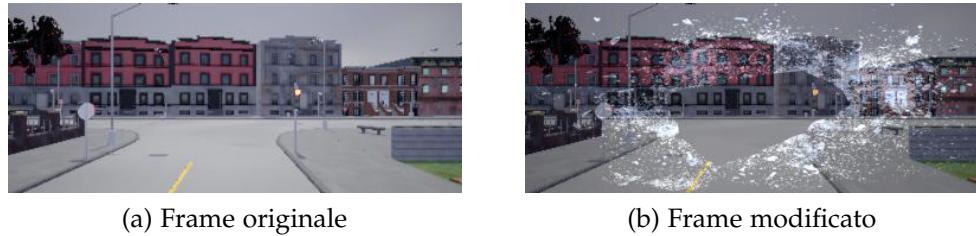


Figura 34.: Fallimento Ice - configurazione 2 nella simulazione

No Bayer Filter - Greyscale

Questa tipologia di fallimento riguarda il Filtro di Bayer. La descrizione approfondita è in 4.2.2.3. Le possibilità di un fallimento del genere sono remote, ma verificabili. In particolare, per la simulazione di questo, si presuppone che il filtro non ci sia, piuttosto che questo sia danneggiato. Infatti, viene convertita l’immagine da RGB ad una che presenti soltanto una scala di grigi. Sarebbe questo, quindi, l’effetto che si otterrebbe se non fosse presente il Filtro di Bayer. La sua caratteristica è quella di raggruppare i sensori per i tre colori fondamentali necessari per la sintesi additiva (rosso, verde e blu) in celle di due fotositi per due. Lo schema Bayer prevede che nelle otto celle adiacenti ad ogni fotosito, ve ne siano almeno due di ognuno degli altri colori. Quindi rende possibile ricostruire il valore della luminosità del rosso, ad esempio, in corrispondenza di un elemento verde o blu, deducendolo dagli elementi rossi circostanti [39].

Tralasciando la descrizione approfondita di questo, già trattata preceden-

temente, il risultato, ottenuto mediante il codice A.11, è rappresentato in Fig. 35.



(a) Frame originale

(b) Frame modificato

Figura 35.: Fallimento No Bayer Filter nella simulazione

Dead Pixel

Il fallimento Dead Pixel (4.2.2.4) prende in considerazione il caso in cui uno o più pixel risultano neri nel fotogramma acquisito. Per questo fallimento sono state create molte configurazioni: le prime presenteranno il caso in cui 1, 50, 200 o 1000 pixel, tutti separati l'uno dall'altro, risultano neri nel fotogramma acquisito e le altre terranno conto della possibilità che i pixel falliti siano adiacenti, ovvero a formare delle righe. Nelle ultime configurazioni considerate sono state create delle righe di pixel neri verticali e/o orizzontali, andando a comporre delle "figure" di disturbo sul frame originale in ingresso. In Fig. 36 si può osservare come, nel frame modificato, sia stato posizionato un singolo pixel nero, circa al centro della carreggiata. Il codice utilizzato per simulare questo fallimento è A.4.



(a) Frame originale

(b) Frame modificato

Figura 36.: Fallimento Dead Pixel - configurazione 1 nella simulazione

Per valutare gli effetti di questo tipo di fallimento, come detto, sono state considerate diverse configurazioni, nelle quali si va ad aumentare il numero di pixel neri in ogni frame che il dispositivo di cattura acquisisce. Nella Fig. 37 viene mostrato come il frame modificato presenti 50

pixel neri, disposti come se fossero fissati su una griglia immaginaria sottostante. Il codice utilizzato per simulare questo fallimento è A.5.



Figura 37.: Fallimento Dead Pixel - configurazione 2 nella simulazione

Nella figura sopra si è considerata una griglia di 10 pixel neri in orizzontale per 5 pixel neri in verticale.

Un'altra configurazione testata nella campagna sperimentale è quella dove vengono aggiunti, ad ogni frame acquisito, 200 pixel neri. Questi, come negli esempi precedenti, non sono disposti in modo randomico, ma sempre a griglia. La dimensione della griglia di pixel in questa configurazione è di 20 pixel neri orizzontali per 10 pixel neri verticali (Fig. 38). Il codice utilizzato per la simulazione dei 200 pixel neri nel fotogramma è A.6. Questo script ha la stessa logica del precedente, in quanto la funzione svolta è la medesima, escluso il numero maggiore di righe e colonne su cui vengono posizionati i pixel falliti.

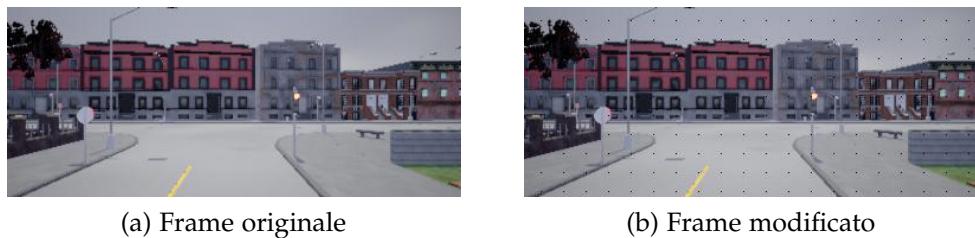


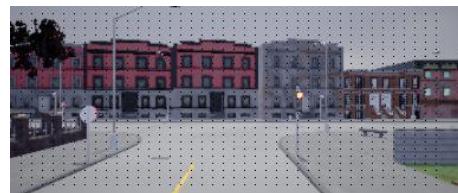
Figura 38.: Fallimento Dead Pixel - configurazione 3 nella simulazione

L'ultima configurazione che considera singoli pixel neri sul frame è quella in Fig. 39. In questa sono stati aggiunti 1000 pixel neri, tutti separati fra loro e sempre disposti su una griglia immaginaria sottostante. Il numero totale di 1000 è stato ottenuto andando a comporre una griglia di 40 pixel

neri orizzontali per 25 pixel neri verticali. Come in precedenza, il codice per l'iniezione di questo fallimento è pressoché lo stesso dei 2 passati, ovvero cambia il numero di righe e colonne dove si collocano i pixel neri (A.7).



(a) Frame originale

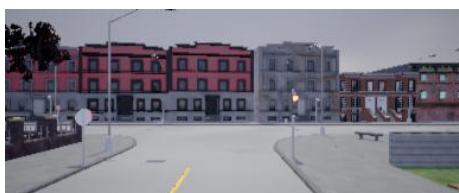


(b) Frame modificato

Figura 39.: Fallimento Dead Pixel - configurazione 4 nella simulazione

Se fino ad adesso sono state presentate configurazioni nelle quali ogni pixel era separato da ogni altro, nei prossimi scenari, come anticipato, si considerano delle righe di pixel neri, verticali e/o orizzontali. Nelle prossime configurazioni quindi si hanno più pixel neri adiacenti, disposti in tutto il frame.

La configurazione mostrata in Fig. 40 rappresenta il fallimento in cui si viene a creare un riga centrale verticale di pixel neri adiacenti. Naturalmente, la posizione di tale riga è stata scelta in modo da valutare se questa potesse o meno influire sulla normale attività del veicolo in marcia.



(a) Frame originale



(b) Frame modificato

Figura 40.: Fallimento Dead Pixel - configurazione 5 nella simulazione

Il codice esempio per la creazione di questo fallimento è A.8.

Da una singola riga verticale, si passa alla configurazione nella quale se

ne considerano 3: due orizzontali di diversa lunghezza ed una verticale (Fig. 41). Tale fallimento può essere ricostruito attraverso il codice in A.8.



Figura 41.: Fallimento Dead Pixel - configurazione 6 nella simulazione

Il prossimo scenario mostrato rappresenta il caso in cui siano presenti 5 linee di pixel neri su ogni frame acquisito. Per coerenza, rispetto ai numeri della precedente configurazione, in questa vengono considerate 3 linee orizzontali e 2 linee verticali. Le linee di pixel aggiunte al frame da codice hanno diversa lunghezza e sono separate le une dalle altre, come mostra la Fig. 42. Parte del codice utilizzato per la realizzazione di questo fallimento è in A.8.



Figura 42.: Fallimento Dead Pixel - configurazione 7 nella simulazione

Altra configurazione considerata è rappresentata in Fig. 43 , nella quale si hanno ben 10 linee, 5 orizzontali e 5 verticali. Le linee inserite in questa hanno coordinate tali da farle rimanere tutte separate fra loro nello spazio del frame. Parte del codice utilizzato per la creazione di tale fallimento è in A.8.



(a) Frame originale



(b) Frame modificato

Figura 43.: Fallimento Dead Pixel - configurazione 8 nella simulazione

Le ultime due configurazioni prese in considerazione vogliono tentare di trarre in inganno il simulatore. Infatti, in Fig. 44 è osservabile come le due linee di pixel neri aggiunte al fotogramma siano oblique e posizionate in modo da ricalcare la corsia dove il veicolo sta viaggiando. L'obiettivo è valutare se il simulatore effettivamente le interpreta come corsia da tenere, ovvero come se il veicolo avesse sempre strada dritta davanti a sé, oppure se, pur tenendo conto di esse, esegue il suo percorso di benchmark senza prestargli troppa attenzione. Parte del codice utilizzato per realizzare tale fallimento è in A.8.



(a) Frame originale



(b) Frame modificato

Figura 44.: Fallimento Dead Pixel - configurazione 9 nella simulazione

L'ultima configurazione considerata è quella in cui, oltre alla carreggiata dritta "disegnata" su ogni frame, si pone, al centro della strada davanti al veicolo, un blocco di pixel neri come a simulare un ostacolo da evitare. Evitare l'ostacolo in questo caso è inteso come fermarsi oppure provare a scansarlo. Naturalmente, se il veicolo provasse a scansare l'ostacolo introdotto tramite pixel neri, non vedrebbe gioventù nella continuazione della propria marcia, in quanto il blocchetto nero rimarrebbe fermo dov'è, essendo un guasto interno al dispositivo. In Fig. 45 è illustrata l'ultima configurazione creata per il fallimento Dead Pixel. La realizzazione

dell'ostacolo centrale e delle linee oblique trova un'esemplificazione nel codice A.8.

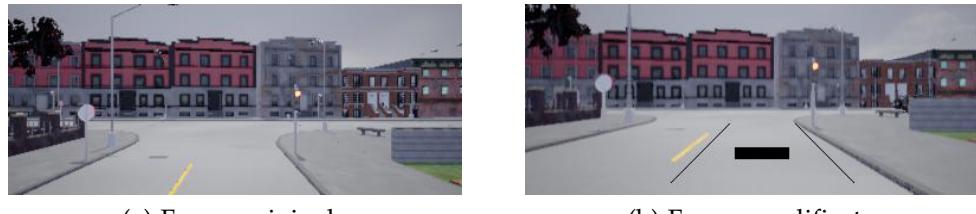


Figura 45.: Fallimento Dead Pixel - configurazione 10 nella simulazione

Banding

Il fallimento Banding è descritto più approfonditamente in 4.2.2.4. La simulazione di questo fallimento è stata introdotta utilizzando il codice A.3, andando a sovrapporre al fotogramma originale, appena acquisito, un'immagine rappresentante una griglia trasparente. In Fig. 46 si può infatti notare come si sia provato a ricreare l'effetto mostrato in 4.2.2.4.

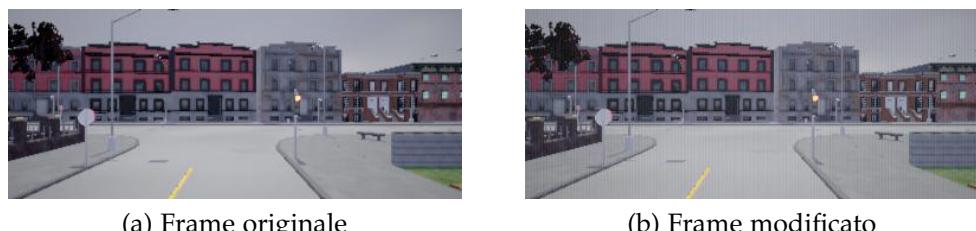


Figura 46.: Fallimento Banding nella simulazione

No Demosaicing

Il fallimento mostrato in Fig. 47 rappresenta il caso in cui l'immagine rimanga grezza (o RAW), ovvero l'impossibilità di ricostruire, grazie a tecniche di interpolazione, tutti i colori dell'immagine. Questa operazione viene spesso effettuata dal firmware stesso della macchina fotografica digitale, se si scatta in JPG. Se si scatta in JPG, infatti, nell'immagine risultante è già stato compiuto il passo di demosaicizzazione. Il formato JPG è ideale per le fotografie, anche se è importante notare che vi è una leggera perdita di qualità dovuta alla compressione. Il fallimento in questione viene trattato in 4.2.2.5.

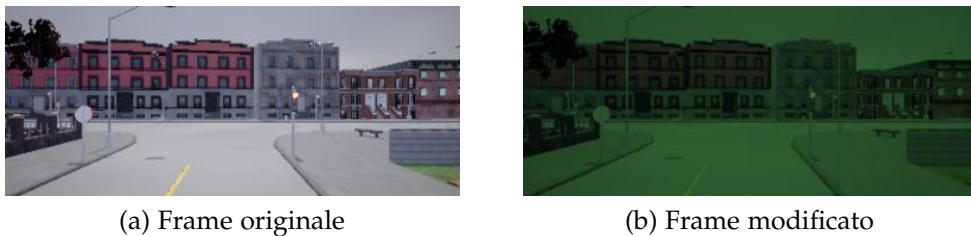


Figura 47.: Fallimento No Demosaicing nella simulazione

La Fig. 47b mostra come nel frame ci sia una prevalenza del colore verde, data dalla composizione del Filtro di Bayer. Nel codice utilizzato per modificare il fotogramma è stata utilizzata una funzione di ridimensionamento dell'immagine (A.2). Questo è stato necessario in quanto il codice inizialmente utilizzato produceva un errore nel simulatore. Infatti, in questo, venivano raddoppiate le dimensioni dell'immagine originale, cosa che il simulatore non poteva gestire. La modifica apportata al codice originario ha prodotto, però, l'effetto di non poter più distinguere tutti i pixel di colori diversi (rosso, verde e blu) nell'immagine risultante. La griglia di Bayer non è più visibile, ma l'immagine risultante è a prevalenza di verde, cosa che si doveva ottenere con questo tipo di trasformazione.

Se non si utilizzasse la funzione per il ridimensionamento del frame, l'immagine finale sarebbe come in Fig. 48.

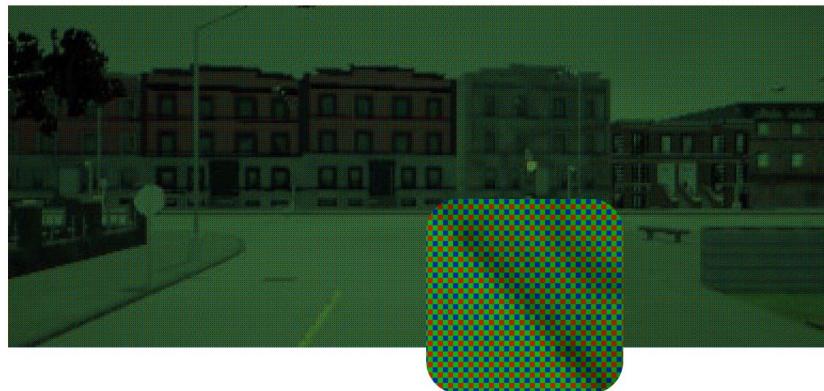


Figura 48.: Fallimento No Demosaicing senza ridimensionamento del frame

No Noise Reduction

Questo fallimento, come descritto in 4.2.2.5, prende in considerazione la casistica in cui la fase di riduzione del disturbo, che può essere adoperata per ogni fotogramma acquisito, non funzionasse, creando quindi immagini poco chiare. Per questo fallimento sono state definite due configurazioni: la prima, nella quale si introduce un rumore di una certa intensità e la seconda, nella quale si riproduce un valore di disturbo più elevato. In Fig. 49 è presentata la prima configurazione.



Figura 49.: Fallimento No Noise Reduction - configurazione 1 nella simulazione

In Fig. 50 si mostra come appare il fotogramma acquisito, sottoposto a introduzione del rumore.

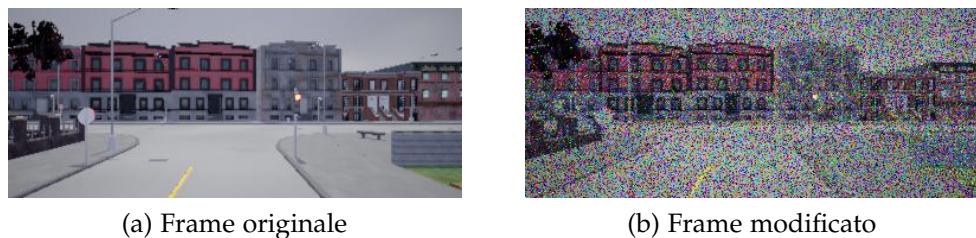


Figura 50.: Fallimento No Noise Reduction - configurazione 2 nella simulazione

Il codice utilizzato per la creazione di questi fallimenti è A.1.

No Sharpness Correction

Come per il fallimento precedente, anche in questo, si è cercato di ricreare una condizione di errore in una delle fasi svolte dall'ISP, precisamente la fase di correzione della nitidezza. Una descrizione più dettagliata di

questo fallimento è in 4.2.2.5.

Nella Fig. 51 viene mostrato come risulta il fotogramma modificato dopo l'utilizzo del codice A.12.

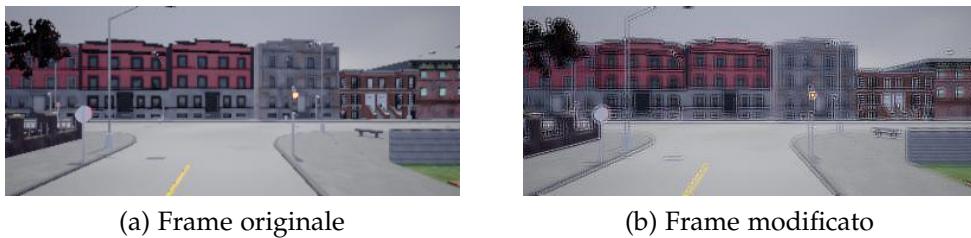


Figura 51.: Fallimento No Sharpness Correction nella simulazione

No Chromatic Aberration Correction

Questo tipo di fallimento è comune a molti dispositivi utilizzati per l'acquisizione delle immagini. Con questo, i frame presentano diversi problemi di colorazione e di definizione dei confini. Una descrizione più accurata è fornita in 4.2.2.5 dove si descrive il fallimento nei dettagli.

In campagna sperimentale si sono scelte due configurazioni: la differenza fra le due sta nel fatto che in una viene aggiunto il cosiddetto blur al fotogramma (la sfocatura), mentre nell'altra no. In Fig. 52 si mostra come risulta il fotogramma dato in input al codice A.14 senza aggiunta di sfocatura.

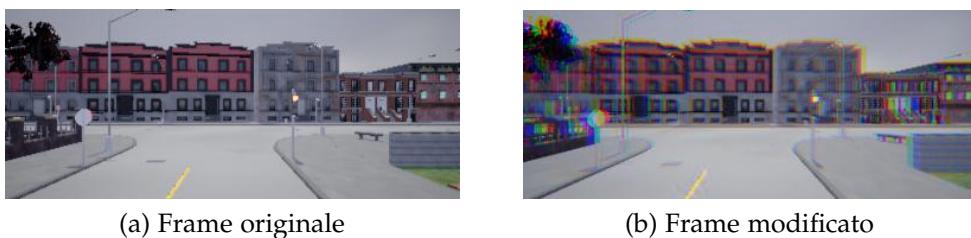


Figura 52.: Fallimento No Chromatic Aberration Correction - configurazione 1 nella simulazione

In Fig. 53, invece, viene mostrata la seconda configurazione, dove il fotogramma viene elaborato dallo stesso codice A.14, ma in questo caso viene effettuata anche la sfocatura.

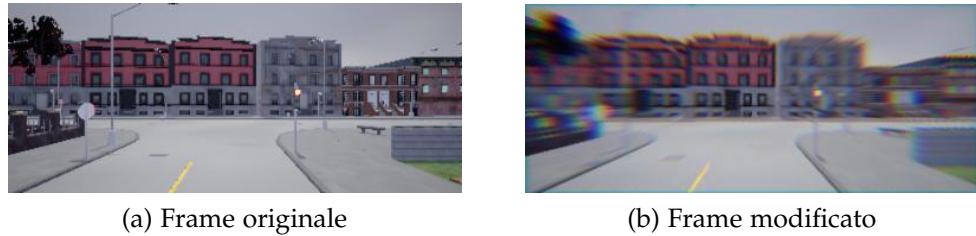


Figura 53.: Fallimento No Chromatic Aberration Correction - configurazione 2 nella simulazione

La differenza, in termini di codice, sta nella riga sottostante:

```
im = add_chromatic(im, strength=2, no_blur=True).
```

In questa con **strength** si imposta l'intensità dell'effetto di aberrazione cromatica, mentre con **no_blur** si decide se aggiungere o meno la sfocatura. Questo metodo è invocato ogni volta che un nuovo fotogramma viene acquisito. La sfocatura inserita non è la stessa che viene aggiunta al fotogramma mediante il codice A.9, precedentemente descritto per l'iniezione del fallimento Blurred nel simulatore.

In Tabella 2 sono riportati i dettagli inerenti l'implementazione dei diversi fallimenti simulati.

Fallimenti	Dettagli implementazione	Configurazioni
Banding	viene utilizzato il Modulo PIL Image [77] e le immagini in [22]	Banding: vengono utilizzati i metodi <i>resize()</i> e <i>blend()</i> per sovrapporre una immagine rappresentante il fenomeno del banding al frame originale
Black	viene impostato ogni pixel con colore nero, utilizzando il Modulo PIL Image [77]	Black: viene usato il metodo <i>load()</i> di PIL per accedere ad ogni pixel e modificarne il colore in nero
White (brightness)	l'aumento di luminosità del frame originale viene introdotto usando i Moduli PIL Image e ImageEnhance [77]	Brightness1: l'aumento di luminosità viene creato usando il metodo <i>ImageEnhance.Brightness(img)</i> con un fattore di 1.5, Brightness2: l'aumento di luminosità viene creato usando il metodo <i>ImageEnhance.Brightness(img)</i> con un fattore di 2.5, White: si usa lo stesso metodo <i>load()</i> del fallimento Black per impostare tutti i pixel di colore bianco
Blurred (and Broken VR)	la sfocatura (blur) viene introdotta utilizzando cv2 [78]	Blurred: viene utilizzato il metodo <i>cv2.blur(img,(12,12))</i>
Broken Lens	ognuno di questi fallimenti è stato simulato utilizzando il Modulo PIL Image [77] e immagini differenti di vetri rotti, sporco, gocce di acqua, condensa, ghiaccio, tutte con uno sfondo trasparente. Le immagini utilizzate per la sovrapposizione sono disponibili in [22].	Broken-Lens1, Broken-Lens2: si usano i metodi <i>resize()</i> e <i>blend()</i> di PIL per sovrapporre due differenti immagini di lenti rotte (vetro infranto con sfondo trasparente)
Condensation		Condensation: si usano i metodi <i>resize()</i> e <i>blend()</i> di PIL per sovrapporre un'immagine rappresentante della condensa su sfondo trasparente
Dirty Internal / Dirty External (e Spots)		Dirty1, Dirty2: si usano i metodi <i>resize()</i> e <i>blend()</i> di PIL per sovrapporre due differenti immagini rappresentanti sporco di vario genere su sfondo trasparente
Ice		Ice1, Ice2: si usano i metodi <i>resize()</i> e <i>blend()</i> di PIL per sovrapporre due immagini rappresentanti ghiaccio su sfondo trasparente
Rain		Rain: si usano i metodi <i>resize()</i> e <i>blend()</i> di PIL per sovrapporre un'immagine rappresentante gocce d'acqua (pioggia) su sfondo trasparente
Dead Pixel	ognuno dei fallimenti DeadPixel viene creato utilizzando cv2 [78] per modificare il colore dei pixel desiderati	DeadPixel1: viene inserito un singolo pixel nero in basso a destra nel frame DeadPixel50: vengono inseriti 50 pixel neri disposti a griglia (5 verticali x 10 orizzontali) DeadPixel200: vengono inseriti 200 pixel neri disposti a griglia (10 verticali x 20 orizzontali) DeadPixel1000: vengono inseriti 1000 pixel neri disposti a griglia (25 verticali x 40 orizzontali) DeadPixel-vertical-central-line: viene inserita un'unica linea verticale al centro del frame, composta da pixel neri adiacenti DeadPixel-3lines-2H-1V: vengono inserite 3 linee nel frame, di cui 2 sono orizzontali ed 1 verticale, composte da pixel neri adiacenti DeadPixel-5lines-3H-2V: vengono inserite 5 linee nel frame, di cui 3 sono orizzontali e 2 verticali, composte da pixel neri adiacenti DeadPixel-10lines-5H-5V: vengono inserite 10 linee nel frame, di cui 5 sono orizzontali e 5 verticali, composte da pixel neri adiacenti DeadPixel-carreggiata: vengono inserite 2 linee oblique nel frame che simulano la carreggiata su cui sta marciando il veicolo DeadPixel-carreggiata-ostacolo: vengono inserite 2 linee oblique nella stessa posizione del fallimento DeadPixel-carreggiata insieme ad un blocco di pixel neri al centro di esse che simula un ostacolo
No Bayer Filter	si utilizza cv2 [78] per l'apertura delle immagini e per le modifiche dei canali dei colori, mentre per la conversione a scala di grigi si utilizza il Modulo PIL Image [77]	Greyscale: apertura dell'immagine con <i>cv2.imread(img)</i> , modifica canale del colore con il metodo <i>cv2.cvtColor(img, cv2.COLOR_BGR2RGB)</i> , conversione a scala di grigi con il metodo <i>.convert('LA')</i>
No Chromatic Aberration Correction	immagini elaborate con il Modulo PIL Image [77], numPy [79] e il codice in [80]	ChromAberr-blur, ChromAberr-noblur: l'effetto viene inserito utilizzando parte del codice in [80], rispettivamente attivando e disattivando la sfocatura (blur)
No Demosaicing	immagini elaborate utilizzando cv2 [78] e ridimensionate con il Modulo PIL Image [77]	NoDemos: lettura immagine con il metodo <i>cv2.imread(img)</i> , elaborazione, ridimensionamento attraverso <i>imgOut.resize(h,w, Image.ANTIALIAS)</i>
No Noise Reduction	speckle noise introdotto utilizzando cv2 [78] e numPy [79]	Noiser: nel metodo <i>np.random.normal</i> sono stati utilizzati i parametri 0, 0.5; dunque <i>np.random.normal(0, 0.5, img.size)</i> Noise2: nel metodo <i>np.random.normal</i> sono stati utilizzati i parametri 0, 1; dunque <i>np.random.normal(0, 1, img.size)</i>
No Sharpness Correction	la nitidezza del frame originale viene modificata usando i Moduli PIL Image e ImageEnhance [77]	Sharpness: il metodo <i>ImageEnhance.Sharpness(img)</i> viene invocato con un fattore pari a -3.5

Tabella 2.: Tabella riassuntiva per i dettagli sull'implementazione dei Fallimenti analizzati

5.3 RISULTATI

I risultati della campagna sperimentale, trattati in questo paragrafo, saranno presentati inizialmente per tutte le tipologie di fallimento definite in sede di FMECA. Dopodiché, sarà affrontato in modo più approfondito il fallimento DeadPixel, in quanto questo presenta il maggior numero di configurazioni. Infatti, sulla base di questo fattore, si potrà discutere in profondità per ogni scenario e per ogni meteo considerato in fase di simulazione.

5.3.1 Success Rate e Numero di Collisioni

Per cominciare è stata svolta un'analisi dei dati che riguarda il Success Rate che ogni fallimento ha ottenuto nelle varie simulazioni eseguite. Questo punteggio indica, fra i fallimenti, quelli andati meglio e di conseguenza quelli andati peggio. Questa analisi è stata fatta rispetto ai tre scenari che si considerano nelle simulazioni, ovvero: FullTown02, StraightTown02 e TurnTown02. Sulla base di questi è stato composto l'istogramma in Fig. 54.

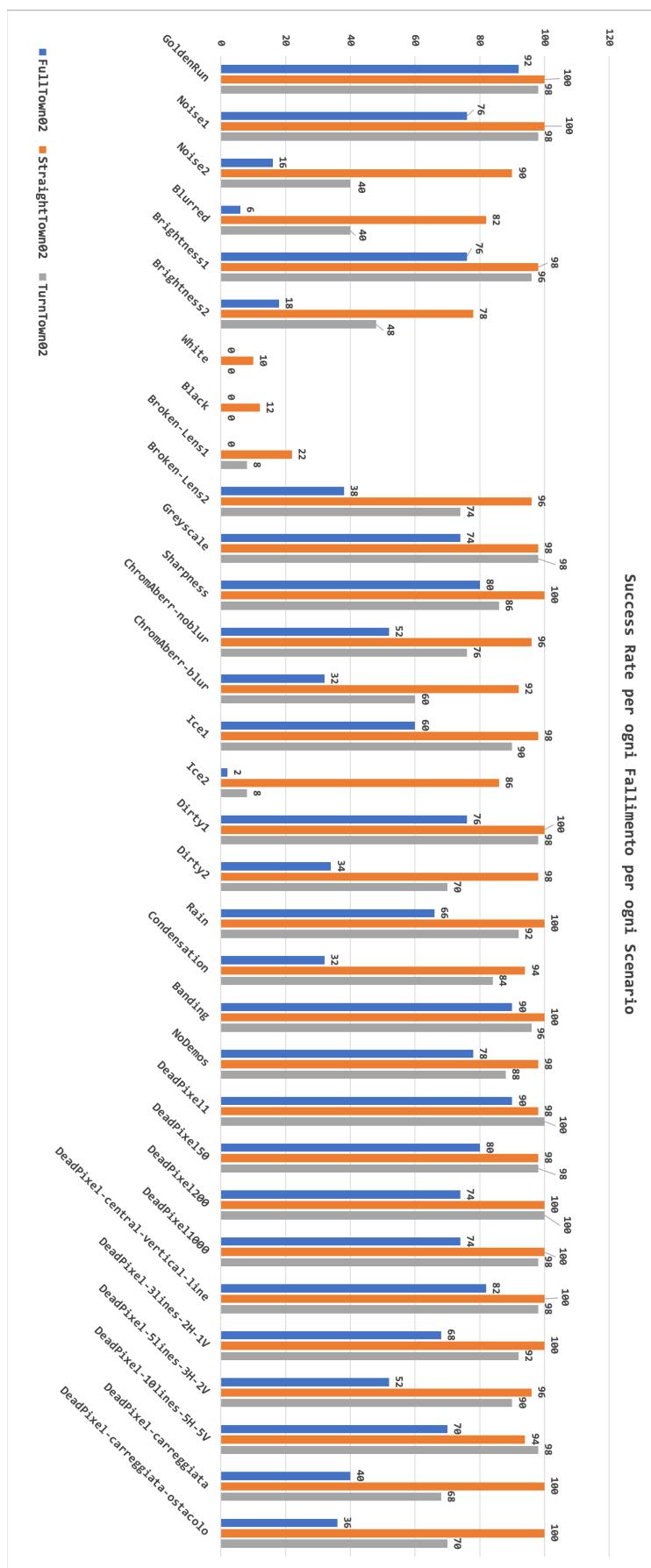


Figura 54.: Success Rate per ogni Fallimento per ogni Scenario

In questo, per ogni fallimento, sull'asse delle ascisse, vengono considerate tre serie di dati, ognuna delle quali inerente ad uno scenario. In particolare, si può notare come alcuni fallimenti creano degli avvallamenti nel percorso dei dati, come nel caso di Blurred, White, Black, Ice2 (la seconda configurazione del fallimento Ice considerata) e Broken-Lens1. Questo viene spiegato dal fatto che, ad esempio in White e Black, l'unica cosa che la telecamera vede è un fotogramma completamente bianco o uno completamente nero. Il fatto che tutti e cinque non presentano una percentuale di successo uguale a 0 nello scenario StraightTown02 è dovuto al fatto che in questo particolare scenario l'auto viene condotta su una strada dritta. Dunque, nel caso in cui il veicolo non trova ostacoli, come veicoli o pedoni che attraversano, ed il computer centrale riesce a condurlo correttamente nella stessa direzione di partenza, si arriverà ad un successo in qualche run.

Per maggiore leggibilità, se si considera il Success Rate sullo scenario FullTown02, ordinando in modo crescente i diversi valori, si ottiene il grafico in Fig. 55.

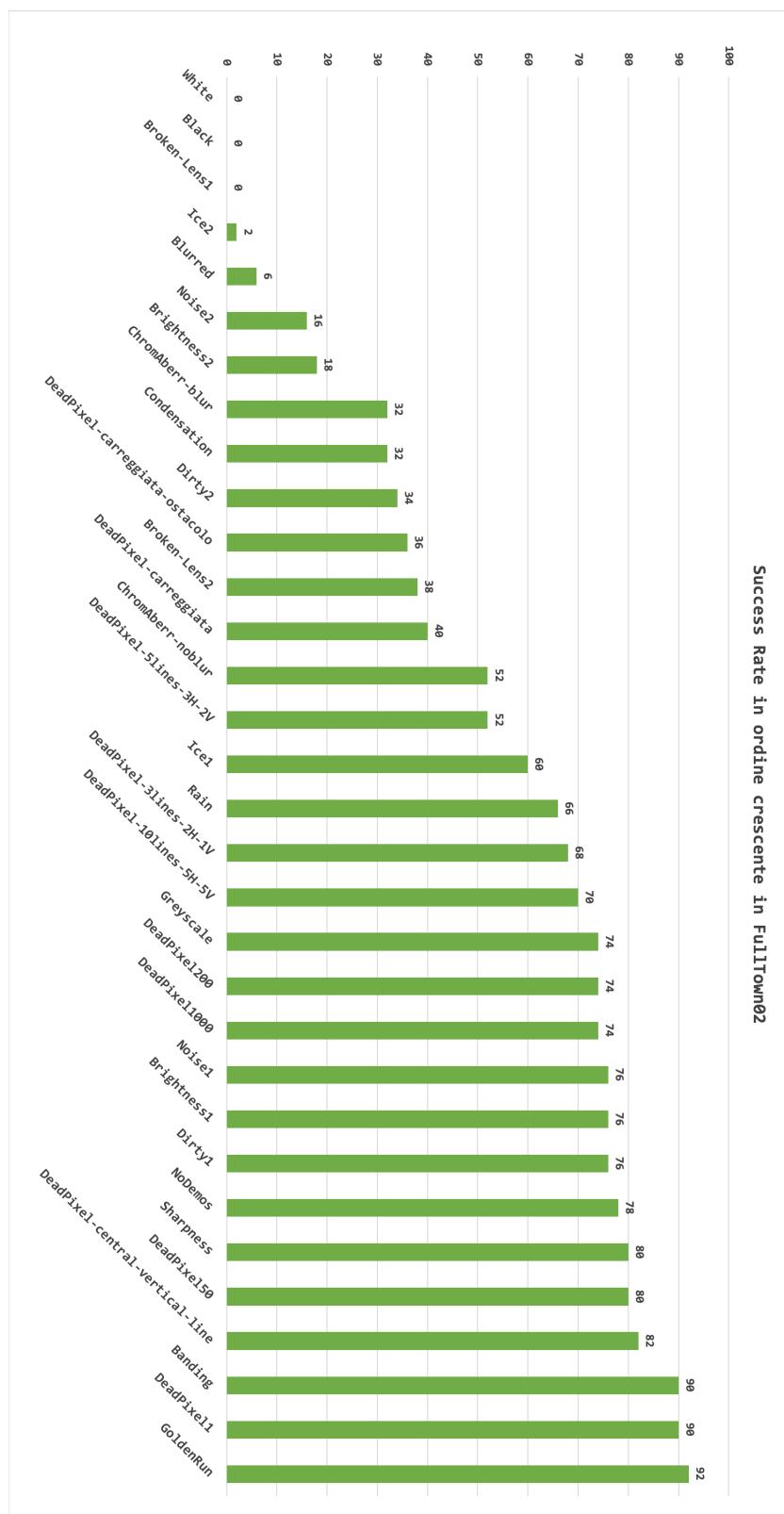


Figura 55.: Success Rate in ordine crescente per ogni Fallimento per FullTown02

Da questo si può notare ancora meglio come gli scenari nominati in precedenza siano effettivamente i più critici dal punto di vista delle run portate a termine con successo. Si è preso in considerazione questo scenario in particolare, in quanto rappresenta al meglio la situazione nella quale un veicolo a guida autonoma si potrà trovare: diverse svolte, sia a destra che a sinistra, semafori, pedoni, altri veicoli ed un percorso più articolato rispetto agli altri due scenari. Inoltre, come desiderato, la Golden Run risulta essere quella con il rate di successo più alto. In questa, infatti, non si iniettano errori nella telecamera.

Un altro dato significativo, fornito dal simulatore, è il numero di collisioni che il veicolo ha avuto in ogni run. Nel nostro caso, queste sono state sommate per ogni scenario considerato. Dunque, si è ottenuta una tabella nella quale per ogni fallimento è indicato il numero di collisioni totali in ogni scenario. Va considerato il fatto che quando si parla di run di simulazione, le iterazioni eseguite per ogni fallimento, ed anche per la Golden Run, sono state 50, come precedentemente detto. Queste sono suddivise nel seguente modo: 17 run vengono eseguite con il meteo W1 (Mezzogiorno Sereno), 17 con il meteo W4 (Mezzogiorno Nuvoloso Bagnato) e, infine, 16 con il meteo W13 (Pioggia Forte al Tramonto). La Fig. 56 mostra, per ogni fallimento, il numero di collisioni verificatesi in ogni scenario.

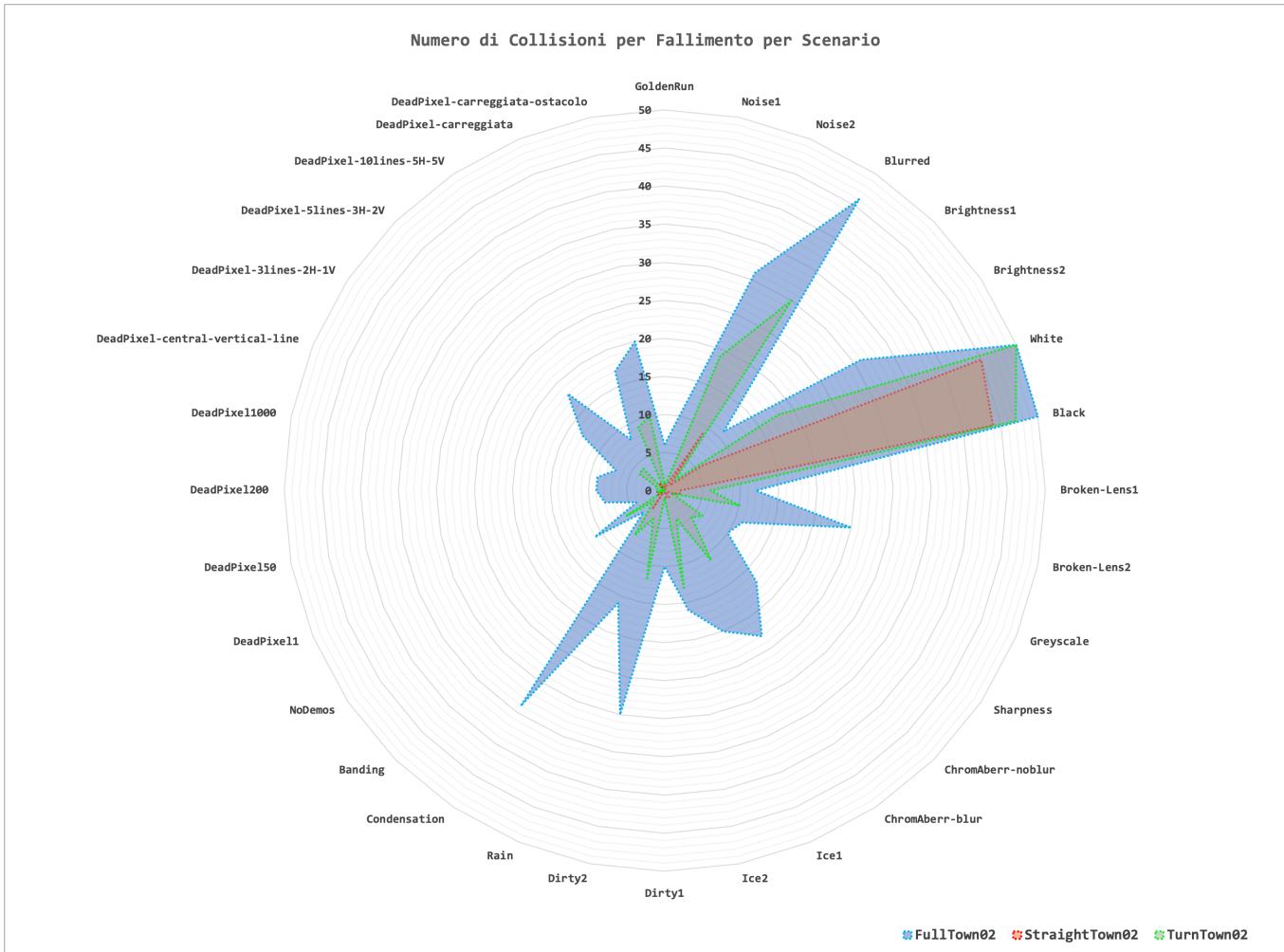


Figura 56.: Grafico Radar per Nr. Collisioni per ogni Fallimento per ogni Scenario

Se, come fatto in precedenza, si estraggono da questo grafico globale i numeri inerenti soltanto allo scenario FullTown02 (il più reale) e si ordinano in modo crescente, si ottiene il risultato presentato in Fig. 57.

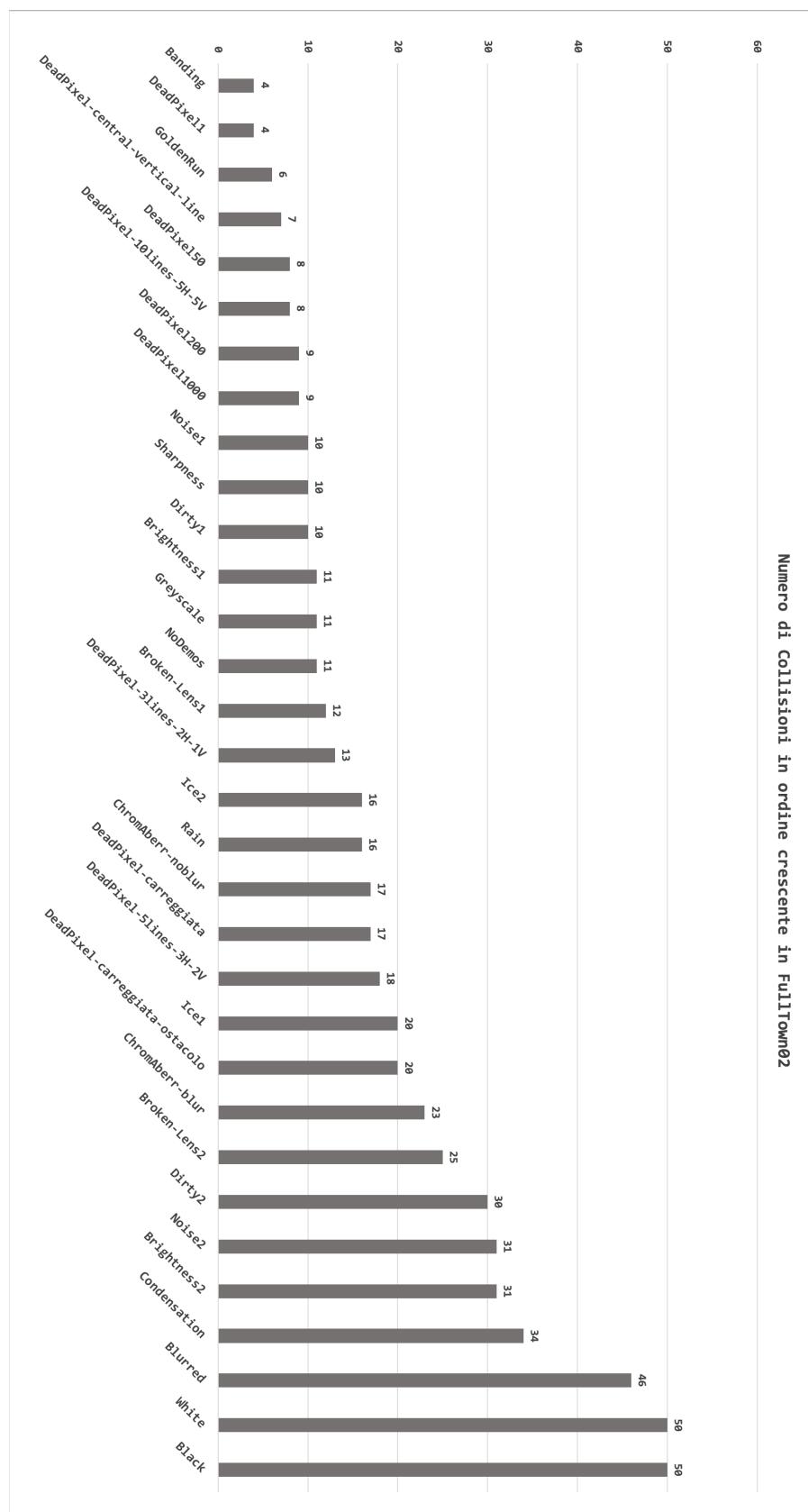


Figura 57.: Numero di Collisioni per ogni Fallimento in FullTown02

Un'altra volta, da questo si può notare come i fallimenti etichettati come più critici, all'inizio del paragrafo 5.3.1, mantengono questo titolo. Dal grafico si può vedere, inoltre, come il numero di collisioni evidenziate da questo siano in linea con quanto evidenziato dal grafico inerente il Success Rate considerato per il solo scenario FullTown02 precedentemente discusso.

Dalla Fig. 56 emerge come anche altri fallimenti siano paragonabili ai più critici discussi precedentemente. Si notano, infatti, dei picchi nei dati che puntano all'etichetta di diversi fallimenti. Tenendo conto che sono state svolte 50 run per fallimento, si può osservare come diversi fallimenti abbiano un numero di collisioni pari o maggiore di 20, nei tre scenari. Questo è importante, dato che il codice è stato modificato in modo da far terminare senza successo la run qualora il veicolo collidesse con qualsiasi cosa. Vengono quindi portati alla luce altri fallimenti pericolosi insieme a quelli evidenziati precedentemente:

- Noise2,
- Blurred,
- Brightness2,
- White,
- Black,
- Broken-Lens2,
- ChromAberr-blur,
- Ice1,
- Dirty2,
- Condensation,
- DeadPixel-carreggiata-ostacolo.

I fallimenti elencati sopra sono stati selezionati tenendo conto del numero di collisioni per ogni fallimento in ogni scenario, impostando un certo valore di soglia. Questo è stato necessario per estrapolare quei fallimenti causa dei success rate più bassi.

5.3.2 *Andamento Run rispetto al Meteo*

Un'altra variabile con cui è stato possibile interpretare i dati è rappresentata dalla variazione del Meteo. Infatti, nelle simulazione eseguite, ogni fallimento è stato valutato su tre tipologie di meteo, oltre che su tre scenari diversi. Dato che per ogni fallimento abbiamo ripetuto lo stesso iter nel condurre la campagna sperimentale, possiamo presentare un'ulteriore elaborazione: l'andamento delle run per ogni fallimento per ogni tipo di meteo. In Fig. 58 è raffigurato il grafico estratto dai dati prodotti in campagna sperimentale. In questo è stata considerata una colonna per ogni meteo, rappresentante il numero di collisioni, per un totale di tre colonne per ogni fallimento iniettato.

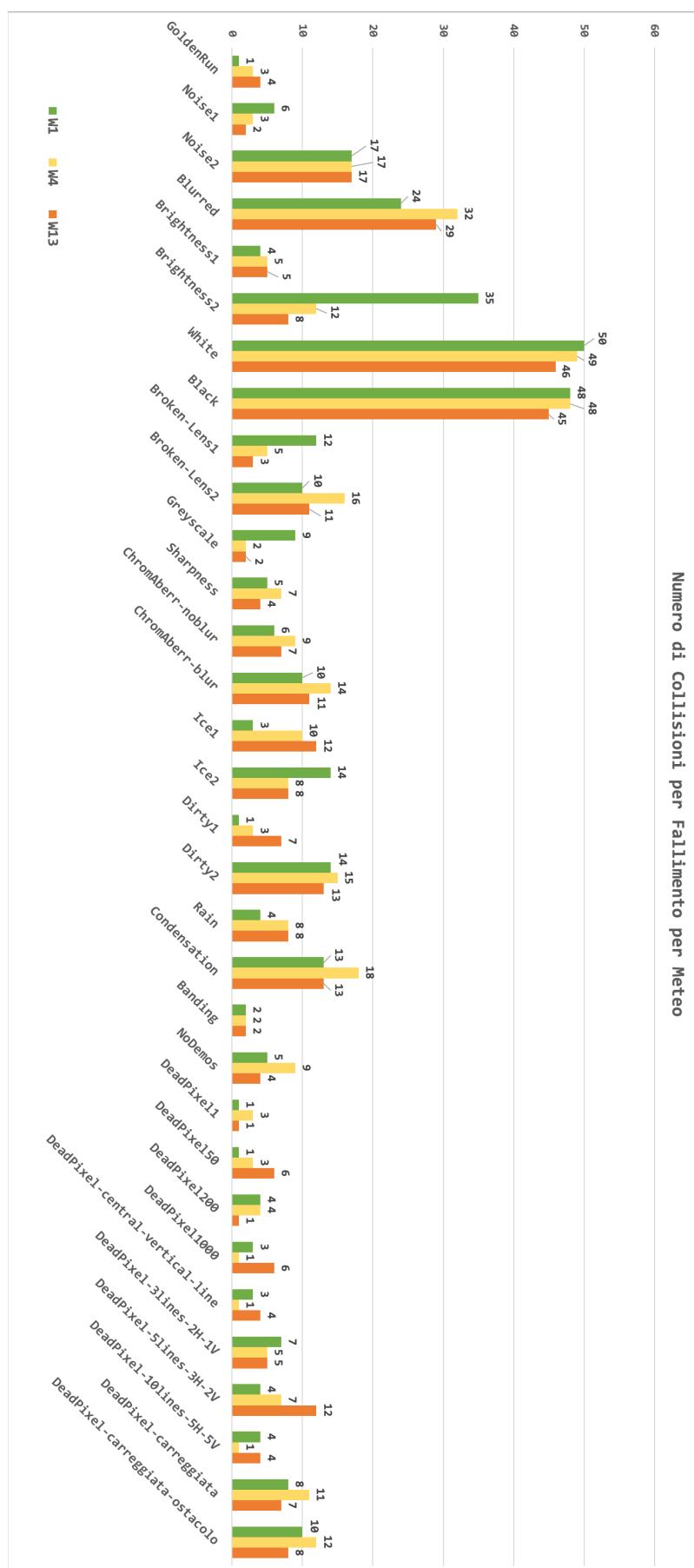


Figura 58.: Grafico Numero di Collisioni per ogni Fallimento per ogni Meteo

Osservando il grafico si nota un andamento simile a quelli presentati precedentemente, in quanto i fallimenti peggiori, anche se in questa rappresentazione si considerano le tre tipologie di meteo, restano gli stessi. Da un'analisi sui numeri è stato evidenziato come, anche se dal grafico si rimarcano gli stessi picchi già visti, la media del numero di collisioni per ogni meteo è pressoché la stessa. In Tabella 3 è riportata la media del numero di collisioni per ogni tipo di meteo.

	W1	W4	W13
Media numero collisioni per meteo	10,56	10,72	9,84

Tabella 3.: Media del numero di Collisioni per Meteo

Inoltre, si possono fare altre considerazioni a riguardo. Infatti, dai dati numerici, si è potuto estrapolare quale fallimento o quali fallimenti sono andati peggio in ogni meteo considerato. Per farlo è stata calcolata la percentuale di collisioni verificatesi per ogni fallimento per ogni meteo. In Tabella 4 si presentano i risultati raccolti per la GoldenRun, mentre in Tabella 5 quelli inerenti i 5 migliori ed i 5 peggiori fallimenti, in termini di percentuale di collisioni per ogni meteo.

	W1	W4	W13
GoldenRun	1,96%	5,88%	8,33%

Tabella 4.: Percentuali Collisioni per Meteo GoldenRun

Top 5 migliori Fallimenti per ogni Meteo		
W1	1, 96%	Dirty1 DeadPixel1
	3, 92%	DeadPixel50 Banding
	5, 88%	Ice1
W4	1, 96%	DeadPixel1000 DeadPixel-central-vertical-line
	3, 92%	DeadPixel-10lines-5H-5V Greyscale Banding
W13	2, 08%	DeadPixel1 DeadPixel200 Noise1
	4, 17%	Greyscale Banding
Top 5 peggiori Fallimenti per ogni Meteo		
W1	98, 04%	White
	94, 12%	Black
	68, 63%	Brightness2
	47, 06%	Blurred
	33, 33%	Noise2
W4	96, 08%	White
	94, 12%	Black
	62, 75%	Blurred
	35, 29%	Condensation
	33, 33%	Noise2
W13	95, 83%	White
	93, 75%	Black
	60, 42%	Blurred
	35, 42%	Noise2
	27, 08%	Condensation

Tabella 5.: Percentuali minime e massime di Collisioni per Meteo

I fallimenti scritti nella tabella sovrastante sono quelli che presentano una uguale percentuale di collisioni su numero di run. Ad esempio, per il valore minimo, nel Meteo W1 se ne ritrovano ben 3, mentre per quanto riguarda la percentuale massima, il fallimento White risulta, come del resto finora, essere il peggiore. Naturalmente in questa tabella sono stati considerati soltanto alcuni dei valori trovati. Se si volessero includere anche gli altri sarebbero ben di più e noteremmo che la distanza (percentuale) fra quelli scritti in tabella e quelli mancanti non sarebbe così elevata. Infatti, il fallimento Black è ad una media di 2.66 punti percentuali da White, quindi molto vicino in termini di numero di collisioni.

Nei numeri presentati sopra sono stati considerati tutti gli scenari uniti, ovvero: prendendo come esempio la Golden Run, in questa si sono registrate 6 collisioni in FullTown02, 1 in StraightTown02 e 1 in TurnTown02; di queste collisioni, 1 si è verificata con il meteo W1, 3 con il meteo W4 e 4 con il meteo W13, per un totale di 8 collisioni in 50 run, suddivise in 17 per W1, 17 per W4 e 16 per W13. Da questi numeri sono state estratte le percentuali presentate.

5.3.3 *Tempi di Simulazione*

Per tempi di simulazione si intendono tutti gli step che CARLA compie per portare a termine una run. Infatti, nel simulatore, non viene considerato il tempo reale, ma degli step che vanno a comporre insieme il numero totale dei frame che danno vita al filmato prodotto da ogni run. Per velocizzare la campagna sperimentale i filmati sono stati disabilitati. Questi sono stati usati per controllare che il fallimento inserito producesse effettivamente un effetto percepibile.

Una volta ottenuti tutti i dati per ogni run, su qualsiasi scenario e per qualsiasi meteo, si è provveduto a svolgere un’analisi anche dei tempi di simulazione. Infatti, se effettivamente il fallimento iniettato rende malfunzionante il dispositivo telecamera, ci si aspetta che il tempo di simulazione scenda con l’aumentare della gravità del fallimento iniettato. Questa diminuzione sarà data dal fatto che le run tenderanno a fallire prima del tempo goal, dato che il veicolo sarà più soggetto a collidere con soggetti mobili e immobili dell’ambiente in cui si muove.

Qui di seguito (Fig. 59) viene presentato il caso del fallimento Noise,

confrontato con i risultati ottenuti nella Golden Run.

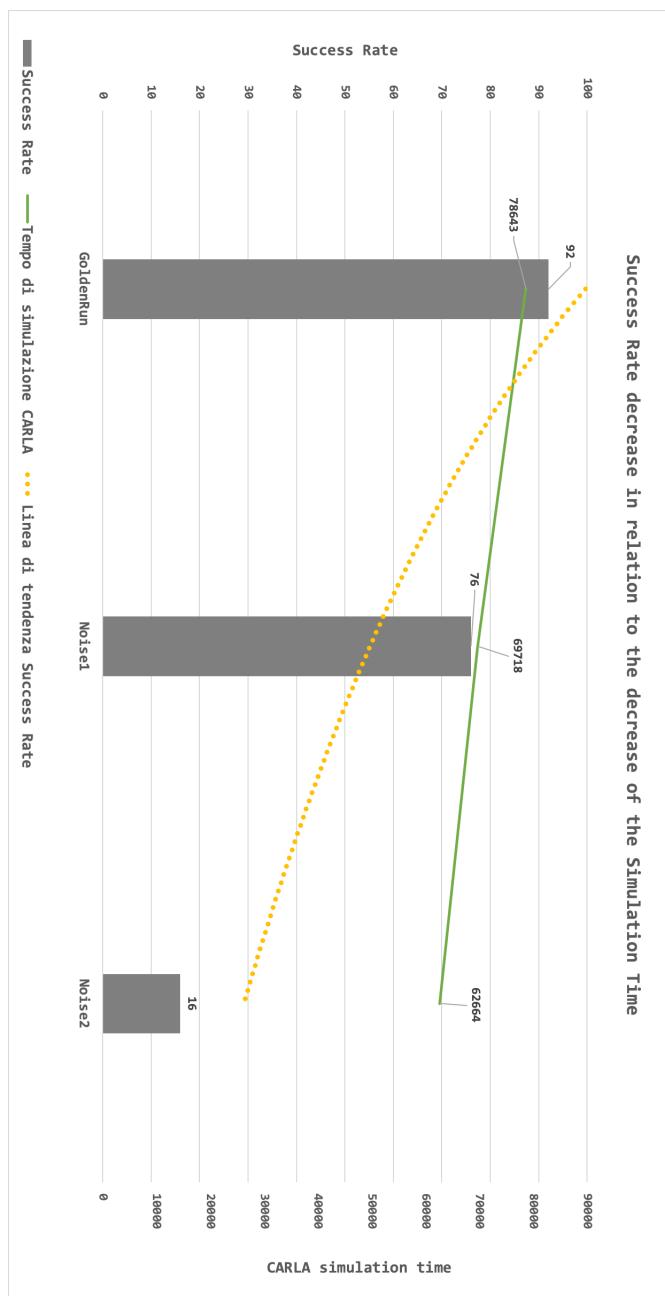


Figura 59.: Relazione fra Tempo di Simulazione CARLA e Success Rate

Nel grafico, come detto, sono state confrontate le due configurazioni di Noise con la Golden Run. In questo, i tempi di simulazione sono stati messi in relazione con i Success Rate così da evidenziare la diminuzione

del tempo e conseguentemente del numero delle run con successo.

Un altro modo per evidenziare la difficoltà del veicolo nel portare a termine le run è mostrata in Fig. 60, nella quale si illustrano i tempi in relazione col numero di collisioni avute.

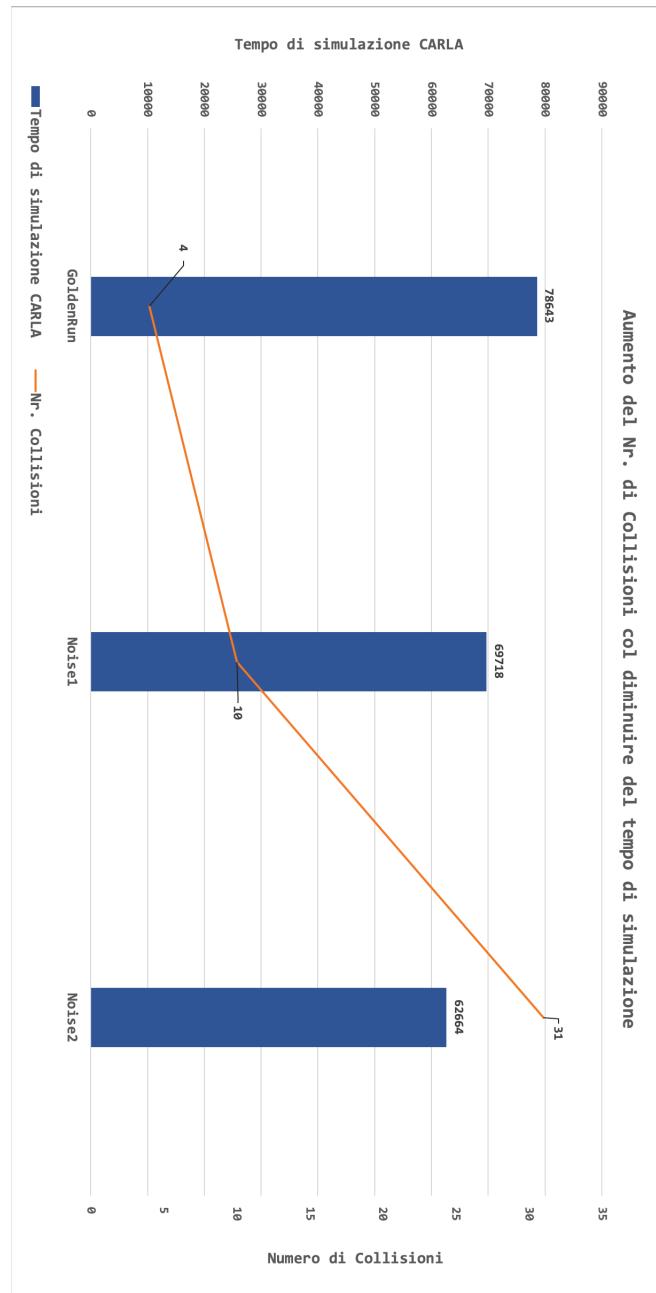


Figura 60.: Relazione fra Numero di Collisioni e Tempo di simulazione CARLA

Da questa figura, come nella precedente, si capisce come la diminuzione del tempo di simulazione sia legata al fatto che le run falliscono prima del goal. Oltre a questo si vede anche come il numero di collisioni va di pari passo con il diminuire del tempo di simulazione CARLA, in quanto il codice del simulatore è stato appositamente modificato per far sì che la run fallisse qualora il veicolo entrasse in collisione con un qualsiasi soggetto nell'ambiente.

Un'importante considerazione da fare riguarda molti dei fallimenti trattati ed anche la GoldenRun. Infatti, se si analizzano i dati come sono registrati nei file di log prodotti, non si notano anomalie. Quando però i dati vengono scorsi e si controllano i numeri, si possono notare dei lunghi tempi morti nei quali il veicolo si ferma. Questo fatto farebbe pensare ad una sorta di indecisione che il simulatore ha in diverse situazioni. Dato che gli altri soggetti dell'ambiente guidano in modo "perfetto", il problema riscontrato è sicuramente nel veicolo guidato dalla visione artificiale che abbiamo analizzato. In diverse run, per diversi fallimenti, sono stati notati dei tempi di simulazione anomali, tanto da far risultare la media di questi tempi più alta di altre medie legate a fallimenti meno severi. L'anomalia notata fa sì che ad un certo punto della run il veicolo si fermi per non ripartire più. In un primo momento si è pensato ad una sorta di meccanismo safe per far sì che il veicolo non si trovasse in situazioni di pericolo e non collidesse. Da un'analisi dei video, però, è stato notato come il veicolo attendesse lo scadere del time-out fermo in quanto davanti a sé trovava altri veicoli, in più di un caso, coinvolti in un incidente. Fin dal principio sono stati esclusi i semafori, dato che la durata di questi non arriva mai ai tempi di stop anomali rilevati.

La fermata del veicolo la si può osservare guardando la colonna velocità del file di log. In questa, nei periodi di immobilità del veicolo anomali, si possono notare delle minime variazioni nella velocità, come se il veicolo provasse a "cambiare posizione" o ad accelerare di poco per cambiare la "vista" della videocamera. Per minime variazioni si intendono numeri compresi in un intervallo $[0.001, 0.02]$ m/s. Considerando che il simulatore lavora in 5fps, poche righe del file di log equivalgono ad uno stato di immobilità del veicolo.

Un'ultima considerazione si può fare tirando in causa il fallimento No Chromatic Aberration Correction. Questo prevede due configurazioni,

una in cui non viene aggiunto l'effetto sfocato e l'altra in cui questo è presente. Nella configurazione col blur è stata notata un'anomalia in termini di tempo reale di simulazione (ore). Da questo si può mostrare come il simulatore abbia i suoi step e questi siano ben distinti dallo scorrere del tempo reale. In Fig. 61, infatti, viene rappresentato perfettamente quanto detto.

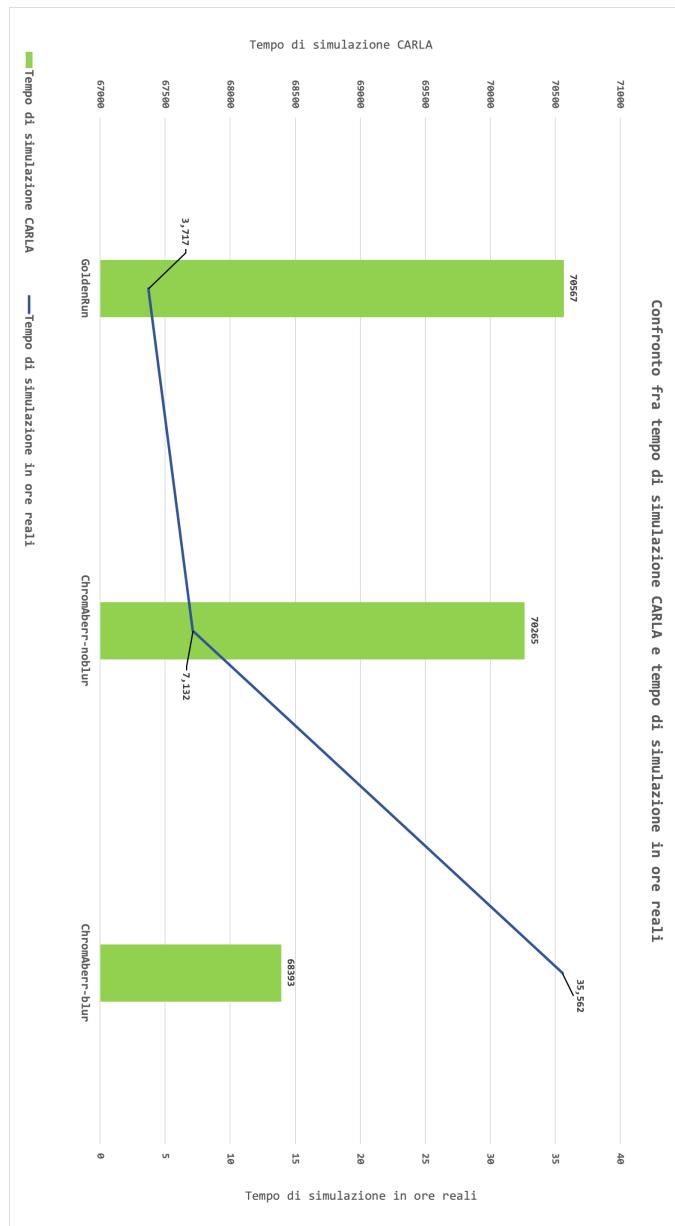


Figura 61.: Tempi di Simulazione CARLA e in ore reali del Fallimento No Chromatic Aberration Correction

Più il fallimento viene iniettato nelle sue configurazioni più severe e più questo dovrebbe abbassare il tempo totale di simulazione. Nel caso sopra, si nota anche una cosa in più: sebbene la simulazione con la seconda configurazione del fallimento No Chromatic Aberration Correction duri, in ore reali, 5 volte di più delle altre, ha il tempo di simulazione CARLA minore, come desiderabile. Questo è dovuto al fatto che il simulatore "aspetta" che ogni passo sia stato fatto, per quanto questo possa essere lungo, prima di andare al prossimo. Nel caso specifico, i passi sono stati allungati dall'introduzione dell'effetto sfocato al frame, prima di inviarlo al comparto decisionale.

5.3.4 *Approfondimento sul Fallimento DeadPixel*

Il fallimento DeadPixel è quello che presenta più configurazioni, ben 10, nelle quali si rappresentano diverse combinazioni, anche estremizzate, di quello che potrebbe succedere al dispositivo di acquisizione durante la sua vita. Come illustrato in 5.2.2, DeadPixel ha diverse varianti: un pixel nero, 50, 200, fino ad arrivare a righe orizzontali, verticali e/o oblique che hanno l'obiettivo di far fallire il dispositivo.

L'approfondimento condotto su questo fallimento ha tenuto conto dei tempi di simulazione, dei success rate, del meteo e del numero di run con collisione. Per quanto concerne i tempi di simulazione, la Fig. 62 illustra i diversi tempi delle configurazioni a confronto con quello della Golden Run.

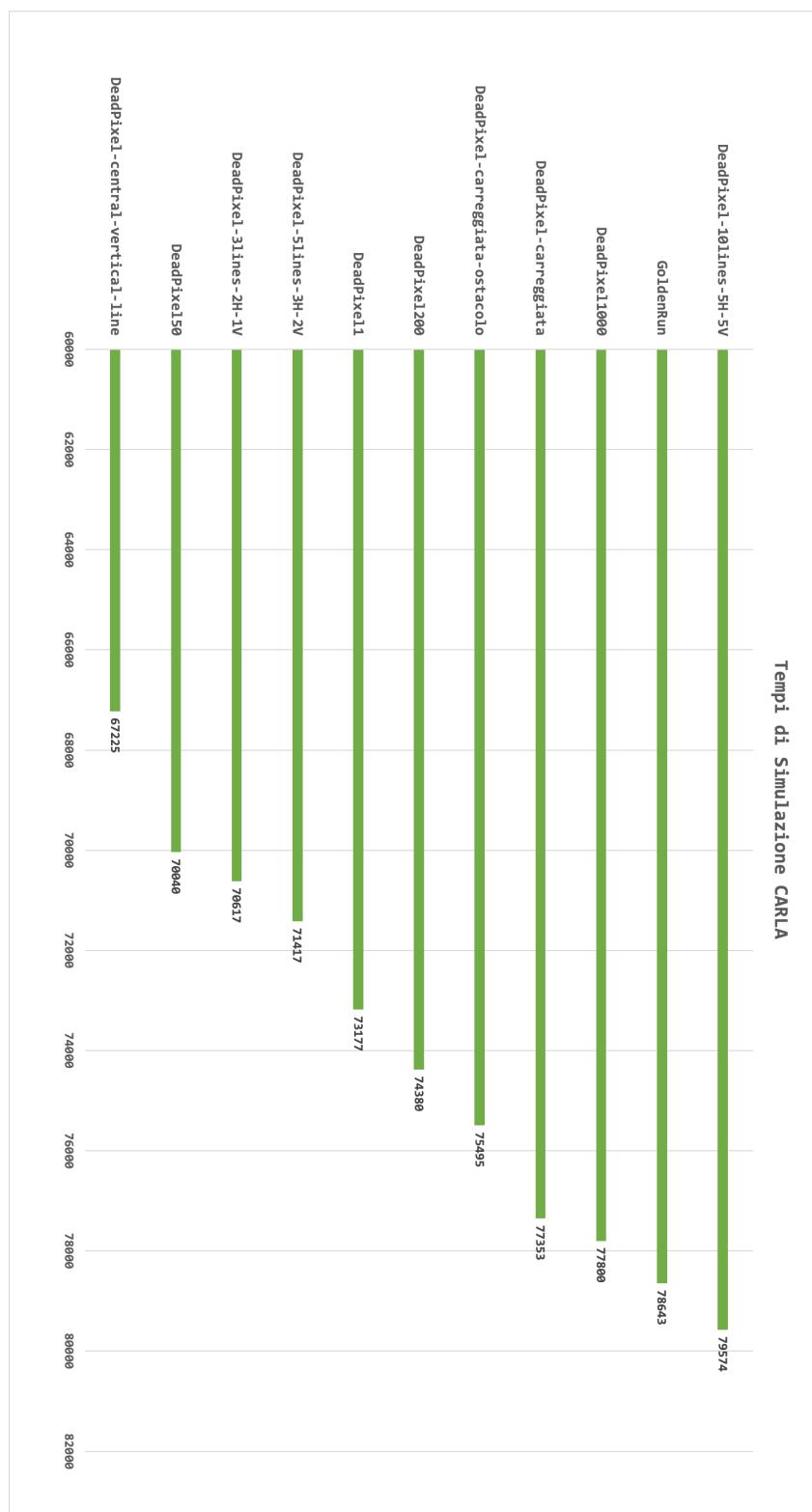


Figura 62.: Tempi di Simulazione del Fallimento DeadPixel nelle sue configurazioni - FullTown02

La Golden run non risulta essere quella col tempo di simulazione CARLA maggiore, anche se di poco. Questo di per sé non è un problema, in quanto la bontà delle run verrà poi misurata in termini di Success Rate, dove la Golden Run è al primo posto. Dunque, se i tempi vengono messi in relazione con il Success Rate e/o il numero di run con collisione, se ne trae un'evidenza diversa. Nella Fig. 63 è mostrato come ad un tempo di simulazione maggiore sono associati i Success Rate più alti, ovvero i fallimenti che hanno creato meno disagio al dispositivo e al sistema auto in generale.

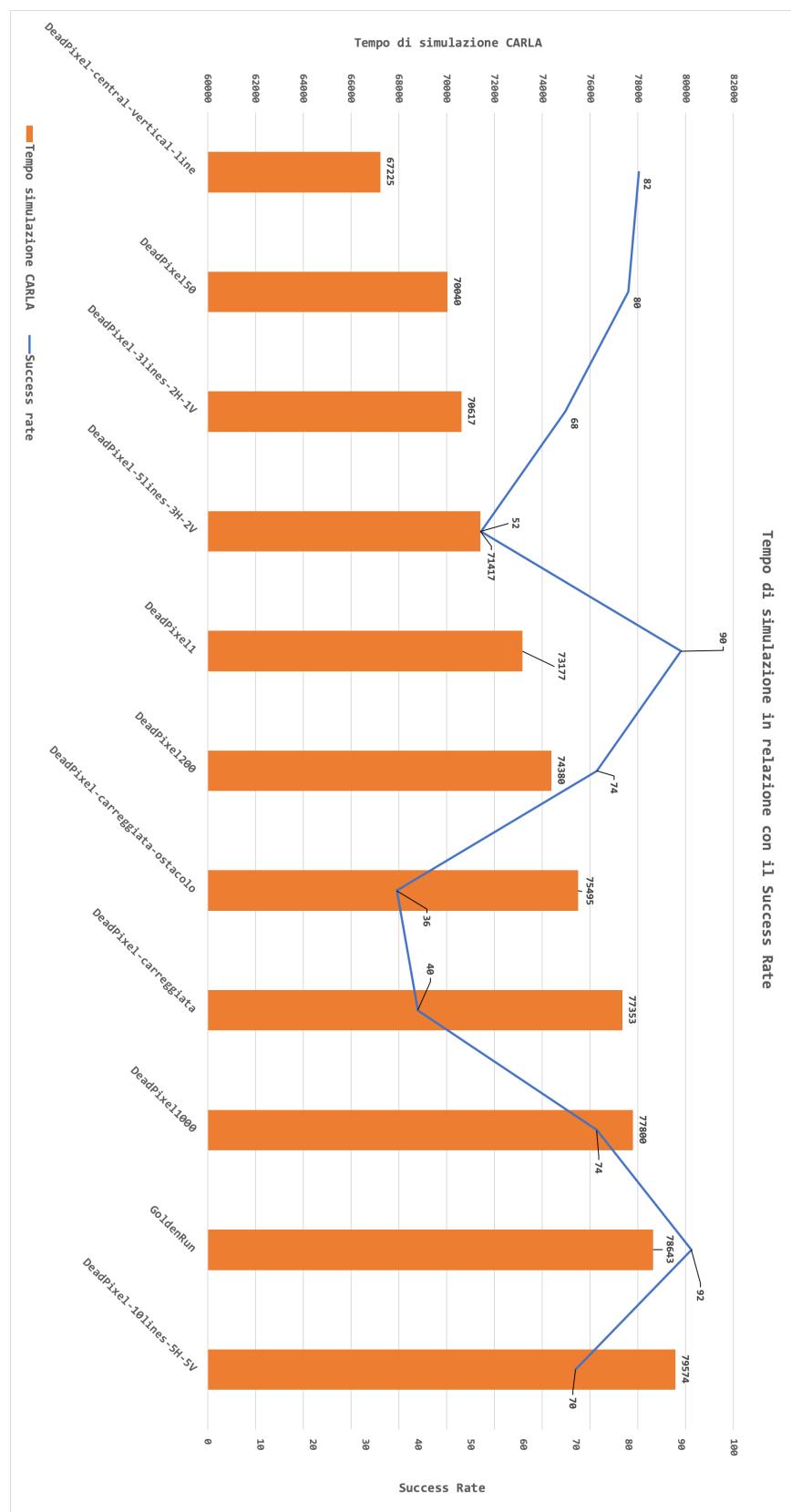


Figura 63.: Tempi di Simulazione in relazione con i Success Rate - FullTown02

In questa figura sono mostrati i fallimenti in ordine crescente per tempo di simulazione CARLA, dunque i picchi verso il basso, come nel caso di DeadPixel-carreggiata e DeadPixel-carreggiata-ostacolo, possono essere intesi come un comportamento più attento da parte dell'auto. Infatti, questi fallimenti in particolare cercano di indurre in errore il veicolo, cercando di far interpretare alla telecamera una carreggiata sempre dritta, come mostrato in Fig. 44 e in Fig. 45.

Le immagini illustrate precedentemente sono estratte dai dati riguardanti lo scenario FullTown02, essendo quello più completo e vicino alla realtà, per eterogeneità di comportamenti e soggetti attivi nell'ambiente di marcia.

Un altro aspetto considerato per il fallimento DeadPixel è il Meteo e come questo possa influire o meno sulle prestazioni. In Fig. 64, si mostra come il meteo W13 risulta essere quello dove si verificano più collisioni.

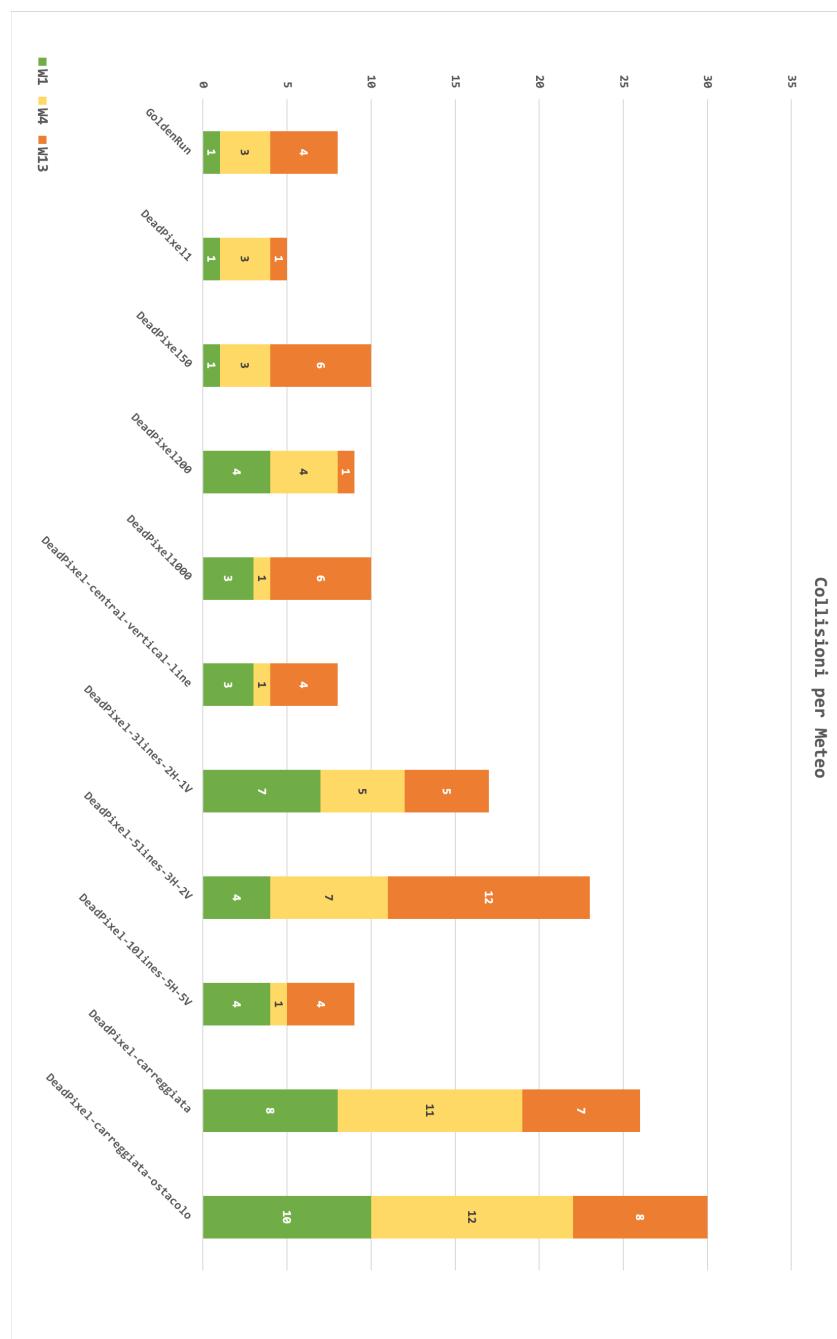


Figura 64.: Nr. Collisioni per ogni Meteo - Scenari uniti

Se si prova a suddividere per Meteo si può constatare come:

- per il meteo W1 le run andate meglio sono quelle di GoldenRun, DeadPixel11 e DeadPixel50;

- per il meteo W4 le run andate meglio sono DeadPixel1000, DeadPixel-central-vertical-line e DeadPixel-10lines-5H-5V;
- infine, per il meteo W13, DeadPixel1 e DeadPixel200.

Per il fallimento DeadPixel è stato usato anche un altro approccio per tentare di analizzare la mole di dati derivanti dalle simulazioni. Per semplicità, per queste considerazioni, si sono usati i soli dati relativi allo scenario FullTown02. Per questo sono stati analizzati i tempi medi che per ognuna delle configurazioni di DeadPixel, in relazione con la Golden Run, sono stati registrati. Oltre a calcolare i tempi medi, questi sono stati suddivisi per meteo: dunque si è calcolato il tempo medio che, ad esempio DeadPixel1, ha registrato per W1, per W4 e per W13. Il risultato finale è stata la Tabella 6.

	W1	W4	W13
GoldenRun	1777,59	1426,82	1174,31
DeadPixel1	1587,82	1402,29	1396,56
DeadPixel50	1424,29	1468,00	1304,44
DeadPixel200	1465,59	1503,65	1493,94
DeadPixel1000	1635,00	1792,00	1221,31
DeadPixel-central-vertical-line	1521,35	1546,06	1236,06
DeadPixel-3lines-2H-1V	1320,41	1410,41	1512,06
DeadPixel-5lines-3H-2V	1672,76	1453,76	1141,63
DeadPixel-10lines-5H-5V	1985,94	1582,00	1182,44
DeadPixel-carreggiata	1875,41	1338,29	1420,00
DeadPixel-carreggiata-ostacolo	1752,94	1271,41	1505,06

Tabella 6.: Tempi Medi run suddivisi per Meteo

Nella trattazione iniziale di questi risultati, si è parlato di tempi morti nei quali il veicolo sembrava fermarsi senza più muoversi fino allo scadere del time-out. Queste run effettivamente non risultano come dei fallimenti, in quanto, per far fallire la run, il veicolo deve collidere. Ma cosa succede se, analizzando tutti i dati, si toglie il tempo in cui il sistema auto sembra in uno **"stato di immobilità"**, che termina con lo scadere del time-out?

Il risultato è presentato in Tabella 7, dove i tempi scritti in arancio sono quelli modificati in fase di revisione.

	W1	W4	W13
GoldenRun	1594,29	1426,82	1174,31
DeadPixel1	1587,82	1402,29	1396,56
DeadPixel50	1326,88	1468,00	1186,50
DeadPixel200	1170,06	1336,24	1381,38
DeadPixel1000	1347,76	1471,94	1221,31
DeadPixel-central-vertical-line	1521,35	1439,18	1236,06
DeadPixel-3lines-2H-1V	1127,76	1152,24	1328,44
DeadPixel-5lines-3H-2V	1173,59	1245,47	1141,63
DeadPixel-10lines-5H-5V	1539,53	1446,88	1182,44
DeadPixel-carreggiata	875,53	911,35	1043,56
DeadPixel-carreggiata-ostacolo	1081,00	1010,06	982,75

Tabella 7.: Tempi Medi run suddivisi per Meteo dopo la revisione

Se per ogni tipologia di meteo si considera la media globale, calcolata sul solo scenario FullTown02, ci si aspetta che al peggiorare delle condizioni meteorologiche tendono a verificarsi più fallimenti e quindi i tempi di simulazione si abbassano. I risultati presentati nella Tabella 8 confermano questo trend.

	W1	W4	W13
Tempi Medi	1638,10	1472,25	1326,16
Tempi Medi post-revisione	1304,14	1300,95	1206,81

Tabella 8.: Tempi Medi pre e post revisione

5.3.5 Riepilogo

Nell'osservare i risultati ottenuti sono stati evidenziati i fallimenti più critici per il sistema, ovvero quelli che, considerando le varie metriche utilizzate, sono andati peggio. Di questi può essere fatta un'elencazione:

- White,

- Brightness2,
- Black,
- Blurred,
- Broken-Lens1 e Broken-Lens2,
- Noise2,
- Ice1 e Ice2,
- Dirty2,
- Condensation,
- DeadPixel-carreggiata-ostacolo
- e ChromAberr-blur.

Naturalmente, tutti gli altri fallimenti mancanti da questo elenco non sono da considerarsi innocui per il sistema, ma semplicemente meno dannosi. Tuttavia, stiamo parlando di un veicolo autonomo e dunque anche la minima incomprensione di quello che si trova al di fuori dell'abitacolo può causare effetti fatali per passeggeri ed altri soggetti della strada.

In generale, nel trattare questi risultati è stato dimostrato che:

- al peggiorare del Meteo, nella maggior parte dei casi, le run sono peggiorate dal punto di vista del Success Rate e quindi del Numero di Collisioni;
- al peggiorare del Meteo, le run tendenzialmente si sono accorciate;
- quando il tempo di simulazione è risultato fra i più alti, in più casi, anche il Success Rate era fra i più alti;
- il tempo di simulazione CARLA è indipendente dal tempo reale di simulazione misurato in ore, come mostrato per il fallimento No Chromatic Aberration Correction in Fig. 61;
- i fallimenti più critici sono risultati essere quelli che effettivamente sono percepiti come tali anche dall'occhio umano e in fase congetturale, con delle sorprese per quanto riguarda quelli andati meglio (Tabella 5);

- l’approfondimento sul fallimento DeadPixel, essendo quello con il maggior numero di configurazioni, ha dato modo di fare un’analisi più approfondita per vedere quali sfumature dello stesso fallimento fossero più critiche per il sistema; la Fig. 63 e la Fig. 64 mostrano come le configurazioni costruite per trarre in inganno l’agente addestrato, come DeadPixel-carreggiata e DeadPixel-carreggiata-ostacolo, sono risultate essere effettivamente le più critiche a scenari uniti, dal punto di vista del numero di collisioni e delle altre metriche.

6

STATO DELL'ARTE

In base ai dati disponibili nessun lavoro di ricerca considera un modello completo di fallimenti di una telecamera per veicoli, compresa un'analisi degli effetti e dei rischi per la sicurezza, nonché la creazione di una libreria software per la replicazione dei fallimenti. Tuttavia, diverse opere hanno affrontato problemi simili o inerenti.

Le prestazioni, la robustezza e la sicurezza di una fotocamera RGB sono state ampiamente esplorate, ciò nonostante di solito si concentrano su elementi specifici o metriche target e senza affrontare l'insieme completo di fallimenti.

In [81] viene proposto un nuovo modello e un set di dati per la stima del flusso di scene 3D con un'applicazione alla guida autonoma. In questo si sfrutta il fatto che spesso le scene all'aperto si possono decomporre in un piccolo numero di oggetti indipendenti in movimento. Ogni elemento della scena viene rappresentato con i suoi parametri di movimento e ogni superpixel con un piano 3D, nonché un indice per l'oggetto corrispondente. Le applicazioni di questo includono la robotica e la guida autonoma in cui il movimento di oggetti 3D è un input fondamentale per attività di alto livello come la comprensione della scena, evitare degli ostacoli e/o la pianificazione del percorso. A differenza di altri studi, in questo, si sfrutta esplicitamente il fatto che tali scene possono spesso essere considerate come una piccola raccolta di oggetti 3D in marcia che includono il movimento di sfondo causato dalla telecamera montata sul veicolo stesso.

In [82] si propone un nuovo quadro basato sul deep reinforcement learning per l'analisi comparativa del comportamento dei meccanismi di prevenzione delle collisioni nel peggiore dei casi, quando si ha a che fare con un adversarial agent ottimale, addestrato a guidare il sistema verso

stati non sicuri. In questo vengono considerate diverse metriche come: tempo prima della collisione, danno riportato (in percentuale), ecc.

Le condizioni meteorologiche avverse sono molto difficili per la guida autonoma poiché la maggior parte dei sensori all'avanguardia smette di funzionare in modo affidabile. Al fine di sviluppare sensori e algoritmi robusti, fare testing con i sensori attuali in determinate condizioni meteorologiche è di cruciale importanza per la determinazione dell'impatto del maltempo per ciascuno di questi. In [11] viene descritta una metodologia di test e valutazione che aiuta a confrontare nuove tecnologie di sensori con quelli attualmente utilizzati. Nell'articolo viene mostrata la differenza fra un'immagine catturata da un fotocamera CMOS standard ed una catturata da fotocamera gated. Per gated ci si riferisce al principio della gated imaging, che può essere utilizzato per migliorare il contrasto dell'immagine nelle scene in cui l'oggetto di interesse è oscurato da disordine o forti fonti di luce che offuscano o saturano la termocamera (Fig. 65).



(a) Fotocamera CMOS Standard



(b) Fotocamera Gated

Figura 65.: Esempio Fotocamera Gated [11]

Considerando invece i problemi di sicurezza dei sensori, in [20] viene "abbagliato" un sistema di telecamere commerciali utilizzato nei veicoli, con diverse sorgenti luminose. Il lavoro mostra che l'utilizzo di laser o LED potrebbe danneggiare la telecamera. Allo stesso modo, in [83], laser e LED vengono puntati direttamente sulla telecamera: questo può causare danni irreversibili e interrompere le corrispondenti applicazioni autonome. In generale, si osserva che, a causa della vulnerabilità della telecamera causata dalle sue caratteristiche ottiche, è difficile costruire un sistema di telecamere completamente sicuro [19].

Nello studio [84], motivati dalle limitazioni dei benchmark stereo multi-view esistenti, viene presentato un nuovo set di dati per questo compito. Per arrivare a tale obiettivo, sono state registrate molte scene sia interne che esterne, con scanner laser, immagini DSLR ad alta risoluzione e video stereo sincronizzati a bassa risoluzione, con diversi campi visivi. Per quanto riguarda l'allineamento delle immagini con le scansioni laser, viene poi proposta una tecnica robusta che minimizza gli errori fotometrici condizionati sulla geometria. Il benchmark proposto copre una serie diversificata di punti di vista e tipi di scena, che vanno dalle scene naturali agli ambienti interni ed esterni artificiali.

In [85] viene preso in considerazione il rilevamento della superficie stradale. Gli attuali set di dati e benchmark stradali descrivono solo scenari europei e nordamericani, mentre nei paesi emergenti si ha una maggiore accettazione da parte dei consumatori di tecnologie AV (veicoli autonomi) e DAS (driver-assistance systems). Questo documento presenta un set di dati di scenari urbani brasiliani selezionati e un benchmark di rilevamento stradale costituito da dati RADAR, LIDAR e telecamere annotati. Propone, inoltre, una nuova metrica di valutazione basata sull'intersezione dei poligoni. L'obiettivo principale di questo manoscritto è fornire scenari impegnativi per la valutazione dell'algoritmo di rilevamento stradale.

Come detto, i veicoli senza conducente sono attualmente in fase di test su strade pubbliche al fine di esaminare la loro capacità di comportarsi in modo sicuro e affidabile in situazioni del mondo reale. Tuttavia, il funzionamento affidabile a lungo termine dei diversi sensori di un veicolo a guida autonoma e gli effetti di potenziali guasti del sensore nel sistema del veicolo non sono ancora stati testati. In [86] si propone un'architettura

di fusione dei sensori che minimizza l'influenza di un guasto del sensore sul sistema auto. In questo vengono presentati risultati sperimentali che simulano i guasti introducendo dispersione nelle informazioni del sensore dal set di dati KITTI.

Fino ad ora, gran parte dello sforzo di ricerca è stato dedicato allo sviluppo di diversi algoritmi di rilevamento e controllo. Tuttavia, vi sono state ricerche limitate su come gestire gli errori dei sensori in modo efficiente. Un semplice errore nel sensore può portare a un guasto imprevisto nell'intera funzione di guida autonoma. In questi casi viene consigliato quindi di rispedire il veicolo al costruttore per la riparazione, che si riassume in una perdita di tempo e di denaro. In [87] si introduce un metodo di correzione del sensore online automatico ed efficiente. Il metodo include quattro funzioni principali: rilevamento degli errori del sensore, insegnamento umano, apprendimento del veicolo e autovalutazione del veicolo. Le applicazioni di questo metodo ai sensori di visione artificiale e radar per ripristinare il controllo adattivo della velocità di crociera e le funzioni di mantenimento della corsia, vengono introdotte in modo dettagliato. I risultati sperimentali dell'elaborato, acquisiti per mezzo di veicoli a guida autonoma in scala 1/10 ad alta fedeltà, illustrano l'efficacia e i vantaggi dell'approccio.

Una delle maggiori sfide aperte nelle auto a guida autonoma è la capacità di rilevare automobili e pedoni per navigare in sicurezza nel mondo. Gli approcci per il rilevamento di oggetti basati sul Deep Learning hanno consentito notevoli progressi nell'uso delle immagini della telecamera per rilevare e classificare gli oggetti. Tuttavia, i tassi di errore allo stato attuale sono ancora troppo elevati per consentire un funzionamento sicuro nel mondo reale. Inoltre, quando si misurano le prestazioni di questi rilevatori di oggetti, lo si fa in fase di test su dati preregistrati. Gli errori che si verificano su nuovi dati non vengono rilevati senza ulteriori etichettature da parte dell'umano. In [88] viene proposto un metodo automatizzato per identificare gli errori commessi dai rilevatori di oggetti. Viene inoltre mostrato come le incoerenze nell'output del rilevatore di oggetti tra una coppia di immagini simili possono essere utilizzate come ipotesi per falsi negativi (rilevamenti mancati). Infine, utilizzando una nuova serie di funzionalità per ciascuna ipotesi, è possibile utilizzare un classificatore binario standard per trovare errori validi. Il metodo proposto può essere utilizzato con qualsiasi rilevatore di oggetti basato su telecamera. Viene mostrato che un rilevatore, un tracker e un classificatore all'avanguardia

addestrati solo su dati sintetici, possono identificare errori validi sul set di dati di tracciamento KITTI con una precisione media di 0,94.

Le reti neurali convoluzionali (CNN) sono comunemente utilizzate per controllare l'angolo di sterzata delle auto a guida autonoma. Il più delle volte, vengono utilizzate più telecamere a lungo raggio. In [89] viene presentato un nuovo modello per generare questi dati e aumentare le etichette usando solo una fotocamera fisheye (grandangolare) a corto raggio. Nel documento si presenta il simulatore e come può essere utilizzato come metrica coerente per la valutazione del controllo end-to-end laterale. Gli esperimenti sono stati condotti su un set di dati personalizzato corrispondente a oltre 10000 km e 200 ore di guida su strada aperta. Inoltre, il modello presentato, viene valutato su scenari di guida nel mondo reale, in una strada aperta e in una pista di prova personalizzata con evitamento ostacoli e curve strette. Infine, viene mostrato che nel simulatore basato su video del mondo reale, il modello finale è in grado di garantire un'autonomia superiore al 99% sulla strada urbana.

In [90] viene presentato un ambiente di simulazione che include strutture virtuali di un'auto progettata per test di guida autonoma, tracciati, ostacoli e ambienti. Un motore di gioco multipiattaforma, Unity 3D, consente all'auto di controllare e provare nuovi tracciati, parametri e calcoli nell'ambiente 3D prima del test in tempo reale. L'ambiente virtuale costruisce il dominio in modo tale che sia imitata l'attività di un'automobile autentica. Situazioni di guida tipiche sono state utilizzate per analizzare il modo in cui i sensori rispondono quando sono utilizzati in circostanze reali oltre che per confermare gli impatti di altri parametri sulle scene. Inoltre, in questo ambiente sono disponibili diverse opzioni, come: scegliere sensori flessibili, monitorare l'output, implementare qualsiasi algoritmo di guida autonoma, previsione dello sterzo, deep learning e apprendimento end-to-end.

I tradizionali sistemi di navigazione autonomi per il trasporto utilizzano scanner a raggio laser per costruire scene di guida 3D. Gli scanner laser attivi soffrono di una serie di fallimenti, come l'incapacità di rilevare superfici stradali bagnate, superfici scure e oggetti a grandi distanze. Al contrario, le videocamere passive sono immuni da questi fallimenti, ma l'elaborazione dei dati acquisiti è impegnativa. L'elevata grandezza dell'immagine di input richiede metodi analitici molto efficienti per consentire al sistema di funzionare in tempo reale. In [91] viene mostrato

come il riconoscimento degli oggetti è essenziale per un sistema di navigazione per la generalizzazione dei punti di riferimento appresi in nuovi scenari di guida. Viene presentata una rete neurale di apprendimento online per la navigazione interna utilizzando solo telecamere stereo. Una telecamera stereo è un tipo di telecamera con due o più lenti con un separato sensore di immagine per ciascuna. Ciò permette all'apparecchio di simulare la visione umana binoculare e quindi dà la capacità di catturare immagini tridimensionali [92]. Sebbene la rete proposta sia pensata per la navigazione sia interna che esterna, è stata testata solo negli ambienti interni in questo elaborato. Lo sviluppo futuro include la formazione e l'apprendimento in scenari di guida all'aperto.

In [93] viene data una panoramica del recente lavoro di riconoscimento degli oggetti svolto sul veicolo autonomo di Stanford e delle principali sfide lungo il percorso. L'obiettivo finale è fornire sistemi pratici di riconoscimento degli oggetti che consentano nuove applicazioni robotiche come i taxi autonomi che riconoscono i pedoni in arrivo, i robot personali che possono riconoscere oggetti specifici nella tua casa e le attrezzature agricole automatizzate che sono addestrate sul posto per riconoscere le piante e i materiali con cui devono interagire. Inoltre, viene discusso di come potrebbe essere necessario l'apprendimento online per utilizzare le grandi quantità di dati etichettati resi disponibili dall'apprendimento semi-supervisionato basato sul monitoraggio. Viene mostrato come errori di segmentazione e tracciamento sono la fonte più comune di errori di riconoscimento degli oggetti. In un'immagine esempio si mostra come un pedone venga segmentato insieme a un ampio tratto di marciapiede (falso negativo), mentre un'auto ben segmentata è identificata correttamente.

In questo capitolo si è voluta dare un'idea di quanti lavori aperti ci siano su questo argomento e in quante direzioni questi studi possano spaziare. La completa elencazione non sarebbe possibile, in quanto esistono moltissimi elaborati che pur non trattando l'argomento guida autonoma nel dettaglio, studiano particolari comportamenti di componenti che possono essere utilizzati per questa. Dunque, rientrerebbero in questo lungo elenco anche studi su lenti, processori, tipi di architetture, materiali e quant'altro.

7

CONCLUSIONI

La guida autonoma è un contesto nuovo e complesso su cui la ricerca e l'industria si sono concentrati negli ultimi anni. Un veicolo autonomo è un sistema tecnologico complesso, composto a sua volta da tecnologie molto differenti che devono lavorare bene insieme. Esistono diverse applicazioni in fase avanzata, a partire dai software e sensori che già vengono utilizzati in ambito automobilistico per l'assistenza alla guida. Inoltre, si trovano mezzi autonomi operativi in ambienti particolari: cantieri, porti, impianti industriali, dove la complessità dell'interazione fra macchine e umani è ridotta rispetto a una strada e il processo di apprendimento da parte dei veicoli autonomi è semplificato [94].

Come mostrato nel lavoro di Tesi, sono molte le tecnologie coinvolte in questo campo: diverse tipologie di sensori, algoritmi per la manipolazione e l'interpretazione dei dati ed in particolare le applicazioni di Intelligenza Artificiale e di Machine Learning. Un veicolo autonomo deve sempre sapere dove si trova e, non da meno, deve sempre sapere cosa fare. Infatti, la possibilità che il sistema si possa trovare in uno stato inconsistente, di non operabilità, non dovrebbe mai verificarsi, in quanto questo significherebbe il bisogno di intervento umano. Molte variabili influiscono su un sistema a guida autonoma. Tuttavia, negli studi analizzati durante la realizzazione di questo lavoro di Tesi, è stato notato come mancano ancora una chiara definizione di un modello di guasti della telecamera e una libreria software per la riproduzione dei fallimenti evidenziati. Una metodologia automatizzata di iniezione dei fallimenti e verifica delle prestazioni sarebbe di grande aiuto nel momento in cui devono essere fatte valutazioni riguardo applicazioni e sistemi di guida autonomi.

Con il lavoro di Tesi presentato si è voluto identificare un modello di guasti attraverso l'applicazione di una FMECA. Insieme all'individuazione

dei fallimenti sono state trovate anche le mitigazioni esistenti per risolvere un dato fallimento oppure prevenirlo. Dopodiché è stata sviluppata una libreria in codice Python nella quale si riproducono le alterazioni verificabili nei frame catturati da una telecamera. Come mostrato, alcuni dei fallimenti trovati si riconducono ad uno stesso tipo di alterazione (ad esempio Blurred e Broken VR). Per altri, invece, è stato impossibile replicare gli effetti sulla telecamera attraverso la libreria Python, in quanto devono essere considerati aspetti del sistema non modificabili sul simulatore: il fallimento No Action del componente ISP, ad esempio, produce un blocco del sistema di elaborazione, dunque il veicolo dovrebbe poter interrompere la marcia come politica di safe.

L'approccio adottato nella realizzazione dell'analisi è definito come segue:

- creazione e discussione di un modello di guasti per telecamere di veicoli nel dominio della guida autonoma, analizzando i diversi fallimenti, le loro cause e i loro effetti sul sistema,
- ricerca delle mitigazioni esistenti a livello di componente,
- riproduzione degli effetti dei guasti sull'immagine via software,
- conferma degli effetti e valutazione dei rischi associati alla sicurezza utilizzando come riferimento un'applicazione per la guida autonoma.

Quanto detto, viene ottenuto attraverso:

- la definizione di una FMECA (Failure Mode, Effects and Criticality Analysis, [21]) sui componenti di una telecamera RGB, supponendo che la telecamera sia collocata su di un veicolo (insieme ad un'attenta revisione della letteratura),
- lo sviluppo di una libreria di fallimenti in Python, resa pubblicamente disponibile in [22], per alterare le immagini secondo il modello dei guasti,
- l'iniezione di fallimenti nella telecamera frontale di un veicolo attraverso l'uso del simulatore di guida autonoma CARLA [10], dove è in esecuzione l'agente addestrato da [23].

Dall'iniezione dei fallimenti nel simulatore si ottengono molti dati. Per

ogni run eseguita vengono prodotti dei file .csv, ovvero i file di log che permettono di avere molte informazioni sulla marcia del veicolo: in che posizione si trova, che azione sta intraprendendo (accelerare, frenare, svoltare, ecc.), se ha colliso, a che velocità sta andando e così via. Attraverso l'elaborazione di tali file è stato possibile produrre tutti i risultati presentati nel Paragrafo 5.3. L'analisi dei dati svolta ha dato la possibilità di capire quali sono i fallimenti più critici per il sistema e quali meno. Come detto, però, anche quelli che hanno riportato un Success Rate alto o paragonabile a quello della Golden Run sono da considerarsi pericolosi: se si pensa al riconoscimento degli oggetti, in alcuni casi, l'effetto sfocato (blur) applicato a delle immagini in input per essere elaborate fa sì che l'oggetto raffigurato in questa venga scambiato con uno totalmente differente.

Per concludere, da questo lavoro se ne possono sviluppare tanti altri con l'obiettivo di definire al meglio un processo di valutazione delle prestazioni e della solidità di un dispositivo di acquisizione utilizzato in ambito guida autonoma. Con dispositivo di acquisizione, oltre che intendere la telecamera fisica, si vuole includere anche tutto il comparto software e decisionale che permette la marcia del veicolo senza intervento umano.

Alcune domande che possiamo porci partendo da tale lavoro sono:

- una volta modificata l'immagine attraverso l'iniezione del fallimento, è possibile risalire al frame originale corretto, cercando di eliminare il difetto inserito?
- c'è la possibilità di rilevare un frame non corretto, ovvero che presenta anomalie, basandoci su un training precedentemente effettuato per il dispositivo?
- invece di cambiare codice ad ogni run, cercando di posizionare vari artefatti nel frame, sarebbe possibile rendere questo processo automatico, facendo sì che gli artefatti vengano generati automaticamente via software, in base a forme, tipologie, scale di colori e quant'altro apprese precedentemente?
- che risultati otterrebbe un agente addestrato con la totalità dei fallimenti creati per questo lavoro di Tesi?
- i fallimenti creati per il simulatore di guida autonoma CARLA, produrrebbero gli stessi effetti, in termini di success rate ed altre metriche, su altre tipologie di simulatori?

- si potrebbe fare un confronto fra quello che farebbe un umano alla guida di un veicolo e, dalle immagini registrate dalla telecamera montata su tale veicolo, quello che farebbe, in termini di log, un software per la guida autonoma?

A partire dai risultati ottenuti sono stati identificati possibili nuovi contributi, ovvero:

- un rilevatore di guasti in grado di identificare immagini errate,
- l'addestramento di un agente per riconoscere le immagini errate, in modo che possa avvisare il sistema di guida,
- lo sviluppo di uno strumento interattivo per facilitare l'utilizzo della libreria software.

La lista potrebbe essere molto più lunga se si pensa a tutti i modi in cui si potrebbero testare gli apparecchi fisici e di conseguenza i software all'interno di essi. Pensando a quanti tipi di simulatori esistono ed alle diverse variabili che possono cambiare da uno all'altro ci si accorge come diventi difficile anche solo scegliere quale di questi sia il più affidabile o il più vicino alla realtà.

Forse ci vorrà ancora molto tempo prima di vedere un'auto realmente autonoma, che possa affrontare qualunque tipo di situazione e che prenda decisioni oggi considerate difficili o con "più risposte". Comunque, guardando il progresso che la tecnologia ha fatto negli ultimi 25 anni non c'è che aspettarsi delle sorprese.

Appendici

A

CODICE PYTHON PER SIMULAZIONE FALLIMENTI

I codici presentati sotto sono stati utilizzati per la modifica dei frame del simulatore. Ognuno di questi permette l'iniezione di uno o più modi di fallimento. Ognuno di questi è rappresentato e descritto in 5.2.2. Inoltre, la totalità dei codici sviluppati è scaricabile da [22].

Listing A.1: codice Fallimento NOISE

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('image_name.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

#Speckle Noise
gauss = np.random.normal(0,1,img.size)
gauss =
    gauss.reshape(img.shape[0],img.shape[1],img.shape[2]).astype('uint8')
noise = img + img * gauss

plt.figure(num='Fallimento NOISE')
plt.subplot(121),plt.imshow(img),plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(noise),plt.title('Noise')
plt.xticks([]), plt.yticks([])
plt.show()
```

Listing A.2: codice Fallimento NO DEMOSAICING

```
from PIL import Image
from numpy import*
import cv2
from matplotlib import pyplot as plt
```

```

# Open image and put it in a numpy array
srcArray = cv2.imread('image_name.jpg')
srcArray1 = Image.open("image_name.jpg")

w, h, _ = srcArray.shape
# Create target array, twice the size of the original image
resArray = zeros((2*w, 2*h, 3), dtype=uint8)

# Map the RGB values in the original picture according to the BGGR
# pattern#
# Blue
resArray[::2, ::2, 2] = srcArray[:, :, 2]

# Green (top row of the Bayer matrix)
resArray[1::2, ::2, 1] = srcArray[:, :, 1]

# Green (bottom row of the Bayer matrix)
resArray[::2, 1::2, 1] = srcArray[:, :, 1]

# Red
resArray[1::2, 1::2, 0] = srcArray[:, :, 0]

resArray = cv2.cvtColor(resArray, cv2.COLOR_BGR2RGB)
# Save the image
#imgOut = Image.fromarray(resArray, "RGB")
#imgOut.save("nodemos.png")

plt.figure(num='Fallimento DEMOSAICING')
plt.subplot(121), plt.imshow(srcArray1), plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(resArray), plt.title('No Demosaicing')
plt.xticks([]), plt.yticks([])
plt.show()

```

Listing A.3: codice Fallimento BROKEN-LENS/ICE/BANDING/...

```

import numpy as np
from matplotlib import pyplot as plt
from PIL import Image

```

```

img = Image.open("image_name.jpg")

img2 = Image.open("broken1.png").convert(img.mode) # inserire fra
apici il nome del file dell'immagine da sovrapporre a quella
letta con Image.open("...")

img2 = img2.resize(img.size)

img3 = Image.blend(img,img2,0.5) # cambiare valore in base al
fallimento e all'immagine da sovrapporre (valore di
trasparenza/prevalenza)

plt.figure(num='Fallimento BROKEN LENS')
plt.subplot(121),plt.imshow(img),plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img3),plt.title('Broken Lens')
plt.xticks([]), plt.yticks([])
plt.show()

```

Listing A.4: codice Fallimento DEADPIXEL

```

import numpy as np
import cv2
from matplotlib import pyplot as plt
from PIL import Image

img = cv2.imread('image_name1.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

"""

If the 'ndarray' of the image read by OpenCV imread() is converted
to a
'PIL.Image' object and saved, the image with the wrong color is
saved.

If you want to convert 'ndarray' and 'PIL.Image' objects to use
both Pillow
and OpenCV functions, you need to convert BGR and RGB.
"""

img1 = Image.open('image_name.jpg')

```

```

pixels = img1.load()

width, height = img1.size

for i in range(0, width):
    for j in range(0, height):
        if i==150 and j==90:
            pixels[i,j] = (255,0,0) # rosso per notarlo meglio
            nell'immagine

img1 = np.array(img1) # esempio trasformazione da
                      # PIL.JpegImagePlugin.JpegImageFile a 'numpy.ndarray'

plt.figure(num='Fallimento DEADPIXEL Configurazione 1 – 1 pixel Nero')
plt.subplot(121),plt.imshow(img),plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img1),plt.title('DEADPIXEL - Configurazione 1')
plt.xticks([]), plt.yticks([])
plt.show()

```

Listing A.5: codice Fallimento DEADPIXEL50

```

import numpy as np
import cv2
from matplotlib import pyplot as plt
import random

img = cv2.imread('image_name.jpg')

img1 = cv2.imread('image_name.jpg')

h, w, _ = img1.shape

count = 0
w1 = w2 = int((w-70)/10) # in base a questo 10 e al 5 sotto decido
                          # quanti pixel neri inserire
h1 = h2 = int((h-70)/5)
w2 = w3 = w2 - 5
h2 = h3 = h2 - 5
countpixel = 0

```

```

# naturalmente bisogna sempre controllare con un paio di prove (o
# calcolandolo) se si esce dai confini dell'immagine in
# elaborazione

for y in range(0, 5):
    h2 = h2 + (h1*y)
    for x in range(0, 10):
        ...
        print("h2 e w2: " + str(h2) + " e " + str(w2))
        inserimento di randomicita' per quanto riguarda le coordinate
        in cui vengono posti i pixel neri (in fase sperimentale
        considerato un valore fisso, in quanto deve essere
        replicabile e quindi sempre uguale):
        img1[(h2+random.randint(1,20)),(w2+random.randint(1,20))]
            = (0, 0, 0)
        versione per fase sperimentale sotto -> "tutto fisso")
        ...
        img1[h2,w2] = (255, 0, 0) #pixels rossi (per visibilita') in
        coordinate [h2,w2]
        countpixel = countpixel + 1
        w2 = w2 + w1
        if x==9:
            h2 = h3
            w2 = w3

print(countpixel) #valore di controllo per l'inserimento del giusto
# numero di pixel neri

plt.figure(num='Fallimento DEADPIXEL Configurazione 2 – 50 pixel Neri')
plt.subplot(121),plt.imshow(img),plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img1),plt.title('DEADPIXEL –
Configurazione 2')
plt.xticks([]), plt.yticks([])
plt.show()

```

Listing A.6: codice Fallimento DEADPIXEL200

```

import numpy as np
import cv2
from matplotlib import pyplot as plt

```

```

import random

img = cv2.imread('image_name.jpg')

img1 = cv2.imread('image_name.jpg')

h, w, _ = img1.shape

count = 0
w1 = w2 = int(w/20) # in base a questo 20 e al 10 sotto decido
quanti pixel neri inserire
h1 = h2 = int(h/10)
w2 = w3 = w2 - 5
h2 = h3 = h2 - 5
countpixel = 0

# naturalmente bisogna sempre controllare con un paio di prove (o
# calcolandolo) se si esce dai confini dell'immagine in
# elaborazione

for y in range(0, 10):
    h2 = h2 + (h1*y)
    for x in range(0, 20):
        #print("h2 e w2: " + str(h2) + " e " + str(w2))
        img1[h2,w2] = (255, 0, 0) #pixels rossi (per visibilita') in
        coordinate [h2,w2]
        countpixel = countpixel + 1
    w2 = w2 + w1
    if x==19:
        h2 = h3
        w2 = w3

img1 = Image.fromarray(img1) # esempio conversione

print(countpixel) #valore di controllo per l'inserimento del giusto
# numero di pixel neri

plt.figure(num='Fallimento DEADPIXEL Configurazione 3 – 200 pixel
Neri')
plt.subplot(121),plt.imshow(img),plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img1),plt.title('DEADPIXEL –

```

```
    Configurazione 3')
plt.xticks([]), plt.yticks([])
plt.show()
```

Nel codice sovrastante è stato pensato anche di inserire una certa randomicità nella locazione dei pixel neri, introdotta con il seguente codice:

```
img1[(h2+random.randint(1,20)),(w2+random.randint(1,20))] = (0,0,0).
```

Listing A.7: codice Fallimento DEADPIXEL1000

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
import random

img = cv2.imread('image_name.jpg')

img1 = cv2.imread('image_name.jpg')

h, w, _ = img1.shape

count = 0
w1 = w2 = int(w/40)
h1 = h2 = int(h/25)
w2 = w3 = w2 - 5
h2 = h3 = h2 - 5
countpixel = 0

# naturalmente bisogna sempre controllare con un paio di prove (o
# calcolandolo) se si esce confini dell'immagine in elaborazione

for y in range(0, 25):
    h2 = h2 + (h1*y)
    for x in range(0, 40):
        #print("h2 e w2: " + str(h2) + " e " + str(w2))
        img1[h2,w2] = (255, 0, 0) #pixels rossi (per visibilita') in
        # coordinate [h2,w2], per pixel neri = (0,0,0)
        countpixel = countpixel + 1
    w2 = w2 + w1
    if x==39:
```

```

h2 = h3
w2 = w3

img1 = Image.fromarray(img1) # esempio conversione

print(countpixel) #valore di controllo per l'inserimento del giusto
                   numero di pixel neri

plt.figure(num='Fallimento DEADPIXEL Configurazione 4 – 1000 pixel
Neri')
plt.subplot(121), plt.imshow(img), plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(img1), plt.title('DEADPIXEL –
Configurazione 4')
plt.xticks([]), plt.yticks([])
plt.show()

```

Listing A.8: codice Fallimento DEADPIXEL-Lines

```

import numpy as np
import cv2
from matplotlib import pyplot as plt
import random
from PIL import Image

img = Image.open('originale.jpg')

img1 = Image.open('originale.jpg')
pixels = img1.load()
width, height = img1.size

#orizzontale
for i in range(0,width):
    for j in range(0,height):
        if i==120 and j==50:
            for k in range(0,20):
                pixels[i+k,j] = (0,0,0)

#verticale
for i in range(0,width):
    for j in range(0,height):
        if i==125 and j==20:#V

```

```

for k in range(0,35):
    pixels[i,j+k] = (0,0,0)

#obliqua a destra
count = 0
for i in range(0,width):
    for j in range(0,height):
        if i>=225 and i<=(225+count) and j>100 and count<45:
            pixels[i,j] = (0,0,0)
            count = count + 1
        i = i + 1

#ostacolo centrale
for i in range(0,width):
    for j in range(0,height):
        if i==177 and j==120:
            for p in range(0,10):
                for k in range(0,46):
                    pixels[i+k,j+p] = (0,0,0)

img1 = np.array(img1)

plt.figure(num='Fallimento DEADPIXEL Configurazioni Linee')
plt.subplot(121),plt.imshow(img),plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img1),plt.title('DEADPIXEL –
Configurazioni Linee')
plt.xticks([]), plt.yticks([])
plt.show()

```

Listing A.9: codice Fallimento BLURRED

```

import cv2
from matplotlib import pyplot as plt

img = cv2.imread('image_name.jpg')

blur = cv2.blur(img,(5,5)) #cambiare valori per aumentare o
                           diminuire sfocatura

#blur e img --> <class 'numpy.ndarray'>

#il codice sotto serve come presentazione per mostrare il fallimento

```

```
plt.figure(num='Fallimento BLURRED')
plt.subplot(121),plt.imshow(img),plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur),plt.title('Sfocata')
plt.xticks([]), plt.yticks([])
plt.show()
```

Listing A.10: codice Fallimento BRIGHTNESS

```
from PIL import Image, ImageEnhance
from matplotlib import pyplot as plt

"""
Brightness: This class can be used to control the
brightness of an image. An enhancement
factor of 0.0 gives a black image.
A factor of 1.0 gives the original image.
"""

img = Image.open("image_name.jpg")

enhancer = ImageEnhance.Brightness(img)

factor = 1
img1 = enhancer.enhance(factor)

factor = 3.5
img2 = enhancer.enhance(factor)

# img --> <class 'PIL.JpegImagePlugin.JpegImageFile'>
# img1 e img2 --> <class 'PIL.Image.Image'>

plt.figure(num='Fallimento BRIGHTNESS')
plt.subplot(121),plt.imshow(img1),plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img2),plt.title('Luminosita' ' Aumentata')
plt.xticks([]), plt.yticks([])
plt.show()
```

Listing A.11: codice Fallimento GREYSCALE

```
import numpy as np
import cv2
from PIL import Image
from matplotlib import pyplot as plt

img = cv2.imread('image_name.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

grey = Image.open('image_name.jpg').convert('LA')
# in alcuni casi .convert('LA') potrebbe non funzionare e l'altra
# conversione
# che ho utilizzato e' .convert('L')

plt.figure(num='Fallimento NO BAYER FILTER')
plt.subplot(121),plt.imshow(img),plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(grey),plt.title('Scala di Grigi')
plt.xticks([]), plt.yticks([])
plt.show()
```

Listing A.12: codice Fallimento SHARPNESS

```
from PIL import Image, ImageEnhance
from matplotlib import pyplot as plt

"""

Sharpness: This class can be used to adjust the sharpness
of an image. An enhancement factor of 0.0 gives a
blurred image, a factor of 1.0 gives the original
image, and a factor of 2.0 gives a sharpened image.
"""

img = Image.open("image_name.jpg") #se apri l'immagine con
# Image.open() non ci sono problemi
# con i canali dei colori, se invece usi cv2.imread()
# devi usare cv2.cvtColor(img, cv2.COLOR_BGR2RGB) per avere
# colori corretti

enhancer = ImageEnhance.Sharpness(img)

factor = 1
img1 = enhancer.enhance(factor)
```

```

...
esempi:

factor = 1 --> img Originale
factor = 3 --> img visibilmente piu' definita (non sempre un bene)
factor = -3.5 --> img sfocata, non nitida (si avvicina molto al
    guasto Blurred)
...

factor = -3.5
img2 = enhancer.enhance(factor)

plt.figure(num='Fallimento SHARPNESS')
plt.subplot(121),plt.imshow(img1),plt.title('Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img2),plt.title('Sharpened')
plt.xticks([]), plt.yticks([])
plt.show()

```

Listing A.13: codice Fallimento WHITE and BLACK

```

from PIL import Image
from matplotlib import pyplot as plt

picture = Image.open("image_name.jpg")
picture1 = Image.open("image_name.jpg")

pixels = picture.load()
width, height = picture.size

for i in range(0,width):
    for j in range(0,height):
        pixels[i,j] = (255, 255, 255) # se si cambia con (0,0,0) si
            ottiene BLACK

#picture e picture1 --> <class 'PIL.JpegImagePlugin.JpegImageFile'>

plt.figure(num='Fallimento WHITE')
plt.subplot(121),plt.imshow(picture1),plt.title('Originale')
plt.xticks([]), plt.yticks([])

```

```
plt.subplot(122),plt.imshow(picture),plt.title('White')
plt.xticks([]), plt.yticks([])
plt.show()
```

Listing A.14: codice Fallimento NO CHROMATIC ABERRATION CORRECTION

```
"""
Kromo V0.3
==== Author ====
Yoonsik Park
park.yoonsik@icloud.com

==== Description ====
Use the command line interface to add chromatic abberation and
lens blur to your images, or import some of the functions below.
```

Il codice seguente e' stato riadattato alle necessita' del lavoro di tesi svolto. Invece che un input da tastiera, adesso, il seguente codice, permette di inserire il file da modificare direttamente all'interno, cosicche' quando il codice viene eseguito, non si debba attendere la vera e propria esecuzione dopo l'inserimento dell'immagine da modificare.

```
"""
```

```
from PIL import Image
import numpy as np
import math
from typing import List

from matplotlib import pyplot as plt

def cartesian_to_polar(data: np.ndarray) -> np.ndarray:
    """Returns the polar form of <data>
    """
    width = data.shape[1]
    height = data.shape[0]
    assert (width > 2)
    assert (height > 2)
    assert (width % 2 == 1)
    assert (height % 2 == 1)
```

```

perimeter = 2 * (width + height - 2)
halfdiag = math.ceil(((width ** 2 + height ** 2) ** 0.5) / 2)
halfw = width // 2
halfh = height // 2
ret = np.zeros((halfdiag, perimeter))

# Don't want to deal with divide by zero errors...
ret[0:(halfw + 1), halfh] = data[halfh, halfw::-1]
ret[0:(halfw + 1), height + width - 2 +
halfh] = data[halfh, halfw:(halfw * 2 + 1)]
ret[0:(halfh + 1), height - 1 + halfw] = data[halfh:(halfh * 2 +
1), halfw]
ret[0:(halfh + 1), perimeter - halfw] = data[halfh::-1, halfw]

# Divide the image into 8 triangles, and use the same
# calculation on
# 4 triangles at a time. This is possible due to symmetry.
# This section is also responsible for the corner pixels
for i in range(0, halfh):
    slope = (halfh - i) / (halfw)
    diagx = ((halfdiag ** 2)/(slope ** 2 + 1)) ** 0.5
    unit_xstep = diagx / (halfdiag - 1)
    unit_ystep = diagx * slope / (halfdiag - 1)
    for row in range(halfdiag):
        ystep = round(row * unit_ystep)
        xstep = round(row * unit_xstep)
        if ((halfh >= ystep) and halfw >= xstep):
            ret[row, i] = data[halfh - ystep, halfw - xstep]
            ret[row, height - 1 - i] = data[halfh + ystep, halfw -
                xstep]
            ret[row, height + width - 2 + i] = data[halfh + ystep,
                halfw + xstep]
            ret[row, height + width + height - 3 - i] = data[halfh -
                ystep, halfw + xstep]
        else:
            break

# Remaining 4 triangles
for j in range(1, halfw):
    slope = (halfh) / (halfw - j)
    diagx = ((halfdiag ** 2)/(slope ** 2 + 1)) ** 0.5
    unit_xstep = diagx / (halfdiag - 1)

```

```

unit_ystep = diagx * slope / (halfdiag - 1)
for row in range(halfdiag):
    ystep = round(row * unit_ystep)
    xstep = round(row * unit_xstep)
    if (halfw >= xstep and halfh >= ystep):
        ret[row, height - 1 + j] = data[halfh + ystep, halfw -
            xstep]
        ret[row, height + width - 2 - j] = data[halfh + ystep,
            halfw + xstep]
        ret[row, height + width + height - 3 + j] = data[halfh
            - ystep, halfw + xstep]
        ret[row, perimeter - j] = data[halfh - ystep, halfw -
            xstep]
    else:
        break
return ret

def polar_to_cartesian(data: np.ndarray, width: int, height: int)
-> np.ndarray:
    """Returns the cartesian form of <data>.

    <width> is the original width of the cartesian image
    <height> is the original height of the cartesian image
    """
    assert (width > 2)
    assert (height > 2)
    assert (width % 2 == 1)
    assert (height % 2 == 1)
    perimeter = 2 * (width + height - 2)
    halfdiag = math.ceil(((width ** 2 + height ** 2) ** 0.5) / 2)
    halfw = width // 2
    halfh = height // 2
    ret = np.zeros((height, width))

    # Don't want to deal with divide by zero errors...
    ret[halfh, halfw:-1] = data[0:(halfw + 1), halfh]
    ret[halfh, halfw:(halfw * 2 + 1)] = data[0:(halfw + 1),
    height + width - 2 + halfh]
    ret[halfh:(halfh * 2 + 1), halfw] = data[0:(halfh + 1), height -
        1 + halfw]
    ret[halfh::-1, halfw] = data[0:(halfh + 1), perimeter - halfw]

```

```

# Same code as above, except the order of the assignments are
# switched

for i in range(0, halfh):
    slope = (halfh - i) / (halfw)
    diagx = ((halfdiag ** 2)/(slope ** 2 + 1)) ** 0.5
    unit_xstep = diagx / (halfdiag - 1)
    unit_ystep = diagx * slope / (halfdiag - 1)
    for row in range(halfdiag):
        ystep = round(row * unit_ystep)
        xstep = round(row * unit_xstep)
        if ((halfh >= ystep) and halfw >= xstep):
            ret[halfh - ystep, halfw - xstep] = \data[row, i]
            ret[halfh + ystep, halfw - xstep] = \data[row, height -
                1 - i]
            ret[halfh + ystep, halfw + xstep] = \data[row, height +
                width - 2 + i]
            ret[halfh - ystep, halfw + xstep] = \data[row, height +
                width + height - 3 - i]
        else:
            break

for j in range(1, halfw):
    slope = (halfh) / (halfw - j)
    diagx = ((halfdiag ** 2)/(slope ** 2 + 1)) ** 0.5
    unit_xstep = diagx / (halfdiag - 1)
    unit_ystep = diagx * slope / (halfdiag - 1)
    for row in range(halfdiag):
        ystep = round(row * unit_ystep)
        xstep = round(row * unit_xstep)
        if (halfw >= xstep and halfh >= ystep):
            ret[halfh + ystep, halfw - xstep] = \data[row, height -
                1 + j]
            ret[halfh + ystep, halfw + xstep] = \data[row, height +
                width - 2 - j]
            ret[halfh - ystep, halfw + xstep] = \data[row, height +
                width + height - 3 + j]
            ret[halfh - ystep, halfw - xstep] = \data[row,
                perimeter - j]
        else:
            break

```

```

# Repairs black/missing pixels in the transformed image
for i in range(1, height - 1):
    for j in range(1, width - 1):
        if ret[i, j] == 0:
            ret[i, j] = (ret[i - 1, j] + ret[i + 1, j]) / 2
return ret

def get_gauss(n: int) -> List[float]:
    """Return the Gaussian 1D kernel for a diameter of <n>
    Referenced from: https://stackoverflow.com/questions/11209115/
    """
    sigma = 0.3 * (n/2 - 1) + 0.8
    r = range(-int(n/2), int(n/2)+1)
    new_sum = sum([1 / (sigma * math.sqrt(2*math.pi)) *
                  math.exp(-float(x)**2/(2*sigma**2)) for x in r])
    # Ensure that the gaussian array adds up to one
    return [(1 / (sigma * math.sqrt(2*math.pi)) *
             math.exp(-float(x)**2/(2*sigma**2))) / new_sum for x in r]

def vertical_gaussian(data: np.ndarray, n: int) -> np.ndarray:
    """Performs a Gaussian blur in the vertical direction on <data>.
    Returns
    the resulting numpy array.

    <n> is the radius, where 1 pixel radius indicates no blur
    """
    padding = n - 1
    width = data.shape[1]
    height = data.shape[0]
    padded_data = np.zeros((height + padding * 2, width))
    padded_data[padding: -padding, :] = data
    ret = np.zeros((height, width))
    kernel = None
    old_radius = -1
    for i in range(height):
        radius = round(i * padding / (height - 1)) + 1
        # Recreate new kernel only if we have to
        if (radius != old_radius):
            old_radius = radius
            kernel = np.tile(get_gauss(1 + 2 * (radius - 1)), (width,

```

```

    1)).transpose()
ret[i, :] = np.sum(np.multiply(padded_data[padding + i -
    radius + 1:padding + i + radius, :], kernel), axis=0)
return ret

def add_chromatic(im, strength: float = 1, no_blur: bool = False):
    """Splits <im> into red, green, and blue channels, then performs
    a
    1D Vertical Gaussian blur through a polar representation.
    Finally,
    it expands the green and blue channels slightly.
    <strength> determines the amount of expansion and blurring.
    <no_blur> disables the radial blur
    """
    r, g, b = im.split()
    rdata = np.asarray(r)
    gdata = np.asarray(g)
    bdata = np.asarray(b)
    if no_blur:
        # channels remain unchanged
        rfinal = r
        gfinal = g
        bfinal = b
    else:
        rpolar = cartesian_to_polar(rdata)
        gpolar = cartesian_to_polar(gdata)
        bpolar = cartesian_to_polar(bdata)

    bluramount = (im.size[0] + im.size[1] - 2) / 100 * strength
    if round(bluramount) > 0:
        rpolar = vertical_gaussian(rpolar, round(bluramount))
        gpolar = vertical_gaussian(gpolar, round(bluramount*1.2))
        bpolar = vertical_gaussian(bpolar, round(bluramount*1.4))

    rcartes = polar_to_cartesian(rpolar, width=rdata.shape[1],
        height=rdata.shape[0])
    gcartes = polar_to_cartesian(gpolar, width=gdata.shape[1],
        height=gdata.shape[0])
    bcartes = polar_to_cartesian(bpolar, width=bdata.shape[1],
        height=bdata.shape[0])

```

```

rfinal = Image.fromarray(np.uint8(rcartes), 'L')
gfinal = Image.fromarray(np.uint8(gcartes), 'L')
bfinal = Image.fromarray(np.uint8(bcartes), 'L')

# enlarge the green and blue channels slightly, blue being the
# most enlarged
gfinal = gfinal.resize((round((1 + 0.018 * strength) *
    rdata.shape[1]), round((1 + 0.018 * strength) *
    rdata.shape[0])), Image.ANTIALIAS)
bfinal = bfinal.resize((round((1 + 0.044 * strength) *
    rdata.shape[1]), round((1 + 0.044 * strength) *
    rdata.shape[0])), Image.ANTIALIAS)

rwidth, rheight = rfinal.size
gwidth, gheight = gfinal.size
bwidth, bheight = bfinal.size
rhdiff = (bheight - rheight) // 2
rwdiff = (bwidth - rwidth) // 2
ghdiff = (bheight - gheight) // 2
gwdiff = (bwidth - gwidth) // 2

# Centre the channels
im = Image.merge("RGB", (rfinal.crop((-rwdiff, -rhdiff, bwidth -
    rwdiff, bheight - rhdiff)),gfinal.crop((-gwdiff, -ghdiff,
    bwidth - gwdiff, bheight - ghdif)),bfinal))

# Crop the image to the original image dimensions
return im.crop((rwdiff, rhdiff, rwidth + rwdiff, rheight +
    rhdif))

def add_jitter(im, pixels: int = 1):
    """Adds a small pixel offset to the Red and Blue channels of
    <im>,
    resulting in a classic chromatic fringe effect. Very cheap
    computationally.
    <pixels> how many pixels to offset the Red and Blue channels
    """
    if pixels == 0:
        return im.copy()
    r, g, b = im.split()
    rwidth, rheight = r.size
    gwidth, gheight = g.size

```

```

        bwidth, bheight = b.size
        im = Image.merge("RGB", (r.crop((pixels, 0, rwidth + pixels,
            rheight)),g.crop((0, 0, gwidth, gheight)),b.crop((-pixels,
            0, bwidth - pixels, bheight))))
        return im

def blend_images(im, og_im, alpha: float = 1, strength: float = 1):
    """Blends original image <og_im> as an overlay over <im>, with
    an alpha value of <alpha>. Resizes <og_im> with respect to
    <strength>,
    before adding it as an overlay.
    """
    og_im.putalpha(int(255 * alpha))
    og_im = og_im.resize((round((1 + 0.018 * strength) *
        og_im.size[0]), round((1 + 0.018 * strength) *
        og_im.size[1])), Image.ANTIALIAS)

    hdiff = (og_im.size[1] - im.size[1]) // 2
    wdiff = (og_im.size[0] - im.size[0]) // 2
    og_im = og_im.crop((wdiff, hdiff, wdiff + im.size[0], hdiff +
        im.size[1]))
    im = im.convert('RGBA')

    final_im = Image.new("RGBA", im.size)
    final_im = Image.alpha_composite(final_im, im)
    final_im = Image.alpha_composite(final_im, og_im)
    final_im = final_im.convert('RGB')
    return final_im

if __name__ == '__main__':
    ...
    parte commentata per input da codice e non da tastiera

import argparse
parser = argparse.ArgumentParser(
description="Apply chromatic aberration and lens blur to images")
parser.add_argument("filename", help="input filename")
parser.add_argument("-s", "--strength", type=float, default=1.0,
help="set blur/aberration strength, defaults to 1.0")

```

```

parser.add_argument("-j", "--jitter", type=int, default=0,
                    help="set color channel offset pixels, defaults to 0")
parser.add_argument("-y", "--overlay", type=float, default=0.0,
                    help="alpha of original image overlay, defaults to 0.0")
parser.add_argument(
    "-n", "--noblur", help="disable radial blur", action="store_true")
parser.add_argument(
    "-o", "--out", help="write to OUTPUT (supports multiple formats)")
parser.add_argument(
    '-v', '--verbose', help="print status messages",
    action="store_true")
args = parser.parse_args()

ifile = args.filename
im = Image.open(ifile)
im1 = Image.open(ifile)
...
im = Image.open("image_name.jpg")

# Ensure width and height are odd numbers
if (im.size[0] % 2 == 0 or im.size[1] % 2 == 0):
    if (im.size[0] % 2 == 0):
        im = im.crop((0, 0, im.size[0] - 1, im.size[1]))
        im.load()
    if (im.size[1] % 2 == 0):
        im = im.crop((0, 0, im.size[0], im.size[1] - 1))
        im.load()

og_im = im.copy()
img = Image.open("image_name.jpg")
im = add_chromatic(im, strength=2, no_blur=True)# metodo da
      invocare per aggiungere effetto
#modificare i valori per ottenere effetti piu' o meno calcati
      (vedere descrizione sotto il metodo)

# im --> <class 'PIL.Image.Image'>
# img --> <class 'PIL.JpegImagePlugin.JpegImageFile'>

plt.figure(num='Fallimento NO CHROMATIC ABERRATION
CORRECTION')
plt.subplot(121), plt.imshow(img), plt.title('Originale')
plt.xticks([]), plt.yticks([])

```

```
plt.subplot(122),plt.imshow(im),plt.title('No Chromatic Aberration  
Correction')  
plt.xticks([]), plt.yticks([])  
plt.show()
```

B

TABELLA FMECA VIDEOCAMERA

La tabella seguente è il prodotto dell’analisi condotta su di una Videocamera, pensando al fatto che questa possa essere utilizzata in ambito Self-Driving. Naturalmente, l’analisi non è stata eseguita su una Videocamera realmente esistente, in quanto non in possesso di un vero e proprio dispositivo da poter utilizzare per un’analisi sicuramente più appropriata e veritiera. Ad ogni modo, si è cercato di rendere il più concreto possibile il lavoro svolto, pensando ad ogni possibile casistica osservabile.

La FMECA eseguita è scaricabile da [42].

Figura 66.: FMECA della Videocamera

FMECA		System:	Date:
ID	Component	Subsystem:	Prepared by:
Severity levels with respect to Detectability (IV = Minor, III = Major, II = Critical, I = Catastrophic)			
DESCRIPTION OF THE FAILURE	EFFECTS OF FAILURE	Existing mitigations	Detectability/easiness to implement
1 Obiettivo BLURRED	Macro-Failure Mode Immagine non a fuoco	Macro-Failure Mode Local Global Le fasi di elaborazione successive non riescono ad elaborare/controllare questo fenomeno se non corretto	Le fasi di elaborazione successive non riescono ad elaborare/controllare questo fenomeno se non corretto
2 Obiettivo BLACK	Macro-Failure Mode Immagine nera	Macro-Failure Mode Local Global Le fasi di elaborazione successive non riescono ad elaborare/controllare questo fenomeno se non corretto	Le fasi di elaborazione successive non riescono ad elaborare/controllare questo fenomeno se non corretto
3 Obiettivo WHITE	Macro-Failure Mode Immagine bianca	Macro-Failure Mode Local Global Le fasi di elaborazione successive non riescono ad elaborare/controllare questo fenomeno se non corretto	Le fasi di elaborazione successive non riescono ad elaborare/controllare questo fenomeno se non corretto
4 Obiettivo BROKEN VR	Macro-Failure Mode Immagine mossa	Macro-Failure Mode Local Global Le fasi di elaborazione successive non riescono ad elaborare/controllare questo fenomeno se non corretto	Le fasi di elaborazione successive non riescono ad elaborare/controllare questo fenomeno se non corretto
5 Obiettivo BROKEN LENS	Macro-Failure Mode Immagine con linee di vario tipo	Macro-Failure Mode Local Global Le fasi di elaborazione successive all'acquisizione presentano problemi in base alla fattura della linea (grandi, piccole, multiple...) e la telecamera processa un'immagine che presenta linee di vario tipo (single o multiple)	Le fasi di elaborazione successive all'acquisizione presentano problemi in base alla fattura della linea (grandi, piccole, multiple...) e la telecamera processa un'immagine che presenta linee di vario tipo (single o multiple)

Purtroppo in questo caso la fotocamera, o videocamera che sia, acquisisce delle immagini prive di informazione, dunque è pressoché impossibile rialzare al fotogramma corretto che contiene dati con cui il computer centrale può prendere le decisioni. In questo caso, a livello di componente non può essere trovata una mitigazione, in quanto si deve più ragionare a livello di sistema. Infatti, una soluzione potrebbe essere quella di mettere un controllore di fotogrammi che, dopo l'acquisizione, decide se l'immagine contiene informazioni utili oppure no.

Anche in questo caso ci possiamo ricordare ai paper (che non sono gli unici sull'argomento) riportati per l'immagine Blurred, dunque: "Blind motion deblurring from a single image using sparse approximation" e "Separable Kernel for Image Deblurring". Anche qui, la mitigazione non può essere trovata a livello di componenti, dato che l'obiettivo non può svolgere tutti i controlli che possono essere fatti dopo l'acquisizione del fotogramma da controllori interni che provvedono al rilevamento e correzione (quando possibile) di frammenti di questo tipo.

L - M

I - III

M - H

I - II

L

II - III

IV

6 Obiettivo	DIRTY-INT	Immagine sporca da detriti interni	Telecamera processa un'immagine non fedele alla realtà, che presenta varie macchie di vario genere	Le fasi di elaborazione successive all'acquisizione potrebbero avere dei problemi in base al tipo di sporco ammucchiato all'interno dell'obiettivo fra una lente e l'altra	L - M	IV	
7 Obiettivo	DIRTY-EXT	Immagine sporca da detriti esterni	Telecamera processa un'immagine non fedele alla realtà, che presenta varie macchie di vario genere	Le fasi di elaborazione successive all'acquisizione potrebbero avere dei problemi in base al tipo di sporco depositatosi sulla lente (all'esterno)	L	IV	
8 Obiettivo	FLARE	Immagine che presenta il cosiddetto flare, fenomeno dovuto al riflesso del sole o altre sorgenti luminose sulle lenti interne all'obiettivo	Telecamera processa immagini con macchie di vario colore	Le fasi successive di elaborazione, seppur correttamente, saranno influenzate dalla presenza di queste macchie	M	II - III	
9 Obiettivo	RAIN	Immagine che presenta macchie dovute a gocce d'acqua	Telecamera processa piccole macchie per lo più bianche	I modi per rilevare ed eliminare le macchie di questo tipo (ad esempio quando si guarda un vetro su cui è appena piovuto) sono vari: si va dall'apparato meccanico che cerca di togliere le macchie in questione con delle specie di tergilavastili per la lente arrivando alla soluzione software nella quale si cerca di eliminare tali macchie in fase di elaborazione. Per il secondo caso il paper "Restoring An Image Taken Through a Window Covered with Dirt or Rain". La soluzione proposta da questi paper è comunque da considerare a livello di sistema, in quanto elaborazione per correggere l'immagine non avviene nello stesso componente Obiettivo, ma dopo la fase di acquisizione.	L - M	IV	
10 Obiettivo	CONDENSATION	Atomi sulle lenti	Telecamera ancora funzionante, ma ogni immagine che viene acquisita presenta debolezza modifica che la rendono inutilizzabile	Le fasi successive all'acquisizione dovranno elaborare immagini nella maggior parte del caso prive di significato (contenuto) finché il fenomeno non si attenua	Ci sono vari metodi con cui ci si propone di rilevare e poi rimuovere anche automaticamente queste tipologie di macchie dovute a sorgenti luminose di vario genere. In particolare si cerca di rimuoverle in: "Automated Lens Flare Removal".	M	II - III
11 Obiettivo	HEAT	Possibile evaporaione della lubrificazione delle parti mobili	Telecamera funzionante, ma impossibilità ad esempio di utilizzare lo zoom (se esistente) ed altri componenti che hanno lo scopo di rendere l'immagine più chiara possibile (strumenti per la messa a fuoco ecc.)	Le fasi successive all'acquisizione potrebbero dover processare immagini non chiare, con la possibilità di interpretarle erroneamente	In questo caso, una possibile mitigazione è quella di utilizzare guarnizioni e altri componenti del corpo macchina soggetti a deformazione di vario genere, a trova di calore (temperature estreme). Se non sono utilizzati componenti di questo genere si può incorrere in una rottura non sempre riparabile. Una soluzione si ritrova nel campo della videosorveglianza, in cui l'azienda Axis Communications, leader nel settore, fabbrica telecamere a prova di temperature estremistiche come i AXI-Q60-C PTZ, dotata di raffreddamento attivo integrato, che permette una maggiore affidabilità in ambienti ad alta temperatura (queste telecamere possono operare in un doppio involucro che crea uno spazio d'aria vuoto fra il contenitore primario e quello secondario ("Noncondensing camera housing window assembly"). Un'altra soluzione può essere trovata nel brevetto "Video camera unit, protective enduse and power circuit for same, particularly for use in vehicles", in quanto pensato proprio per un uso su veicolo.	L	III - IV

TABELLA FMECA VIDEOCAMERA

12	Obiettivo	SAND	Possibile corrosione dei componenti esterni dell'obiettivo (percentuale di sabbia nella sabbia), con conseguente entrata di agenti esterni	Determinati materiali ad una temperatura di molto inferiore allo zero potrebbero rompersi in modo anomalo, smettendo di essere isolanti contro gli agenti esterni	Possibile malfunzionamento della telecamera a temperature estreme, a maggior radice se la temperatura fa sì che si rompano componenti isolanti	La telecamera potrebbe non acquisire immagini esatte in quanto la sabbia, specialmente nell'obiettivo, causerebbe, ad esempio, il blocco dell'otturatore e/o altri componenti che hanno lo scopo di rendere l'immagine il più chiara possibile (strumenti per la messa a fuoco, zoom ecc.)	Le fasi successive all'acquisizione potrebbero dover processare immagini non chiare, con la possibilità di interpretarle erroneamente	Per quanto riguarda il problema della sabbia, non c'è alcuna cosa che, una volta danneggiata, una giunzione o un componente isolante, permetta di risolvere la situazione aspramente, sciogliendola o qual'altro. L'unico modo è progettare l'innovazione della telecamera in modo tale che la sabbia, anche se depositata su questo obiettivo, non ne danneggi la robustezza e non entri. La mitigazione deve essere considerata per tutto l'apparecchio, quindi a livello di sistema, dato che gli stessi problemi che si riscontrano per il corpo macchina (senza obiettivo) si ritrovano anche per l'obiettivo.	M - H	I - II
13	Obiettivo	ICE	Possibile deformazione dei componenti esterni dell'obiettivo	Nel caso in cui il ghiaccio strato di ghiaccio formi una coltre più o meno spessa sulla lente esterna dell'obiettivo le immagini, che fa sì che l'immagine sia completamente inutilizzabile	Se l'acqua penetra all'interno dell'obiettivo può bloccare molte componenti: azionate probabilmente non acquisite più immagini o acquisite completamente senza significato (contenuto)	Le fasi successive all'acquisizione saranno influenzate da tale coltre (immagini senza significato per il computer centrale)	Per quanto riguarda la lente esterna ricoperta da una coltre di ghiaccio, una soluzione utilizzabile potrebbe essere quella che è presente in ogni macchina: il lunotto posteriore termico. Si tratta di un semplice finestino dotato di resistenze elettriche, inserite o applicate, con la funzione di evitare fenomeni di condensa interna. Questo potrebbe essere preso in considerazione come punto di partenza per lo sviluppo di materiali riciclabili alla stessa maniera, ma, possibilmente, ionizzati, attraversati dalla resistenza. Questo soluzioni un po' futuristiche è necessaria in quanto la telecamera non si può permettere di processare immagini con linee orizzontali fisse, in quanto potrebbero provocare un comportamento indesiderato del veicolo in fase di decisione di manovra.	M - H	I - II	
14	Obiettivo	ICE	Possibile danneggiamento dei circuiti e componenti meccaniche interne	Se il salmastro penetra all'interno dell'obiettivo può corraderne materiali vari e circuiti interni, quindi influenzare l'acquisizione o a non permettere affatto di riceverne	Se l'obiettivo desse la possibilità di acquisire le immagini, queste, come detto, non controrrebbero le informazioni necessarie per far controllare l'auto dal computer centrale	Oggi sul mercato esistono telecamere di ogni tipo che permettono di andare a diversi metri sott'acqua, senza che succeda alcunché all'obiettivo ed al corpo macchina. La tecnologia per ovviare a questo fallimento è esistente e possibile da utilizzare.	L	IV		
15	Obiettivo	WATER	Possibile danneggiamento delle componenti esterne dell'obiettivo (conseguentemente anche quelli interni)	Se il salmastro penetra all'interno dell'obiettivo può corraderne materiali vari e circuiti interni, quindi influenzare l'acquisizione o a non permettere affatto di riceverne	La foto che viene acquisita potrebbe presentare macchie di vario genere, data l'infiltrazione di altri agenti all'interno dell'obiettivo. Le fasi successive dovranno processare un'immagine diversa dalla realtà	Per questo fallimento, l'unico modo di prevenirlo, è quello di costruire il corpo esterno ed anche l'obiettivo (componentistica esterna in generale) con materiali e tecnologie che non favoriscono la corrosione. A questo proposito, il paper "Method for the prevention of fouling and/or corrosion of structures in seawater, brackish water and/or fresh water", propone un dispositivo e un metodo per prevenire incrostazioni e corrosione delle superfici esposte di una struttura a contatto con acqua marina. Questo sembra un punto di partenza da prendere in considerazione. Anche per questo fallimento, andranno a riguardare tutto l'apparecchio, non un componente in particolare (mitigazione a livello di sistema).	H	I - II		
16	Obiettivo	BRACKISH/SALT WATER	Possibile danneggiamento dei parti che presentano intercapedini	Possibile rottura di tali pezzi con conseguente infiltrazione di agenti vari (acquisizione immagini "sporcate", sporco, lenti; immagini traslate, tagliate, ecc.)	Le fasi successive all'acquisizione di tali immagini sporcate saranno influenzate da questa condizione	Una soluzione comune, proposta inizialmente per telecamere di sorveglianza, potrebbe essere quella illustrata in "Condensation Prevention Camera Device", in questo documento si propone una soluzione al formarsi della condensa. La soluzione/mitigazione è comune a tutto il dispositivo (telecamera (corpo e obiettivo). Questa prevede l'utilizzo di un contenitore secondario per la telecamera con quale si favorisce lo scorrere dell'aria attraverso l'inenefra che viene a formarsi fra i due involucri. Quando la temperatura dell'aria esterna si riduce bruscamente, la condensa può apparire sulle pareti dell'obiettivo di una telecamera. La condensa, o umidità, provoca il degrado delle immagini e, se penetra all'interno del dispositivo, anche il degrado delle parti elettroniche. Inoltre, in questo progetto si propone anche una componente riscaldante per la telecamera, in modo che la temperatura si mantenga sempre la stessa e non si abbia lo sbalzo termico che favorisce il formarsi della condensa. Sulla stessa principio si basa un'altra soluzione a doppio involucro che crea uno spazio d'aria vuoto fra il contenitore primario e quello secondario ("Noncondensing security camera housing window assembly").	L	III		
17	Obiettivo	WIND	Possibile danneggiamento dei parti che presentano intercapedini	Le fasi successive all'acquisizione non disporranno di tutti i dati attesi e potrebbero non avvenire; se si hanno più videocamere sul veicolo, la parte di ambiente esterno sovvenuta dalla telecamera/affetta/da questo fallimento non verrà elaborata	Una soluzione comune, proposta inizialmente per telecamere di sorveglianza, potrebbe essere quella illustrata in "Condensation Prevention Camera Device", in questo documento si propone una soluzione al formarsi della condensa. La soluzione/mitigazione è comune a tutto il dispositivo (telecamera (corpo e obiettivo). Questa prevede l'utilizzo di un contenitore secondario per la telecamera con quale si favorisce lo scorrere dell'aria attraverso l'inenefra che viene a formarsi fra i due involucri. Quando la temperatura dell'aria esterna si riduce bruscamente, la condensa può apparire sulle pareti dell'obiettivo di una telecamera. La condensa, o umidità, provoca il degrado delle immagini e, se penetra all'interno del dispositivo, anche il degrado delle parti elettroniche. Inoltre, in questo progetto si propone anche una componente riscaldante per la telecamera, in modo che la temperatura si mantenga sempre la stessa e non si abbia lo sbalzo termico che favorisce il formarsi della condensa. Sulla stessa principio si basa un'altra soluzione a doppio involucro che crea uno spazio d'aria vuoto fra il contenitore primario e quello secondario ("Noncondensing security camera housing window assembly").	L	III - IV			
18	Corpo macchina	CONDENSATION	Degrado elettronica interna	Telecamera potrebbe non svolgere il suo lavoro nel modo corretto, nei casi pegiori si ha un blocco dell'apparecchio	Le fasi successive all'acquisizione non disporranno di tutti i dati attesi e potrebbero non avvenire; se si hanno più videocamere sul veicolo, la parte di ambiente esterno sovvenuta dalla telecamera/affetta/da questo fallimento non verrà elaborata	Una soluzione comune, proposta inizialmente per telecamere di sorveglianza, potrebbe essere quella illustrata in "Condensation Prevention Camera Device", in questo documento si propone una soluzione al formarsi della condensa. La soluzione/mitigazione è comune a tutto il dispositivo (telecamera (corpo e obiettivo). Questa prevede l'utilizzo di un contenitore secondario per la telecamera con quale si favorisce lo scorrere dell'aria attraverso l'inenefra che viene a formarsi fra i due involucri. Quando la temperatura dell'aria esterna si riduce bruscamente, la condensa può apparire sulle pareti dell'obiettivo di una telecamera. La condensa, o umidità, provoca il degrado delle immagini e, se penetra all'interno del dispositivo, anche il degrado delle parti elettroniche. Inoltre, in questo progetto si propone anche una componente riscaldante per la telecamera, in modo che la temperatura si mantenga sempre la stessa e non si abbia lo sbalzo termico che favorisce il formarsi della condensa. Sulla stessa principio si basa un'altra soluzione a doppio involucro che crea uno spazio d'aria vuoto fra il contenitore primario e quello secondario ("Noncondensing security camera housing window assembly").	M - H	I - II		

TABELLA FMECA VIDEOCAMERA

19	Corpo macchina	HEAT	Degrado parti non resistenti ad alte temperature (guarnizioni)	Telecamera funzionante, ma se le guarnizioni si rompono, il dispositivo smetterebbe di funzionare, lasciando la strada libera a umidità e polvere, fino al fallimento della componentistica interna.	In questo caso, una possibile mitigazione è quella di utilizzare guarnizioni e altri componenti del corpo macchina soggetti a deformazione di vario genere, a prova di calore (temperatura estrema); se non sono utilizzati componenti di questo genere si può incorrere in una rottura non sempre riparabile. Una soluzione si ritrova nel campo della videosorveglianza, in cui la azienda Axis Communications, leader nel settore, fabbrica telecamere a prova di temperatura desertiche come la AXN-Q60-C PTZ, dotata di raffreddamento attivo integrato, che permette una maggiore affidabilità in ambienti ad alta temperatura ([+20 C°/+75 C°]; soddisfano lo standard militare MIL-STD-810G).	L	III - IV	
20	Corpo macchina	SAND	Possibile corrosione di guarnizioni (percentuale di sale nella sabbia), con conseguente entrata di agenti esterni	Telecamera funzionante, ma se le guarnizioni cedessero, lascerebbero strada libera a umidità e aria, libera a umidità e polvere, fino al fallimento della componentistica interna.	Al cedimento di pezzi fondamentali per l'isolamento, all'interno del dispositivo potrebbero comparire deilettori per la componentistica interna, con conseguente elaborazione di alcuna immagine	M - H	I - II	
21	Corpo macchina	ICE	Possibile deformazione dei sottocomponenti esterni del corpo macchina	Determinati materiali ad una temperatura di molto inferiore allo zero potrebbero rompersi in modo anomalo, smettendo di essere isolanti contro gli agenti esterni	Possibile malfunzionamento della telecamera a temperature estreme, a maggior ragione se la temperatura fa sì che si rompano materiali isolanti	H	I	
22	Corpo macchina	WATER	Possibile danneggiamento dei circuiti interni	Se l'acqua arriva all'interno del corpo macchina, la telecamera smette di funzionare e non acquisisce più immagini	Blocco totale della telecamera	L	IV	
23	Corpo macchina	BRACKISH/SALT WATER	Possibile danneggiamento di guarnizioni e altri componenti esterni (conseguentemente anche quelli interni)	Possibile danneggiamento delle guarnizioni e dei materiali isolanti che potrebbe causare l'infiltrazione di piccoli acqua. Se il salinastro arriva alle componenti meccaniche e/o ai circuiti interni, la corrosione può bloccare le funzioni di acquisizione (e non solo: blocco totale)	Per questo fallimento, l'unico modo di prevenirlo, è quello di costruire il corpo esterno ed anche l'obiettivo (componentistica esterna in genere) con materiali e tecnologie che non favoriscono la corrosione. A questo proposito, il paper "Method for the prevention of fouling and/or corrosion of structures in seawater, brackish water and/or fresh water", propone un dispositivo e un metodo per prevenire incrostazioni e corrosione delle superfici esposte di una struttura a contatto con acqua marina. Questo sembra un punto di partenza da prendere in considerazione. Anche per questo fallimento, le mitigazioni andranno a riguardare tutto l'apparecchio, non un componente in particolare (mitigazione a livello di sistema).	H	I	
24	Corpo macchina	ELECTRICAL OVERLOAD	Sovraccarico di energia elettrica nei circuiti interni	Il dispositivo non acquisisce più le immagini come dovrebbe o non le acquisisce affatto (l'aumento eccessivo e temperatura nei conduttori potrebbe determinare rottura ed altre conseguenze nel corpo macchina)	Il dispositivo smette di funzionare o ritrova nelle fasi successive all'acquisizione (spagliata), in uno stato in cui le immagini non sono processabili	Esistono due tipi di protezione per le unità elettriche che devono essere considerate: il primo riguarda la protezione dei cavi elettrici che alimentano i circuiti, da un sovraccarico superiore alla loro capacità di carico; l'altro tipo riguarda la protezione da sovraccarico dei singoli apparecchi e apparecchiature elettriche collegate a un circuito di alimentazione. Nel nostro caso, la prima soluzione è da tenere presente, con l'isolamento dei vari cavi e circuiti (ed i controllo del loro amperaggio). Qua la mitigazione deve essere considerata a livello di sistema, in quanto l'energia elettrica è fondamentale per il funzionamento di ogni componente, e dunque in ognuno si ritrovano circuiti soggetti a questo fallimento.	L - M	I - II
25	Filtro di Bayer	NO BAYER FILTER	Immagine che presenta colori non corretti	Il dispositivo acquisisce un'immagine con colori non corretti (scalati a grigi)	Le fasi successive dovranno processare un'immagine con colori errati	In questo caso ci si concentra sulla affidabilità del componente, più questa è alta e più il pezzo non sarà incline a fallire. Per sistemi all'avanguardia, dotati di una grande potenza e velocità di ricalco (come ci si aspetta di trovare sulle auto a guida autonoma), potrebbe non essere un problema valutare più output, controllando se il risultato ottenuto sia il medesimo.	M - H	I - II

26	Sensore	SPOTS	Immagini con piccole o medie ombreggiature di forma per lo più circolare	Telecamera processa immagini che presentano piccole o medie ombreggiature	Le fasi successive all'acquisizione saranno influenzate dalla presenza tali ombreggiature	La soluzione a questo problema deriva dal mondo della fotografia digitale. Infatti, su molte macchine produttrici di DSR o Mirrorless è installato un sistema chiamato "Dust reduction system". Questo serve per rimuovere la polvere dal sensore di immagine. Ogni volta che si sostituiscono gli obiettivi, la polvere potrebbe penetrare nel corpo macchina e depositarsi sul sensore di immagine. La polvere può essere generata da molti mobili intorno o può essere mossa da correnti d'aria all'interno della telecamera. Alcuni sistemi puliscono il sensore vibrando a una frequenza molto elevata, tra 100 hertz e 50 kilohertz. Se questo non avesse gli effetti aspettati, allora è necessario procedere manualmente alla pulitura del sensore. Inoltre, esistono molti programmi di fotografia come Adobe Lightroom e Adobe Photoshop che hanno la funzionalità di "rimozione macchie", che fanno di controllare una macchia e sostituirla con una texture più simile possibile selezionata all'interno della stessa foto. Per una soluzione automatizzata via software sono stati trovati alcuni paper o brevetti che vogliono risolvere tale problema: "An optical model of the appearance of blemishes in digital photographs" (algoritmo di processamento delle immagini) "Detection and removal of blemishes in digital images utilizing original images or defocused scenes", "Infrared camera sensor dust reduction system" (dove il sensore di immagine è protetto da uno schermo di vetro che filtra i raggi infrarossi). In questo caso, se si utilizza un approccio o quando possibile è manuale, per la pulizia superficiale del sensore, si può pulire con un panno nero, correggerlo andando a mitigare il livello di componente, mentre quando ci viene ad utilizzare strumenti software per rimuovere le imperfezioni dell'immagine in questo caso ombreggiature per lo più circolari) si parla di mitigazione a livello di sistema, in quanto ci si prevede una elaborazione interna post-acquisizione.	L - M	II - III
27	Sensore	DEAD PIXEL	Immagini che presentano uno o più difetti di colore di dimensione di un pixel	Telecamera processa immagini contenenti uno o più pixel neri (raramente di altro colore)	Le fasi successive all'acquisizione saranno influenzate dalla presenza di questi pixel difetti	Esistono vari modi per verificare che uno o più pixel siano danneggiati. Uno di questi, nel campo della fotografia digitale è accendere lo schermo LCD per il live view e controllare se si vede il difetto anche dopo l'acquisizione. In fase di presentazione. Altri modi, che fanno riferimento al campo Automotive, prevedono un rilevamento automatico di tali malfunzionamenti e provano a correggere l'errore sul fotogramma acquisito via software. Un paper che si propone di fare questo è "Real-Time Photo Sensor Dead Pixel Detection for Embedded Devices". In questo ci si occupa della rilevazione del problema. Un'altra possibile soluzione è data da "On-chip dead pixel correction in a CMOS imaging sensor", dove si utilizza un approccio o quando possibile è manuale, per la pulizia superficiale del sensore, si può utilizzare un software per rimuovere il pixel nero, correggerlo andando a sostituire il suo segnale con il segnale del pixel immediatamente precedente nell'array.	L	II - III
28	Sensore	BRIGHT LINE	Immagini che presentano linee (luminose e meno intense) e/o verticali e/o orizzontali	Telecamera processa immagini/fotogrammi che presentano linee verticali e/o orizzontali	Le fasi successive all'acquisizione saranno influenzate dalla presenza di tali linee	Non esistono mitigazioni per questo genere di fallimenti. Infatti, in tale caso il danno può essere causato dai cosiddetti lidar o altre ipologie di laser (non visibili dall'occhio umano). Si trova la dimostrazione di questo danneggiamento in vari articoli online. Fra questi si trova quello di un fotografo che ha provato il "Real-Time Photo Sensor Dead Pixel Detection for Embedded Devices". In questo ci si occupa della rilevazione del problema. Un'altra possibile soluzione è data da "On-chip dead pixel correction in a CMOS imaging sensor", dove si utilizza un approccio o quando possibile è manuale, per la pulizia superficiale del sensore, si può utilizzare un software per rimuovere il pixel nero, correggerlo andando a sostituire il suo segnale con il segnale del pixel immediatamente precedente nell'array.	L	III - IV
29	Sensore	BANDING	Immagini che presentano un insieme di righe parallele in secondo piano	Telecamera processa immagini che presentano un insieme di linee parallele visibili maggiormente sui colori più chiari	Le fasi successive all'acquisizione saranno influenzate dalla presenza di tali effetti	Le soluzioni a questo problema si trovano soprattutto nel mondo della fotografia. Infatti, il banding è un fenomeno di creazione di strie all'interno dell'immagine che non sono viste di buon occhio dai fotografi professionisti (e non professionisti). Queste si creano, come detto, soprattutto in condizioni nelle quali si passa da un colore chiaro ad uno scuro. In questo scalare della tonalità si viene a formare il problema definito banding. Il problema viene a creare quando non sono disponibili toni sufficienti per ricevere una gradazione uniforme, motivo per cui è più comune nelle immagini a 8 bit e con immagini pesantemente compresse. Le soluzioni vengono presentate in " https://www.dpmag.com/how-to/tip-of-the-week/detecting-repairing-banding/ ", nel quale vengono dati diversi consigli: "rimostra fra tutti è quello di convertire l'immagine in una a 16 bit; il secondo è quello di non comprimere troppo il file. Infine, nei casi più gravi si richiede un trattamento più attento, andando ad utilizzare un po' di rumore. Mitigazione poco probabile a livello di componente, dato che il fotogramma necessiterà di varie elaborazioni, che in alcuni casi non bastano a sanitizzare l'immagine.	L	II - III
30	ISP	NO DEMOSAICING	Immagine in formato grigio [RAW]	Il dispositivo acquisisce un'immagine in formato RAW (non ancora sottoposta a demosaicing), dunque contiene solo un valore rosso, verde o blu in ogni pixel	L'immagine arriva in una forma non corretta alle fasi successive, se l'algoritmo di demosaicing non ha dei problemi durante la sua esecuzione il dispositivo si può bloccare in uno stato inconsistente	Questa è la prima fase di elaborazione e la più importante. L'algoritmo di Demosaicing è l'architettura utilizzata per farlo operare devono avere un'affidabilità massima pena la mancanza corretta l'interpretazione della scena. L'algoritmo descritto nel paper "An Efficient Method for Demosaicing", si propone come una valida alternativa a molti altri, svolgendo un buon lavoro anche con immagini problematiche dal punto di vista della ricostruzione dei colori. Interessante anche il brevetto "Demosaicing RGB sensor", nel quale si fa riferimento ad un motore di Demosaicing. Questo può essere configurato per generare, mediante una tecnica addittiva selettiva per colore, un'immagine policromatica basata sui valori di pixel monocromatici catturati e sui valori di pixel monocromatici stimati.	M - H	II - III
31	ISP	INCOMPLETE DEMOSAICING	Immagine parzialmente sottoposta a Demosaicizzazione	Immagine (nella sua rappresentazione a colori), presenta ancora parti di immagine con pixel o blu o verde o rossi (RAW parziale)	Questa è la prima fase di elaborazione e la più importante. L'algoritmo di Demosaicing e l'architettura utilizzata per farlo operare devono avere un'affidabilità massima pena la mancanza corretta l'interpretazione della scena. L'algoritmo descritto nel paper "An Efficient Method for Demosaicing", si propone come una valida alternativa a molti altri, svolgendo un buon lavoro anche con immagini problematiche dal punto di vista della ricostruzione dei colori. Interessante anche il brevetto "Demosaicing RGB sensor", nel quale si fa riferimento ad un motore di Demosaicing. Questo può essere configurato per generare, mediante una tecnica addittiva selettiva per colore, un'immagine policromatica basata sui valori di pixel monocromatici catturati e sui valori di pixel monocromatici stimati.	M - H	II - III	
32	ISP	NO NOISE REDUCTION	Immagine che presenta rumore	L'immagine acquisita non viene ripulita dal rumore completamente	Con entrambi i fallimenti il risultato è un'immagine non ben definita che presenta del rumore. Questo può essere mal interpretato dall'elaboratore centrale, il quale potrebbe percepire oggetti in situazioni non vere. Di seguito in che effetti questo processo di sanificazione dell'immagine ne esiste una grande varietà, con diverse qualità. La soluzione principale per risolvere questo problema è sempre l'affidabilità del processore, il quale non deve saltare questa fase, né deve effettuarla parzialmente. Per quanto riguarda la riduzione del rumore, il paper "Noise Reduction for Image Signal Processor in Digital Camera" sembra essere perfetto per il nostro caso: infatti, si tratta di un algoritmo adatto per l'ISP nel quale non deve essere ricreato il modello di rumore ad ogni immagine, ma una sola volta. Pensando a quante operazioni deve fare il processore, è raccomandabile risparmiare quanto più tempo computazionale possibile se ce n'è la possibilità.	M - H	II - III	

TABELLA FMECA VIDEOCAMERA

175

33	ISP	INCOMPLETE NOISE REDUCTION	Immagini che non viene completamente ripulita dal rumore	L'immagine acquisita presenta dei bordi non ben definiti e contorni imprecisi e poco chiarezza (non nitida)	Le fasi di elaborazione successive processeranno un'immagine nella quale non è stata effettuata completamente la rimozione del rumore	M	II - III
34	ISP	NO SHARPNESS	Immagini che presentano dettagli imprecisi e poca chiarezza (non nitida)	L'immagine acquisita presenta bordi non ben definiti e contorni imprecisi e poco chiarezza (non nitida)	L'immagine non arriva alle fasi successive; il sistema si può bloccare completamente	M - H	II - III
35	ISP	INCOMPLETE SHARPNESS	Immagini che presentano dettagli imprecisi e poca chiarezza (non nitida)	L'immagine è stata solo elaborata parzialmente da questa fase, dunque alcune porzioni di fotogramma non sono state trattate per aggiustare la nitidezza dei dettagli (bordi, ecc.)	Le fasi di elaborazione successive dovranno elaborare un'immagine che non è stata rifinita da questo processo	M	II - III
36	ISP	NO LENS DISTORTION CORRECTION	Immagini che sembra mappata attorno ad una sfera (o barrel distortion)	L'immagine acquisita è deformata, presenta forme tondeggianti assimmetriche innaturali	Le fasi successive non saranno ragguaglia in quanto l'immagine si blocca in questa fase di elaborazione; il sistema può bloccarsi in questa fase non poter elaborare le immagini passanti per tale procedura	M	II - III
37	ISP	INCOMPLETE LENS DISTORTION CORRECTION	Immagini che sembra parzialmente mappata attorno ad una sfera (o barrel distortion)	L'immagine acquisita è deformata, presenta forme tondeggianti assimmetriche innaturali	Le fasi successive, se l'immagine non viene corretta, la processoranno così come è stata ottenuta, elaboratore centrale, alla fine, dovrà interpretare un'immagine che presenta errori radiali asimmetriche soggette a errori di interpretazione	M	III.
38	ISP	NO CHROMATIC ABBERRATION CORRECTION	Immagini che presentano come "strange" contorni di vario colore (viola per lo più) e una sorta di stocatura	Le fasi successive a questa procedura non potranno elaborare il fotogramma corrente perché bloccato in tali fasce non rispondente al sistema può andare in blocco (il elaboratore centrale non elaborerà nessuna delle immagini passanti per tale fase)	Per la mitigazione di tale problema dovremo fare contare sull'affidabilità del processore che si utilizza. Infatti, di algoritmi e metodi per la riduzione della distorsione delle lenti o la loro correzione via software (dopo una misurazione dei loro coefficienti di distorsione); ce ne sono moltissimi come ad esempio: "lens distortion correction for digital image correlation by measuring rigid body displacement", "Line-Based Correction of Radial Lens Distortion" e anche "True multi-distortion correction for digital image correlation by measuring rigid body displacement".	L - M	III.
39	ISP	INCOMPLETE CHROMATIC ABBERRATION CORRECTION	Immagini che presentano come "strange" contorni di vario colore (viola per lo più) e una sorta di stocatura	Le fasi successive dovranno elaborare un'immagine con questo difetto, che si protrarrà fino all'elaboratore distorsione cromatica). Uno di questi è discusso nel paper "Automatic Removal of Chromatic Aberration from a Single Image", che mostra in che modo può essere eliminato questo disturbo dall'immagine processata. In questo si fa riferimento ad una singola immagine, altro metodo interessante è descritto nel paper "Removing chromatic aberration by digital image processing", nel quale, non solo si corregge l'aberrazione cromatica, ma si considera anche un detector che la rileva.	Questo è un fallimento che può portare dei cambiamenti di colore, fondamentali per quanto riguarda la segnaletica stradale. Dunque, sempre tenendo conto dell'affidabilità del processore, esistono molti metodi e algoritmi che, in modo automatico, risolvono il problema dell'aberrazione cromatica (chiamata anche distorsione cromatica). Uno di questi è discusso nel paper "Automatic Removal of Chromatic Aberration from a Single Image", che mostra in che modo può essere eliminato questo disturbo dall'immagine processata. In questo si fa riferimento ad una singola immagine, altro metodo interessante è descritto nel paper "Removing chromatic aberration by digital image processing", nel quale, non solo si corregge l'aberrazione cromatica, ma si considera anche un detector che la rilevi.	L - M	III.

40	ISP	NO ACTION	L'ISP non risponde L'immagine acquisita non viene elaborata (rimane in formato (RAW)) L'immagine non arriva alle fasi successive del processo di elaborazione (il computer centrale non disporrà di dati provenienti dalla telecamera); il sistema si può bloccare completamente	<p>Il fallimento peggiore che può accadere, dove il processore non risponde e l'immagine acquisita resta "grezza" e non viene inviata al computer centrale (non si passa dalle fasi intermedie). Anche qui, l'affidabilità del processore è il primo aspetto da tenere in considerazione e forse anche l'unico, in quanto, come detto, se il processore non è affidabile e tende a sbagliare o a non rispondere, il sistema Auto non potrà cominciare. Le mitigazioni per questo non possono essere elencate, in quanto l'affidabilità di un componente così importante si basa su molti altri rispetti. La cosa più importante da fare è tenere presente che, eseguendo l'impegno su un veicolo a guida autonoma per l'elaborazione di immagini che servono per la visione artificiale, la sua progettazione deve essere eseguita facendo riferimento ai più alti standard di sicurezza e affidabilità. Una possibile mitigazione, che riguarda il sistema nella sua interezza, può essere quella di sottoporre il processore a dei test periodici (periodi brevi) per valutare il suo corretto funzionamento. Oltre a questo potrebbe essere necessario aumentare l'affidabilità di un componente così fondamentale e considerando l'utilizzo di più processori; se non vengono rilevati problemi possono anche funzionare tutti con obiettivo della divisione del carico di lavoro; mentre nel momento in cui venga rilevato un qualsiasi errore in uno o più di questi, i rimanenti potrebbero farsi carico del lavoro che non può essere svolto da quelli falliti. In questo caso, anche se rimanesse uno, ma questo fosse in grado di elaborare tutta la mole di dati che deriva dall'acquisizione dei fotogrammi, il sistema Auto potrebbe continuare la sua marcia, anziché doversi fermare per un errore che non permette più al computer centrale di prendere decisioni.</p>	H	I

BIBLIOGRAFIA

- [1] Il Sole 24 Ore, 2018. <https://www.ilsole24ore.com/art/guida-autonoma-rivoluzione-che-procede-piccoli-passi-AEAorNNG>. (Cited on pages 3 and 25.)
- [2] Goodyear. <https://images.app.goo.gl/ScboEGhGDmZgxrLu5>. (Cited on pages 3 and 28.)
- [3] Cadence Community, 2018. https://community.cadence.com/cadence_blogs_8/b/breakfast-bytes/posts/autos182. (Cited on pages 3 and 29.)
- [4] SZ Interactive, 2019. <https://images.app.goo.gl/zWUwbhAuoXixeFUe7>. (Cited on pages 3 and 43.)
- [5] Medium, 2019. <https://images.app.goo.gl/ACF4ejmFFmgfKktb9>. (Cited on pages 3 and 44.)
- [6] Jit Ray Chowdhury, 2019. <https://images.app.goo.gl/wwYNLZqDuCpu8xP56>. (Cited on pages 3 and 67.)
- [7] Wikipedia, “Colour banding,” 2020. https://en.wikipedia.org/wiki/Colour_banding. (Cited on pages 3 and 70.)
- [8] Marco Togni, “Iso – guida agli iso per foto perfette,” <https://www.marcotogni.it/iso/>. (Cited on pages 3 and 72.)
- [9] LightroomCafé - il blog italiano di Lightroom, “Aberrazione cromatica e come correggerla in lightroom 4.1,” 2012. www.lightroomcafe.it. (Cited on pages 3 and 75.)
- [10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” in *Conference on Robot Learning (CoRL)*, 2017. (Cited on pages 4, 13, 79, 81, 82, 83, and 142.)
- [11] M. Bijelic, T. Gruber, and W. Ritter, “Benchmarking image sensors under adverse weather conditions for autonomous driving,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1773–1779, 2018. (Cited on pages 5 and 136.)
- [12] S.-C. Lin, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, “The architectural implications of autonomous driving: Constraints and acceleration,” in *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018. (Cited on page 11.)

- [13] A. A. C. Premebida, G. Melotti, "Rgb-d object classification for autonomous driving perception," 2019. (Cited on page 11.)
- [14] J. W. Z. Zheng, X. He, "Approaching camera-based real-world navigation using object recognition," 2015. (Cited on page 11.)
- [15] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammler, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 163–168, 2011. (Cited on page 11.)
- [16] K. Pei, Y. Cao, J. Yang, and S. Jana, "Towards practical verification of machine learning: The case of computer vision systems," *arXiv preprint arXiv:1712.01785*, 2017. (Cited on page 12.)
- [17] W. Wu, H. Xu, S. Zhong, M. R. Lyu, and I. King, "Deep validation: Toward detecting real-world corner cases for deep neural networks," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 125–137, 2019. (Cited on page 12.)
- [18] M.-I. Nicolae, M. Sinn, M. N. Tran, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. M. Molloy, *et al.*, "Adversarial robustness toolbox vo. 4.0," *arXiv preprint arXiv:1807.01069*, 2018. (Cited on page 12.)
- [19] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin, "The security of autonomous driving: Threats, defenses, and future directions," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 357–372, 2020. (Cited on pages 12 and 137.)
- [20] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, p. 2015, 2015. (Cited on pages 12 and 137.)
- [21] Wikipedia, "FMEA," 2016. <https://it.wikipedia.org/wiki/FMEA>. (Cited on pages 12, 35, and 142.)
- [22] F. Secci, "Python Image Failures," 2020. https://github.com/francescosecci/Python_Image_Failures. (Cited on pages 13, 87, 106, 142, and 147.)

- [23] D. Chen, B. Zhou, V. Koltun, and P. Krahenbuhl, "Learning by cheating," *In Conference on Robot learning (CoRL)*, 2019. (Cited on pages 13, 84, 85, 86, 87, and 142.)
- [24] Wikipedia, "Sistema critico," Jan 2019. https://it.wikipedia.org/wiki/Sistema_critico. (Cited on page 15.)
- [25] A. Bondavalli, *L'Analisi Quantitativa dei Sistemi Critici*. Esculapio, 2011. (Cited on pages 15, 16, 19, 20, and 21.)
- [26] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 11–33, Jan 2004. (Cited on pages 17, 18, 19, 21, and 22.)
- [27] Wikipedia, "Dependability," Jan 2019. <https://en.wikipedia.org/wiki/Dependability>. (Cited on pages 19, 20, and 21.)
- [28] E. Awad, S. Dsouza, R. Kim, J. Schulz, J. Henrich, A. Shariff, J.-F. Bonnefon, and I. Rahwan, "The moral machine experiment," *Nature*, vol. 563, pp. 59–64, 2018. (Cited on page 23.)
- [29] Wikipedia, "Problema del carrello ferroviario," 2020. https://it.wikipedia.org/wiki/Problema_del_carrello_ferroviario. (Cited on page 23.)
- [30] Wikipedia, "Tesla Autopilot," 2020. https://en.wikipedia.org/wiki/Tesla_Autopilot. (Cited on page 26.)
- [31] P. Das, *Risk analysis of autonomous vehicle and its safety impact on mixed traffic stream*. Theses and Dissertations, 2018. (Cited on pages 29, 30, 31, 32, and 34.)
- [32] Wikipedia, "Visione artificiale," 2020. https://it.wikipedia.org/wiki/Visione_artificiale. (Cited on page 29.)
- [33] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 1802–1811, PMLR, 2019. (Cited on pages 32 and 33.)
- [34] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks

- on deep learning visual classification," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634, 2018. (Cited on page 34.)
- [35] R. M. A. Caminiati, P. Di Denia, "Risk management," 2007. (Cited on page 35.)
- [36] J. Huang, J.-X. You, H.-C. Liu, and M.-S. Song, "Failure mode and effect analysis improvement: A systematic literature review and future research agenda," *Reliability Engineering and System Safety*, vol. 199, 2020. (Cited on pages 36 and 37.)
- [37] A. Bouti and D. A. Kadi, "A State-of-the-Art Review of FMEA/FMECA," *International Journal of Reliability, Quality and Safety Engineering*, vol. 1, no. 4, pp. 515–543, 1994. (Cited on page 36.)
- [38] Wikipedia, "Obiettivo fotografico," 2020. https://it.wikipedia.org/wiki/Obiettivo_fotografico. (Cited on page 40.)
- [39] Wikipedia, "Schema bayer," Feb 2017. https://it.wikipedia.org/wiki/Schema_Bayer. (Cited on pages 41 and 95.)
- [40] Redazione di Elettronica Open Source, "I sensori di immagine," 2018. (Cited on page 45.)
- [41] Wikipedia, "Image processor," 2020. https://en.wikipedia.org/wiki/Image_processor. (Cited on page 45.)
- [42] F. Secci, "FMECA Videocamera," 2020. https://github.com/francescosecci/Python_Image_Failures/blob/master/Documenti/FMECA.xlsx. (Cited on pages 47 and 169.)
- [43] J. Cai, H. Ji, C. Liu, and Z. Shen, "Blind motion deblurring from a single image using sparse approximation," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 104–111, June 2009. (Cited on pages 48 and 50.)
- [44] L. Fang, H. Liu, F. Wu, X. Sun, and H. Li, "Separable kernel for image deblurring," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2885–2892, June 2014. (Cited on pages 48 and 50.)
- [45] J. Gu, R. Ramamoorthi, P. Belhumeur, and S. Nayar, "Removing image artifacts due to dirty camera lenses and thin occluders," *ACM Transactions on Graphics*, Dec. (Cited on page 51.)

- [46] D. Eigen, D. Krishnan, and R. Fergus, "Restoring an image taken through a window covered with dirt or rain," in *2013 IEEE International Conference on Computer Vision*, pp. 633–640, Dec 2013. (Cited on pages 51 and 52.)
- [47] Ficosa Corporation, "Sensor and camera cleaning." <https://www.ficosa.com/products/underhood/sensor-and-camera-cleaning/>. (Cited on page 51.)
- [48] F. Chabert, "Automated lens flare removal," *Technical report, Stanford University, Department of Electrical Engineering*, 2015. (Cited on page 52.)
- [49] Clarifii, "Clarifii: Water repellent and anti fog." <https://www.clarifiiclean.com/products/clarifii>. (Cited on page 53.)
- [50] M. Kondou, "Condensation prevention camera device," 2016. (Cited on pages 54 and 59.)
- [51] D. F. Loiacono, "Noncondensing security camera housing window assembly," 2010. (Cited on pages 54 and 60.)
- [52] B. Englander, "Video camera unit, protective enclosure and power circuit for same, particularly for use in vehicles," 1995. (Cited on page 54.)
- [53] A. Talbert, M. B. Moore, and N. Roy, "Lens heater," 1948. (Cited on page 57.)
- [54] W. J. Riffe and J. D. Carter, "Method for the prevention of fouling and/or corrosion of structures in seawater, brackish water and/or fresh water," 1994. (Cited on pages 58 and 62.)
- [55] A. Zamfir, A. Drimbarean, M. Zamfir, V. Buzuloiu, E. Steinberg, and D. Ursu, "An optical model of the appearance of blemishes in digital photographs," in *Digital Photography III* (R. A. Martin, J. M. DiCarlo, and N. Sampat, eds.), vol. 6502, pp. 166 – 177, International Society for Optics and Photonics, SPIE, 2007. (Cited on page 65.)
- [56] E. Steinberg, P. Bigioi, and A. Zamfir, "Detection and removal of blemishes in digital images utilizing original images of defocused scenes," 2007. (Cited on page 65.)

- [57] C. Cho, T. Chen, W. Wang, and C. Liu, "Real-time photo sensor dead pixel detection for embedded devices," in *2011 International Conference on Digital Image Computing: Techniques and Applications*, pp. 164–169, Dec 2011. (Cited on page 66.)
- [58] K. Dong, "On-chip dead pixel correction in a cmos imaging sensor," 2009. (Cited on page 66.)
- [59] M. Zhang, "Man's \$1,998 camera fried by self-driving car laser," 2019. (Cited on page 67.)
- [60] Z. Kleinman, "Driverless car laser ruined camera," 2019. (Cited on page 67.)
- [61] Wikipedia, "Dithering," Aug 2019. <https://it.wikipedia.org/wiki/Dithering>. (Cited on page 70.)
- [62] W. Sawalich, "Identifying and repairing banding," 2016. <https://www.dpmag.com/how-to/tip-of-the-week/identifying-repairing-banding/>. (Cited on page 70.)
- [63] Cambridge in Colour, a learning community for photographers, "Raw file format," <https://www.cambridgeincolour.com/tutorials/raw-file-format.htm>. (Cited on page 71.)
- [64] B. Grossmann and Y. C. Eldar, "An efficient method for demosaicing," in *2004 23rd IEEE Convention of Electrical and Electronics Engineers in Israel*, pp. 436–439, Sep 2004. (Cited on page 71.)
- [65] L. Shi and I. Ovsiannikov, "Demosaicing rgbz sensor," 2015. (Cited on page 71.)
- [66] Y. Baek, D. Cho, J. Lee, and W. Kim, "Noise reduction for image signal processor in digital cameras," in *2008 International Conference on Convergence and Hybrid Information Technology*, pp. 474–481, Aug 2008. (Cited on page 73.)
- [67] G. Albani Lattanzi, "Guida alla nitidezza," <https://www.giovannilattanzi.it/guida-alla-nitidezza/>. (Cited on page 73.)
- [68] S. Yoneyama, H. Kikuta, A. Kitagawa, and K. Kitamura, "Lens distortion correction for digital image correlation by measuring rigid body displacement," *Optical Engineering*, vol. 45, no. 2, pp. 1 – 9, 2006. (Cited on page 74.)

- [69] B. Prescott and G. McLean, "Line-based correction of radial lens distortion," *Graphical Models and Image Processing*, vol. 59, no. 1, pp. 39 – 47, 1997. (Cited on page 74.)
- [70] H. S. Sawhney and R. Kumar, "True multi-image alignment and its application to mosaicing and lens distortion correction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 235–243, March 1999. (Cited on page 74.)
- [71] Wikipedia, "Aberrazione cromatica," Jun 2019. https://it.wikipedia.org/wiki/Aberrazione_cromatica. (Cited on page 75.)
- [72] S. B. Kang, "Automatic removal of chromatic aberration from a single image," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2007. (Cited on page 76.)
- [73] S.-W. Chung, B.-K. Kim, and W.-J. Song, "Removing chromatic aberration by digital image processing," *Optical Engineering*, vol. 49, no. 6, pp. 1 – 10, 2010. (Cited on page 76.)
- [74] D. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Neural Information Processing Systems (NIPS)*, 1988. (Cited on page 80.)
- [75] Epic Games, "Unreal engine 4." <https://www.unrealengine.com>. (Cited on page 81.)
- [76] CARLA Documentation. https://carla.readthedocs.io/en/stable/carla_settings/. (Cited on page 87.)
- [77] PIL 3.0. <https://pillow.readthedocs.io/en/3.0.x/>. (Cited on page 106.)
- [78] CV2 filtering. <https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html>. (Cited on page 106.)
- [79] NumPy. <https://numpy.org/>. (Cited on page 106.)
- [80] Y. Park, "Realistic lens blur/chromatic aberration filter." <https://github.com/yoonsikp/kromo>. (Cited on page 106.)
- [81] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. (Cited on page 135.)

- [82] V. Behzadan and A. Munir, "Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles," *IEEE Intelligent Transportation Systems Magazine*, pp. 1–1, 2019. (Cited on page 135.)
- [83] C. Yan, "Can you trust autonomous vehicles : Contactless attacks against sensors of self-driving vehicle," 2016. (Cited on page 137.)
- [84] T. Schops, J. L. Schonberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. (Cited on page 137.)
- [85] P. Y. Shinzato, T. C. dos Santos, L. A. Rosero, D. A. Ridel, C. M. Massera, F. Alencar, M. P. Batista, A. Y. Hata, F. S. Osório, and D. F. Wolf, "Carina dataset: An emerging-country urban scenario benchmark for road detection systems," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 41–46, 2016. (Cited on page 137.)
- [86] M. Realpe, B. Vintimilla, and L. Vlacic, "Multi-sensor fusion module in a fault tolerant perception system for autonomous vehicles," *Journal of Automation and Control Engineering*, vol. 4, pp. 430–436, 12 2016. (Cited on page 137.)
- [87] L. Guo, S. Mangani, Y. Liu, and Y. Jia, "Automatic sensor correction of autonomous vehicles by human-vehicle teaching-and-learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8085–8099, 2018. (Cited on page 138.)
- [88] M. S. Ramanagopal, C. Anderson, R. Vasudevan, and M. Johnson-Roberson, "Failing to learn: Autonomously identifying perception failures for self-driving cars," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3860–3867, 2018. (Cited on page 138.)
- [89] M. Toromanoff, E. Wirbel, F. Wilhelm, C. Vejarano, X. Perrotton, and F. Moutarde, "End to end vehicle lateral control using a single fisheye camera," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3613–3619, 2018. (Cited on page 139.)
- [90] S. Hossain, A. R. Fayjie, O. Doukhi, and D.-j. Lee, "Caias simulator: Self-driving vehicle simulator for ai research," in *Intelligent Computing & Optimization* (P. Vasant, I. Zelinka, and G.-W. Weber, eds.),

(Cham), pp. 187–195, Springer International Publishing, 2019. (Cited on page 139.)

- [91] Z. Zheng, X. He, and J. Weng, “Approaching camera-based real-world navigation using object recognition,” *Procedia Computer Science*, vol. 53, pp. 428 – 436, 2015. INNS Conference on Big Data 2015 Program San Francisco, CA, USA 8-10 August 2015. (Cited on page 139.)
- [92] Wikipedia, “Stereo camera,” 2019. https://it.qwe.wiki/wiki/Stereo_camera. (Cited on page 140.)
- [93] A. Teichman and S. Thrun, “Practical object recognition in autonomous driving and beyond,” in *Advanced Robotics and its Social Impacts*, pp. 35–38, 2011. (Cited on page 140.)
- [94] Aspen Institute Italia, “Auto a guida autonoma, una sfida per gli algoritmi,” 2019. (Cited on page 141.)