



Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Μονάδα Επεξεργασίας Σήματος
και Βιοϊατρικής Τεχνολογίας

Διπλωματική Εργασία

Υλοποίηση διεπαφής έξυπνου καθρέπτη με
δυνατότητες εκτίμησης πόζας για εφαρμογές
εξατομικευμένης άσκησης και ευεξίας

Ζαχαρόπουλος Φίλιππος 8559
filipposz@ece.auth.gr

Παπαγεωργίου Δημήτριος 8884
dim_papag@windowslive.com

Επιβλέποντες:

Χατζηλεοντιάδης Λεόντιος
Καθηγητής Α.Π.Θ.

Ντράχα Αναστασία
Υποψήφια Διδάκτορας

Χατζηδημητρίου Στέλιος
Μεταδιδακτορικός Ερευνητής

10 Μαρτίου 2022

ΕΥΧΑΡΙΣΤΙΕΣ

Πρώτο από όλους θα θέλαμε να ευχαριστήσουμε τον Στέλιο Χατζηδημητρίου και την Αναστασία Ντραχά, οι οποίοι μας βοήθησαν καθόλη τη διάρκεια της διπλωματικής, μας κατεύθυναν και ήταν συνεχώς διαθέσιμοι για να απαντήσουν στις ερωτήσεις μας. Επίσης, θα θέλαμε να ευχαριστήσουμε τον κ. Λεόντιο Χατζηλεοντιάδη που μας έδωσε τη δυνατότητα να αναλάβουμε αυτή τη διπλωματική αλλά και την Μονάδα Επεξεργασίας Σήματος και Βιοϊατρικής Τεχνολογίας για την στήριξη σε υλικό που μας παρείχαν.

Πέραν των επιστημονικών συνεργατών, θα θέλαμε να ευχαριστήσουμε τους φίλους και τις φίλες μας για τη συμπαράστασή τους και για όλες τις όμορφες στιγμές που ζήσαμε στην περίοδο των φοιτητικών μας χρόνων. Τέλος, θα θέλαμε να ευχαριστήσουμε τους γονείς μας και τα αδέρφια μας για την στήριξη και τα εφόδια που μας έδωσαν και που ήταν στο πλευρό μας όποτε τους χρειαστήκαμε.

Περίληψη

Η μείωση του κόστους ενσωματωμένων συσκευών αλλά και η πρόοδος σχετικών τεχνολογιών του Internet of Things έχουν προκαλέσει μια έκρηξη στην ανάπτυξη έξυπνων συσκευών. Οι συσκευές αυτές αποδεικνύονται χρήσιμες για πλήθος ανθρώπων διότι έρχονται να αυτοματοποιήσουν τις εργασίες τους και να βελτιώσουν την ποιότητα ζωής τους.

Ο έξυπνος καθρέφτης αποτελεί μια τέτοιου είδους συσκευή, η οποία έρχεται να επαυξήσει τις δυνατότητες του κλασσικού καθρέφτη. Με την προσθήκη οθόνης πίσω από μια ανακλαστική, εν μέρη διαφανή, επιφάνεια είναι εφικτή η προβολή πληροφοριών παράλληλα με το είδωλο του ανθρώπου, ενώ με χρήση μικροϋπολογιστή και αισθητήρων ο καθρέφτης μπορεί να αποκτήσει λογική για την επίτευξη χρήσιμων λειτουργιών για τον χρήστη.

Η ανάπτυξη, όμως, εφαρμογών που αφορούν καθημερινές δραστηριότητες, όπως ενημέρωση, λήψη υπενθυμίσεων και συγχρονισμός ηλεκτρονικού ταχυδρομείου, δεν ικανοποιούνται ιδιαίτερα εύκολα από μια έξυπνη συσκευή όπως είναι ο καθρέφτης. Η ικανοποίηση των παραπάνω εφαρμογών καλύπτεται αποδοτικότερα και ευκολότερα από άλλες συσκευές όπως το κινητό ή ο υπολογιστής. Για τον λόγο αυτό μέχρι σήμερα ο έξυπνος καθρέφτης δεν αποτελεί ένα διαδεδομένο εμπορικό προϊόν.

Ένας τομέας, ωστόσο, που ο έξυπνος καθρέφτης μπορεί να αξιοποιηθεί είναι αυτός της υγείας και της ευεξίας. Το παραπάνω στηρίζεται στο γεγονός ότι ο άνθρωπος χρησιμοποιεί τον απλό καθρέφτη για να λάβει ανατροφοδότηση σχετικά με τον εαυτό του και την κατάσταση στην οποία βρίσκεται η υγεία του, λόγου χάρη βλέπει πόσο χλωμός είναι ή πόσο ορθά εκτελεί μία άσκηση. Επομένως, ο έξυπνος καθρέφτης μπορεί να εγκατασταθεί σε χώρους γυμναστηρίων, κλινικών αλλά και να λειτουργήσει ως ένας προσωπικός βοηθός υγείας στο σπίτι.

Τα δύο βασικά προβλήματα που γεννά η χρήση του έξυπνου καθρέφτη αποτελούν η ύπαρξη ενός ευρέως διαδεδομένου λειτουργικού συστήματος και η εύκολη αλληλεπίδραση του ανθρώπου μαζί του. Από την μία πλευρά, η υλοποίηση ενός καθολικά αποδεκτού λειτουργικού θα βοηθήσει στην εύκολη ανάπτυξη και επέκταση εφαρμογών πάνω στον έξυπνο καθρέφτη, με τον ίδιο τρόπο που συνέβαλλε το android στην επέκταση των έξυπνων κινητών τηλεφώνων. Από την άλλη πλευρά, η εύκολη χρήση και επικοινωνία με τον έξυπνο καθρέφτη μπορούν να συμβάλλουν στην εκτεταμένη υιοθέτηση του.

Η παρούσα διπλωματική υλοποιεί ένα ελαφρύ αρθρωτό λειτουργικό σύστημα πάνω στο οποίο είναι εφικτό να αναπτύξει κανείς εφαρμογές για τον έξυπνο καθρέφτη. Επιπρόσθετα, η αλληλεπίδραση μεταξύ χρήστη και έξυπνου καθρέφτη γίνεται με χρήση φωνητικών εντολών, τις οποίες αναγνωρίζει το σύστημα και εκτελεί τις επιθυμητές ενέργειες. Πέρα από αυτό, υλοποιήθηκε και μια εφαρμογή για τον έλεγχο της ορθότητας μιας άσκησης η οποία βασίζεται σε τεχνολογία εκτίμησης πόζας. Τέλος, προτείνονται διάφορες επεκτάσεις και ιδέες για την περεταίρω ανάπτυξη και βελτίωση του έξυπνου καθρέφτη.

Title

Implementation of interface for smart mirror with capabilities of poze estimation for personalized exercise and well-being applications

Abstract

Zacharopoulos Filippos & Papageorgiou Dimitrios
Signal Processing and Biomedical Technology Unit
Electrical & Computer Engineering Department,
Aristotle University of Thessaloniki, Greece
February 2022

The cost reduction of embedded devices as well as the progress in technologies pertaining the Internet of Things (*IoT*) have caused a blast in smart devices development. These devices are proving to be useful for many people since they can automate their tasks and improve their quality of life.

The smart mirror constitutes such a smart device that comes to augment the capabilities of the classic mirror. By adding a monitor behind a reflective, partially transparent, surface it is feasible to show information in parallel with the human idol, while using a microprocessor and sensors the mirror can obtain logic for achieving useful operations for the user.

The development, however, of applications that concern daily activities such as news briefing, getting reminders and receiving emails are not very easily satisfied from a device such as the smart mirror. The gratification of the above tasks is covered more efficiently and easier by other devices like a smart phone or a computer. For this reason the smart mirror isn't commercially spread up to this date.

Nevertheless, the smart mirror can be utilized in the fields of health and well-being. The previous statement is backed by the fact that a human is using the classic mirror to receive feedback regarding himself and the status of their current health, for example they are watching how pale their skin is or how correct they execute on physical exercise. Therefore, the smart mirror can be installed in gym spaces, medical clinics and also operate like a personal health assistant at home.

The two major problems that arise from the use of the smart mirror are the lack of existence of a widely used operating system and the easy interaction between the human. On one hand, the implementation of a universally accepted operating system will help easy the development and extension of applications on top of the smart mirror, much like the same way that android has helped with the development of smart phones. On the other hand, easy usage and communication with the smart mirror can contribute in its extended adoption.

This diploma thesis implements a lightweight modular operating system on top of which one can develop applications for the smart mirror. Additionally, the interaction

between human and smart mirror is achieved using voice commands, which the system is capable of recognizing and executing the desired actions. Apart from that, an application for controlling the correctness of a physical exercise was developed, which relies on pose estimation technology. Lastly, we propose various extensions and ideas for further development and improvement of the smart mirror.

Περιεχόμενα

Ευχαριστίες	i
Περίληψη	iii
Abstract	v
1 Εισαγωγή	1
1.1 Περιγραφή του Προβλήματος	2
1.2 Σκοπός - Συνεισφορά της Διπλωματικής Εργασίας	3
1.3 Διάρθρωση της Αναφοράς	3
2 Θεωρητικό Υπόβαθρο	5
2.1 Διαδίκτυο των Πραγμάτων	5
2.1.1 Πώς δουλεύει το IoT	5
2.2 Λειτουργικά Συστήματα	8
2.2.1 Το ΛΣ ως Διεπαφή Χρήστη/Υπολογιστή	8
2.2.2 Το ΛΣ ως Διαχειριστής Πόρων	10
2.2.3 Open Graphics Library	11
2.3 Ανάπτυξη Λογισμικού	12
2.3.1 Τεχνολογία Λογισμικού	14
2.3.2 Ροές διαδικασίας Τεχνολογίας Λογισμικού	15
2.3.3 Μοντέλα ανάπτυξης λογισμικού	17
2.4 Εκτίμηση Πόζας	20
2.4.1 Μεθοδολογίες εκτίμησης πόζας	21
2.4.2 Μοντελοποίηση ανθρώπινου σώματος	22
2.4.3 Κατηγοριοποιήσεις λύσεων του προβλήματος εκτίμησης πόζας	24
3 Εργαλεία	32
3.1 Kivy	32
3.1.1 Αρχιτεκτονική του Kivy	33
3.2 Wit.ai	35
3.3 Χειρισμός Φωνής	36
3.3.1 SpeechRecognition	37
3.3.2 Text To Speech	37
3.4 Απεικόνιση του ανθρώπινου σώματος	38
3.4.1 SMPL: Skinned multi-person Linear Model	38
3.4.2 OpenGL Shading Language ES	40
3.5 Εκτίμηση 3D ανθρώπινης πόζας	40
3.5.1 NumPy	41

3.5.2	OpenCV	42
3.5.3	Tensorflow	42
3.5.4	Μοντέλο Human Mesh Recovery	45
4	Υλοποίηση Λειτουργικού Καθρέφτη	51
4.1	Model-View-Controller	51
4.2	Απαιτήσεις Συστήματος	52
4.2.1	Λειτουργικές Απαιτήσεις	53
4.2.2	Μη Λειτουργικές Απαιτήσεις	55
4.3	Μοντελοποίηση Κλάσεων	58
4.3.1	SmartMirrorApp	59
4.3.2	MainPage	61
4.3.3	Controller	62
4.3.4	Speech	65
4.3.5	Bot	67
4.3.6	Action	68
4.3.7	MirrorClock	70
4.3.8	Weather	72
5	Υλοποίηση Εκτίμησης Πόζας	75
5.1	Απαιτήσεις Συστήματος	76
5.1.1	Μη Λειτουργικές Απαιτήσεις	79
5.2	Μοντελοποίηση Κλάσεων εκτίμησης πόζας	81
5.2.1	ExercisorScreen	82
5.2.2	ExerciseController	84
5.2.3	Controls	86
5.2.4	Actions	91
5.2.5	MLThreads	99
5.2.6	Renderer	105
6	Συμπεράσματα και Μελλοντικές Επεκτάσεις	111
6.1	Ανάπτυξη Εφαρμογών	111
6.1.1	Δομή Φακέλων	111
6.1.2	Εγκατάσταση Οθόνης	112
6.1.3	Ρυθμίσεις	113
6.1.4	Φωνητικές Εντολές	114
6.2	Συμπεράσματα	114
6.3	Μελλοντικές επεκτάσεις	115
Βιβλιογραφία		117
Α'	Ορισμοί	119
Α'.1	Συναρτήσεις ενεργοποίησης	119
Α'.2	Επίπεδα νευρωνικών δικτύων	122
Α'.3	Υπόλοιπα	125

Κατάλογος Σχημάτων

2.1	Αρχιτεκτονική 3 Επιπέδων	6
2.2	Αρχιτεκτονική 5 Επιπέδων	7
2.3	Δομή Γλικού και Λογισμικού Υπολογιστή	9
2.4	Διαδικασία απεικόνισης γραφικών της OpenGL	12
2.5	Παραδείγματα πρωτόγονων σχημάτων	12
2.6	Σχετική κατανομή κόστους υλικού/λογισμικού (Πηγή: [1])	13
2.7	Καμπύλη αποτυχίας για το λογισμικό	14
2.8	Ροές διαδικασίας λογισμικού	16
2.9	Αυξητικό Μοντέλο	18
2.10	Σπειροειδές Μοντέλο	19
2.11	Παράδειγμα εκτίμησης πόζας	21
2.12	Μοντέλο Σκελετού	23
2.13	Μοντελοποίηση SMPL	24
2.14	Μοντελοποίηση DensePose	24
2.15	Άρθρα εκτίμησης πόζας ανά χρόνο	25
2.16	Μέθοδος εκτίμησης πόζας βασισμένη στον εντοπισμό	27
2.17	Αρχιτεκτονική simple baseline για την εκτίμηση πόζας	28
2.18	Lifting from the deep: Παράδειγμα αρχιτεκτονικής ανύψωσης 2D σε 3D	29
2.19	Ordinal Depth Supervision: Παράδειγμα αρχιτεκτονικής ανύψωσης 2D σε 3D	29
2.20	Αρχιτεκτονική του μοντέλου SPIN	30
2.21	Αρχιτεκτονική του μοντέλου στο άρθρο <i>Learning to Estimate 3D Human Pose and Shape from a Single Color Image</i>	31
3.1	Kivy	32
3.2	Αρχιτεκτονική της βιβλιοθήκης Kivy	33
3.3	Wit.ai	35
3.4	Πλέγματα ανθρώπινου σώματος με χρήση του μοντέλου SMPL	38
3.5	Ο σκελετός του ανθρώπινου σώματος του μοντέλου SMPL	39
3.6	Λογότυπο της βιβλιοθήκης NumPy	41
3.7	Λογότυπο της βιβλιοθήκης OpenCV	42
3.8	Λογότυπο της βιβλιοθήκης Tensorflow	42
3.9	Παράδειγμα υπολογιστικού γράφου	43
3.10	Παράδειγμα τανυστή	44
3.11	Αρχιτεκτονική του μοντέλου Human Mesh Recovery	46
3.12	Αρχιτεκτονική του δικτύου εκτίμησης Human Mesh Recovery - Tensorboard	47

3.13 Tensorboard - Αρχιτεκτονική του resnet_v2_50	48
3.14 Αρχιτεκτονική του residual unit του resnet_v2_50	49
3.15 Tensorboard - Αρχιτεκτονική της μονάδας 3D_module	50
4.1 Αρχιτεκτονική MVC	51
4.2 Παράδειγμα αναπαράστασης κλάσης στη UML	58
4.3 UML της κλάσης SmartMirrorApp	59
4.4 UML της κλάσης MainPage	61
4.5 UML της κλάσης Controller	62
4.6 UML της κλάσης Speech	65
4.7 UML της κλάσης Bot	67
4.8 UML της κλάσης Action	68
4.9 UML της κλάσης MirrorClock	70
4.10 Γραφική αναπαράσταση του Widget MirrorClock	71
4.11 UML της κλάσης Weather	72
4.12 Γραφική αναπαράσταση του Widget Weather	73
5.1 UML διάγραμμα της κλάσης ExercisorScreen	82
5.2 Εποπτικό UML διάγραμμα σχέσεων του ExercisorScreen.	83
5.3 Οι βασικές λειτουργίες του Exercisor	84
5.4 UML διάγραμμα της κλάσης ExerciseController	84
5.5 UML διάγραμμα των κλάσεων τύπου AbstractControls και των σχέσεων τους.	86
5.6 Η γραφική αναπαράσταση των κλάσεων τύπου AbstractControls.	87
5.7 Οι τρόποι απεικόνισης του ανθρώπου.	87
5.8 Οι διαθέσιμοι διάλογοι της κλάσης EditorControls.	90
5.9 Εποπτικό UML διάγραμμα των κλάσεων τύπου AbstractActions και των σχέσεων τους.	91
5.10 UML διάγραμμα της κλάσης AbstractAction.	92
5.11 UML διάγραμμα της κλάσης PredictAction.	93
5.12 UML διάγραμμα της κλάσης PlaybackAction.	94
5.13 Γραφική απεικόνιση της λειτουργίας παιχνιδιού. Το ανθρώπινο πλέγμα είναι η απεικόνιση της άσκηση αναφοράς ενώ με τον σκελετό απεικονίζεται η εκτίμηση πόζας του χρήστη. Στο κίτρινο ορθογώνιο φαίνεται το widget ProgressCounter.	95
5.14 UML διάγραμμα της κλάσης PlayAction.	96
5.15 UML διάγραμμα της κλάσης MLThread.	99
5.16 UML διάγραμμα της κλάσης HMRThread.	101
5.17 UML διάγραμμα της κλάσης SMPThread.	103
5.18 UML διάγραμμα της κλάσης Renderer και των σχέσεων της.	105
6.1 Αναπαράσταση δομής του καθρέφτη	112
A'. 1 Παράδειγμα συνάρτησης argmax	120
A'. 2 Η συνάρτηση ReLU	121
A'. 3 Επίπεδο Fully Connected	122
A'. 4 Αριθμητικό παράδειγμα συνελικτικού επιπέδου	125
A'. 5 Τύποι επιπέδων pooling	126

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

Α'. 6 Παράδειγμα κατευθυνόμενου άκυκλου γράφου	126
Α'. 7 Κάθετα διανύσματα κορυφών πολύεδρου	127
Α'. 8 Αντανάκλαση του φωτός στο κάθετο διάνυσμα	127

1

Εισαγωγή

Τα τελευταία χρόνια τόσο η μείωση του κόστους παραγωγής όσο και η εξουκείωση του ευρέος κοινού με την τεχνολογία έχουν προωθήσει σημαντικά την ανάπτυξη και τη διάδοση των έξυπνων συσκευών. Οι έξυπνες συσκευές βρίσκουν εφαρμογή και αποδεικύονται ολοένα και πιο χρήσιμες σε πολλαπλούς κλάδους. Από την περίθαλψη ασθενών στα νοσοκομεία, την αποφυγή κίνησης στους δρόμους (έξυπνη πόλη), τη βελτίωση των συστημάτων του αυτοκινήτου για καλύτερη οδηγική εμπειρία και αυξημένη ασφάλεια, μέχρι ακόμα και την κουζίνα ενός σπιτιού, οι συσκευές αυτές έχουν τη δυνατότητα να επιδράσουν θετικά στην ποιότητα ζωής μας. Συνδυάζοντας υλικό και λογισμικό εκτελούν εργασίες ή συλλέγουν και αναπαριστούν πληροφορίες, από τις οποίες επωφελούμαστε ατομικά και συλλογικά.

Ειδικότερα οι έξυπνες συσκευές για το σπίτι έχουν προσελκύσει το ενδιαφέρον τόσο εταιρειών παραγωγής συσκευών για το σπίτι, οι οποίες επιθυμούν να αναβαθμίσουν τα προϊόντα τους, όσο και μηχανικών, οι οποίοι καινοτομούν δημιουργώντας τέτοιες συσκευές. Δικαιολογημένα συμβαίνει αυτό, εφόσον τέτοιες συσκευές μπορούν να συνεισφέρουν στην αυτοματοποίηση και κατά συνέπεια βελτιστοποίηση καθημερινών εργασιών, βοηθώντας έτσι στην εξοικονόμηση χρόνου, καθώς και να προσφέρουν οι ίδιες εργαλεία για την οργάνωση χρόνου και σε αρκετές περιπτώσεις να λειτουργήσουν ως μέσο ψυχαγωγίας για τους χρήστες. Για παράδειγμα, ένα έξυπνος θερμοστάτης μπορεί να ρυθμίζει την θερμοκρασία ενός χώρου σε μια προκαθορισμένη από τον άνθρωπο τιμή δημιουργώντας ένα ευχάριστο περιβάλλον και εξοικονομώντας ενέργεια.

Μία οικιακή, κυρίως, συσκευή η οποία μπορεί να παρέχει μια μεγάλη γκάμα εφαρμογών στον χρήστη είναι ο έξυπνος καθρέφτης. Οι έξυπνοι καθρέφτες έρχονται να επαυξήσουν την ανακλαστική επιφάνεια του κλασσικού καθρέφτη με υλικό και λογισμικό, τα οποία θα παρέχουν πληροφορίες στον χρήστη είτε μέσω του διαδικτύου είτε μέσω αισθητήρων. Ίσως το πιο διαδεδομένο πρότζεκτ πάνω στον

έξυπνο καθρέφτη αποτελεί το MagicMirror¹ το οποίο είναι μία αρθρωτή (modular) πλατφόρμα πάνω στην οποία μπορεί κανείς να αναπτύξει και να εγκαταστήσει εφαρμογές για την επαύξηση των δυνατοτήτων του απλού καθρέφτη. Ένα ακόμη εγχείρημα, το οποίο έδωσε αρκετή έμπνευση και στην παρούσα διπλωματική, είναι ο έξυπνος καθρέφτης² της ομάδας Hacker Shack. Αξίζει επίσης να σημειωθεί ότι με μια απλή αναζήτηση στο διαδίκτυο συναντάει εύκολα κανείς και άλλες εφαρμογές πάνω στον έξυπνο καθρέφτη. Συμπεραίνουμε, επομένως, ότι ο έξυπνος καθρέφτης αποτελεί ελκυστικό αντικείμενο ενασχόλησης.

1.1 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Οι περισσότερες εφαρμογές που έχουν αναπτυχθεί πάνω στον έξυπνο καθρέφτη αυτή τη στιγμή αφορούν κυρίως στην ενημέρωση (news and social media feeds), στον συγχρονισμό ηλεκτρονικού ημερολογίου για λήψη υπενθυμίσεων και στην προβολή του καιρού αποτελώντας αρωγό σε καθημερινές δραστηριότητες. Όμως, οι απαιτήσεις χρηστών των παραπάνω εφαρμογών ικανοποιούνται ευκολότερα και αποτελεσματικότερα από άλλες συσκευές, όπως smartphone, καθιστώντας τον έξυπνο καθρέφτη μη ελκυστικό εμπορικό προϊόν.

Βασική αιτία των παραπάνω αποτελεί το εγγενές πρόβλημα αλληλεπίδρασης με τον καθρέφτη, δηλαδή η έλλειψη αποδοτικών συσκευών εισόδου. Η επιλογή που φαντάζει λογικότερη ως είσοδο αποτελεί η φωνή του χρήστη δεδομένου ότι η επαφή του με τον καθρέφτη γίνεται από απόσταση. Όμως, στην ανίχνευση φωνής ελλοχεύουν δύο κύρια προβλήματα. Αρχικά, η αβεβαιότητα των συνθηκών του περιβάλλοντος στο οποίο τοποθετείται ο καθρέφτης δυσχεραίνει την ικανότητα του συστήματος να ανιχνεύει καθαρά τον ήχο. Επιπλέον, οι σημερινές τεχνολογίες αναγνώρισης φωνής και κατανόησης πρόθεσης του χρήστη επιδέχονται βελτίωση.

Επίσης, έως και σήμερα δεν έχει υλοποιηθεί κάποιο λειτουργικό σύστημα, το οποίο να χρησιμοποιείται καθολικά, κανονικοποιώντας έτσι τον τρόπο ανάπτυξης εφαρμογών πάνω στον έξυπνο καθρέφτη. Επομένως, οι λειτουργίες του κάθισε συστήματος καθορίζονται από τον εκάστοτε κατασκευαστή μειώνοντας με αυτόν τον τρόπο την επεκτασιμότητα του έξυπνου καθρέφτη.

Παρόλα αυτά, ο έξυπνος καθρέφτης αποτελεί πρόσφορο έδαφος για την ανάπτυξη εφαρμογών που επαυξάνουν την βασική του λειτουργία, την προβολή του ειδώλου. Συνεχώς ο άνθρωπος αξιοποιεί έναν καθρέφτη προκειμένου να πάρει ανατροφοδότηση σχετικά με την εμφάνισή του, ώστε να συμπεράνει πόσο υγιής φαίνεται ή να αποκτήσει αυτοπεποίθηση, αλλά και σχετικά με την ορθότητα της στάσης και της κίνησής του όταν αθλείται, όπως για παράδειγμα στα γυμναστήρια. Ως εκ τούτου, ο έξυπνος καθρέφτης μπορεί να αποδειχθεί χρήσιμος συμπληρώνοντας τις εφαρμογές του απλού καθρέφτη στους τομείς της υγείας και της ευεξίας.

¹<https://github.com/MichMich/MagicMirror>

²<https://github.com/HackerShackOfficial/Smart-Mirror>

1.2. ΣΚΟΠΟΣ - ΣΥΝΕΙΣΦΟΡΑ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Δυστυχώς, όμως, μέχρι και σήμερα δεν έχει αναπτυχθεί ικανός αριθμός εφαρμογών για τον έξυπνο καθρέφτη που αφορούν στην υγεία.

1.2 ΣΚΟΠΟΣ - ΣΥΝΕΙΣΦΟΡΑ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η παρούσα διπλωματική έχει δύο πρωταρχικούς στόχους, την ανάπτυξη ενός modular λειτουργικού συστήματος το οποίο θα επιτρέπει στους χρήστες να αλληλεπιδράσουν με τον έξυπνο καθρέφτη και την ανάπτυξη μιας εφαρμογής που αφορά στον έλεγχο της ορθότητας μιας άσκησης ενός αθλούμενου. Ο έλεγχος του καθρέφτη από τον χρήστη επιτυγχάνεται μέσω φωνητικών εντολών, ενώ ο έλεγχος της ορθότητας μέσω αποθηκευμένων ασκήσεων, καθορισμένες από ειδικούς, με τις οποίες συγκρίνεται η κίνηση του χρήστη.

Αρχικά, υλοποιήθηκε ένα ελαφρύ γραφικό περιβάλλον για την εμφάνιση πληροφοριών γύρω από το είδωλό του χρήστη. Στη συνέχεια, το λογισμικό εμπλουτίστηκε με λογική για ανάγνωση φωνητικών εντολών και αναγνώριση πρόθεσης με σκοπό την εκτέλεση των κατάλληλων πράξεων, όπως ανανέωση μιας ρύθμισης ή προβολή μιας νέας πληροφορίας. Τέλος, δόθηκε ιδιαίτερη μνεία στην επεκτασιμότητα του συστήματος, δηλαδή στην δυνατότητα εύκολης εγκατάστασης εξωτερικών εφαρμογών από τον χρήστη οι οποίες επαυξάνουν τις λειτουργίες του καθρέφτη.

Όσον αφορά τον δεύτερο στόχο της διπλωματικής, έγινε υλοποίηση μιας εφαρμογής για την εκτίμηση της ορθότητας μιας άσκησης. Η εφαρμογή αντιλαμβάνεται τον χρήστη μέσω κάμερας και κάνει εκτίμηση της πόζας του κατά τη διάρκεια εκτέλεσης της άσκησης. Τα δεδομένα της πόζας συγκρίνονται με ένα "ground truth", το οποίο ορίζεται από κάποιουν ειδικό γυμναστή ή φυσιοθεραπευτή, και δίνεται ανατροφοδότηση στον χρήστη σε πραγματικό χρόνο σχετικά με την ακρίβεια εκτέλεσης της άσκησης. Επιπλέον, παρέχεται η δυνατότητα στον χρήστη να αποθηκεύσει την άσκησή του με σκοπό να την αναπαράξει ξανά στο μέλλον.

1.3 ΔΙΑΡΘΡΩΣΗ ΤΗΣ ΑΝΑΦΟΡΑΣ

Η διάρθρωση της παρούσας διπλωματικής εργασίας είναι η εξής:

- **Κεφάλαιο 2:** Αναλύεται το θεωρητικό υπόβαθρο.
- **Κεφάλαιο 3:** Παρουσιάζονται τα διάφορα εργαλεία που χρησιμοποιήθηκαν στις υλοποιήσεις.
- **Κεφάλαιο 4:** Παρουσιάζεται η υλοποίηση του λειτουργικού συστήματος του καθρέφτη που αναπτύχθηκε.
- **Κεφάλαιο 5:** Παρουσιάζεται η υλοποίηση της εφαρμογής για την εκτίμηση πόζας.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

- **Κεφάλαιο 6:** Παρουσιάζονται τα τελικά συμπεράσματα και προτείνονται θέματα για μελλοντική μελέτη, αλλαγές και επεκτάσεις.

2

Θεωρητικό Υπόβαθρο

2.1 ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ

Το Διαδίκτυο των Πραγμάτων (μτφ. **Internet of Things**, εν συντομίᾳ IoT) είναι ένα σύνολο φυσικών αντικειμένων τα οποία έχουν αισθητήρες, λογισμικό, υπολογιστική ισχύ κτλ. και τα οποία ανταλλάσσουν δεδομένα μεταξύ τους μέσω του Διαδικτύου ή ενός οποιουδήποτε άλλου δικτύου επικοινωνίας. Τα αντικείμενα αυτά μπορεί να είναι ένας άνθρωπος με βηματοδότη στην καρδιά, ένα έξυπνο ρολόι που μετράει παλμούς, ένας καθρέφτης που διαβάζει δεδομένα από αισθητήρες και εμφανίζει πληροφορίες για την κατάσταση του χρήστη.

2.1.1 Πώς δουλεύει το IoT

Ένα σύστημα IoT περιέχει πολλές φυσικές μονάδες (hardware) οι οποίες μπορούν να χωριστούν στις ακόλουθες κατηγορίες [2]:

- Αισθητήρες & Ενεργοποιητές (Sensors & Actuators)
- Μονάδες Επεξεργασίας (Processing Units)
- Μονάδες Αποθήκευσης (Storage Units)
- Μονάδες Επικοινωνίας (Communication Units)

Σε ό,τι αφορά το λογισμικό των IoT συσκευών, δεν υπάρχει κάποια συγκεκριμένη αρχιτεκτονική που να ακολουθείται καθολικά. Έχουν προταθεί αρκετές, ενώ

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

παρακάτω παρουσιάζονται 2 από τις πιο διαδεδομένες, η αρχιτεκτονική 3 επιπέδων και η αρχιτεκτονική 5 επιπέδων [3].

Αρχιτεκτονική 3 επιπέδων

Η αρχιτεκτονική 3 επιπέδων είναι από τις πιο βασικές και πρώιμες αρχιτεκτονικές που αναπτύχθηκαν και απεικονίζεται στο Σχήμα 2.1.



Σχήμα 2.1: Αρχιτεκτονική 3 Επιπέδων

Όπως προκύπτει και από το όνομα αποτελείται από 3 επίπεδα:

- **Επίπεδο Αντίληψης:** Το επίπεδο αντίληψης είναι το φυσικό επίπεδο (hardware) το οποίο περιέχει αισθητήρες για λήψη πληροφοριών και ανίχνευση παραμέτρων από τον περιβάλλοντα χώρο.
- **Επίπεδο Δικτύου:** Το επίπεδο δικτύου είναι υπεύθυνο για την διασύνδεση με άλλες συσκευές, έξυπνα πράγματα κτλ. Χρησιμοποιείται επίσης για την μετάδοση και επεξεργασία δεδομένων των αισθητήρων.
- **Επίπεδο Εφαρμογής:** Το επίπεδο εφαρμογής είναι υπεύθυνο για την παροχή υπηρεσιών συγκεκριμένης εφαρμογής (application specific services) στον χρήστη. Καθορίζει τις διάφορες εφαρμογές στις οποίες μπορεί να αναπτυχθεί το IoT, όπως το έξυπνο σπίτι, ο έξυπνος καθρέφτης κ.α.

Αρχιτεκτονική 5 επιπέδων

Η αρχιτεκτονική των 3 επιπέδων περιγράφει την βασική ιδέα πίσω από το IoT, αλλά στην πράξη δεν επαρκή. Για το λόγο αυτό, υπάρχουν αρχιτεκτονικές με περισσότερα επίπεδα στην βιβλιογραφία. Μία από αυτές είναι η αρχιτεκτονική 5 επιπέδων που φαίνεται στο Σχήμα 2.2.



Σχήμα 2.2: Αρχιτεκτονική 5 Επιπέδων

Στην περίπτωση αυτή, τα επίπεδα Αντίληψης και Εφαρμογής παραμένουν ίδια με αυτά της αρχιτεκτονικής 3 επιπέδων ενώ επεξηγούνται και τα υπόλοιπα 3 επίπεδα:

- Επίπεδο Μεταφοράς:** Το επίπεδο μεταφοράς, μεταδίδει τα δεδομένα των αισθητήρων από το επίπεδο αντίληψης στο επίπεδο επεξεργασίας και αντίστροφα μέσω δικτύων όπως 3G, LAN, RFID, NFC κτλ.
- Επίπεδο Επεξεργασίας:** Το επίπεδο επεξεργασίας, γνωστό και ως middleware, αποθηκεύει, αναλύει και επεξεργάζεται μεγάλο όγκο δεδομένων που προέρχονται από το επίπεδο μεταφοράς. Προσφέρει μια ευρεία γκάμα υπηρεσιών

στα χαμηλότερα επίπεδα. Χρησιμοποιεί επίσης πολλές τεχνολογίες όπως βάσεις δεδομένων, υπολογιστική νέφους και ενότητες επεξεργασίας μεγάλων δεδομένων.

- **Επίπεδο Επιχείρησης:** Το επίπεδο επιχείρησης διαχειρίζεται όλο το IoT σύστημα, συμπεριλαμβανομένων των εφαρμογών του, των επιχειρηματικών του μοντέλων και της ιδιωτικότητας των χρηστών.

2.2 ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Σύμφωνα με το [4]: Τα **Λειτουργικά Συστήματα** (εν συντομίᾳ **ΛΣ**) είναι προγράμματα που ελέγχουν την εκτέλεση προγραμμάτων εφαρμογών και δρουν ως διεπαφή ανάμεσα στις εφαρμογές και το υλικό του υπολογιστή. Θα λέγαμε ότι έχουν τρεις στόχους:

- **Ευκολία:** Τα ΛΣ κάνουν ευκολότερη τη χρήση ενός υπολογιστή.
- **Αποτελεσματικότητα:** Τα ΛΣ επιτρέπουν την αποτελεσματική χρήση των πόρων ενός υπολογιστικού συστήματος.
- **Ικανότητα Εξέλιξης:** Τα ΛΣ πρέπει να είναι κατασκευασμένα με τέτοιο τρόπο, ώστε να επιτρέπουν την αποτελεσματική ανάπτυξη, τον έλεγχο και την εισαγωγή νέων λειτουργιών συστήματος, χωρίς να παρεμβαίνουν στην παροχή υπηρεσιών.

2.2.1 Το ΛΣ ως Διεπαφή Χρήστη/Υπολογιστή

Ένας υπολογιστής είναι ένα σύστημα που αποτελείται από το υλικό (hardware) και το λογισμικό (software). Το υλικό είναι ο φυσικός εξοπλισμός (οθόνες, μνήμες, εκτυπωτές κτλ) ενώ το λογισμικό είναι μια συλλογή προγραμμάτων που επιτρέπουν στο υλικό να λειτουργεί [5]. Το λογισμικό διαιρείται σε 2 κατηγορίες: στα προγράμματα εφαρμογών και στα λειτουργικά συστήματα. Το υλικό και το λογισμικό που αποτελούν ένα υπολογιστικό σύστημα μπορεί να αναπαρασταθεί από μια ιεραρχική δομή ή μια δομή επιπέδων, όπως φαίνεται στο Σχήμα 2.3.

Το ΛΣ, επομένως, «κάθεται» ανάμεσα στο υλικό και στις εφαρμογές που χρησιμοποιεί ο χρήστης ο οποίος συνήθως δεν ενδιαφέρεται για λεπτομέρειες του υλικού του υπολογιστή. Δρα, δηλαδή, ως ένας διαμεσολαβητής που διευκολύνει την επικοινωνία του χρήστη με τον υπολογιστή χωρίς ο πρώτος να χρειάζεται να γνωρίζει την γλώσσα του τελευταίου, ενώ είναι υπεύθυνο για την ομαλή λειτουργία των προγραμμάτων εφαρμογών. Για παράδειγμα, προκειμένου να εμφανιστεί το κείμενο «Χαίρε Κόσμε» στην οθόνη, πρέπει μερικές εκατοντάδες πίξελ να λειτουργήσουν σε συγκεκριμένες θέσεις. Αυτό μπορεί να γίνει με το να διαβάζει κανείς τις προδιαγραφές του υλικού και να γράψει κώδικα που χειρίζεται τα σωστά μπιτς στην



Σχήμα 2.3: Δομή Γλικού και Λογισμικού Υπολογιστή

μνήμη κάτι που είναι αρκετά επίπονο. Οι περισσότεροι χρήστες όμως θέλουν απλά να γράψουν την εντολή `print("Hello World")` χωρίς να νοιαστούν για περαιτέρω λεπτομέρειες. Εκεί είναι που μπαίνει το ΛΣ για να δώσει λύση [6].

Πιο αναλυτικά, μερικές από τις περιοχές στις οποίες το ΛΣ παρέχει υπηρεσίες περιγράφονται παρακάτω σύμφωνα με το [4]:

- **Στην ανάπτυξη προγραμμάτων:** Το ΛΣ παρέχει πληθώρα υπηρεσιών, όπως επεξεργαστές κειμένου και διορθωτές λαθών (debuggers), προκειμένου να βοηθήσει τον προγραμματιστή στην ανάπτυξη εφαρμογών. Συνήθως, οι υπηρεσίες αυτές έχουν τη μορφή βιοηθητικών προγραμμάτων, τα οποία αν και δεν αποτελούν αυστηρά τμήμα του πυρήνα του λειτουργικού συστήματος, παρέχονται μαζί με το ΛΣ και αναφέρονται ως εργαλεία ανάπτυξης προγραμμάτων εφαρμογών.
- **Στην εκτέλεση προγραμμάτων:** Για την εκτέλεση ενός προγράμματος απαιτείται πλήθος βημάτων. Εντολές και δεδομένα πρέπει να φορτωθούν στην κύρια μνήμη, συσκευές Εισόδου/Εξόδου (Ε/Ε) και αρχικοποιηθούν και διάφοροι άλλοι πόροι πρέπει να ετοιμαστούν. Το ΛΣ διαχειρίζεται αυτές τις υποχρεώσεις χρονοδρομολόγησης για λογαριασμό του χρήστη.
- **Στην πρόσβαση σε συσκευές Εισόδου/Εξόδου(Ε/Ε):** Κάθε συσκευή Ε/Ε απαιτεί το δικό της ιδιαίτερο σύνολο εντολών ή σημάτων ελέγχου για τη λειτουργία

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

της. Το ΛΣ παρέχει ενιαία διεπαφή που αποκρύπτει αυτές τις λεπτομέρειες, έτσι ώστε οι προγραμματιστές να έχουν πρόσβαση σε τέτοιες συσκευές χρησιμοποιώντας απλές αναγνώσεις (reads) και εγγραφές (writes).

- **Στην ελεγχόμενη πρόσβαση σε αρχεία:** Το ΛΣ πρέπει να αναπαριστά τη λεπτομερή κατανόηση, όχι μόνο της φύσης των συσκευών Ε/Ε (οδηγού δίσκου, οδηγού ταινίας), αλλά και της δομής των δεδομένων που βρίσκονται σε αρχεία στο αποθηκευτικό μέσο. Επίσης, στην περίπτωση συστημάτων πολλαπλών χρηστών, το ΛΣ μπορεί να παρέχει μηχανισμούς προστασίας για τον έλεγχο της πρόσβασης σε αρχεία.
- **Στην πρόσβαση στο σύστημα:** Σε ό,τι αφορά διαμοιραζόμενα ή δημόσια συστήματα, το ΛΣ ελέγχει συνολικά την πρόσβαση στο σύστημα και σε συγκεκριμένους πόρους του. Η λειτουργία πρόσβασης πρέπει να παρέχει προστασία των πόρων και των δεδομένων από μη εξουσιοδοτημένους χρήστες και να διευθετεί θέματα συγκρούσεων και διεκδίκησης πόρων.
- **Στην ανίχνευση σφαλμάτων και στην απόκριση:** Διάφορα σφάλματα μπορούν να προκύψουν κατά τη διάρκεια λειτουργίας ενός υπολογιστικού συστήματος. Μεταξύ αυτών περιλαμβάνονται εσωτερικά και εξωτερικά σφάλματα υλικού, όπως σφάλματα μνήμης ή δυσλειτουργία κάποιας συσκευής, καθώς και διάφορα σφάλματα λογισμικού, όπως η διαίρεση με το μηδέν και η προσπάθεια προσπέλασης απαγορευμένης θέσης μνήμης. Σε κάθε περίπτωση, το ΛΣ πρέπει να παρέχει μια απόκριση, η οποία απαλείφει τη συνθήκη σφάλματος, με το μικρότερο δυνατό αντίκτυπο στις εφαρμογές που εκείνη την ώρα εκτελούνται. Απόκριση μπορεί να αποτελεί η λήξη του προγράμματος που προκάλεσε το σφάλμα, η επανέναρξη λειτουργίας ή ακόμα και η απλή αναφορά σφάλματος στην εφαρμογή.
- **Στην λογιστική:** Ένα καλό ΛΣ συλλέγει στατιστικά χρήσης διάφορων πόρων και παρακολουθεί παραμέτρους απόδοσης, όπως είναι ο χρόνος απόκρισης. Σε κάθε σύστημα, οι πληροφορίες αυτές είναι χρήσιμες για τη λήψη αποφάσεων σχετικών με μελλοντικές αναβαθμίσεις και με τις ρυθμίσεις του συστήματος, ώστε να επιτυγχάνεται η βελτίωση της απόδοσης. Σε συστήματα πολλαπλών χρηστών, οι πληροφορίες αυτές μπορούν να χρησιμοποιηθούν και για λόγους χρέωσης.

2.2.2 Το ΛΣ ως Διαχειριστής Πόρων

Όπως αναφέρεται και παραπάνω το ΛΣ είναι μεταξύ άλλων υπεύθυνο για τον έλεγχο των πόρων του υπολογιστή, δηλαδή τις μνήμες, τους δίσκους, τον επεξεργαστή κτλ. Αναφέρθηκε, επίσης, ότι το ΛΣ είναι και αυτό ένα λογισμικό και άρα λειτουργεί με τον ίδιο τρόπο που λειτουργεί οποιαδήποτε άλλη εφαρμογή, δηλαδή είναι μια ακολουθία εντολών που εκτελούνται από τον επεξεργαστή. Κατά την φάση εκτέλεσης, το ΛΣ κατανέμει τον χρόνο του επεξεργαστή και τους πόρους του υπολογιστή στα προγράμματα που πρόκειται να τρέξουν. Για να εκτελεστεί όμως μια εφαρμογή ή ένα πρόγραμμα ο επεξεργαστής πρέπει να διακόψει την

εκτέλεση του ΛΣ, ενώ μετά το πέρας των εντολών επαναφέρει το ΛΣ στον έλεγχο προκειμένου να γίνουν οι απαραίτητες προετοιμασίες για τις επόμενες εργασίες [4].

Ακριβώς επειδή το ΛΣ βοηθάει την εκτέλεση σχεδόν κάθε προγράμματος πρέπει να είναι και αρκετά αποδοτικό. Για παράδειγμα, τα προγράμματα εφαρμογών δημιουργούν αντικείμενα και πίνακες συνέχεια οπότε είναι σημαντικό αυτή η δουλειά να γίνεται γρήγορα και με εξοικονόμηση πόρων. Οποιοδήποτε κέρδος του ΛΣ, είτε σε ταχύτητα είτε σε μνήμη, μπορεί να επηρεάσει δραματικά την απόδοση του υπολογιστικού συστήματος [6].

2.2.3 Open Graphics Library

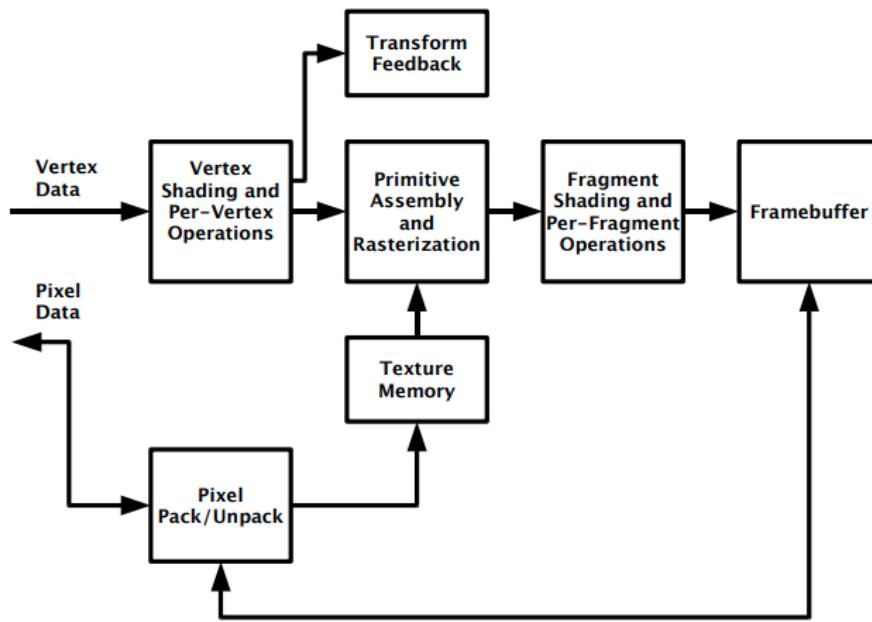
Αντίστοιχα με το ΛΣ αλλά ανεξάρτητο από αυτό, η **Open Graphics Library**[7] (εν συντομίᾳ **OpenGL**) αποτελεί μία *Διεπαφή Προγραμματισμού Εφαρμογών* για τη βέλτιστη χρήση των πόρων της κάρτας γραφικών με σκοπό την οικονομική και γρήγορη απόδοση 2D και 3D γραφικών. Η διεπαφή περιλαμβάνει ένα σύνολο εντολών οι οποίες επιτρέπουν στον χρήστη να καθορίσει τα απαραίτητα αντικείμενα και διεργασίες για την παραγωγή υψηλής ποιότητας έγχρωμων εικόνων δισδιάστατων ή τρισδιάστατων αντικειμένων.

Τρόπος λειτουργίας της OpenGL

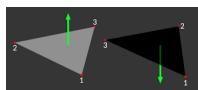
Η OpenGL είναι υπεύθυνη για την επεξεργασία των δεδομένων στην μνήμη της κάρτας γραφικών, την εγγραφή δεδομένων στον καταχωρητή που αποδίδει τα πίξελ της οθόνης (framebuffer) και την ανάγνωση του. Ο framebuffer απαρτίζεται από ένα σύνολο πίξελ διατεταγμένων σε ένα δισδιάστατο πίνακα. Κάθε στοιχείο του πίνακα αποτελείται από έναν αριθμό bits (ανάλογα με την υλοποίηση της OpenGL) τα οποία καθορίζουν το χρώμα, το βάθος και το στένσιλ για κάθε πίξελ.

Στο Σχήμα 2.4 φαίνεται η διαδικασία απεικόνισης γραφικών της OpenGL. Αρχικά, δέχεται ως είσοδο τα δεδομένα των κορυφών των αντικειμένων προς προβολή, τα οποία συνθέτονται σε πρωτόγονα σχήματα. Με άλλα λόγια, οι κορυφές μεταμορφώνονται σε γεωμετρικά πρωτόγονα σχήματα, συνήθως τρίγωνα ή πολύγωνα, Σχήμα 2.5α, τα οποία μέσω της διαδικασίας ψηφίδωσης, διαμορφώνουν ένα πλέγμα, παράγοντας περιπλοκότερα σχήματα, όπως το πολύεδρο στο Σχήμα 2.5β. Προαιρετικά, τα αποτελέσματα από αυτά τα στάδια δύναται να ανατροφοδοτήσουν ενδιάμεσες μνήμες, όπου αποθηκεύονται για μετέπειτα χρήση.

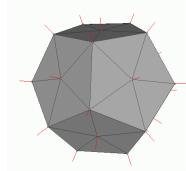
Τα τελικά πρωτόγονα σχήματα περικόπτονται από έναν καθορισμένο όγκο για να προετοιμαστούν για το στάδιο της ψηφιοποίησης. Η διαδικασία αυτή παράγει σειρές από διευθύνσεις του framebuffer συνοδευόμενες από τιμές που περιγράφουν τη δισδιάστατη απεικόνιση των αρχικών τρισδιάστατων πρωτόγονων σχημάτων.



Σχήμα 2.4: Διαδικασία απεικόνισης γραφικών της OpenGL



(α') Πρωτόγονο σχήμα τρίγωνο. Φαίνονται οι τρεις κορυφές του τριγώνου καθώς και το κάθετο διάνυσμα στην επιφάνεια του.



(β') Τρισδιάστατο δωδεκάεδρο πλέγμα που απαρτίζεται από τρίγωνα.

Σχήμα 2.5: Παραδείγματα πρωτόγονων σχημάτων

Κάθε τμήμα που παράγεται με αυτό τον τρόπο υπόκειται σε περαιτέρω διεργασίες ξεχωριστά. Οι διεργασίες περιλαμβάνουν υπό συνθήκη ενημέρωση των τιμών σε συνάρτηση με εισερχόμενες ή αποθηκευμένες τιμές του βάθους ή του στένσιλ (επιπλέον τιμές που καθορίζουν μετέπειτα επεξεργασία κατά την απεικόνιση), ανάμειξη των εισερχόμενων χρωμάτων με αποθηκευμένα χρώματα ή άλλες διεργασίες στις τιμές των τμημάτων. Τέλος, ο framebuffer επικαιροποιείται με τις τιμές των τμημάτων για κάθε πίξελ, προβάλλοντας έτσι την τελική εικόνα στην οθόνη.

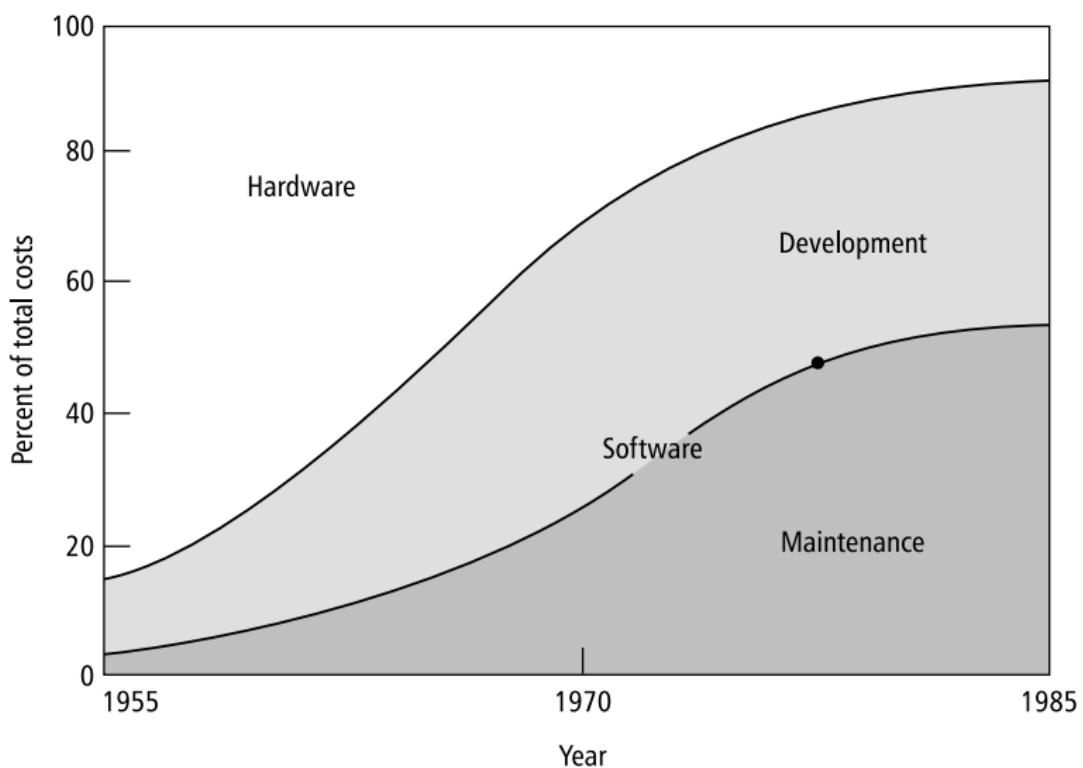
2.3 ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ

Στη σύγχρονη πραγματικότητα, το λογισμικό έχει κυριαρχήσει στην καθημερινή ζωή, καθιστώντας το μία από τις σημαντικότερες τεχνολογίες. Παίζοντας ταυτό-

2.3. ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ

χρονα το ρόλο προϊόντος και εργαλείου, αποτελεί τη βάση της επιστημονικής έρευνας και της επίλυσης προβλημάτων μηχανικής, καθιστώντας δυνατή την δημιουργία καινούργιων και την επέκταση υπαρχουσών τεχνολογιών (π.χ. γενετική μηχανική και τηλεπικοινωνίες αντίστοιχα). Ταυτόχρονα, έχει διεισδύσει σε συστήματα κάθε είδους: μεταφοράς, ιατρικά, τηλεπικοινωνιών, στρατιωτικά, βιομηχανικά, φυσαγωγίας κ.α. αλλάζοντας τον τρόπο που αντιλαμβανόμαστε και αλληλεπιδράμε με τον κόσμο.

Έτσι, τα λογισμικά προγράμματα αντιμετωπίζουν ολοένα και περισσότερα προβλήματα της καθημερινής ζωής, αυξάνοντας την αναγκαιότητα και το κόστος τους. Πράγματι, το συνολικό κόστος λογισμικού υπολογίζεται στα €500 δισεκατομμύρια στην Αμερική και διπλάσιο παγκοσμίως [8]. Αυτό αναφέρεται τόσο στο κόστος ανάπτυξης του λογισμικού όσο και συντήρησής του αφού έχει παραδοθεί στον πελάτη. Ταυτόχρονα, το κόστος του υλικού έχει μειωθεί δραματικά, αποτελώντας λιγότερο από το 20% των συνολικών εξόδων ενός συστήματος όπως φαίνεται στο Σχήμα 2.6.

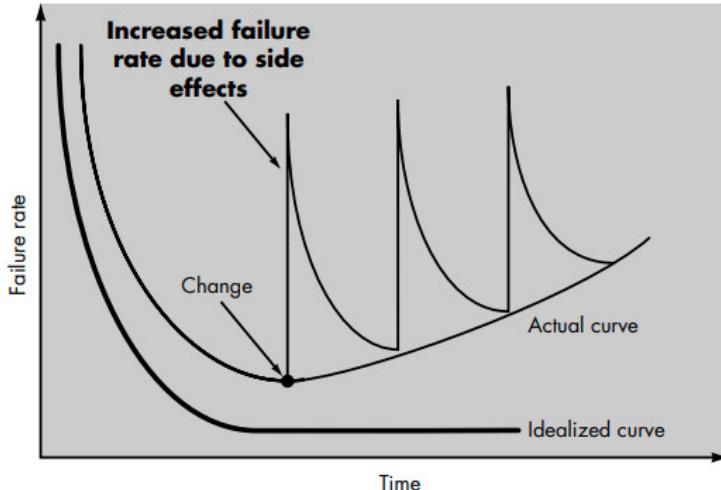


Σχήμα 2.6: Σχετική κατανομή κόστους υλικού/λογισμικού (Πηγή: [1])

Η διαφορά κόστους έγκειται στο γεγονός ότι τα σύγχρονα προγράμματα λογισμικού είναι μεγάλα και περίπλοκα, απαιτώντας ομάδες υψηλής εξειδίκευσης, δεν έχουν περιορισμούς (δηλαδή έχουν περισσότερους βαθμούς ελευθερίας) και τέλος επειδή υφίσταται συνεχόμενες αλλαγές. Μάλιστα, στη διάρκεια ζωής του λογισμικού οι αλλαγές αυτές ενδέχεται να εισάγουν σφάλματα, αυξάνοντας τον κίνδυνο αποτυχίας του συστήματος, όπως φαίνεται στο Σχήμα 2.7. Επιπλέον, σε αντίθεση με την αποτυχία του υλικού, όπου αντιμετωπίζεται με αντικατάσταση του χαλασμένου

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

μέρους, η αποτυχία του λογισμικού έγκειται σε σφάλμα σχεδιασμού, καθιστώντας τη συντήρηση του σημαντικά πιο περίπλοκη και ακριβή διαδικασία.



Σχήμα 2.7: Καμπύλη αποτυχίας για το λογισμικό

Εντούτοις, η αυξανόμενη εξάρτηση των καθημερινών δραστηριοτήτων από περίπλοκα συστήματα λογισμικού απαιτεί την εύρωστη και αλάνθαστη λειτουργία τους καθ' όλη τη διάρκεια ζωής του λογισμικού, το οποίο όμως αλλάζει συνεχώς αυξάνοντας τον κίνδυνο αποτυχίας του. Τοιουτοτρόπως, για την μείωση του ρίσκου αποτυχίας και την απλούστευση της ανάπτυξης και συντήρησης υψηλής ποιότητας έργων λογισμικού, έπρεπε να τυποποιηθούν οι μέθοδοι και διαδικασίες που διέπουν ολόκληρο τον κύκλο ζωής του. Αυτές οι μεθοδολογίες εμπεριέχονται και αναλύονται από την επιστήμη της τεχνολογίας λογισμικού.

2.3.1 Τεχνολογία Λογισμικού

Η τεχνολογία λογισμικού είναι η συστηματική, πειθαρχημένη και μετρήσιμη προσέγγιση με σκοπό την ανάπτυξη, λειτουργία και συντήρηση υψηλής ποιότητας έργων λογισμικού. Οι διαδικασίες της τεχνολογίας λογισμικού παρέχουν ένα ευρύτερο πλαίσιο, μέσα στο οποίο εφαρμόζονται οι τεχνικές μέθοδοι, παράγονται προϊόντα δουλειάς (μοντέλα, έγγραφα, δεδομένα, αναφορές κ.α.), εξακριβώνονται σημεία σταθμοί, εξασφαλίζεται η ποιότητα και γίνεται κατάλληλη διαχείριση των αλλαγών των απαιτήσεων.

Οι μέθοδοι της τεχνολογίας λογισμικού είναι το τεχνικό εγχειρίδιο για την ανάπτυξη λογισμικού. Οι μέθοδοι εγκολπώνουν ένα ευρύ σύνολο λειτουργιών συμπεριλαμβανομένων την ανάλυση απαιτήσεων, σχεδιασμού, δημιουργία προγράμματος, δοκιμής και υποστήριξης. Οι μέθοδοι της τεχνολογίας λογισμικού βασίζονται σε βασικές αρχές και πρακτικές που διέπουν κάθε τομέα της τεχνολογίας και περιλαμβάνουν δραστηριότητες μοντελοποίησης και άλλες περιγραφικές τεχνικές.

Πλαίσιο διαδικασίας

Το πλαίσιο διαδικασίας της τεχνολογίας λογισμικού απαρτίζεται από δραστηριότητες που είναι κοινές για οποιοδήποτε έργο λογισμικού ανεξάρτητα από το μέγεθος και την περιπλοκότητα του. Ένα γενικό πλαίσιο διαδικασίας περιλαμβάνει πέντε κύριες δραστηριότητες [9]:

- **Επικοινωνία (Communication):** Πριν την έναρξη οποιασδήποτε τεχνικής εργασίας, είναι άκρως σημαντική η επικοινωνία και συνεργασία με τον πελάτη (και άλλα ενδιαφερόμενα μέλη). Η πρόθεση είναι η κατανόηση των στόχων των ενδιαφερόμενων μελών για το έργο λογισμικού και η συλλογή απαιτήσεων, βοηθώντας τον καθορισμό των λειτουργιών και των χαρακτηριστικών του λογισμικού.
- **Προγραμματισμός (Planning):** Ένα έργο λογισμικού είναι ένα περίπλοκο ταξίδι και η διαδικασία του προγραμματισμού είναι ο χάρτης ο οποίος οδηγεί την ομάδα ανάπτυξης. Το σχέδιο του έργου αποσαφηνίζει την απαιτούμενη δουλειά, περιγράφοντας τις τεχνικές εργασίες προς διεκπεραίωση, τα πιθανά ρίσκα, τους απαιτούμενους πόρους, τα προϊόντα προς παραγωγή και το πρόγραμμα εργασίας.
- **Μοντελοποίηση (Modelling):** Δημιουργούνται μοντέλα ώστε να γίνει εφικτή η καλύτερη κατανόηση των απαιτήσεων του λογισμικού και του σχεδιασμού ο οποίος δύναται να ικανοποιήσει αυτές τις απαιτήσεις. Γίνεται εμφανής η αρχιτεκτονική μορφή του έργου, ο τρόπος αλληλεξάρτησης των μεμονωμένων μερών και πολλά άλλα χαρακτηριστικά, στη προσπάθεια κατανόησης του προβλήματος και του τρόπου επίλυσης του.
- **Κατασκευή (Construction):** Η δραστηριότητα αυτή περιλαμβάνει την υλοποίηση του έργου παράγοντας κώδικα (είτε με χειροκίνητο ή αυτόματο τρόπο) και τον έλεγχο του κώδικα για την ανίχνευση σφαλμάτων.
- **Εγκατάσταση (Deployment):** Το λογισμικό ως ολόκληρη οντότητα ή σε επιμέρους διαδοχικά στάδια παραδίνεται στον πελάτη ο οποίος αξιολογεί το παραδοθέν προϊόν και παρέχει ανατροφοδότηση.

Σε πολλά έργα λογισμικού, οι δραστηριότητες πλαισίου εφαρμόζονται επαναληπτικά καθώς αναπτύσσεται το έργο. Κάθε επανάληψη παράγει μία επαύξηση του λογισμικού η οποία θέτει στην διάθεση των ενδιαφερόμενων μελών ένα υποσύνολο των τελικών χαρακτηριστικών και λειτουργιών του λογισμικού. Τοιουτοτρόπως, το λογισμικό ολοκληρώνεται σταδιακά με το πέρας κάθε επανάληψης.

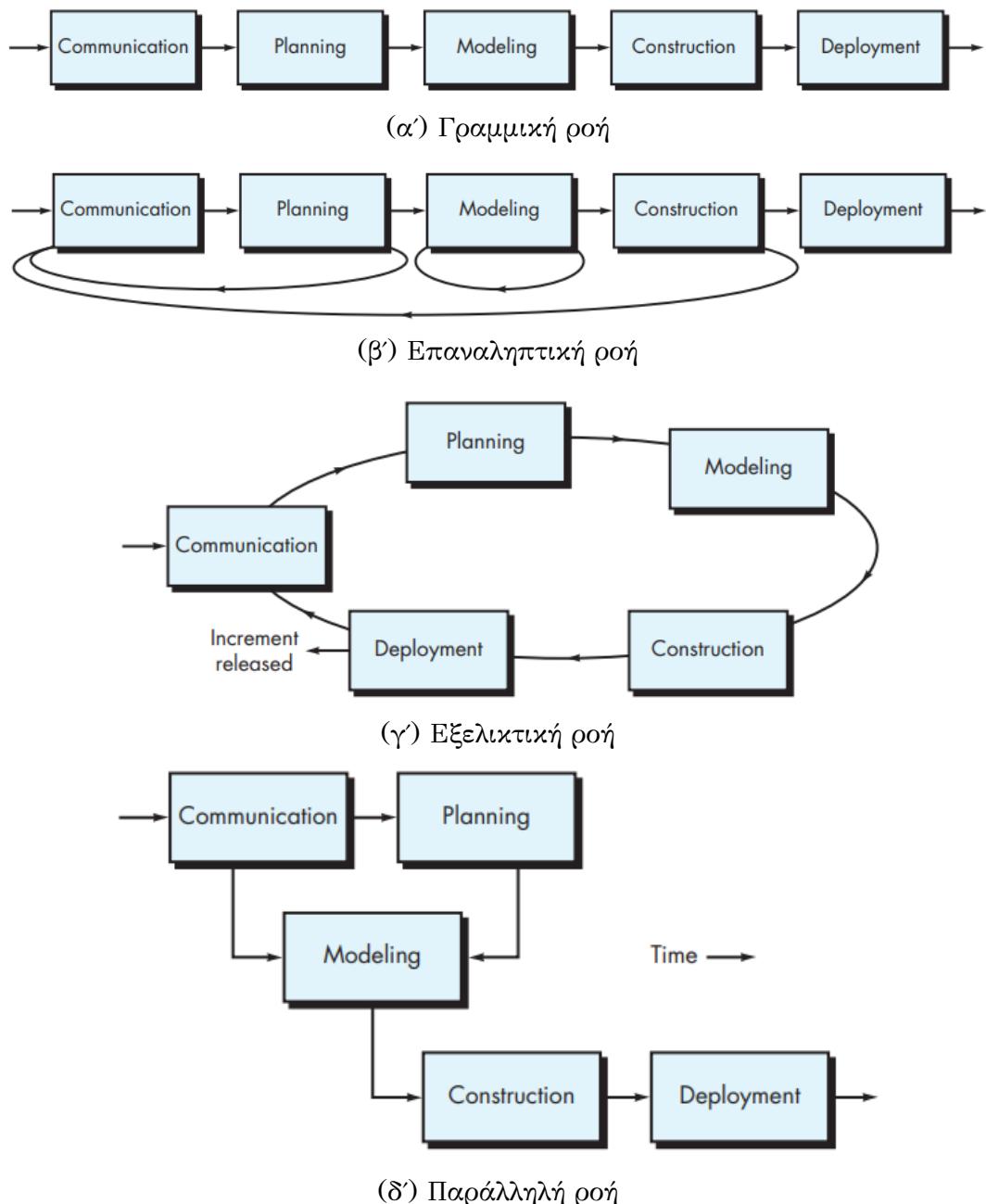
2.3.2 Ροές διαδικασίας Τεχνολογίας Λογισμικού

Μία διαδικασία ορίζεται ως το άθροισμα των δραστηριοτήτων, ενεργειών και εργασιών που πρέπει να διεκπεραιωθούν με σκοπό την παραγωγή ενός προϊόντος εργασίας. Όπως αναφέρθηκε παραπάνω, το γενικό πλαίσιο διαδικασίας του λογισμικού αποτελείται από πέντε δραστηριότητες σε συνδυασμό με άλλες ενέργειες

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

- όπως παρακολούθηση και έλεγχος έργου, εκτίμηση και διαχείριση ρίσκου, διασφάλιση ποιότητας, ρύθμιση παραμέτρων κ.α. - που εφαρμόζονται καθ' όλη την διαδικασία.

Η ροή διαδικασίας περιγράφει τον τρόπο με τον οποίο οι δραστηριότητες πλαισίουν, οι ενέργειες και οι εργασίες εντός κάθε δραστηριότητας οργανώνονται σε σχέση με την αλληλουχία τους και τον χρόνο όπως φαίνεται στο Σχήμα 2.8.



Σχήμα 2.8: Ροές διαδικασίας λογισμικού

Η γραμμική ροή εκτελεί τις δραστηριότητες διαδοχικά, ξεκινώντας την κάθε δραστηριότητα με το πέρας της προηγούμενης (2.8α'). Η επαναληπτική ροή επα-

ναλαμβάνει μία ή περισσότερες δραστηριότητες πριν προχωρήσει στην επόμενη (2.8β'). Η εξελικτική ροή εκτελεί τις δραστηριότητες κυκλικά. Στο πέρας της διεκπεραίωσης των δραστηριοτήτων παράγεται σταδιακά η εφαρμογή ολοένα και πιο ολοκληρωμένη (2.8γ'). Η παράλληλη ροή (2.8δ') εκτελεί μία ή περισσότερες δραστηριότητες παράλληλα με άλλες δραστηριότητες (για παράδειγμα η μοντελοποίηση μιας πτυχής του λογισμικού μπορεί να γίνεται ταυτόχρονα με την κατασκευή μιας άλλης πτυχής του).

2.3.3 Μοντέλα ανάπτυξης λογισμικού

Μοντέλο Καταρράκτη

Το μοντέλο ανάπτυξης καταρράκτη ακολουθεί την γραμμική ροή, δηλαδή οι δραστηριότητες από την επικοινωνία μέχρι την εγκατάσταση διεκπεραιώνονται διαδοχικά. Το μοντέλο αυτό επιλέγεται όταν οι απαιτήσεις του λογισμικού είναι ξεκάθαρα ορισμένες και επαρκώς σταθερές.

Το μοντέλο καταρράκτη είναι το παλαιότερο μοντέλο της τεχνολογίας λογισμικού. Εντούτοις, με την εξέλιξη της τεχνολογίας και των απαιτήσεων έχει παρουσιάσει πληθώρα προβλημάτων.

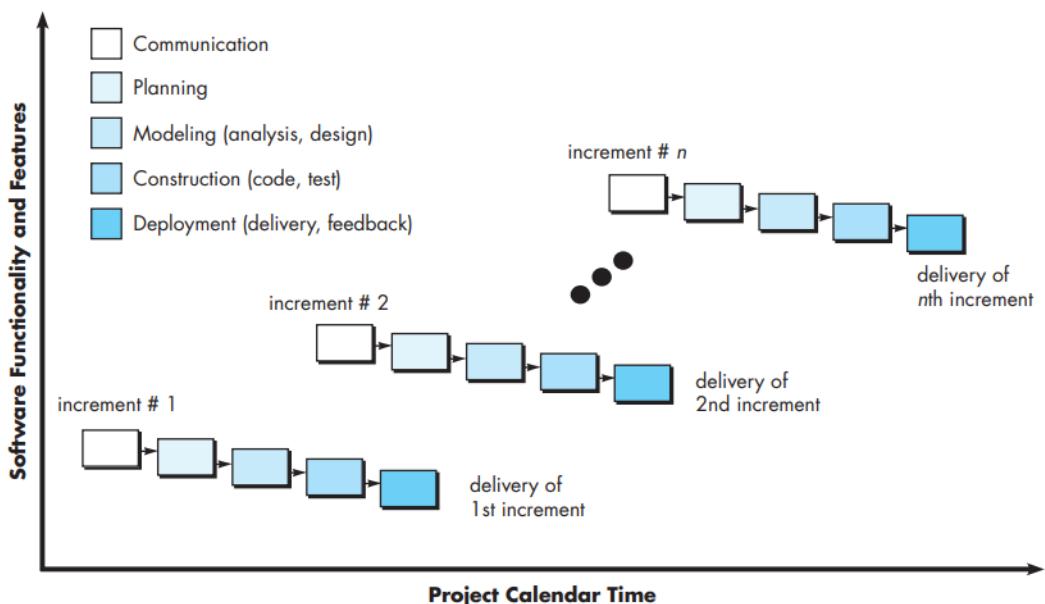
- Τα έργα λογισμικού σπάνια ακολουθούν την προτεινόμενη γραμμική ροή. Τοιουτοτρόπως, οι αλλαγές δεν είναι εύκολα διαχειρίσιμες και μπορεί να προκαλέσουν σύγχυση.
- Συνήθως είναι δύσκολο για τον πελάτη να καθορίσει όλες τις απαιτήσεις αναλυτικά. Το μοντέλο καταρράκτη όμως το απαιτεί και έτσι δεν μπορεί να διαχειριστεί την αβεβαιότητα που έγκειται στην έναρξη οποιουδήποτε έργου λογισμικού.
- Μία λειτουργική έκδοση του λογισμικού γίνεται διαθέσιμη αρκετά αργά στην διάρκεια ζωής του έργου. Έτσι, από την μία ο πελάτης πρέπει να είναι υπομονετικός και από την άλλη μία αστοχία σχεδιασμού που θα ανιχνευτεί κατά την αξιολόγηση της λειτουργικής έκδοσης μπορεί να είναι καταστροφική.

Αυξητικό Μοντέλο

Το αυξητικό μοντέλο συνδυάζει την γραμμική και την παράλληλη ροή διαδικασιών. Όπως φαίνεται στο Σχήμα 2.9, το αυξητικό μοντέλο χωρίζει τις λειτουργίες και τα χαρακτηριστικά του λογισμικού σε μικρές επαυξήσεις και για κάθε επαύξηση διεκπεραιώνει γραμμικά τις δραστηριότητες, στο πέρας των οποίων παραδίδεται η λειτουργία του λογισμικού.

Οι πρώτες επαυξήσεις υλοποιούν τις κυριότερες λειτουργίες του λογισμικού, ικανοποιώντας τις αρχικές απαιτήσεις. Το προϊόν παραδίδεται στον πελάτη ή υπό-

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ



Σχήμα 2.9: Αυξητικό Μοντέλο

κειται σε λεπτομερή αξιολόγηση. Τοιουτοτρόπως, αναπτύσσεται ένα σχέδιο για τις επόμενες επαυξήσεις. Το σχέδιο αφορά τις αλλαγές του κύριου προϊόντος, με σκοπό την καλύτερη ικανοποίηση των απαιτήσεων του πελάτη, και την ανάπτυξη επιπλέον χαρακτηριστικών και λειτουργιών. Η διαδικασία επαναλαμβάνεται μετά από την παράδοση κάθε επαύξησης έως ότου ολοκληρωθεί το λογισμικό.

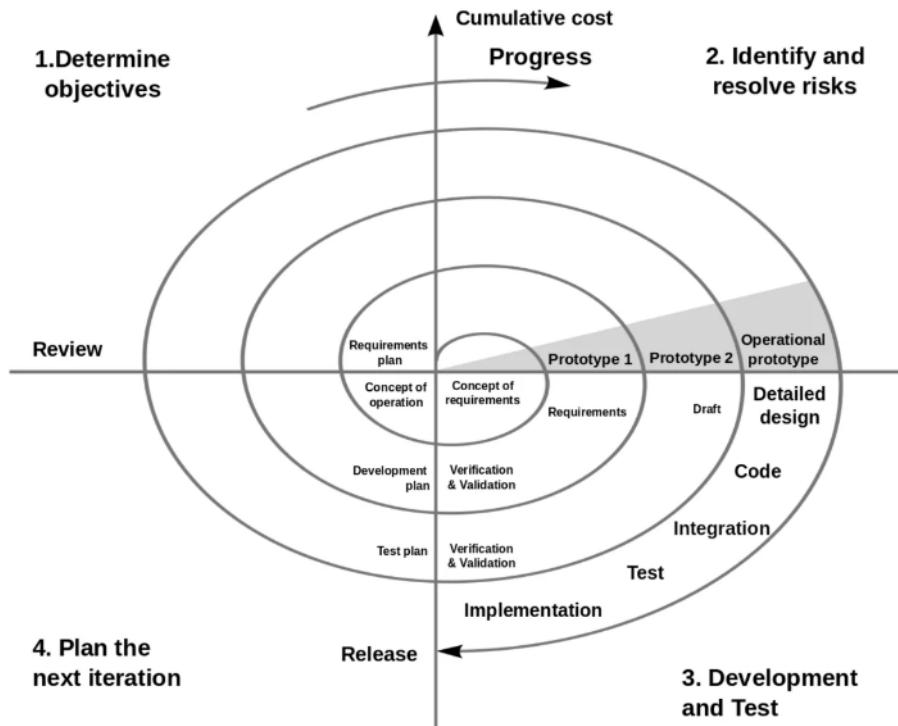
Το αυξητικό μοντέλο είναι ιδανικό όταν υπάρχουν κάποιες πλήρης και καλά καθορισμένες απαιτήσεις με χαλαρά συζευγμένα μέρη. Έτσι, οι αρχικές επαυξήσεις μπορούν να προγραμματιστούν και να παραλληλοποιηθούν επαρκώς.

Σπειροειδές Μοντέλο

Το σπειροειδές μοντέλο είναι ένα εξελικτικό μοντέλο με κύριο γνώμονα το ρίσκο. Όπως και στο αυξητικό μοντέλο το λογισμικό παραδίδεται σε επαναλήψεις, όμως σε αντίθεση με αυτό τα βήματα δεν είναι δραστηριότητες αλλά φάσεις, με σκοπό την αντιμετώπιση του προβλήματος με το μεγαλύτερο ρίσκο να προκαλέσει αποτυχία.

Όπως φαίνεται στο Σχήμα 2.10, οι φάσεις σε κάθε επανάληψη είναι:

1. Εξακρίβωση του προβλήματος με το μεγαλύτερο ρίσκο, καθορισμός των στόχων και των εναλλακτικών λύσεων.
2. Αξιολόγηση και σχεδιασμός των πιθανών λύσεων και προσδιορισμός των ρίσκων.
3. Ανάπτυξη μίας λύσης, επιβεβαίωση καταλληλότητας και δοκιμή της
4. Προετοιμασία για την επόμενη επανάληψη με βάση την ανατροφοδότηση από



Σχήμα 2.10: Σπειροειδές Μοντέλο

την παραδοθείσα επαύξηση.

Το σπειροειδής μοντέλο μπορεί να διαχειριστεί την αβεβαιότητα εξαιρετικά καλά. Έτσι, είναι ιδανικό για έργα λογισμικού με ασαφή απαιτήσεις και για έργα που βρίσκονται στην έρευνα και ανάπτυξη.

Ευέλικτο Μοντέλο

Στη σύγχρονη πραγματικότητα, η εργασία στην παραγωγή λογισμικού έχει ταχύς ρυθμούς και είναι υποκείμενο σε συνεχή ροή αλλαγών (χαρακτηριστικών, λειτουργιών και πληροφοριακού περιεχόμενου). Έτσι, το ευέλικτο μοντέλο αποτελεί μία λογική εναλλακτική στις παραδοσιακές μεθόδους της τεχνολογίας λογισμικού.

Το μοντέλο ενθαρρύνει συνεχόμενες επαναλήψεις από ανάπτυξη και δοκιμή των λειτουργιών. Σε κάθε επανάληψη παράγεται μία επαύξηση του λογισμικού αποτελούμενη από ένα μικρό σύνολο από λειτουργίες πλήρως ολοκληρωμένες. Επίσης, η κάθε επανάληψη είναι σχεδιασμένη ώστε να είναι μικρή, διαχειρίσιμη και ολοκληρώσιμη σε λίγες εβδομάδες. Συμπεριλαμβάνει των πελάτη στην διαδικασία της ανάπτυξης και ελαχιστοποιεί τα έγγραφα χορηγιμοποιώντας ανεπίσημη επικοινωνία.

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Το ευέλικτο μοντέλο αν και αποτελεί ρεαλιστική προσέγγιση στην ανάπτυξη λογισμικού, βρίσκεται σε μειονεκτική θέση όταν το λογισμικό είναι σύνθετο. Επίσης, αδυνατεί να διαχειριστεί τις μεταβιβάσεις λόγω των λίγων εγγράφων, αλλά από την άλλη μπορεί να διαχειριστεί τις μεταβαλλόμενες απαιτήσεις.

Οι πιο συνηθισμένες ευέλικτες μεθοδολογίες είναι:

- **Scrum:** Αποτελείται από επαναλήψεις που λέγονται sprints. Κάθε sprint διαρκεί 2 με 4 εβδομάδες και πριν την έναρξη του γίνεται προγραμματισμός. Αφού οριστούν οι δραστηριότητες και οι εργασίες για το sprint δεν μεταβάλλονται κατά την διάρκειά του.
- **Extreme Programming (XP):** Σε αυτό το μοντέλο η επανάληψη διαρκεί 1 με 2 εβδομάδες. Χαρακτηριστικά αυτού του μοντέλου είναι ο προγραμματισμός σε ζεύγη, η ανάπτυξη οδηγούμενη από έλεγχο (test-driven development), ο αυτοματοποιημένος έλεγχος, η απλή σχεδίασμη λογισμικού και τέλος οι μικρές εκδόσεις με συνεχόμενη ενσωμάτωση στο σύστημα.
- **Kanban:** Το μοντέλο Kanban επικεντρώνεται στην οπτικοποίηση, και αν υπάρχουν επαναλήψεις είναι πολύ μικρές. Στο πίνακα Kanban παριστάνονται όλες οι δραστηριότητες και εργασίες του έργου, συνοδευόμενες από τα υπεύθυνα άτομα για την διεκπεραίωσή τους και την πρόοδό τους.

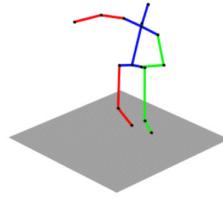
2.4 ΕΚΤΙΜΗΣΗ ΠΟΖΑΣ

Η εκτίμηση πόζας είναι μία διεργασία της υπολογιστικής όρασης όπου εκτιμάται η πόζα ενός αντικειμένου ή ανθρώπου σε μία εικόνα ή βίντεο. Το πρόβλημα της εκτίμησης πόζας περιλαμβάνει, επίσης, τον καθορισμό της θέσης και του προσανατολισμού της κάμερας σε σχέση με το αντικείμενο ή τον άνθρωπο.

Συνήθως αυτό γίνεται με την αναγνώριση, εκτίμηση θέσης και παρακολούθησης ενός αριθμού σημείων κλειδιών του αντικειμένου ή του ανθρώπου. Για τα αντικείμενα, αυτά μπορεί να είναι γωνίες ή άλλα σημαντικά χαρακτηριστικά ενώ για τον άνθρωπο τα σημεία κλειδιά αναπαριστούν κύριες αρθρώσεις όπως οι αγκώνες ή τα γόνατα.

Χαρακτηριστική είναι η διάχριση μεταξύ της 2D και της 3D εκτίμησης πόζας. Στην δισδιάστατη εκτίμηση πόζας εκτιμούνται οι θέσεις των σημείων στο 2D χώρο σε σχέση με το πλαίσιο της εικόνας ή του βίντεο. Αντιθέτως, η 3D εκτίμηση πόζας προβλέπει τις συντεταγμένες των σημείων κλειδιών σε ένα τρισδιάστατο σύστημα συντεταγμένων όπως φαίνεται στο Σχήμα 2.11.

Επιπλέον, η εκτίμηση πόζας διαφοροποιείται ως προς την αναγνώριση ενός ή περισσότερων αντικειμένων. Οι δύο προσεγγίσεις αποκαλούνται μεμονωμένη εκτίμηση πόζας, όπου αναγνωρίζεται και παρακολουθείται μόνο ένα αντικείμενο ή άνθρωπος, και πολλαπλή εκτίμηση πόζας, όπου αναγνωρίζονται και παρακολουθούνται

(α') 2D σημεία με συντεταγμένες (x_i, y_i) (β') 3D σημεία με συντεταγμένες (x_i, y_i, z_i)

Σχήμα 2.11: Παράδειγμα εκτίμησης πόζας

πολλαπλά.

2.4.1 Μεθοδολογίες εκτίμησης πόζας

Το πρόβλημα της εκτίμησης πόζας μπορεί να λυθεί με ποικίλους τρόπους αναλόγως με την διαρρύθμιση των αισθητήρων εικόνας και την επιλογή της μεθοδολογίας. Διακρίνονται τρεις κλάσεις μεθοδολογιών [10]:

- **Αναλυτικές ή γεωμετρικές μέθοδοι:** Απαιτούν την ρύθμιση της κάμερας ώστε η αντιστοίχηση των τρισδιάστατων σημείων της σκηνής και των δισδιάστατων σημείων της εικόνας να είναι γνωστή. Τοιουτοτρόπως, γνωρίζοντας την γεωμετρία του αντικειμένου η προβολή του στην εικόνα της κάμερας είναι μια γνωστή συνάρτηση της πόζας του αντικειμένου. Έτσι, το πρόβλημα της εκτίμησης πόζας λύνεται αναγνωρίζοντας τα σημεία κλειδιά και στην συνέχεια λύνοντας το σύνολο των εξισώσεων που αντιστοιχίζουν τις τρισδιάστατες συντεταγμένες των σημείων με τις δισδιάστατες συντεταγμένες της εικόνας.
- **Γενετικοί αλγόριθμοι:** Στην περίπτωση που η πόζα του αντικειμένου ή του ανθρώπου δεν απαιτεί τον υπολογισμό της σε πραγματικό χρόνο μπορεί να χρησιμοποιηθεί ένας γενετικός αλγόριθμος. Η πόζα περιγράφεται από μεταβλητές, όπως για παράδειγμα οι περιστροφές των αρθρώσεων από μία στάση αναφοράς, οι οποίες χρησιμοποιούνται ως παράμετροι εισόδου του γενετικού, ορίζοντας έτσι τη συνάρτηση καταλληλότητας (fitness function) ως το σφάλμα της προβολής των εκτιμώμενων τρισδιάστατων σημείων κλειδιών από τις πραγματικές συντεταγμένες τους στο δισδιάστατο πλαίσιο της εικόνας.
- **Μηχανική μάθηση:** Οι μέθοδοι μηχανικής μάθησης χρησιμοποιούν ένα σύστημα τεχνητής νοημοσύνης το οποίο μαθαίνει την αντιστοίχηση των 2D χαρακτηριστικών της εικόνας με τις παραμέτρους μοντελοποίησης που περιγράφουν με σαφήνεια την πόζα. Εν συντομίᾳ, ένα επαρκώς μεγάλο σύνολο από φωτογραφίες ή βίντεο αντικειμένων ή ανθρώπων σε διαφορετικές πόζες χρησιμοποιείται ως είσοδος στο σύστημα κατά την διάρκεια της φάσης εκμάθησης. Με το πέρας της εκμάθησης, το σύστημα μπορεί να εκτιμήσει την πόζα του αντικειμένου ή ανθρώπου δεδομένης της 2D εικόνας του.

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Στη συγκεκριμένη εφαρμογή, απαιτείται η εκτίμηση της ανθρώπινης πόζας στον τρισδιάστατο χώρο σε πραγματικό χρόνο. Το πρόβλημα αυτό κρίνεται άκρως περίπλοκο, με την δυσκολία του να έγκειται στο γεγονός ότι η εκτίμηση της τρισδιάστατης πόζας από μία δισδιάστατη εικόνα είναι εξ ορισμού κακώς ορισμένο πρόβλημα. Αυτό γίνεται εμφανές αν σκεφτούμε ότι μία δισδιάστατη πόζα μπορεί να προκύψει από πολυάριθμες τρισδιάστατες πόζες. Ταυτόχρονα, υπάρχουν πολλές επιπλέον προκλήσεις όπως η αβεβαιότητα του περιβάλλοντος, του φόντου, του φωτισμού, η κίνηση της κάμερας, οι γρήγορες κινήσεις, οι μεταβολές των ρούχων και των σκιών που μπορεί να κάνουν την μορφή και το σχήμα του ανθρώπου να αλλάξει δραματικά σε συνάρτηση με τον χρόνο. Επιπλέον, οι ανθρώπινες αρθρώσεις είναι μικρές, οριακά εμφανής και έχουν πολλούς βαθμούς ελευθερίας δυσχεραίνοντας ακόμα περισσότερο το πρόβλημα της εκτίμησης πόζας. Ως εκ τούτου, η καταλληλότερη μεθοδολογία για την επίλυση του προβλήματος είναι η *Bαθιά Μηχανική Μάθηση*, ικανοποιώντας τόσο την απαίτηση της σχετικά ακριβής εκτίμησης των σημείων κλειδιών αλλά και διεξάγοντας την εκτίμηση σε πραγματικό χρόνο. Στη συνέχεια του κεφαλαίου λοιπόν, θα αναλύσουμε το θεωρητικό υπόβαθρο που συναντάει κανείς στις διάφορες προσεγγίσεις επίλυσης του προβλήματος εκτίμησης ανθρώπινης πόζας.

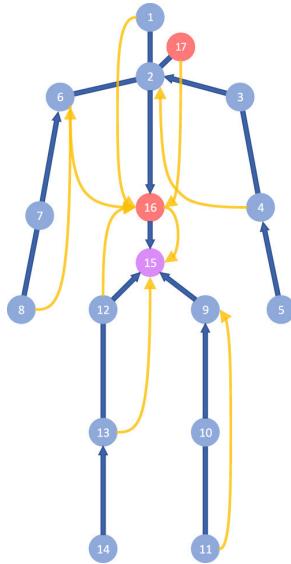
2.4.2 Μοντελοποίηση ανθρώπινου σώματος

Όπως αναφέρθηκε, το ανθρώπινο σώμα είναι περίπλοκο με πολλαπλές αρθρώσεις πολλών βαθμών ελευθερίας και μεγάλου εύρους κίνησης. Τοιουτοτρόπως, κρίνεται αναγκαία η σαφής μοντελοποίηση της υποβόσκουσας δομής του, μειώνοντας αφενός την εγγενή αβεβαιότητα του προβλήματος εκτίμησης πόζας και αφετέρου αποτελώντας το μέσο για την περιγραφή της τελικής πόζας.

Οι διαφορετικές μέθοδοι εκτίμησης πόζας χρησιμοποιούν διαφορετικές μοντελοποιήσεις του ανθρώπινου σώματος ανάλογα με τις ανάγκες της εκάστοτε μεθόδου. Εντούτοις, οι κατά κόρον χρησιμοποιούμενες μοντελοποιήσεις είναι τα σκελετικά μοντέλα (*skeleton models*) και τα μοντέλα σχήματος (*shape models*). Ταυτόχρονα, αξίζει να σημειωθεί μία καινούργια μορφή αναπαράστασης του ανθρώπινου σώματος που βασίζεται στην αναπαράσταση των σημείων της επιφάνειας του σώματος. [11]

Μοντέλο σκελετού

Το Μοντέλο σκελετού είναι μια δενδρική δομή αποτελούμενη από σημεία κλειδιά του ανθρώπινου σώματος, συνδέοντας τις φυσικές παρακείμενες αρθρώσεις, τα σημεία κλειδιά, με ακμές (για παράδειγμα η ακμή του βραχίονα συνδέει τις αρθρώσεις-σημεία κλειδιά του καρπού και του αγκώνα). Ανάλογα με την μοντελοποίηση, ορίζονται οι γονείς πρώτης τάξης του κάθε σημείου κλειδιού ή ενδεχομένως και οι γονείς δεύτερης τάξης, όπως φαίνεται στο Σχήμα 2.12, ανάλογα με τις ανάγκες του χρησιμοποιούμενου μοντέλου μηχανικής μάθησης.



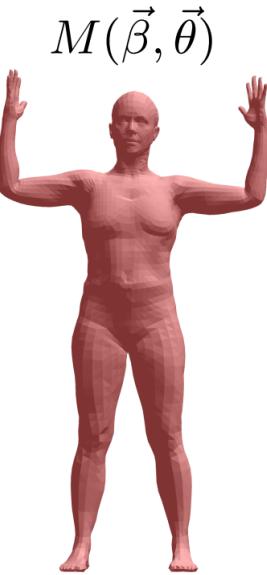
Σχήμα 2.12: Μοντέλο Σκελετού ανθρώπινου σώματος. Το σημείο κλειδί αναφοράς είναι το 15. Οι μπλε ακμές δείχνουν τους γονείς πρώτης τάξης του κάθε σημείου κλειδιού, ενώ οι κίτρινες ακμές τους γονείς δεύτερης τάξης (παραλείπονται κάποιοι για ευχρίνεια της εικόνας).

Μοντέλο σχήματος

Η μοντελοποίηση που χρησιμοποιείται στην πλειοψηφία των άρθρων εκτίμησης πόζας των τελευταίων χρόνων είναι το *skinned multi-person linear model* (SMPL) [12]. Το μοντέλο σχήματος αναπαριστά το ανθρώπινο σώμα ως ένα τριγωνικό πλέγμα καθορίζοντας έτσι πλήρως το σχήμα του σε οποιαδήποτε πόζα, όπως φαίνεται στο Σχήμα 2.13. Το μοντέλο SMPL περιγράφει το ανθρώπινο σώμα με 6890 κορυφές που σχηματίζουν το τριγωνικό πλέγμα. Οι θέσεις των κορυφών τριγώνων υπολογίζονται από τις παραμέτρους σχήματος β , που καθορίζουν τις αναλογίες του σώματος όπως το ύψος και τα κιλά, και τις παραμέτρους πόζας θ που περιγράφουν τις σχετικές 3D γωνίες περιστροφής των αρθρώσεων από μία πόζα αναφοράς.

Επιφανειακό μοντέλο

Μία ακόμα πιο πρόσφατη μοντελοποίηση του ανθρώπινου σώματος για το πρόβλημα της εκτίμησης πόζας, αποτελεί το επιφανειακό μοντέλο. Αναπτύχθηκε με την εικασία ότι μία αραιή συσχέτιση της εικόνας με τα σημεία κλειδιά του ανθρώπινου σώματος δεν είναι αρκετή για να αντικατοπτρίσει τη πλήρη κατάσταση του. Τοιουτοτρόπως, στο άρθρο [11], προτείνεται η επιφανειακή μοντελοποίηση, αποκαλούμενη DensePose, όπου εγκαθιδρύεται μία πυκνή συσχέτιση των πίξελ της εικόνας με μία επιφανειακή αντιπροσώπευση του ανθρώπινου σώματος, όπως φαίνεται στο Σχήμα 2.14



Σχήμα 2.13: *Μοντελοποίηση SMPL*. Το ανθρώπινο σώμα περιγράφεται από τις παραμέτρους σχήματος $\vec{\beta}$ και πόζας $\vec{\theta}$.

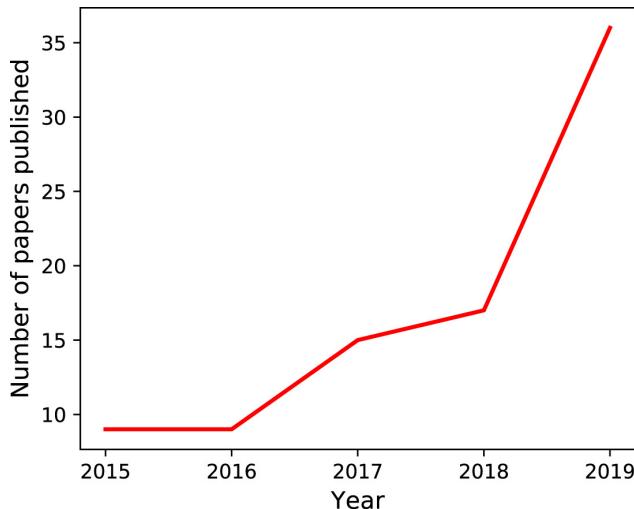


Σχήμα 2.14: *Μοντελοποίηση DensePose*. Στα αριστερά φαίνεται η επιφανειακή τρισδιάστατη αναπαράσταση του ανθρώπινου σώματος. Αντιστοιχίζοντας τα σημεία της εικόνας δεξιά στην αριστερή επιφάνεια γίνεται δυνατή η εκτίμηση της πόζας.

2.4.3 Κατηγοριοποιήσεις λύσεων του προβλήματος εκτίμησης πόζας

Για την επίλυση του προβλήματος εκτίμησης πόζας τα τελευταία χρόνια παρουσιάζονται ολοένα και περισσότερα άρθρα όπως φαίνεται στο Σχήμα 2.15. Προφανώς, η κάθε προσέγγιση επίλυσης του προβλήματος έχει μοναδικά χαρακτηριστικά, ικανοποιώντας διαφορετικές απαιτήσεις, αλλά υπάρχουν κοινές κατευθυντήριες γραμμές πάνω στις οποίες κινούνται οι ερευνητές. Με αυτόν τον τρόπο, γίνεται δυνατή η κατηγοριοποίηση των λύσεων του προβλήματος ως προς τα διάφορα χαρακτηριστικά τους.

Η γενικότερη κατηγοριοποίηση αφορά την μορφή της εισόδου του μοντέλου μηχανικής μάθησης. Με άλλα λόγια, το μοντέλο μπορεί να χρησιμοποιεί για είσοδο μόνο μία εικόνα ή πολλαπλές εικόνες από διαφορετικές οπτικές (της ίδιας χρονικής στιγμής). Αντίθετα, για την αξιοποίηση της χρονικής πληροφορίας, μπορεί να



Σχήμα 2.15: Αριθμός άρθρων εκτίμησης ανθρώπινης πόζας (κατακόρυφος άξονας) ανά χρόνο (οριζόντιος άξονας)

χρησιμοποιηθεί ως είσοδος αλληλουχίες εικόνων, είτε μεμονωμένων είτε από διαφορετικές οπτικές. Προφανώς, οι πιο περίπλοκες είσοδοι απαιτούν περισσότερους πόρους τόσο για την ανάκτηση των εικόνων όσο και για την επεξεργασία τους, μειώνοντας ωστόσο τις ασάφειες του προβλήματος.

Επιπλέον, οι μεθοδολογίες της εκτίμησης πόζας μπορούν να διαχωριστούν ανάλογα με την ικανότητα τους στην πρόβλεψη πόζας ενός ή περισσότερων ανθρώπων. Στην πρώτη περίπτωση, ορίζεται η πρόβλεψη της πόζας μόνο για έναν άνθρωπο μέσα στην εικόνα και είναι σαφώς η απλούστερη. Στην περίπτωση εκτίμησης πόζας περισσότερων ανθρώπων το πρόβλημα δυσχεραίνεται λόγω του πολύ μεγαλύτερου χώρου καταστάσεων, της απόκρυψης ή και σύγχυσης των σημείων κλειδιών και της αναγκαιότητας διαφοροποίησης τους.

Τοιουτορόπως, συγκεκριμένα για το πρόβλημα εκτίμησης πόζας πολλαπλών ανθρώπων, διακρίνουμε την κατηγοριοποίηση με βάση τον τρόπο ομαδοποίησης των σημείων κλειδιών στους ανθρώπους στους οποίους ανήκουν. Οι δύο κύριες προσεγγίσεις είναι η από κάτω προς τα πάνω και η από πάνω προς τα κάτω.

- **Από κάτω προς τα πάνω:** Το μοντέλο ανιχνεύει κάθε εμφάνιση ενός συγκεκριμένου σημείου κλειδιού (όπως όλα τα αριστερά χέρια) σε μία εικόνα και στην συνέχεια προσπαθεί να ομαδοποιήσει τα σημεία κλειδιά με βάση τους ανθρώπους στους οποίους ανήκουν.
- **Από πάνω προς τα κάτω:** Αντίθετα, σε αυτή την μέθοδο, το μοντέλο αρχικά χρησιμοποιεί έναν ανιχνευτή αντικειμένων, καθορίζοντας έτσι ένα πλαίσιο οριοθέτησης γύρω από κάθε άνθρωπο, και έπειτα εκτιμά την θέση των σημείων κλειδιών του κάθε ανθρώπου εσωτερικά σε κάθε περικομμένη περιοχή.

Μία άλλη πιθανή κατηγοριοποίηση σχετίζεται με τα χρησιμοποιούμενα στάδια προβλέψεων έως ότου προκύψει η τελική εκτίμηση πόζας. Αναλυτικότερα, θα ανα-

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

λύσουμε τις μεθόδους από τρεις κατηγορίες: εκτίμηση τρισδιάστατης πόζας απευθείας από εικόνες, ανύψωση από δισδιάστατες σε τρισδιάστατες προβλέψεις και μεθόδους βασισμένες στο μοντέλο SMPL.

Εκτίμηση πόζας απευθείας από εικόνες

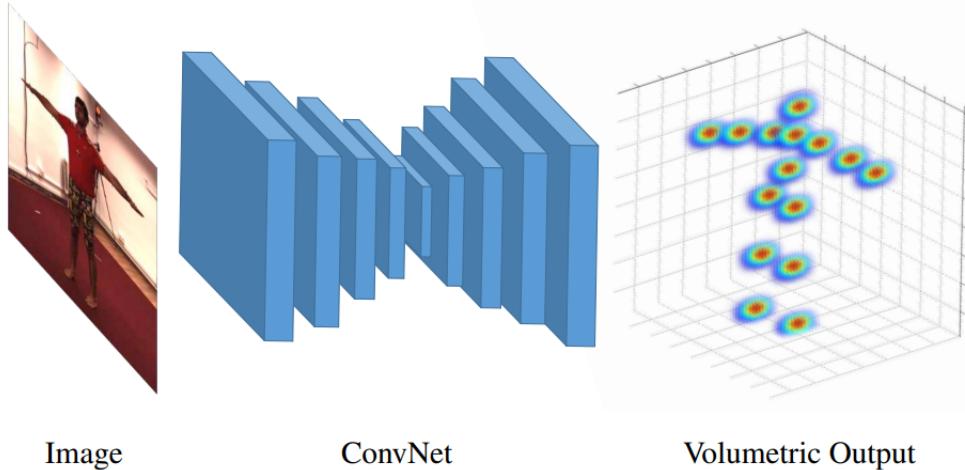
Η πιο άμεση μεθοδολογία για την εκτίμηση της τρισδιάστατης πόζας είναι ο σχεδιασμός συνεχόμενων δικτύων για την πρόβλεψη των 3D συντεταγμένων των σημείων κλειδιών ή αρθρώσεων. Οι μέθοδοι της απευθείας εκτίμησης από εικόνες μπορούν να διαχωριστούν περαιτέρω σε δύο κλάσεις: μέθοδοι βασισμένες στον εντοπισμό και μέθοδοι βασισμένες στην παλινδρόμηση. Αξίζει να σημειωθεί ότι έχουν γίνει προσπάθειες για ενοποίηση των δύο αυτών προσεγγίσεων.

Οι μέθοδοι βασισμένες στον εντοπισμό προβλέπουν έναν πιθανοτικό θερμικό χάρτη για κάθε σημείο κλειδί και παίρνοντας την μέγιστη πιθανότητα του χάρτη καθορίζουν την θέση του σημείου κλειδιού. Για παράδειγμα στο [13] αναπαριστάται ο τρισδιάστατος χώρος σε έναν όγκο και εκπαιδεύεται ένα μοντέλο Συνελικτικού Νευρωνικού Δικτύου (*Convolutional Neural Network, CNN* στο καθεξής) ώστε να προβλέπονται οι ογκομετρικοί πιθανοτικοί θερμικοί χάρτες για κάθε σημείο κλειδί, όπως φαίνεται στο Σχήμα 2.16. Παρόμοιες μέθοδοι βασίζονται σε επιπλέον βήματα για την μετατροπή των θερμικών χαρτών στην τελική εκτίμηση των θέσεων των σημείων κλειδιών, συνήθως παίρνοντας την μέγιστη πιθανότητα του χάρτη, δηλαδή εφαρμόζοντας την συνάρτηση argmax (βλ. στο παράρτημα τον ορισμό Α'.1). Βέβαια, η διαδικασία αυτή δεν είναι διαφορίσιμη, αποτελώντας τροχοπέδη στον μηχανισμό εκμάθησης των νευρωνικών δικτύων. Ταυτόχρονα, η ακρίβεια των προβλεπόμενων σημείων κλειδιών είναι ανάλογη της διακριτότητας των θερμικών χαρτών ο οποίοι έχουν εγγενή αδυναμία στη χωρική γενίκευση. Για την αύξηση της ακρίβειας των προβλέψεων, οι παραγόμενοι θερμικοί χάρτες απαιτούν αύξηση της χωρικής διακριτότητας, αυξάνοντας όμως τετραγωνικά τις απαιτήσεις σε υπολογιστική ισχύ και κατανάλωση μνήμης.

Από την άλλη μεριά, το πρόβλημα εκτίμησης πόζας μπορεί να θεωρηθεί διαισθητικά ως ένα πρόβλημα παλινδρόμησης, όπου εκτιμούνται οι συντεταγμένες θέσεων των σημείων κλειδιών ως προς την θέση ενός σημείου κλειδιού αναφοράς³. Σε αυτή την προσέγγιση, γίνεται εμφανής η καταλληλότητα επιλογής των μοντέλων σκελετού ή παρεμφερή μοντελοποιήσεις για την αναπαράσταση του ανθρώπινου σώματος. Για παράδειγμα, για να ενσωματώσουν προηγούμενη γνώση για την δομή του ανθρώπινου σώματος, στο [14], εισάγουν ένα κινηματικό μοντέλο σκελετού, αποτελούμενο από αρθρώσεις και κόκαλα, τα οποία κόκαλα έχουν σταθερό μήκος και μπορούν να περιστραφούν γύρω από τις αρθρώσεις. Εντούτοις, το σταθερό μήκος των κοκάλων δεν ανταποκρίνεται επαρκώς στη μεταβλητότητα του ανθρώπινου σκελετού, γεγονός που υποβαθμίζει την ικανότητα γενικοποίησης του μοντέλου. Από την άλλη μεριά, στο [15] υποθέτουν ότι στην εκτίμηση πόζας είναι πιο λογική

³Στο μοντέλο σκελετού SMPL, για παράδειγμα, το σημείο αναφοράς, σημείο με δείκτη (*index*) 0, είναι η λεκάνη

η εκτίμηση της θέσεως των κοκάλων αντί για τις αρθρώσεις διότι η αναπαράσταση των κοκάλων καθιστά ευκολότερη την εκμάθηση του μοντέλου βαθιάς μάθησης και αντικατοπτρίζουν καλύτερα τους γεωμετρικούς περιορισμούς του ανθρώπινου σκελετού. Φυσικά, αυτή η προσέγγιση απαιτεί την μετατροπή των δεδομένων εκπαίδευσης στην σχετική μορφή αναπαράστασης τις θέσεις των κοκάλων.



Σχήμα 2.16: Μέθοδος εκτίμησης πόζας βασισμένη στον εντοπισμό. Η εικόνα ως είσοδος περνάει από την αρχιτεκτονική του CNN και παράγει πιθανοτικούς χάρτες για κάθε σημείο κλειδί στον 3D χώρο.

Εν κατακλείδι, οι μέθοδοι εντοπισμού με θερμικούς χάρτες, αν και πιο αποδοτικοί από τις μεθόδους παλινδρόμησης, έχουν κάποια μειονεκτήματα. Η διαδικασία επιλογής της μέγιστης πιθανότητας δεν είναι διαφορίσιμη και δεν επιτρέπει την από άκρη σε άκρη εκμάθηση του μοντέλου, σε αντίθεση με τις μεθόδους παλινδρόμησης όπου αυτή η εκμάθηση είναι εφικτή.

Προς αντιμετώπιση των προαναφερθέντων προβλημάτων έχουν γίνει προσπάθειες συνδυασμού των δύο παραπάνω μεθόδων. Για παράδειγμα, στα άρθρα [16], [17], χρησιμοποιείται η συνάρτηση soft-argmax (αναλυτικότερα βλ. ορισμό [Α'.1](#)) για την μετατροπή των χαρτών χαρακτηριστικών (feature maps), ή των θερμικών χαρτών, σε συντεταγμένες των σημείων κλειδιών, έχοντας ως αποτέλεσμα ένα πλήρως διαφορίσιμο σύστημα. Με παρόμοια λογική με την συνάρτηση soft-argmax, στο άρθρο [18], εισάγουν ένα καινούργιο επίπεδο, αποκαλούμενο διαφορίσιμη χωρική σε αριθμητική μετατροπή (differentiable spatial to numerical transform, DSNT), για να διατηρήσουν την από άκρη σε άκρη δυνατότητα εκμάθησης του μοντέλου και ταυτόχρονα βελτιώνοντας την ικανότητα γενικοποίησης του.

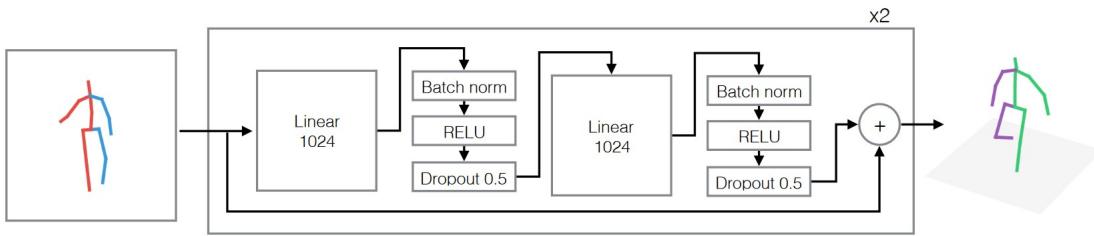
Ανύφωση προβλέψεων από 2D σε 3D

Εμπνευσμένοι από την ραγδαία εξέλιξη των αλγορίθμων για εκτίμηση της 2D πόζας, αρκετοί ερευνητές προσπάθησαν να χρησιμοποιήσουν τα αποτελέσματα της

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2D εκτίμησης πόζας για να βελτιώσουν την ικανότητα γενικοποίησης σε δεδομένα αποκτημένα υπό καθημερινές συνθήκες.

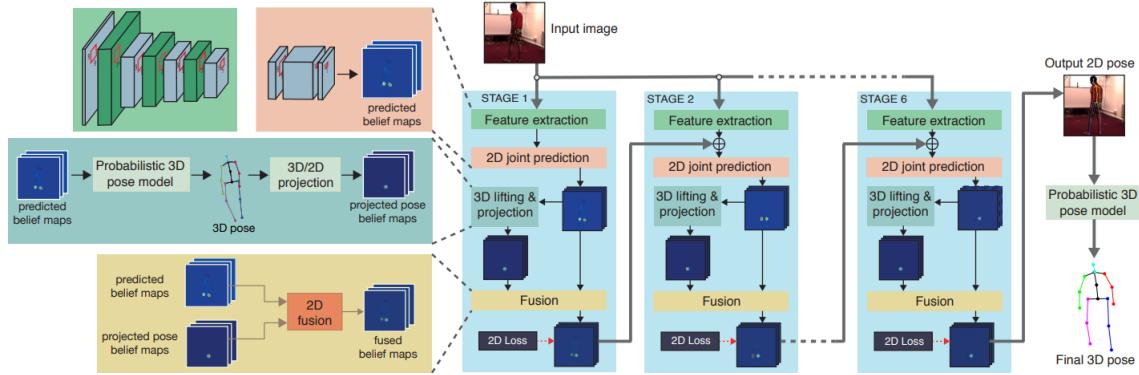
Χαρακτηριστικό παράδειγμα αποτελεί η έρευνα στο [19] όπου προτείνεται μία αρχιτεκτονική, όπως φαίνεται στο Σχήμα 2.17, επικεντρωμένη στην ανύψωση της 2D πόζας σε 3D με ένα απλό αλλά αποδοτικό νευρωνικό δίκτυο, εμφυσώντας ενδιαφέρον για περαιτέρω έρευνα στην ανύψωση της 2D πόζας στον 3D χώρο.



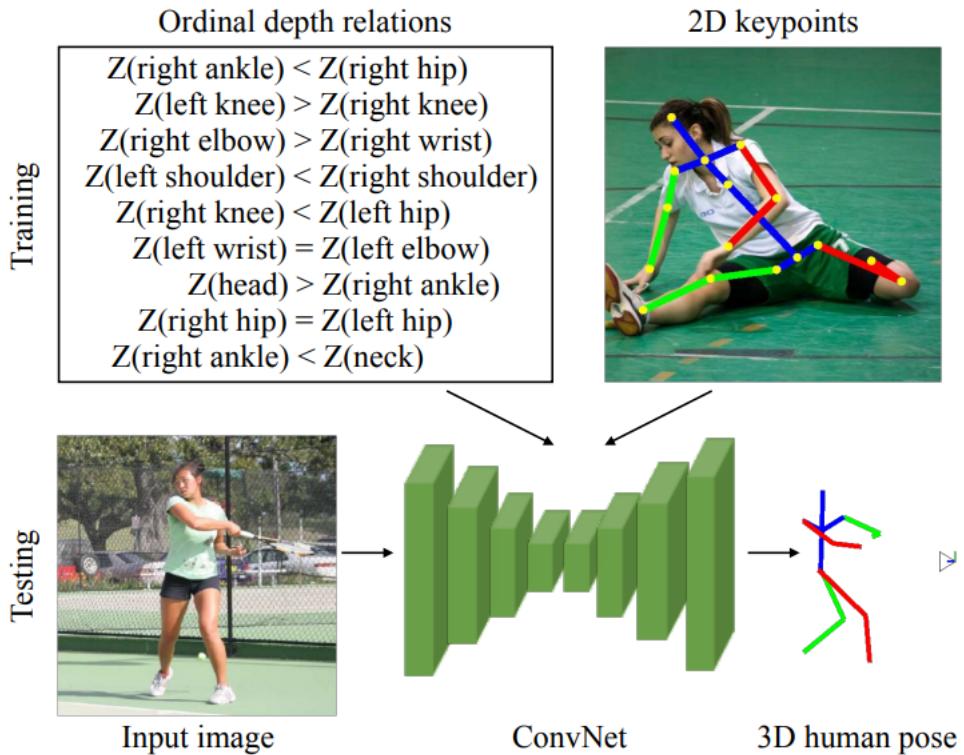
Σχήμα 2.17: Αρχιτεκτονική simple baseline για την εκτίμηση πόζας. Ακρογωνιαίος λίθος του νευρωνικού αποτελεί ένα γραμμικό επίπεδο, ακολουθούμενο από ένα επίπεδο κανονικοποίησης παρτίδας (batch normalization) (βλ. A'.2), ένα επίπεδο ενεργοποίησης με συνάρτηση ReLU ((βλ. A'.1)) και ένα επίπεδο εγκατάλειψης μονάδων εισόδου (dropout) (βλ. A'.2). Η είσοδος του συστήματος είναι ένας πίνακας με τις 2D θέσεις των σημείων κλειδιών και η έξοδος είναι οι θέσεις των σημείων κλειδιών σε 3D συντεταγμένες.

Για την αντιμετώπιση της δυσκολίας παραγωγής και σχολιασμού εξολοκλήρου 3D συνόλων δεδομένων σε διάφορες έρευνες χρησιμοποιούνται επιπλέον πληροφορίες ως ενδιάμεση επίβλεψη του συστήματος. Ένα είδος πληροφορίας που χρησιμοποιείται συχνά είναι η προβολή της προβλεπόμενης 3D πόζας στον 2D χώρο της εικόνας. Για παράδειγμα στο [20], αρχικά εκτιμάται η 2D πόζα η οποία χρησιμοποιείται για την εκτίμηση της 3D πόζας. Έπειτα, οι προβλέψεις της 3D πόζας προβάλλονται στον 2D χώρο, συνδυάζονται με τις αρχικές 2D προβλέψεις και τέλος συγκρίνονται με τους 2D σχολιασμούς του συνόλου δεδομένων (όπως φαίνεται στο Σχήμα 2.18), εκπαιδεύοντας έτσι το σύστημα. Αξίζει να σημειωθεί ότι στο στάδιο ανύψωσης των 2D προβλέψεων σε 3D ενσωματώνεται ένα επιπλέον επίπεδο στο CNN όπου αναπαριστώνται οι 3D γεωμετρικοί περιορισμοί του σώματος, ενός μοντέλου σκελετού (όπως αναπτύχθηκε στην ενότητα 2.4.2), εξασφαλίζοντας ότι οι προβλεπόμενες πόζες βρίσκονται στον χώρο των επιτρεπόμενων εκ φύσεως ποζών.

Αντίστοιχα για την αποφυγή σχολιασμού των 3D συνόλων δεδομένων, στο [21], προτείνεται μία εναλλακτική, πιο ασθενής μέθοδος επίβλεψης του συστήματος. Πιο συγκεκριμένα, για την επίβλεψη χρησιμοποιούνται οι ήδη υπάρχοντες σχολιασμοί των 2D σημείων κλειδιών σε συνδυασμό με τις σχέσεις βάθους (πιο κοντά - πιο μακριά) των σημείων κλειδιών, όπως φαίνεται στο Σχήμα 2.19. Μάλιστα, παρουσιάζεται η ευελιξία αυτής της μεθόδου επίβλεψης, ενσωματώνοντάς την σε διαφορετικές διαρρυθμίσεις νευρωνικών δικτύων όπου επιτυγχάνονται ανταγωνιστικά αποτελέσματα με την επίβλεψη με πραγματικούς 3D σχολιασμούς του συνόλου δεδομένων.



Σχήμα 2.18: *Lifting from the deep*: Παράδειγμα αρχιτεκτονικής ανύψωσης 2D σε 3D. Κάθε στάδιο παράγει ως έξοδο ένα σύνολο 2D πιθανοτικών χαρτών για κάθε σημείο κλειδί, οι οποίοι μαζί με την αρχική εικόνα αποτελούν την είσοδο του επόμενου σταδίου. Κάθε στάδιο μαθαίνει να συνδυάζει τους πιθανοτικούς χάρτες της 2D εκτίμησης πόζας με τους προβαλλόμενους χάρτες από την 3D εκτίμηση πόζας.



Σχήμα 2.19: *Ordinal Depth Supervision*: Παράδειγμα αρχιτεκτονικής ανύψωσης 2D σε 3D. Όταν εκλείπουν οι 3D σχολιασμοί του συνόλου δεδομένου, μπορούν να χρησιμοποιηθούν οι 2D σχολιασμοί των σημείων κλειδιών σε συνδυασμό με τις σχέσεις βάθους των σημείων κλειδιών.

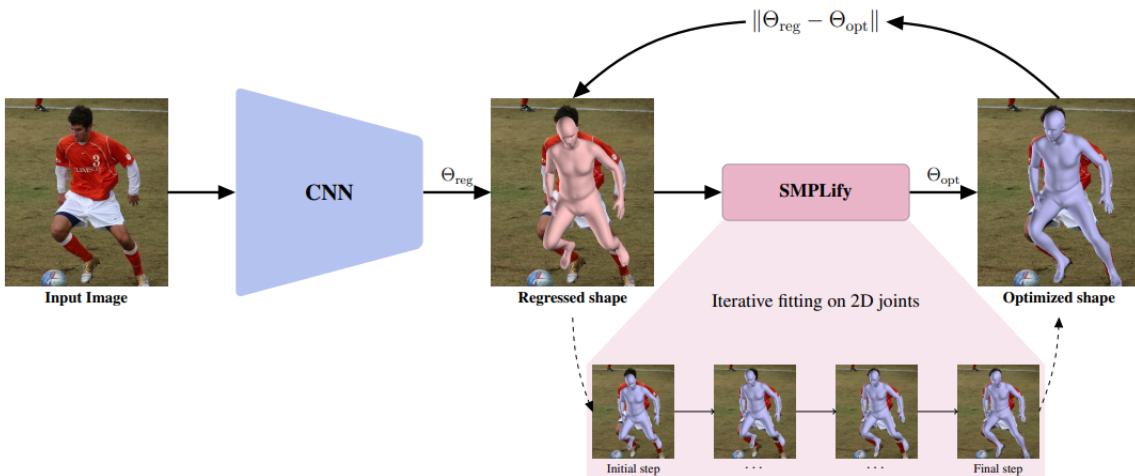
Μέθοδοι βασισμένες στο μοντέλο σχήματος SMPL

Οι μέθοδοι εκτίμησης πόζας με το μοντέλο SMPL (όπως περιγράφηκε στην ενότητα 2.4.2) βασίζονται στην εκτίμηση των παραμέτρων πόζας β και σχήματος θ .

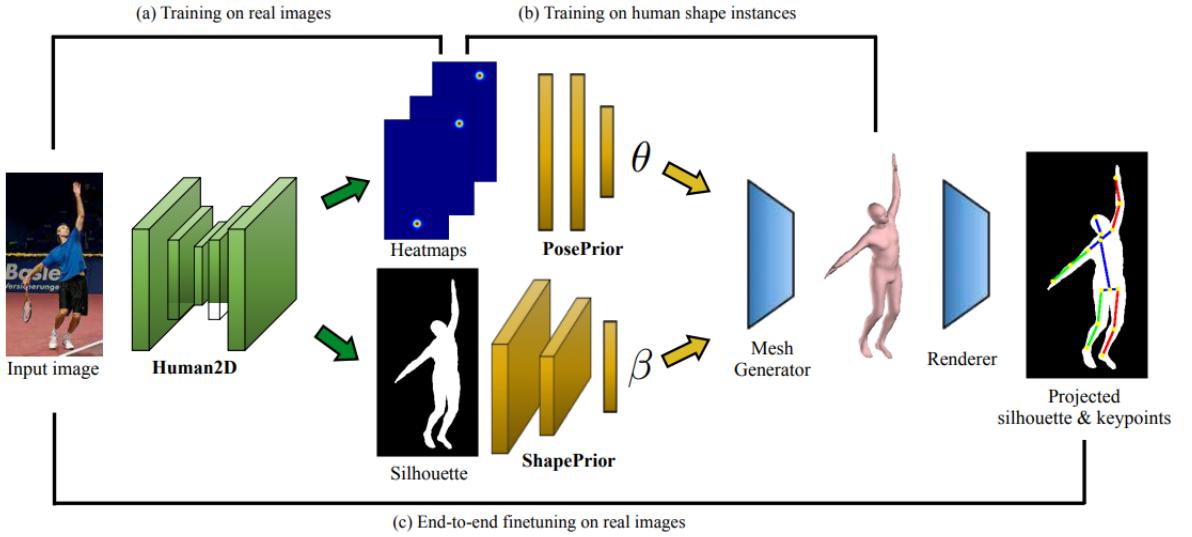
ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Δεδομένου ότι το μοντέλο SMPL εμπεριέχει a priori γνώση του ανθρώπινου σώματος, το πρόβλημα μπορεί να επιλυθεί με σχετικά μικρό αριθμό δεδομένων.

Η εκτίμηση της 3D πόζας με το μοντέλο SMPL μπορεί να επιλυθεί με δύο διαφορετικές προσεγγίσεις. Από την μία, οι μέθοδοι βασισμένες σε βελτιστοποίηση, όπου το μοντέλο SMPL εφαρμόζεται στα 2D δεδομένα σε έναν επαναληπτικό βρόγχο, έχουν ως αποτέλεσμα ακριβή αντιστοιχία της εικόνας με το μοντέλο αλλά συνήθως είναι αργές και εξαρτιούνται από την αρχικοποίηση του προβλήματος. Αντίθετα, οι παλινδρομικές μέθοδοι, όπου εκτιμούνται οι παράμετροι του SMPL απευθείας με νευρωνικά δίκτυα, τείνουν να παρέχουν ικανοποιητικά, όχι όμως ακριβή, αποτελέσματα ενώ απαιτούν μεγαλύτερο όγκο δεδομένων για επίβλεψη. Τοιουτοτρόπως, αρκετές μέθοδοι έχουν προσπαθήσει να συνδυάσουν τις δύο μεθόδους. Για παράδειγμα στο SMPLify, [22], αρχικά γίνεται πρόβλεψη των 2D σημείων κλειδιών με ένα νευρωνικό δίκτυο και έπειτα το μοντέλο SMPL εφαρμόζεται επαναληπτικά σε αυτά, ελαχιστοποιώντας την διαφορά των προβαλλόμενων σημείων κλειδιών του SMPL με τα προβλεπόμενα. Αντίστοιχα, στο SPIN (SMPL oPtimization IN the loop), [23], ένα νευρωνικό δίκτυο προβλέπει τις παραμέτρους του SMPL, οι οποίες αρχικοποιούν το βρόγχο βελτιστοποίησης, όπως φαίνεται στο Σχήμα 2.20, με αποτέλεσμα η εφαρμογή του μοντέλου στα 2D δεδομένα να είναι πιο γρήγορη και ακριβής. Ταυτόχρονα, η τέλεια εφαρμογή του μοντέλου στα πίξελ της εικόνας από την διαδικασία της βελτιστοποίησης, χρησιμοποιείται μετέπειτα ως ισχυρή επίβλεψη για την εκμάθηση του νευρωνικού δικτύου. Κατά τη διαδικασία αυτή, καλύτερες εκτιμήσεις του νευρωνικού αθούν τη βελτιστοποίηση σε πιο ακριβής λύσεις και εξ ακολούθως πιο ακριβής εφαρμογές του SMPL στην εικόνα παρέχουν στο νευρωνικό καλύτερο σήμα επίβλεψης, καθιστώντας την διαδικασία να αυτο-βελτιώνεται φυσικά.



Σχήμα 2.20: Αρχιτεκτονική του μοντέλου SPIN: Ένα συνελικτικό νευρωνικό δίκτυο εκτιμάει τις παραμέτρους του μοντέλου SMPL, οι οποίες αρχικοποιούν το πρόβλημα βελτιστοποίησης.



Σχήμα 2.21: Αρχιτεκτονική του μοντέλου στο άρθρο *Learning to Estimate 3D Human Pose and Shape from a Single Color Image*

Οι πρόσφατες μέθοδοι εκτίμησης πόζας βασισμένες στο μοντέλο SMPL εκτιμούνε τις παραμέτρους του με χρήση δικτύων που διεκπεραιώνουν διαφορετικές λειτουργίες. Για παράδειγμα, στο [24] προτείνεται η χρήση του *Human Mesh Recovery* (εν συντομίᾳ HMR), ενός από άκρη σε άκρη δικτύου όπου μαθαίνεται η αντιστοίχηση από τα πίξελ της εικόνας στις παραμέτρους του μοντέλου SMPL, χρησιμοποιώντας αρχικά έναν κωδικοποιητή για την εξαγωγή των χαρακτηριστικών της εικόνας, έναν παλινδρομητή για την εκτίμηση των παραμέτρων SMPL και τέλος έναν διακριτή για την επιβεβαίωση της ρεαλιστικότητας της εκτιμώμενης πόζας. Αντίστοιχα, στο [25] αρχικά γίνεται η εκτίμηση των 2D σημείων κλειδιών με χρήση θερμικών χαρτών και της σιλουέτας του ανθρώπου, όπως φαίνεται στο Σχήμα 2.21. Στη συνέχεια, οι θερμικοί χάρτες χρησιμοποιούνται από ένα δίκτυο για την εκτίμηση των παραμέτρων πόζας β του μοντέλου SMPL ενώ ένα άλλο δίκτυο χρησιμοποιεί τη σιλουέτα για την εκτίμηση των παραμέτρων σχήματος θ . Τέλος, η εκπαίδευση από άκρη σε άκρη του δικτύου επιτυγχάνεται με την σύγκριση της προβολής των 3D σημείων κλειδιών με τα πραγματικά σχολιασμένα 2D σημεία κλειδιά και συγκρίνοντας την προβολή της σιλουέτας με σχολιασμένες 2D μάσκες. Ένα από τα πλεονεκτήματα αυτών των μεθόδων αποτελεί η εκπαίδευσή τους με 2D σχολιασμένα δεδομένα καθώς τα σύνολα δεδομένων με σχολιασμούς στον τρισδιάστατο χώρο είναι περιορισμένα.

3

Εργαλεία

3.1 Kivy



Σχήμα 3.1: Kivy

Το Kivy⁴ είναι μια βιβλιοθήκη ανοιχτού κώδικα της Python για την γρήγορη ανάπτυξη εφαρμογών που χρησιμοποιούν γραφικά για υλοποίηση καινοτόμων διεπαφών χρηστών. Επιλέξαμε την εν λόγω βιβλιοθήκη για την υλοποίηση της διεπαφής του καθρέφτη επειδή είναι:

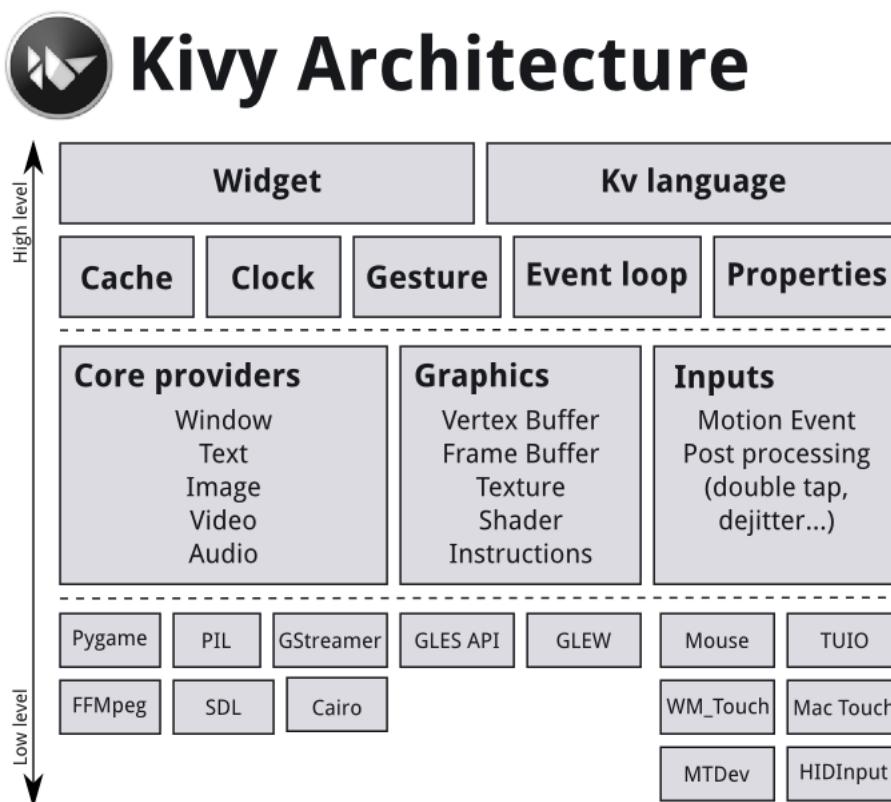
- **Cross Platform:** Το Kivy τρέχει σε διαφορετικά λειτουργικά συστήματα και πλατφόρμες όπως Windows, Linux, Android, OS X, Raspberry Pi με τον ίδιο κώδικα.
- **GPU Accelerated:** Η μηχανή γραφικών έχει υλοποιηθεί πάνω στην OpenGL ES 2 χρησιμοποιώντας μια σύγχρονη και γρήγορη γραμμή γραφικών.

⁴<https://kivy.org/>

- **Εύκολη στη Χρήση:** Περιλαμβάνει μια ευρεία γκάμα από widgets ενώ χρησιμοποιεί μια απλή ενδιάμεση γλώσσα (kv) για τον εύκολο σχεδιασμό widget από τον χρήστη.

3.1.1 Αρχιτεκτονική του Kivy

Το Kivy αποτελείται από πολλά δομικά κομμάτια όπως φαίνεται στο Σχήμα 3.2. Κάποια βασικά από αυτά τα δομικά κομμάτια περιγράφονται στη συνέχεια [26].



Σχήμα 3.2: Αρχιτεκτονική της βιβλιοθήκης Kivy

Core και Input Providers

Ένας Core Provider είναι ένα κομμάτι κώδικα που λειτουργεί ως διαμεσολαβητής ανάμεσα στο ΛΣ και το Kivy. Η βασική ιδέα είναι να χωριστούν εργασίες όπως το άνοιγμα ενός παραθύρου, η εμφάνιση μια εικόνας ή κειμένου, η αναπαραγωγή ήχου κτλ. έτσι ώστε το API να είναι εύχρηστο και επεκτάσιμο. Το βασικότερο πλεονέκτημα όμως είναι ότι με αυτόν τον τρόπο επιτρέπεται να χρησιμοποιηθούν συγκεκριμένοι providers ανάλογα με τα σενάρια χρήσης της εφαρμογής. Για παράδειγμα, τα OS X, Linux και Windows χρησιμοποιούν διαφορετικούς providers

ΚΕΦΑΛΑΙΟ 3. ΕΡΓΑΛΕΙΑ

για τις παραπάνω εργασίες. Με τη χρήση εξειδικευμένων providers για κάθε πλατφόρμα είναι εφικτή η πλήρης αξιοποίηση του ΛΣ με αποδοτικό τρόπο, ενώ είναι εφικτή η μεταφορά της εφαρμογής σε άλλες πλατφόρμες αρκετά εύκολα.

Με αντίστοιχο τρόπο δουλεύουν και οι providers εισόδων. Αν κάποιος θέλει να υποστηρίξει μια νέα συσκευή εισόδου χρειάζεται απλά να γράψει μία κλάση η οποία διαβάζει τα δεδομένα της συγκεκριμένης συσκευής και τα μεταφράζει και γεγονότα του Kivy (Kivy Events).

Γραφικά

Το γραφικό API του Kivy είναι μια αφαιρετικότητα της OpenGL. Στο χαμηλότερο επίπεδο το Kivy δίνει hardware accelerated εντολές σχεδιασμού χρησιμοποιώντας OpenGL. Αλλά επειδή το να γράφει κανείς σε OpenGL είναι επίπονο, ειδικά για τους νέους χρήστες, είναι εφικτό να σχεδιαστούν γραφικά χρησιμοποιώντας μεταφορικές εντολές (Canvas, Rectangle κτλ.) που δεν υπάρχουν στην OpenGL.⁵

Πυρήνας

Ο κώδικας του πακέτου του πυρήνα παρέχει συχνά χρησιμοποιούμενα χαρακτηριστικά όπως:

- Το **ρολόι** που χρησιμοποιείται για να προγραμματιστούν χρονικά γεγονότα.
- Την **cache** σε περίπτωση που κάποιος θέλει να αποθηκεύσει κάτι που χρησιμοποιείται συχνά.
- Την **αναγνώριση χειρονομίας** για αποθήκευση και ανίχνευση διαφόρων σχημάτων, όπως κύκλοι, ενώ μπορεί να εκπαιδευτεί για σχήματα που καθορίζει ο χρήστης.
- Την **γλώσσα Kivy** η οποία χρησιμοποιείται για να περιγραφούν με ευκολία και αποδοτικότητα οι διεπαφές χρηστών
- Τις **ιδιότητες** οι οποίες είναι κλάσεις που συνδέουν τον κώδικα ενός widget με την περιγραφή της διεπαφής του χρήστη.

UIX

Το συγκεκριμένο πακέτο περιέχει τα πιο συχνά χρησιμοποιούμενα widgets και layouts για την γρήγορη ανάπτυξη μια διεπαφής χρήστη.

- Τα **widgets** είναι στοιχεία διεπαφής τα οποία εισάγονται στο πρόγραμμα και παρέχουν κάποια λειτουργία όπως πχ. κουμπιά, λίστες, επιγραφές κ.α.

⁵Στην εφαρμογή εκτίμησης πόζας χρησιμοποιείται η διεπαφή προγραμματισμού της OpenGL ES για την απόδοση του πλέγματος του ανθρώπινου σώματος όπως αναπτύσσεται στην ενότητα 3.4

- Τα layouts χρησιμοποιούνται για την διάταξη των widgets.

3.2 WIT.AI



Σχήμα 3.3: Wit.ai

Το Wit⁶ είναι ένα framework ανοιχτού κώδικα που επιτρέπει του χρήστες να αλληλεπιδρούν με την εφαρμογή μέσως φωνής και βασίζεται στη χρήση ισχυρού NLP (Natural Language Processing). Η λειτουργία του Wit μπορεί να χωριστεί σε 2 βασικές κατηγορίες, την **ταξινόμηση πρόθεσης** (intent classification) και την **εξαγωγή οντοτήτων** (entity extraction).

Το Wit επιτρέπει μια εφαρμογή να καταλάβει τους χρήστες. Αρχικά μπορεί ο προγραμματιστής μέσω της πλατφόρμας να δώσει κάποιες **εκφράσεις** (utterances) ως παραδείγματα στις οποίες ορίζει την πρόθεση και τις οντότητες που περιέχουν προκειμένου να εκπαιδευτεί το μοντέλο. Για παράδειγμα μπορούμε να δώσουμε ως έκφραση την παρακάτω πρόταση:

«What's the temperature?»

και να ορίσουμε ως πρόθεση **temperature_get**. Με αυτόν τον τρόπο μαθαίνουμε στο Wit ότι με αυτήν την έκφραση ο χρήστης δηλώνει την πρόθεσή του να μάθει την θερμοκρασία. Όταν επομένως ρωτήσει ο χρήστης το Wit για την θερμοκρασία (είτε με την παραπάνω έκφραση είτε με μια παρόμοια όπως «I would like to know the temperature») το Wit θα προσπαθήσει να προβλέψει την πρόθεση του χρήστη δίνοντας και μία μετρική της σιγουριάς (confidence) από 0 έως 1. Όσα περισσότερα παραδείγματα δώσουμε τόσο μεγαλύτερη γίνεται η σιγουριά και η ικανότητα του Wit για την πρόβλεψη της πρόθεσης.

Έστω ότι δίνεται ως είσοδος στο Wit η έκφραση «set the temperature to 70 degrees». Η απάντηση που θα δοθεί θα έχει την εξής μορφή:

```
{
  "text": "set the temperature to 70 degrees",
  "intents": [
    {
      "id": "226127658493500",
      "name": "temperature_get",
      "confidence": 0.9953
    }
  ]
}
```

⁶<https://wit.ai/>

ΚΕΦΑΛΑΙΟ 3. ΕΡΓΑΛΕΙΑ

```
],  
  "entities": [],  
  "traits": []  
}
```

Η απάντηση όμως δεν είναι ικανοποιητική αφού η πρόθεση του χρήστη είναι να αλλάξει η θερμοκρασία και όχι να πάρει την θερμοκρασία κάτι που είναι λογικό αφού το Wit καταλαβαίνει μόνο μία πρόθεση. Μέσω της πλατφόρμας όμως είναι εύκολο να αλλάξουμε την πρόθεση μιας ήδη δοσμένης έκφρασης και να ξαναεκπαίδευτεί το μοντέλο θέτοντας αυτή τη φορά την πρόθεση `temperature_set`. Δίνεται επίσης η επιλογή να οριστεί ένα κατώφλι σιγουριάς έτσι ώστε να μην επιστρέφονται προθέσεις κάτω από αυτό το όριο.

Στο παραπάνω παράδειγμα, θέλουμε να αποσπάσουμε μαζί με την πρόθεση και το νούμερο της επιθυμητής θερμοκρασίας. Κάτι τέτοιο είναι εύκολο μέσω της πλατφόρμας όπου μπορούμε να ορίσουμε το `70 degrees` να είναι μια οντότητα (πχ `temperature_degrees`). Έτσι στην απάντηση που θα λάβει η εφαρμογή από το Wit θα μπορεί να εξάγει και τον αριθμό των βαθμών έτσι ώστε να προβεί στην κατάλληλη πράξη για να θέσει την θερμοκρασία στην επιθυμητή τιμή.

Η επιλογή του Wit έγινε ανάμεσα σε αρκετά αντίστοιχα εργαλεία ανοιχτού κώδικα που υπάρχουν διαθέσιμα όπως το spaCy⁷ ή το Rasa⁸. Ο βασικός λόγος επιλογής του Wit είναι η ευκολία εκπαίδευσης και προσαρμογής του μέσω της πλατφόρμας. Επίσης, η ενσωμάτωση στην εφαρμογή ήταν πολύ εύκολη με τη χρήση της βιβλιοθήκης⁹ που παρέχει το Wit για Python, το documentation ήταν αρκετά κατατοπιστικό και η δομή των δεδομένων στην απάντηση του Wit το έκαναν εύκολο στην χρήση. Τέλος, είναι δυνατό κανείς να κατεβάσει τα δεδομένα ενός ήδη εκπαιδευμένου μοντέλου, και να δημιουργήσει ένα με βάση αυτά όποτε και να μπορεί στη συνέχεια να το επεκτείνει με δικές του εντολές και να το προσαρμόσει στις ανάγκες των δικών του εφαρμογών, βιοηθώντας κατά αυτόν τον τρόπο στην επεκτασιμότητα της διεπαφής του καθρέφτη.

3.3 ΧΕΙΡΙΣΜΟΣ ΦΩΝΗΣ

Για την αλληλεπίδραση του χρήστη με τον καθρέφτη επιλέξαμε ως μέσο την φωνή του. Το κομμάτι της φωνής μπορεί να χωριστεί σε δύο μέρη, την είσοδο κατά την οποία θέλουμε να μετατρέψουμε τη φωνή σε κείμενο ούτως ώστε η εφαρμογή να καταλάβει την εντολή και να εκτελέσει την κατάλληλη ενέργεια (**Speech to Text**) και την έξοδο όπου σε αρκετές περιπτώσεις θα θέλαμε ο καθρέφτης να μιλάει στον χρήστη δίνοντας του πληροφορίες και κατά την οποία μετατρέπουμε το κείμενο σε φωνή (**Text to Speech**).

⁷<https://spacy.io/>

⁸<https://rasa.com/open-source/>

⁹<https://github.com/wit-ai/pywit>

3.3.1 SpeechRecognition

Το SpeechRecognition¹⁰ είναι μια βιβλιοθήκη για την εκτέλεση αναγνώρισης φωνής όπου υποστηρίζει διάφορες μηχανές και APIs. Για τον καθρέφτη χρησιμοποιήθηκε το API της Google¹¹.

Για την λειτουργία της αναγνώρισης φωνής, απαιτείται η δημιουργία 2 αντικειμένων, ένα τύπου **Microphone** και ένα τύπου **Recognizer**. Το πρώτο αντικείμενο περιέχει μεθόδους για την είσοδο φωνής μέσω του μικροφώνου αλλά και παραμετροποίησης διαφόρων τιμών όπως η μέγιστη διάρκεια που θα ακούει το μικρόφωνο, τιμές σχετικές με τον θόρυβο φόντου κτλ, ενώ επιστρέφει ένα αντικείμενο που μπορεί να χρησιμοποιηθεί ως πηγή (source) στη συνέχεια.

Το δεύτερο αντικείμενο, περιέχει μεθόδους για την ανάγνωση του ήχου και την μετατροπή του σε κείμενο μέσω του επιθυμητού υποστηριζόμενου API. Πιο συγκεκριμένα, μέσω της συνάρτησης **listen** μπορούμε να ηχογραφήσουμε από την πηγή (στον καθρέφτη η πηγή μας είναι το μικρόφωνο) και να αποθηκεύσουμε τα δεδομένα σε κατάλληλη μορφή για την αναγνώρισή τους αργότερα. Αυτό επιτυγχάνεται περιμένοντας το επίπεδο ενέργειας του ήχου να ξεπεράσει ένα κατώφλι (ο χρήστης ξεκίνησε να μιλάει) και σταματάει όταν η ενέργεια πέσει χαμηλά ή όταν δεν υπάρχει άλλη είσοδος ήχου. Η μέθοδος **listen()** περιέχει επίσης κάποιες χρήσιμες παραμέτρους για τον καθορισμό του μέγιστου χρόνου που θα περιμένει η εφαρμογή μέχρι ο χρήστης αρχίζει να μιλάει (**timeout**) και τον καθορισμό του μέγιστου χρόνου που θα περιμένει την φράση να τελειώσει (**phrase_timeout**). Επομένως αν και οι 2 παράμετροι είναι ορισμένοι ο συνολικός χρόνος που θα ακούει η εφαρμογή είναι **timeout + phrase_timeout**.

Τέλος, τα δεδομένα που επιστρέφονται από την συνάρτηση **listen()** δίνονται ως είσοδο στην κατάλληλη συνάρτηση αναγνώρισης για το επιθυμητό API. Στην εφαρμογή του καθρέφτη επιλέξαμε την μέθοδο **recognize_google()** η οποία στέλνει τα δεδομένα στο API της Google και αυτό με τη σειρά του επιστρέφει σε κείμενο τα λόγια που ειπώθηκαν από τον χρήστη.

3.3.2 Text To Speech

Για την υλοποίηση της μετατροπής κειμένου σε φωνή χρησιμοποιήθηκαν δύο βιβλιοθήκες, η **gTTS**¹² και η **pydub**¹³.

Η **gTTS** είναι μια βιβλιοθήκη που επιτρέπει την αλληλεπίδραση της εφαρμογής με το Text-To-Speech API¹⁴ της Google και αποθηκεύει τον ήχο ομιλίας σε ένα

¹⁰https://github.com/Uberi/speech_recognition

¹¹<https://cloud.google.com/speech-to-text>

¹²<https://gtts.readthedocs.io/en/latest/>

¹³<https://pydub.com/>

¹⁴<https://cloud.google.com/text-to-speech>

προσωρινό mp3 αρχείο. Στη συνέχεια, γίνεται χρήση της pydub βιβλιοθήκης για την αναπαραγωγή του ήχου στον χρήστη.

3.4 ΑΠΕΙΚΟΝΙΣΗ ΤΟΥ ΑΝΘΡΩΠΙΝΟΥ ΣΩΜΑΤΟΣ

Στην εφαρμογή εκτίμησης πόζας, οι προβλέψεις του νευρωνικού δικτύου, δηλαδή το ανθρώπινο μοντέλο με την πόζα του χρήστη, πρέπει να απεικονίζονται στην οθόνη του καθρέφτη. Προς επίτευξη αυτού του σκοπού, απαιτείται μία περιγραφεί του ανθρώπινου σώματος καθώς και ενός αποδοτικού τρόπου παραγωγής των γραφικών. Έτσι, σε αυτό το κεφάλαιο θα αναλυθεί ο τρόπος χρήσης του μοντέλου SMPL για την μοντελοποίηση του ανθρώπινου σώματος αλλά και η χρήση της OpenGL για την απεικόνιση αυτού.

3.4.1 SMPL: Skinned multi-person Linear Model



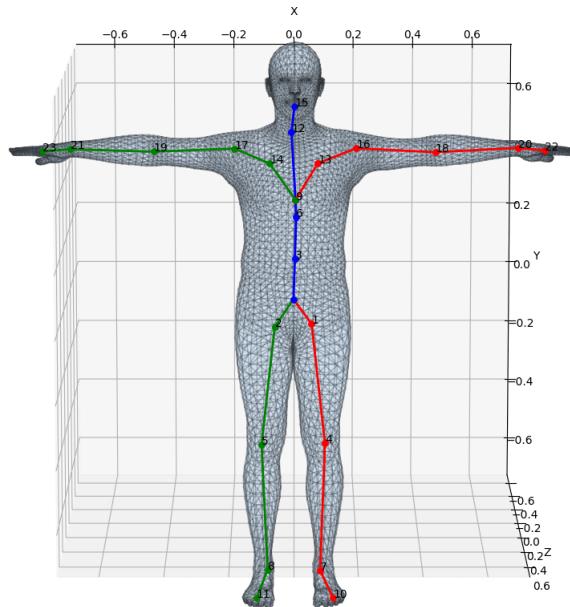
Σχήμα 3.4: Πλέγματα ανθρώπινου σώματος με χρήση του μοντέλου SMPL

Όπως αναφέρθηκε στην ενότητα 2.4.2, το μοντέλο SMPL¹⁵ αποτελεί μία μοντελοποίηση του ανθρώπινου σώματος. Το ανθρώπινο σώμα θεωρείται ότι αποτελείται από την μορφή, η διαφορά των σωμάτων σε ύψος, πάχος και αναλογίες σώματος, και την πόζα, δηλαδή τον τρόπο με οποίο παραμορφώνεται το σώμα με την κίνηση των αρθρώσεων. Η μορφή του σώματος περιγράφεται από τις παραμέτρους $\vec{\beta} \in \mathbb{R}^{10}$. Αντίστοιχα, η πόζα περιγράφεται από τις παραμέτρους $\vec{\theta} \in \mathbb{R}^{3K}$, οι οποίες είναι οι σχετικές 3D περιστροφές των $K = 23$ αρθρώσεων σε αναπαράσταση άξονα-γωνίας. Το μοντέλο SMPL, $M(\vec{\theta}, \vec{\beta}) \in \mathbb{R}^{3 \times N}$, είναι μια διαφορική συνάρτηση όπου με είσοδο τις παραμέτρους $\vec{\beta}$ και $\vec{\theta}$ παράγει ένα τριγωνικό πλέγμα του ανθρώπινου σώματος με $= 6980$ κορυφές. Αυτό προκύπτει από την διαμόρφωση των κορυφών αναφοράς σύμφωνα με τις παραμέτρους $\vec{\beta}$ και $\vec{\theta}$, στην συνέχεια περιστρέφοντας τις αρθρώσεις

¹⁵<https://smpl.is.tue.mpg.de/>

3.4. ΑΠΕΙΚΟΝΙΣΗ ΤΟΥ ΑΝΘΡΩΠΙΝΟΥ ΣΩΜΑΤΟΣ

σύμφωνα με τα $\vec{\theta}$ από την πόζα αναφοράς με χρήση του μπροστινού κινηματικού μοντέλου, και τέλος παραμορφώνοντας την επιφάνεια με γραμμική μίξη του δέρματος. Επιπλέον, το μοντέλο παράγει τα 3D σημεία κλειδιά του ανθρώπινου σκελετού (βλ. ενότητα 2.4.2), $X(\vec{\theta}, \vec{\beta}) \in \mathbb{R}^{3 \times P}$, υπολογίζοντας τα με γραμμική παλινδρόμηση από τις τελικές κορυφές του πλέγματος, όπως φαίνεται στο Σχήμα 3.5.



Σχήμα 3.5: Ο σκελετός του ανθρώπινου σώματος του μοντέλου SMPL

Στην εφαρμογή εκτίμησης πόζας, η έξοδος του βαθιού νευρωνικού δικτύου είναι το σύνολο των παραμέτρων του μοντέλου SMPL, $\vec{\beta}$ και $\vec{\theta}$, και εξ' ακολούθως το πλήρες ανθρώπινο πλέγμα αλλά και ο σκελετός. Με αυτό τον τρόπο, καθιστάτε δυνατή η λεπτομερής απεικόνιση του ανθρώπινου σώματος στην οθόνη παρέχοντας πολύ καλύτερη γραφική διεπαφή για τον χρήστη, τόσο για αισθητικούς όσο και για πρακτικούς λόγους, που δεν θα ήταν δυνατό να επιτευχθεί μόνο με μια μοντελοποίηση σκελετού.

Επιπλέον, το μοντέλο SMPL είναι ευέλικτο και ιδανικό για εφαρμογές όπου η απεικόνιση του σώματος είναι υψηστης σημασίας. Αναλυτικότερα, οι παράμετροι πόζας $\vec{\beta}$, όντας οι σχετικές 3D γωνίες περιστροφής των αρθρώσεων, υφίσταται εύκολη επεξεργασία για τον καθορισμό της πόζας. Ταυτόχρονα, το τελικό ανθρώπινο πλέγμα μπορεί να χρησιμοποιηθεί απευθείας για απεικόνιση στην οθόνη, με χρήση για παράδειγμα της OpenGL, μπορεί να γίνει επεξεργασία και την μορφή και την όψη του, και τελικά να ενσωματωθεί εύκολα με τα υπόλοιπα γραφικά της εφαρμογής.

3.4.2 OpenGL Shading Language ES

Όπως αναφέρθηκε παραπάνω, για την απεικόνιση του ανθρώπινου σώματος απαιτείται η απεικόνιση σύνθετων τρισδιάστατων πλεγμάτων. Τοιουτοτρόπως, για την απεικόνιση στην οθόνη χρησιμοποιούνται οι χαμηλότερου επιπέδου εντολές της OpenGL μέσω της διεπαφής προγραμματισμού που παρέχει το Kivy. Πιο συγκεκριμένα, για τον προσδιορισμό της θέσης, του χρώματος και του φωτισμού της κάθε κορυφής του πλέγματος χρησιμοποιείται η γλώσσα προγραμματισμού OpenGL Shading Language για Ενσωματώμενα Συστήματα (εν συντομίᾳ *GLSL ES*)¹⁶.

Η *GLSL ES* είναι μία διεπαφή προγραμματισμού εφαρμογής (Application Programming Interface, εν συντομίᾳ *API*) σκίασης υψηλού επιπέδου, αποτελώντας ένα υποσύνολο της OpenGL, και παρέχει παρόμοια σύνταξη και συναρτήσεις με την γλώσσα προγραμματισμού C. Το API επιτρέπει τον προγραμματισμό του καταχωρητή απόδοσης των πίξελ στην οθόνη, του *framebuffer* όπως αναλύθηκε στην ενότητα 2.2.3. Χαρακτηριστικό της *GLSL ES*, που την κάνει ιδιαίτερη για παραγωγή γραφικών σε ενσωματωμένα συστήματα, αποτελεί η ανεξαρτησία της από την εκάστοτε γλώσσα προγραμματισμού, η διαπλατφορμικότητα της αλλά και η απλοϊκότητα της στα στάδια παραγωγής γραφικών.

Τα επίπεδα επεξεργασίας της *GLSL ES* καθορίζονται από τους σκιαστές, οι οποίοι είναι προγράμματα που τρέχουν στην κάρτα γραφικών. Κάθε σκιαστής αντιστοιχεί σε ένα συγκεκριμένο τμήμα της διαδικασίας παραγωγής γραφικών. Επιπλέον, οι σκιαστές είναι απομονωμένοι, δηλαδή ο μόνος τρόπος επικοινωνίας μεταξύ τους είναι μέσω των εισόδων και εξόδων τους και ειδικών ομοιόμορφων μεταβλητών (*uniform variables*) οι οποίες είναι καθολικές και προσβάσιμες από όλους τους σκιαστές σε οποιαδήποτε φάση της διαδικασίας παραγωγής γραφικών.

Αναλυτικότερα, στη *GLSL ES* υφίστανται δύο επίπεδα επεξεργασίας που υλοποιούνται από τον σκιαστή κορυφής (*vertex shader*) και τον σκιαστή τμήματος (*fragment shader*). Ο *vertex shader* δέχεται ως είσοδο μία κορυφή την φορά και τα δεδομένα σχετικά με αυτή, όπως τη θέση της στον τρισδιάστατο χώρο και το κανονικό της διάνυσμα (*vertex normal* βλ. παράρτημα A.3). Η έξοδος του *vertex shader* είναι η θέση της εκάστοτε κορυφής στις συντεταγμένες της οθόνης. Από την άλλη μεριά, ο *fragment shader* δέχεται ως είσοδο τα πίξελ της οθόνης που αντιστοιχούν σε ένα πρωτόγονο σχήμα, όπως υπολογίστηκαν από τον *vertex shader*, και καθορίζει το χρώμα, το βάθος και ενδεχομένως το στένσιλ για αυτό το τμήμα.

3.5 ΕΚΤΙΜΗΣΗ 3D ΑΝΘΡΩΠΙΝΗΣ ΠΟΖΑΣ

Η εκτίμηση της 3D ανθρώπινης πόζας είναι μία περίπλοκη διαδικασία, απαιτώντας έναν τρόπο λήψης των καρέ εισόδου από την κάμερα του χρήστη ή από ένα

¹⁶<https://www.khronos.org/opengles/>

βίντεο εισόδου, την επεξεργασία των καρέ σε συγκεκριμένη μορφή, μία βιβλιοθήκη μηχανικής μάθησης και φυσικά το μοντέλο του βαθιού νευρωνικού δικτύου που διεξάγει τις εκτιμήσεις. Επομένως, στην συνέχεια του κεφαλαίου θα αναπτυχθούν ακριβώς αυτά τα εργαλεία που φέρουν σε πέρας τις αναφερθείσες διαδικασίες.

3.5.1 NumPy



Σχήμα 3.6: Λογότυπο της βιβλιοθήκης NumPy

Η βιβλιοθήκη Numerical Python (εν συντομίᾳ *NumPy*)¹⁷ χρησιμοποιείται για την διαχείριση μεγάλων, πολλών διαστάσεων λιστών και πινάκων, προσφέροντας επίσης μία εκτενή συλλογή μαθηματικών συναρτήσεων υψηλού επιπέδου για την επεξεργασία αυτών. Η NumPy είναι βιβλιοθήκη ανοιχτού κώδικα και χρησιμοποιείται κατά κόρων από την επιστημονική κοινότητα με πληθώρα συνεισφορών.

Η NumPy χρησιμοποιεί την υλοποίηση αναφοράς CPython της Python, η οποία όμως όντας διερμηνέας δεν βελτιστοποιεί τον κώδικα. Τοιουτοτρόπως, αλγόριθμοι που τρέχουν πάνω σε αυτόν τείνουν να είναι πιο αργοί από αντίστοιχους που έχουν μεταγλωττιστεί. Η NumPy αντιμετωπίζει την αργή εκτέλεση παρέχοντας πολυδιάστατες δομές πινάκων και συναρτήσεων επεξεργασίας τους, οι οποίοι είναι αποδοτικοί. Προς επίτευξη αυτού του σκοπού, η NumPy:

- Παρέχει πίνακες κοινών τύπων δεδομένων που στοιβάζονται πυκνά στην μνήμη. Οι λίστες της Python μπορούν να έχουν διαφορετικούς τύπους δεδομένων θέτοντας περιορισμούς καθώς διεξάγονται υπολογισμοί.
- Χωρίζει τις διεργασίες σε υποδιεργασίες και τις επεξεργάζεται παράλληλα.
- Χρησιμοποιεί εσωτερικά τη διεπαφή ξένων συναρτήσεων (Foreign Language Interface, FFI) της CPython, μέσω της οποίας επιτρέπεται η εκτέλεση ρουτινών και η χρήση υπηρεσιών διαφορετικής γλώσσας, όπως της C.

Η NumPy χρησιμοποιείται στην εκτίμηση πόζας εκτενώς τόσο για την αποθήκευση των διάφορων δεδομένων σε αποδοτικές δομές όσο και για την γρήγορη επεξεργασία τους. Μερικά παραδείγματα χρήσης της NumPy αποτελούν η αποθήκευση και επεξεργασία της έγχρωμης εικόνας καρέ σε έναν πίνακα διαστάσεων MxNx3 (όπου M ο αριθμός των πίξελ μήκους, N τα πίξελ πλάτους και τα 3 κανάλια χρώματος RGB¹⁸), καθώς και η αποθήκευση και επεξεργασία της εξόδου του

¹⁷<https://numpy.org/>

¹⁸Στο πρότυπο χρώματος RGB ένα οποιοδήποτε χρώμα μπορεί να παραχθεί συνδυάζοντας προσθετικά τις τιμές από τα τρία βασικά χρώματα κόκκινου (Red), πράσινου (Green) και μπλε (Blue).

ΚΕΦΑΛΑΙΟ 3. ΕΡΓΑΛΕΙΑ

νευρωνικού σε δύο πίνακες διαστάσεων 6980x3 (οι 3D συντεταγμένες των 6980 κορυφών πλέγματος) και 24x3 (οι συντεταγμένες των 24 σημείων κλειδιών).

3.5.2 OpenCV



Σχήμα 3.7: Λογότυπο της βιβλιοθήκης OpenCV

Η Open Source Computer Library (εν συντομίᾳ *OpenCV*)¹⁹ είναι μία βιβλιοθήκη προγραμματιστικών συναρτήσεων γραμμένων σε C++, παρέχοντας όμως APIs και σε άλλες γλώσσες όπως Python, Java και Matlab/Octave. Η OpenCV είναι μία διαπλατφορμική βιβλιοθήκη ανοιχτού κώδικα παρέχοντας πλήθος αλγορίθμων για υπολογιστική όραση και επεξεργασία εικόνας.

Η OpenCV χρησιμοποιείται στην εφαρμογή εκτίμησης πόζας κυρίως για την δυνατότητα λήψης εικόνων καρέ από κάποια πηγή. Συγκεκριμένα, η OpenCV παρέχει την κλάση *VideoCapture* η οποία όταν αρχικοποιείται δέχεται ως είσοδο την πηγή, η οποία μπορεί να είναι είτε η κάμερα του χρήστη είτε ένα αρχείο βίντεο. Έπειτα, καλώντας την συνάρτηση *read()* του αντικειμένου της κλάσης λαμβάνεται μία εικόνα καρέ από την πηγή, η οποία κατά την διαδικασία της αποσφαλμάτωσης προβάλλεται σε ειδικό παράθυρο με χρήση της συνάρτησης *imshow()* της OpenCV.

3.5.3 Tensorflow



Σχήμα 3.8: Λογότυπο της βιβλιοθήκης Tensorflow

Το Tensorflow²⁰ είναι μία πλατφόρμα από άκρη σε άκρη για μηχανική μάθηση. Αποτελείται από ένα ευρύ, ευέλικτο οικοσύστημα εργαλείων, βιβλιοθηκών και πό-

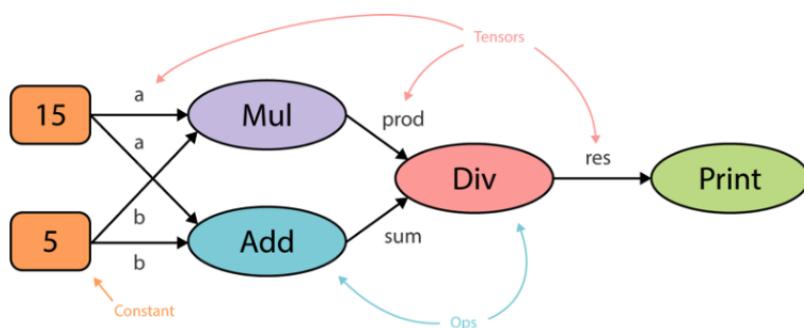
¹⁹<https://opencv.org/>

²⁰<https://www.tensorflow.org/>

ρους από την κοινότητα που επιτρέπει στους ερευνητές να εξελίξουν την τελευταία λέξη της τεχνολογίας στη μηχανική μάθηση και στους προγραμματιστές να αναπτύξουν και να ανεβάσουν με ευκολία εφαρμογές μηχανικής μάθησης.

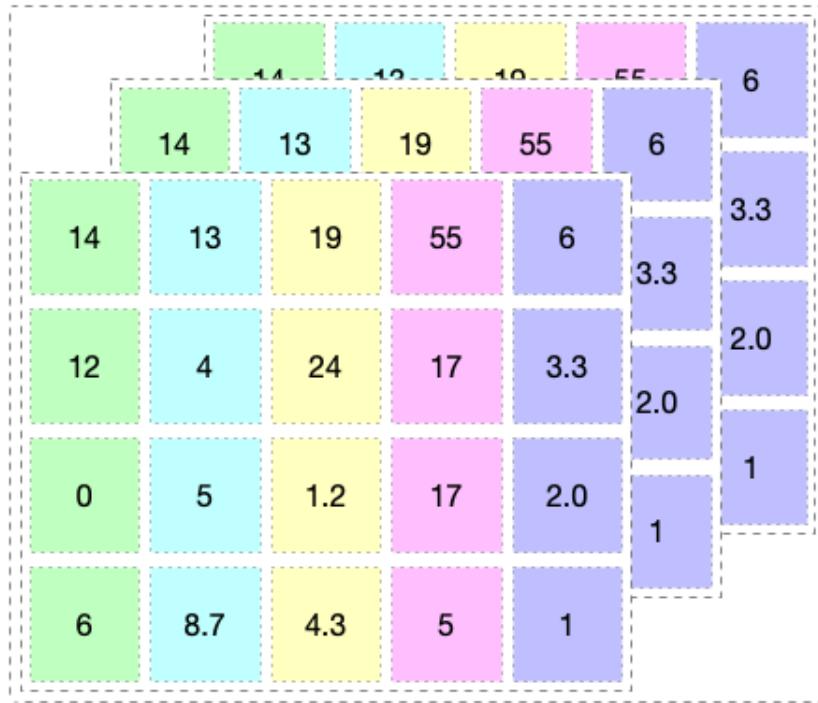
Το Tensorflow εκτελεί πράξεις με τανυστές (*tensors*), οι οποίοι με απλά λόγια είναι πολυδιάστατοι πίνακες όπως φαίνεται στο Σχήμα 3.10. Η αλληλουχία των υπολογισμών στους τανυστές περιγράφεται από έναν στατικό κατευθυνόμενο άκυκλο γράφο (βλ. παράρτημα A.3), ονομαζόμενο υπολογιστικός γράφος (computational graph), ο οποίος καθορίζεται πριν την εκτέλεση του μοντέλου. Η επικοινωνία του μοντέλου με την εφαρμογή επιτυγχάνεται μέσω των αντικειμένων *Session*, που επιτρέπει την εκτέλεση ενός γράφου ή τμήμα αυτού διανέμοντας τους πόρους του συστήματος και αποθηκεύοντας τιμές μεταβλητών και αποτελεσμάτων, και των αντικειμένων *Placeholder*, που είναι τανυστές οι οποίοι αντικαθίστανται από συγκεκριμένες τιμές κατά την εκτέλεση του μοντέλου. Για παράδειγμα ο παρακάτω κώδικας θα παρήγαγε τον υπολογιστικό γράφο που φαίνεται στο Σχήμα 3.9.

```
a = 15
b = 5
prod = a * b
sum = a + b
result = prod / sum
print(result)
```



Σχήμα 3.9: Παράδειγμα υπολογιστικού γράφου. Οι τανυστές *a* και *b* περνάνε από τους κόμβους υπολογισμού *Mul* και *Add* παράγοντας τους τανυστές *prod* και *sum* αντίστοιχα.

Με αυτό τον τρόπο, ο υπολογιστικός γράφος παράγεται με στατικό τρόπο πριν την εκτέλεση του κώδικα. Το κύριο πλεονέκτημα του στατικού γράφου είναι ότι επιτρέπει παραλληλισμό και χρονοπρογραμματισμό βασισμένο στις εξαρτήσεις των τανυστών καθιστώντας την εκπαίδευση του νευρωνικού δικτύου πιο γρήγορη και αποδοτική. Από την άλλη μεριά, ο στατικός γράφος θέτει περιορισμούς κατά την εκτέλεση του μοντέλου, απαιτώντας το μήκος της εισόδου να είναι σταθερό, το



Σχήμα 3.10: Παράδειγμα τανυστή

οποίο είναι αρκετά περιοριστικό σε ορισμένες περιπτώσεις που απαιτείται δυναμική μορφή εισόδου. Για παράδειγμα, σε ένα μοντέλο ανάλυσης συναισθήματος από κείμενο θα έπρεπε να ορισθεί σταθερό μήκος προτάσεων (γεμίζοντας τις μικρότερες προτάσεις με μηδενικά) μειώνοντας ενδεχομένως την απόδοση του μοντέλου.

Ένα άλλο χαρακτηριστικό του Tensorflow, που αποδεικνύει την ωριμότητα του ως πλατφόρμα, αποτελεί το πλαίσιο λογισμικού TensorFlow Serving. Αυτό είναι ένα ευέλικτο, υψηλής απόδοσης σύστημα σερβιρίσματος μοντέλων μηχανικής μάθησης. Το TensorFlow Serving επιτρέπει το εύκολο και γρήγορο στήσιμο ενός HTTP ή gRPC server όπου ανεβάζεται το εκπαιδευμένο μοντέλο. Στην συνέχεια, παρέχεται ένα RESTful API με το οποίο γίνεται δυνατή η χρήση του μοντέλου για εξαγωγή προβλέψεων. Τοιουτοτρόπως, σε αντίστοιχες εφαρμογές με περιορισμένους υπολογιστικούς πόρους μπορεί να αποδεσμευτεί το μοντέλο μηχανικής μάθησης από την υπόλοιπη εφαρμογή, επιτρέποντας την εκτέλεση αλγορίθμων σε συστήματα που δεν θα ήταν διαφορετικά εφικτό.

Επιπλέον, το Tensorflow προσφέρει το Tensorboard το οποίο είναι ένα εργαλείο για οπτικοποίηση του νευρωνικού δικτύου και των αποτελεσμάτων εκπαίδευσής του. Πιο συγκεκριμένα, το Tensorboard μπορεί να προβάλει τους υπολογιστικό γράφο του δικτύου, να φτιάξει γραφήματα των μεταβλητών, να προβάλει ιστογράμματα και κατανομές σχετικές με την εκπαίδευση και να παράγει σύνοψη μετρικών έτσι όπως ορίστηκαν στον κώδικα με χρήση της μονάδας `summary` του Tensorflow.

3.5.4 Μοντέλο Human Mesh Recovery

Η εκτίμηση της 3D ανθρώπινης πόζας επιτεύχθηκε με αρωγό το Human Mesh Recovery (εν συντομίᾳ HMR)²¹[24]. Το μοντέλο HMR είναι ένα από άκρη βαθύ νευρωνικό δίκτυο για την ανάκτηση του 3D πλέγματος του ανθρώπινου σώματος από μία έγχρωμη εικόνα, χρησιμοποιώντας το μοντέλο ανθρώπινου σώματος SMPL, όπως περιγράφηκε στην ενότητα 3.4.1. Έτσι, η έξοδος του HMR είναι ολιστική, εκτιμώντας ολόκληρο το 3D σώμα ακόμα και σε περιπτώσεις όπου η εικόνα περικόπτεται ή ο άνθρωπος αποκρύβεται, καθιστώντας την λειτουργία του μοντέλου εύρωστη σε συνθήκες αβέβαιου περιβάλλοντος.

Αξίζει να σημειωθεί ότι υπάρχει πληθώρα ερευνών για την ανάλυση της ανθρώπινης πόζας από μία εικόνα. Εντούτοις, η πλειοψηφία των ερευνών επικεντρώνεται στην ανάκτηση των 3D θέσεων των σημείων κλειδιών ή αρθρώσεων του σώματος [13], [20], [17]. Μία τέτοια προσέγγιση όμως δεν περιγράφει πλήρως την πόζα, καθώς οι θέσεις των σημείων κλειδιών δεν περιορίζουν πλήρως τους βαθμούς ελευθερίας των αρθρώσεων. Για παράδειγμα, γνωρίζοντας τις θέσεις των σημείων κλειδιών δεν είναι εφικτή η εκτίμηση του προσανατολισμού των άκρων και του κεφαλιού, ένα πρόβλημα που αντιμετωπίζεται με την χρήση του HMR και συγκεκριμένα της έξόδου σε παραμέτρους SMPL.

Επιπλέον, υπάρχουσες μέθοδοι για την ανάκτηση του ανθρώπινου πλέγματος χρησιμοποιούν προσεγγίσεις πολλαπλών σταδίων [22], [27], [25]. Αναλυτικότερα, αρχικά εκτιμούνται οι 2D θέσεις των αρθρώσεων και στη συνέχεια εκτιμούνται οι παράμετροι του 3D μοντέλου. Αυτή η σταδιακή προσέγγιση όμως δεν είναι ιδανική για δύο λόγους. Αφενός τα επιμέρους νευρωνικά δίκτυα επαυξάνουν το χρόνο εκτίμησης και τις απαιτήσεις για υπολογιστική ισχύ και αφετέρου στα διαδοχικά στάδια εκτίμησης χάνονται πολύτιμες πληροφορίες της εικόνας.

Έτσι, το HMR ξεπερνάει τις υπάρχουσες μεθόδους με αρκετούς τρόπους:

- Γίνεται εκτίμηση του 3D ανθρώπινου πλέγματος απευθείας από τα χαρακτηριστικά της εικόνας. Με αυτόν τον τρόπο αποφεύγεται η ανάγκη εκπαίδευσης του νευρωνικού σε διαφορετικά στάδια και διατηρεί τις πληροφορίες της εικόνας.
- Πέρα από τον ανθρώπινο σκελετό το δίκτυο εξάγει το ανθρώπινο πλέγμα, το οποίο περιέχει περισσότερη πληροφορία σχετικά με την πόζα και επιτρέπει την εύκολη απεικόνιση του σε περιβάλλον παιχνιδιού, ικανοποιώντας τις ανάγκες της εφαρμογής.
- Το δίκτυο εκπαίδευται από άκρη σε άκρη, χωρίς ενδιάμεσα στάδια εκτίμησης. Μάλιστα, είναι πιο αποδοτικό τόσο σε ακρίβεια όσο και ταχύτητα σε σχέση με προηγούμενες μεθόδους που έχουν έξοδο το ανθρώπινο πλέγμα [27], [22].
- Το δίκτυο μπορεί να εκπαίδευτεί με σχολιασμένα σύνολα δεδομένων είτε σε

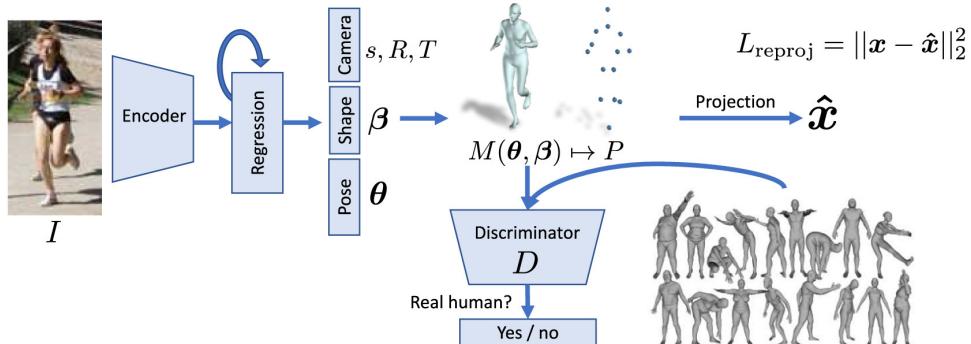
²¹<https://github.com/akanazawa/hmr>

ΚΕΦΑΛΑΙΟ 3. ΕΡΓΑΛΕΙΑ

2D είτε σε 3D. Ακόμα και στην περίπτωση εκπαίδευσης μόνο με 2D δεδομένα η ανακατασκεύαση της πόζας είναι επαρκώς ικανοποιητική.

Αρχιτεκτονική του HMR

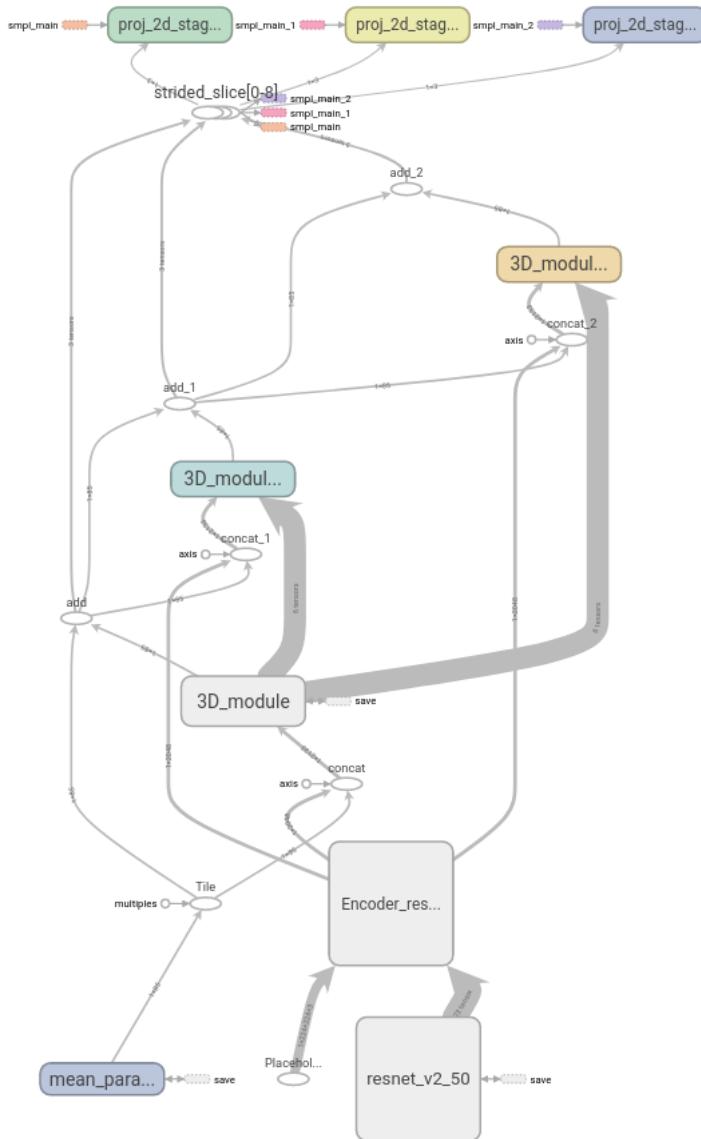
Στο Σχήμα 3.11 φαίνεται μία σύνοψη της αρχιτεκτονικής του HMR, που μπορεί να εκπαιδευτεί από άκρη σε άκρη. Τα χαρακτηριστικά της εικόνας προκύπτουν από ένα συνελικτικό δίκτυο και στέλνονται σε έναν επαναληπτικό παλινδρομητή στόχος του οποίου είναι η εκτίμηση του 3D ανθρώπινου σώματος και η θέση της κάμερας έτσι ώστε τα εκτιμώμενα 3D σημεία κλειδιά να προβάλλονται πάνω στα σχολιασμένα 2D σημεία κλειδιά. Οι εκτιμώμενοι παράμετροι στέλνονται επίσης σε έναν διαχωριστή ο οποίος κρίνει αν είναι φυσικά δυνατή η πόζα. Τοιουτούπως, ο διαχωριστής λειτουργεί ως ασθενής επίβλεψη του δικτύου, βελτιώνοντας την απόδοση του και παροτρύνοντάς το στην εκτίμηση ποζών που βρίσκονται στα πλαίσια των δυνατών.



Σχήμα 3.11: Αρχιτεκτονική του μοντέλου *Human Mesh Recovery*: Η εικόνα περνάει από ένα συνελικτικό κωδικοποιητή, εξάγοντας τα χαρακτηριστικά της. Έπειτα, στέλνεται σε μία μονάδα επαναληπτικής παλινδρόμησης όπου γίνεται εκτίμηση της 3D αναπαράστασης του ανθρώπου, υπολογίζοντας τις παραμέτρους του μοντέλου SMPL. Η μονάδα παλινδρόμησης προσπαθεί να ελαχιστοποιήσει το σφάλμα της προβολής των σημείων κλειδιών. Οι εκτιμώμενες 3D παράμετροι περνάνε επίσης από έναν διαχωριστή D, στόχος του οποίου είναι να επαληθεύσει αν είναι πραγματικά εφικτή η πόζα.

Αναλυτικότερα, η είσοδος του νευρωνικού δικτύου για κάθε καρέ είναι μία εικόνα σμικρυσμένη σε διαστάσεις 224×224 πίξελ, διατηρώντας όμως την αρχική αναλογία διαστάσεων. Η επιλογή των μικρών διαστάσεων της εικόνας είναι απαραίτητη ώστε η εξαγωγή των χαρακτηριστικών της (και ολόκληρη η εκτίμηση πόζας) να γίνεται σε πραγματικό χρόνο.

3.5. ΕΚΤΙΜΗΣΗ 3D ΑΝΘΡΩΠΙΝΗΣ ΠΟΖΑΣ

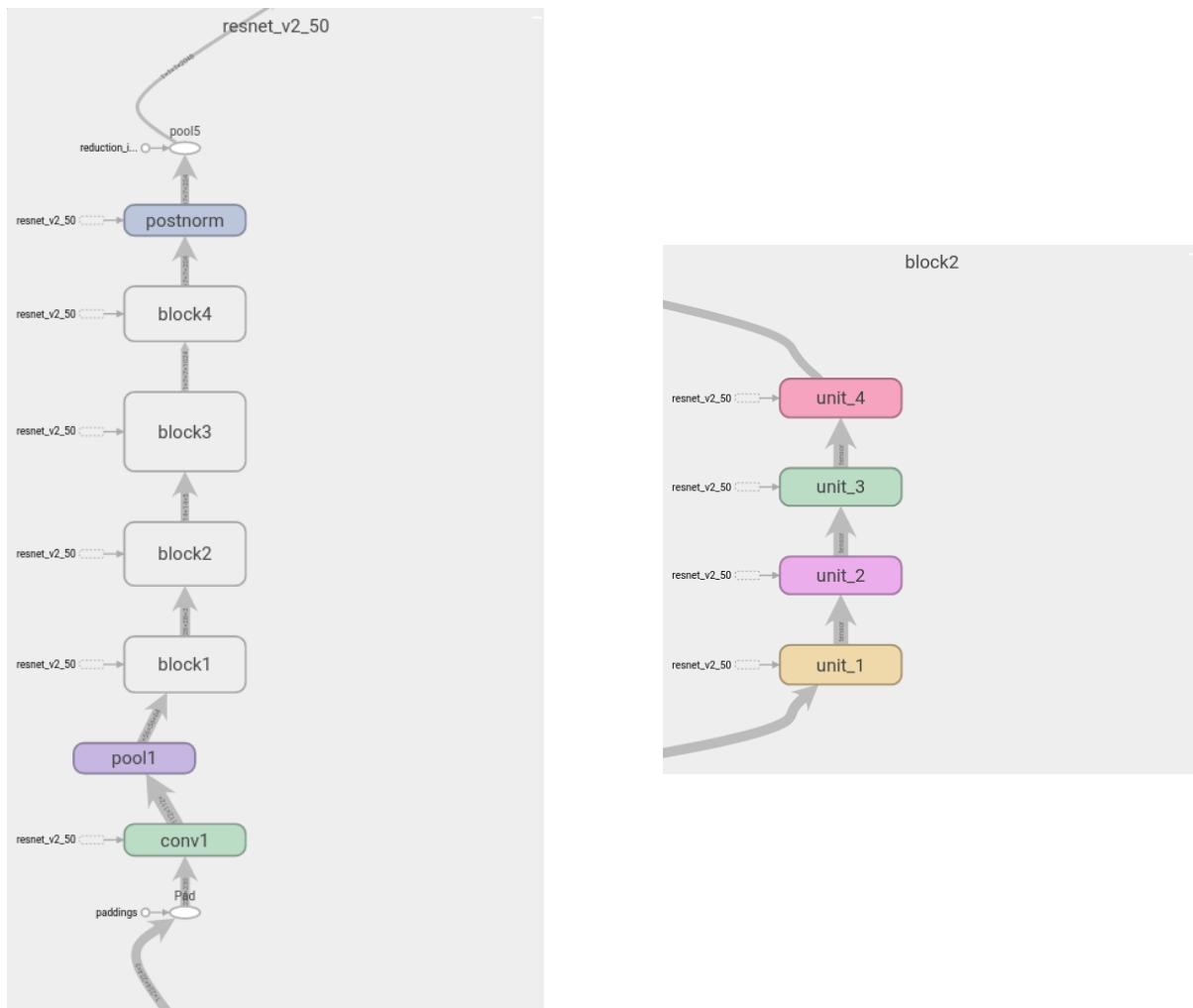


Σχήμα 3.12: Αρχιτεκτονική του δικτύου εκτίμησης του *Human Mesh Recovery* από το *Tensorboard*: Τα χαρακτηριστικά της εικόνας εξάγονται από τον κωδικοποιητή *Encoder_resnet*. Στην συνέχεια τα χαρακτηριστικά μαζί με τις τις παραμέτρους στέλνονται στον παλινδρομητή *3D_module* 3 φορές. Στην πρώτη επανάληψη (*3_module* γκρί) χρησιμοποιούνται οι μέσες τιμές των παραμέτρων, ενώ στις άλλες δύο (μπλε και κίτρινο) χρησιμοποιούνται οι εκτιμώμενοι παράμετροι από τη προηγούμενη επανάληψη. Οι παράμετροι στέλνονται στο μοντέλο SMPL, *smpl_main*, παράγονται οι 3D κορυφές του πλέγματος και τα σημεία κλειδιά και μαζί με τις συντεταγμένες τις κάμερας στέλνονται στα τελικά 3 στάδια *proj_2d_stage*, όπου επιτελείται η προβολή των 3D σημείων κλειδιών σε 2D.

Στη συνέχεια, η εξαγωγή των χαρακτηριστικών της εικόνας γίνεται από την μονάδα *Encoder_resnet*, η οποία είναι το προ-εκπαιδευμένο μοντέλο του *Tensorflow*

ΚΕΦΑΛΑΙΟ 3. ΕΡΓΑΛΕΙΑ

[resnet_v2_50](#) ταξινόμησης εικόνων πάνω στο σύνολο δεδομένων ImageNet²². Με λίγα λόγια, το *resnet_v2_50* είναι ένα παραμένων συνελικτικό δίκτυο (residual convolutional network) αποτελούμενο από 4 μπλοκ με 50 συνολικά επίπεδα, όπως φαίνεται στο Σχήμα 3.13. Αρχικά, υπάρχει ένα επίπεδο convolution²³ και max-pooling με βήμα 2 και διαστάσεις πυρήνων 7×7 και 3×3 αντίστοιχα. Τα 4 μπλοκ αποτελούνται από 3, 4, 6 και 3 παραμένων μονάδες (residual units) αντίστοιχα των οποίων η δομή φαίνεται στο Σχήμα 3.14. Τέλος, υπάρχουν ένα επίπεδο batch normalization, ένα επίπεδο ενεργοποίησης ReLU και ένα average pooling.²⁴ Η έξοδος του *Encoder_resnet* είναι τα χαρακτηριστικά της εικόνας $\phi \in \mathbb{R}^{2048}$.

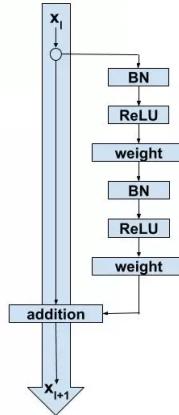


Σχήμα 3.13: Η αρχιτεκτονική του *resnet_v2_50* όπως φαίνεται με χρήση του εργαλείου Tensorboard. Στα αριστερά φαίνονται τα 4 μπλοκ του *resnet_v2_50* ενώ στα δεξιά φαίνονται τα 4 residual units του 2ου μπλοκ.

²²<https://www.image-net.org/>

²³Η επεξήγηση των επιπέδων των νευρωνικών δικτύων γίνεται στο παράρτημα [A.2](#).

²⁴Αναφορικά, στο προ-εκπαίδευμένο δίκτυο *resnet_v2_50* υπάρχει και ένα τελικό επίπεδο πλήρως συνδεδεμένο με 1000 κόμβους για την έξοδο κλάσης του ImageNet, το οποίο όμως δεν χρησιμοποιείται στην εκτίμηση πόζας.



Σχήμα 3.14: Αρχιτεκτονική του residual unit του Resnet v2: Το κάθε residual unit αποτελείται από τα εξής επίπεδα Batch Normalization, ReLU, Convolution, Batch Normalization και Convolution. Επιπλέον, υπάρχει και μία ταυτοτική σύνδεση με το επόμενο residual unit, δίνοντας την δυνατότητα να παραλειφθούν τα αναφερόμενα επίπεδα.

Το δίκτυο ResNet χρησιμοποιεί κατά κόρων επίπεδα Batch Normalization, ρυθμίζοντας την μέση τιμή και διακύμανση των δεδομένων εισόδου σε κάθε επίπεδο και βελτιώνοντας έτσι την απόδοση του δικτύου. Τοιουτοτρόπως, εξαλείφεται το πρόβλημα της μεταβολής των στατιστικών χαρακτηριστικών μεταξύ των δεδομένων εκπαίδευσης και του πραγματικού κόσμου. Στην εφαρμογή της εκτίμησης πόζας, το γεγονός αυτό είναι άκρως σημαντικό καθώς η αβεβαιότητα των συνθηκών περιβάλλοντος, του φωτισμού ή και της ενδυμασίας του χρήστη είναι δεδομένα.

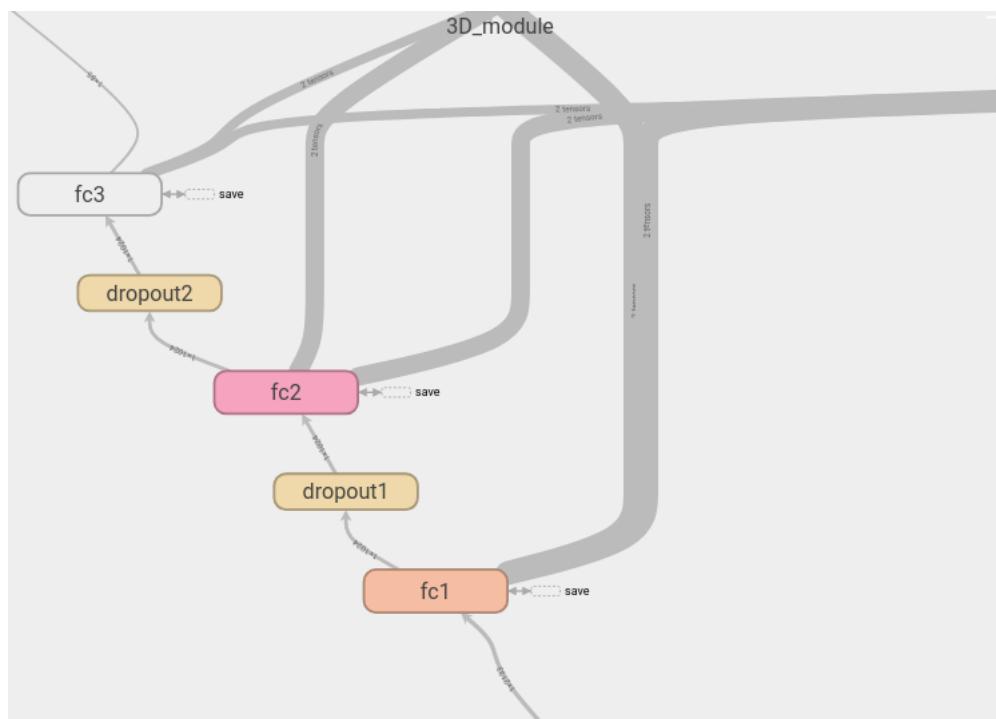
Ένα άλλο χαρακτηριστικό του ResNet είναι η χρήση των ταυτοτικών συνδέσεων μεταξύ των residual units. Από την μία πλευρά, αυτό επιτρέπει στο δίκτυο να μάθει μία ταυτοτική συνάρτηση εξασφαλίζοντας ότι το υψηλότερα επίπεδα θα έχουν τουλάχιστον τόσο καλή απόδοση όσο τα χαμηλότερα επίπεδα και σίγουρα όχι χειρότερη. Από την άλλη πλευρά και ίσως η πιο σημαντική, είναι ότι επιτρέπουν στο ανάδελτα του σφάλματος να ρεύσει προς τα αρχικά επίπεδα μέσω αυτού του εναλλακτικού μονοπατιού. Με αυτόν τον τρόπο, αντιμετωπίζεται το πρόβλημα του εξαφανιζόμενου ανάδελτα σφάλματος (vanishing gradient problem) καθιστώντας την εκπαίδευση βαθιών νευρικών δικτύων δυνατή [28].

Στο Σχήμα 3.15 φαίνεται η μονάδα 3D παλινδρόμησης, η οποία αποτελείται από 2 πλήρως συνδεδεμένα επίπεδα με 1024 κόμβους το καθένα και ένα επίπεδο dropout μεταξύ τους, ακολουθούμενα από ένα τελικό πλήρως συνδεδεμένο επίπεδο με 85 κόμβους. Η έξοδος του 3D_module λοιπόν είναι το διάγυσμα $\Theta \in \mathbb{R}^{85}$, το οποίο αποτελείται αναλυτικά από:

- τις παραμέτρους πόζας $\theta \in \mathbb{R}^{72}$, που είναι οι περιστροφές ανά άξονα, δηλαδή οι 3 γωνίες των 24 αρθρώσεων.
- τις παραμέτρους σχήματος $\beta \in \mathbb{R}^{10}$, που καθορίζουν την μορφή του ανθρώπινου σώματος, δηλαδή το ύψος, βάρος και αναλογία οστών.

ΚΕΦΑΛΑΙΟ 3. ΕΡΓΑΛΕΙΑ

- τις 3D συντεταγμένες θέσεως της κάμερας.



Σχήμα 3.15: Tensorboard - Αρχιτεκτονική της μονάδας 3D_module

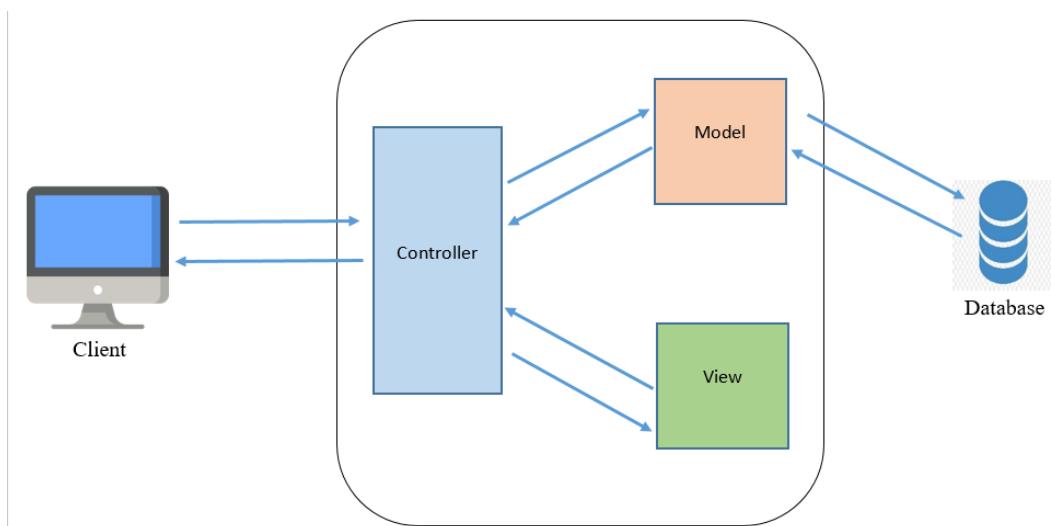
Όπως έχει αναφερθεί, η τελική έξοδος της εκτίμησης πόζας προκύπτει από την εκτέλεση 3 φορών της μονάδας 3D_module, όπου η είσοδος κάθε φορά είναι οι παράμετροι Θ της προηγούμενης επανάληψης και τα χαρακτηριστικά της εικόνας ϕ .

4

Υλοποίηση Λειτουργικού Καθρέφτη

4.1 MODEL-VIEW-CONTROLLER

Το Model-View-Controller (εν συντομίᾳ **MVC**) είναι ένα σχεδιαστικό πρότυπο ευρέως διαδεδομένο, ειδικά στην ανάπτυξη Web εφαρμογών και απεικονίζεται στο Σχήμα 4.1. Όπως προδίδει και το όνομά του, το MVC χρησιμοποιεί 3 είδη κλάσεων οι οποίες περιγράφονται παρακάτω [29].



Σχήμα 4.1: Αρχιτεκτονική MVC

- **Model:** Οι συγκεκριμένες κλάσεις χρησιμοποιούνται για την αλληλεπίδραση με τα δεδομένα, δηλαδή την εισαγωγή, την ανάκτηση και την ενημέρωση των

ΚΕΦΑΛΑΙΟ 4. ΥΛΟΠΟΙΗΣΗ ΛΕΙΤΟΥΡΓΙΚΟΥ ΚΑΘΡΕΦΤΗ

βάσεων δεδομένων της εφαρμογής

- **View:** Υλοποιούν με γραφικό τρόπο την διεπαφή της εφαρμογής μέσω της οποίας αλληλεπιδρούν οι χρήστες.
- **Controller:** Ακούνε τα αιτήματα των χρηστών και εκτελούν τις κατάλληλες ενέργειες. Οι κλάσεις αυτές αλληλεπιδρούν με τις Model και επιλέγουν το κατάλληλο View για να εμφανίσουν στον χρήστη ανάλογα με τις ενέργειες που απαιτεί.

Βλέπουμε επομένως ότι το MVC είναι ένα πρότυπο 3 επιπέδων το οποίο ορίζει εκτός από τα επίπεδα και τις σχέσεις μεταξύ αυτών (Σχήμα 4.1). Πιο συγκεκριμένα, το επίπεδο του Controller αλληλεπιδρά με τον χρήστη λαμβάνοντας δεδομένα τα οποία τα αποστέλλει στο Model επίπεδο αφού πρώτα τα επικυρώσει. Επίσης, ανάλογα με το αίτημα του χρήστη θα ζητήσει τα κατάλληλα δεδομένα από το Model προκειμένου να τα αποστείλει στο επίπεδο View για να δημιουργηθεί η κατάλληλη απεικόνιση για τον χρήστη.

Τέλος, το συγκεκριμένο πρότυπο προσφέρει αρκετά πλεονεκτήματα όπως:

- Μας βοηθάει να ελέγξουμε την πολυπλοκότητα της εφαρμογής χωρίζοντας την σε 3 επίπεδα.
- Επιτρέπει ευκολότερες δοκιμές (testing) της εφαρμογής.
- Διαφορετικοί προγραμματιστές μπορούν να δουλεύουν ταυτόχρονα στα διαφορετικά επίπεδα κάτι που μειώνει τον χρόνο ανάπτυξης της εφαρμογής.

4.2 ΑΠΑΙΤΗΣΕΙΣ ΣΥΣΤΗΜΑΤΟΣ

Οι απαιτήσεις συστήματος εκφράζουν τις ανάγκες και τους περιορισμούς που τίθενται σε ένα σύστημα το οποίο προσπαθεί να λύσει ένα πραγματικό πρόβλημα [30].

Οι απαιτήσεις από το σύστημα χωρίζονται σε δύο κατηγορίες, τις λειτουργικές (ΛΑ) και τις μη λειτουργικές (ΜΛΑ) απαιτήσεις. Οι λειτουργικές απαιτήσεις αντανακλούν τη δυνατότητα εκτέλεσης ενεργειών που παρέχει το σύστημα στον χρήστη, ενώ οι μη λειτουργικές απαιτήσεις είναι αυτές που χαρακτηρίζουν το σύστημα και προσδιορίζουν ποιοτικά χαρακτηριστικά στις λειτουργίες του. Μπορεί, επίσης, να τεθούν περιορισμοί και στόχοι ως προς τη συμπεριφορά του συστήματος, όπως και απαιτήσεις ως προς την εξελιξιμότητά του.

4.2.1 Λειτουργικές Απαιτήσεις

ΛΑ-1

Ο χρήστης πρέπει να μπορεί να δίνει φωνητικές εντολές.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να δίνει φωνητικές εντολής μέσω της φωνής του τις οποίες ο καθρέφτης θα εκτελεί.

User Priority (5/5): Η δυνατότητα φωνητικών εντολών είναι πάρα πολύ σημαντική για τον χρήστη αφού θα είναι ο κύριος τρόπος επικοινωνίας και αλληλεπίδρασής του με τον καθρέφτη.

Technical Priority (5/5): Η δυνατότητα φωνητικών εντολών είναι υψηστης σημασίας για το σύστημα, καθώς πρέπει να είναι σε θέση να αναγνωρίζει τις προθέσεις του χρήστη και να εκτελεί τις κατάλληλες ενέργειες.

ΛΑ-2

Ο χρήστης πρέπει να μπορεί να ανοίγει διαφορετικές εφαρμογές.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να χρησιμοποιεί τις διαφορετικές εφαρμογές που είναι εγκατεστημένες στον καθρέφτη.

User Priority(5/5): Η δυνατότητα χρήστης διαφορετικών εφαρμογών είναι πολύ σημαντική για τον χρήστη, καθώς με αυτόν τον τρόπο μπορεί να εκτελέσει τις διαφορετικές λειτουργίες που του παρέχει ο καθρέφτης

Technical Priority (5/5): Η δυνατότητα χρήσης διαφορετικών εφαρμογών είναι πολύ σημαντική για το σύστημα, αφού θα πρέπει να είναι σε θέση να εκτελεί τις διαφορετικές εφαρμογές που είναι εγκατεστημένες στον καθρέφτη.

ΛΑ-3

Ο χρήστης πρέπει να μπορεί να εγκαθιστά διαφορετικές εφαρμογές.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να εγκαθιστά στον καθρέφτη εφαρμογές που αναπτύσσει αυτός ή κάποιος τρίτος.

User Priority (4/5): Η δυνατότητα εγκατάστασης διαφορετικών εφαρμογών είναι αρκετά σημαντική για τον χρήστη, καθώς με αυτόν τον τρόπο θα μπορεί να επεκτείνει τις δυνατότητες του καθρέφτη.

Technical Priority (5/5): Η δυνατότητα εγκατάστασης διαφορετικών εφαρμογών εί-

ΚΕΦΑΛΑΙΟ 4. ΥΛΟΠΟΙΗΣΗ ΛΕΙΤΟΥΡΓΙΚΟΥ ΚΑΘΡΕΦΤΗ

ναι πολύ σημαντική για το σύστημα, αφού θα πρέπει να δέχεται εύκολα εξωτερικές εφαρμογές και να τις εκτελεί χωρίς προβλήματα.

ΛΑ-4

Ο χρήστης πρέπει να μπορεί να ενεργοποιεί και να απενεργοποιεί τον καθρέφτη με φωνητικές εντολές.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να ενεργοποιεί και να απενεργοποιεί τον καθρέφτη δίνοντάς του κατάλληλες εντολές ανοίγματος/κλεισίματος.

User Priority (5/5): Η δυνατότητα ενεργοποίησης/απενεργοποίησης του καθρέφτη με εντολές είναι πολύ σημαντική για τον χρήστη, καθώς θα μπορεί να ελέγξει πότε ο καθρέφτης να είναι σε λειτουργία

Technical Priority (5/5): Η δυνατότητα ενεργοποίησης/απενεργοποίησης του καθρέφτη με εντολές είναι πολύ σημαντική για το σύστημα, καθώς θα μπορεί να ελεγχθεί η κατάσταση λειτουργίας του.

ΛΑ-5

Ο χρήστης πρέπει να μπορεί να αλλάζει τις ρυθμίσεις του καθρέφτη.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να αλλάζει τις ρυθμίσεις του καθρέφτη είτε μέσω εντολών είτε μέσω κειμένου.

User Priority (5/5): Η δυνατότητα αλλαγής των ρυθμίσεων είναι πολύ σημαντική για τον χρήστη, καθώς θα μπορεί να εξατομικεύσει τις λειτουργίες του καθρέφτη ανάλογα με τα θέλω του.

Technical Priority (5/5): Η δυνατότητα αλλαγής των ρυθμίσεων είναι πολύ σημαντική για το σύστημα, καθώς θα είναι εφικτή η ορθή και εξατομικευμένη λειτουργία των εφαρμογών του καθρέφτη.

ΛΑ-6

Ο χρήστης πρέπει να μπορεί να δει την ημερομηνία και ώρα.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να δει την τωρινή ημερομηνία και την ώρα στην αρχική οθόνη του καθρέφτη

User Priority (5/5): Η δυνατότητα ενημέρωσης για την ημερομηνία και ώρα είναι πολύ σημαντική για τον χρήστη προκειμένου να μπορεί να οργανώσει καλύτερα το

πρόγραμμά του.

Technical Priority (5/5): Η δυνατότητα ενημέρωσης για την ημερομηνία και ώρα είναι πολύ σημαντική για το σύστημα αφού με βάση αυτά μπορεί να επηρεαστεί η λειτουργία του.

ΛΑ-7

Ο χρήστης πρέπει να μπορεί να δει τον καιρό σε ζωντανό χρόνο.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να δει τον καιρό σε ζωντανό χρόνο στην αρχική οθόνη του καθρέφτη.

User Priority (5/5): Η δυνατότητα ζωντανής ενημέρωσης του καιρού είναι πολύ σημαντική για τον χρήστη, καθώς μπορεί να οργανώσει καλύτερα το πρόγραμμά του.

Technical Priority (4/5): Η δυνατότητα ζωντανής ενημέρωσης του καιρού είναι αρκετά σημαντική για το σύστημα αφού μπορεί να χρησιμοποιήσει τις πληροφορίες για βελτίωση της λειτουργίας του.

ΛΑ-8

Ο χρήστης πρέπει να μπορεί να κάνει ερωτήσεις σχετικά με τον καιρό.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να κάνει ερωτήσεις και να ενημερώνεται σχετικά με τον καιρό.

User Priority (4/5): Η δυνατότητα ερωτήσεων σχετικά με τον καιρό είναι αρκετά σημαντική για τον χρήστη αφού μπορεί να ενημερωθεί και να οργανώσει καλύτερα το πρόγραμμά του.

Technical Priority (2/5): Η δυνατότητα ερωτήσεων σχετικά με τον καιρό δεν είναι πολύ σημαντική για την εύρυθμη λειτουργία του συστήματος αφού μπορεί να λειτουργήσει ομαλά και χωρίς αυτήν.

4.2.2 Μη Λειτουργικές Απαιτήσεις

ΜΛΑ-1

Το σύστημα πρέπει να αποκρίνεται στις εντολές του χρήστη σε λιγότερο από 500ms.

ΚΕΦΑΛΑΙΟ 4. ΥΛΟΠΟΙΗΣΗ ΛΕΙΤΟΥΡΓΙΚΟΥ ΚΑΘΡΕΦΤΗ

Περιγραφή: Το σύστημα πρέπει να αναγνωρίζει τις προθέσεις του χρήστη και να ανταποκρίνεται κατάλληλα σε λιγότερο από 500ms.

User Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για τον χρήστη, καθώς θέλει ο καθρέφτης να του προσφέρει μια διαδραστική εμπειρία πραγματικού χρόνου.

Technical Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για το σύστημα, καθώς θα καθορίσει τα τεχνικά χαρακτηριστικά και τη δομή του, προκειμένου να καλύπτεται η απαιτούμενη ταχύτητα απόκρισης (500ms).

ΜΛΑ-2

Το σύστημα πρέπει να ανταποκρίνεται ορθά όταν η εντολή του χρήστη δεν αναγνωρίζεται

Περιγραφή: Το σύστημα πρέπει να χειρίζεται εντολές του χρήστη που δεν αναγνωρίζονται και να μην καταρρέει.

User Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για τον χρήστη, καθώς θέλει ο καθρέφτης να λειτουργεί χωρίς σφάλματα.

Technical Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για το σύστημα, καθώς ο χειρισμός των "κακών" εισόδων θα βοηθήσει τον καθρέφτη να λειτουργεί ορθά χωρίς σφάλματα.

ΜΛΑ-3

Το σύστημα πρέπει να επιτρέπει την προσάρτηση εφαρμογών.

Περιγραφή: Το σύστημα πρέπει να επιτρέπει την προσάρτηση εφαρμογών και την εύκολη ενσωμάτωσή τους στην ροή του προγράμματος.

User Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για τον χρήστη, καθώς η ενσωμάτωση εξωτερικών εφαρμογών θα επεκτείνει τις δυνατότητες που θα του δίνει ο καθρέφτης.

Technical Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για το σύστημα, καθώς θα πρέπει να σχεδιαστεί με τέτοιο τρόπο ώστε να είναι εύκολη η επεκτασιμότητα των εφαρμογών του καθρέφτη.

ΜΛΑ-4

Το σύστημα πρέπει να επιτρέπει την ταυτόχρονη εμφάνιση εφαρμογών και του ειδώλου του χρήστη.

Περιγραφή: Το σύστημα πρέπει να έχει ανακλαστική οθόνη στην οποία ο χρήστης να μπορεί να δει τόσο τις εφαρμογές του καθρέφτη όσο και το είδωλό του.

User Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για τον χρήστη, καθώς θέλει να μπορεί να αλληλεπιδρά με τον καθρέφτη καθώς βλέπει το είδωλό του.

Technical Priority (5/5): Η απαίτηση αυτήν είναι πολύ σημαντική για το σύστημα, καθώς θα καθορίσει το είδος της οθόνης που θα χρησιμοποιηθεί και την τοποθέτηση των εφαρμογών πάνω σε αυτήν.

ΜΛΑ-5

Το σύστημα πρέπει να υποστηρίζει σύνδεση με το Wit.ai.

Περιγραφή: Το σύστημα πρέπει να υποστηρίζει σύνδεση με το Wit.ai προκειμένου να επιτρέπεται η αναγνώριση πρόθεσης των εντολών του χρήστη.

User Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για τον χρήστη, καθώς ο καθρέφτης πρέπει να αναγνωρίζει τις προθέσεις του και να εκτελεί τις κατάλληλες ενέργειες.

Technical Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για το σύστημα, καθώς πρέπει να μπορεί να αναγνωρίζει τις προθέσεις του χρήστη για να εκτέλεση των επιθυμητών ενεργειών.

ΜΛΑ-6

Το σύστημα πρέπει να λειτουργεί μόνο όταν ανιχνεύει πρόσωπο ανθρώπου.

Περιγραφή: Το σύστημα πρέπει να ανιχνεύει την ύπαρξη ανθρώπου μπροστά στον καθρέφτη και να σταματά τη λειτουργία του όταν δεν βρίσκει τίποτα.

User Priority (4/5): Η απαίτηση αυτή είναι σημαντική για τον χρήστη, καθώς δεν επιθυμεί ο καθρέφτης να ανιχνεύει τα λόγια του σε ανύποπτο χρόνο και να πιάνει εντολές χωρίς πρόθεση.

Technical Priority (5/5): Η απαίτηση αυτήν είναι πολύ σημαντική για το σύστημα, καθώς θα πρέπει να ελέγχει συνεχώς την ύπαρξη χρήστη μπροστά στον καθρέφτη και να σταματά τη λειτουργία του. Η απαίτηση αυτή θα βοηθήσει και στην

ΚΕΦΑΛΑΙΟ 4. ΥΛΟΠΟΙΗΣΗ ΛΕΙΤΟΥΡΓΙΚΟΥ ΚΑΘΡΕΦΤΗ

εξοικονόμηση πόρων του συστήματος.

ΜΛΑ-7

Το σύστημα πρέπει να συνδέεται στο διαδίκτυο με ικανοποιητικό εύρος σύνδεσης.

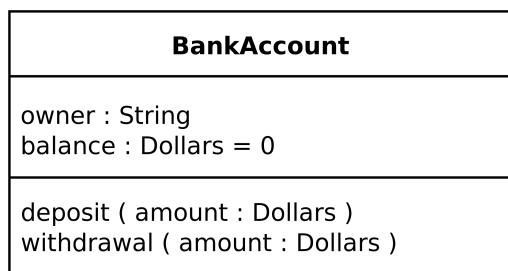
Περιγραφή: Το σύστημα πρέπει να υποστηρίζει σύνδεση στο διαδίκτυο προκειμένου να μπορεί να αλληλεπιδρά με τα διαφορετικά APIs και να λειτουργεί ορθά. Επομένως πρέπει να υπάρχει η απαραίτητη υποστήριξη σύνδεσης στο διαδίκτυο από το υλικό του συστήματος.

User Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για τον χρήστη, καθώς χωρίς σύνδεση διαδικτύου ο καθρέφτης δεν θα είναι σε θέση να εκτελέσει τις εφαρμογές του και να αναγνωρίσει τις εντολές του χρήστη.

Technical Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για το σύστημα, καθώς χρειάζεται σύνδεση στο διαδίκτυο για να λειτουργήσει ο καθρέφτης τις εφαρμογές του.

4.3 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΛΑΣΕΩΝ

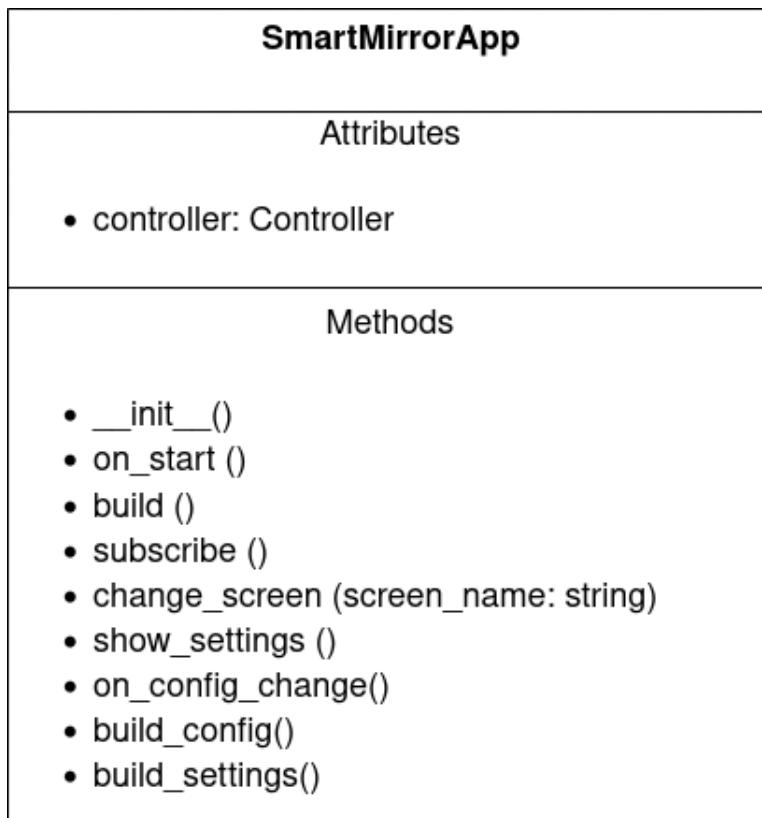
Η οργάνωση του κώδικα έγινε με την μορφή κλάσεων στη γλώσσα Python ενώ όπως έχει αναφερθεί παραπάνω για την γραφική διεπαφή χρησιμοποιήθηκε η βιβλιοθήκη Kivy. Στο παρόν κεφάλαιο θα παρουσιαστούν όλες οι κλάσεις που υλοποιήθηκαν για το λογισμικό του καθρέφτη μαζί με τις μεταβλητές και μεθόδους που ανήκουν σε κάθε κλάση, ενώ θα γίνει και μια περιγραφή του κάθε μέλους. Με τέτοιου είδους διαγράμματα είναι εφικτό να μοντελοποιηθεί το λογισμικό σε ανώτερο επίπεδο και χωρίς να χρειαστεί να κοιτάξουμε άμεσα τον κώδικα [31]. Η περιγραφή των κλάσεων θα γίνει στην γλώσσα UML²⁵ κατά την οποία, όπως φαίνεται στο Σχήμα 4.2, μια κλάση αναπαριστάται από 3 μέρη, το όνομά της, τα χαρακτηριστικά (μεταβλητές) της και τέλος οι μέθοδοί της.



Σχήμα 4.2: Παράδειγμα αναπαράστασης κλάσης στη UML

²⁵https://en.wikipedia.org/wiki/Unified_Modeling_Language

4.3.1 SmartMirrorApp



Σχήμα 4.3: UML της κλάσης SmartMirrorApp

Περιγραφή Κλάσης

Η κλάση αυτή κληρονομεί από την κλάση `kivy.app.App`²⁶ η οποία είναι η βασική κλάση για την δημιουργία μιας Kivy εφαρμογής. Μέσως αυτής της κλάσης ελέγχουμε την γραφική εφαρμογή και όταν όλα είναι έτοιμα ξεκινάμε τον κύκλο ζωής της εφαρμογής καλώντας την μέθοδο `SmartMirrorApp().run()`. **Χαρακτηριστικά Κλάσης**

- `controller`: Αντικείμενο τύπου `Controller` για την ενεργοποίηση του thread ελέγχου.

Μέθοδοι Κλάσης

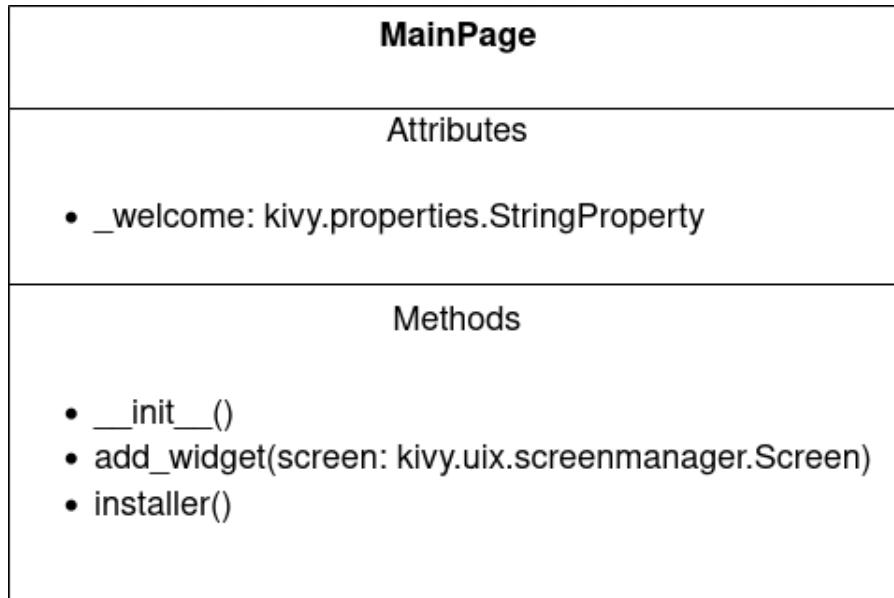
- `__init__()`: Συνάρτηση δόμησης της κλάσης η οποία καλή την συνάρτηση δόμησης της κλάσης `kivy.app.App` μέσω του `super()` και αρχικοποιεί τη μεταβλητή `controller` να είναι `None`.
- `on_start()`: Ειδική μέθοδος της κλάσης `kivy.app.App` που εκτελείται λίγο πριν την εκκίνηση της εφαρμογής (αμέσως μετά την συνάρτηση `build()`). Δημιουργεί το αντικείμενο `Controller` και ξεκινάει το thread ελέγχου.
- `build()`: Ειδική μέθοδος της κλάσης `kivy.app.App` η οποία αρχικοποιεί την

²⁶<https://kivy.org/doc/stable/api-kivy.app.html#kivy.app.App>

εφαρμογή και εκτελείται μόνο μία φορά. Φορτώνει όλες τις εγκατεστημένες εφαρμογές και επιστρέφει το βασικό Widget (ένα στιγμιότυπο της κλάσης MainPage η οποία θα εξηγηθεί παρακάτω).

- **subscribe()**: Μέθοδος η οποία επιστρέφει ένα Python dictionary όπου γίνεται μια αντιστοίχηση ενός intent σε μια συνάρτηση. Η μέθοδος αυτή χρησιμοποιείται προκειμένου να μπορεί να εκτελεστεί η σωστή συνάρτηση όταν γίνει αναγνώριση της πρόθεσης της εντολής του χρήστη από το Wit.ai.
- **change_screen(screen_name: string)**: Συνάρτηση που χρησιμοποείται για να γίνει αλλαγή Screen δηλαδή να φορτώσει μια άλλη εφαρμογή με όνομα screen_name.
- **show_settings()**: Μέθοδος η οποία καλεί την συνάρτηση open_settings της κλάσης kivy.app.App για να ανοίξει το μενού ρυθμίσεων της εφαρμογής.
- **on_config_change()**: Ειδική μέθοδος της κλάσης kivy.app.App η οποία εκτελείται όταν γίνει κάποια αλλαγή ρύθμισης. Για κάθε ένα από τα εγκατεστημένα Widgets ανανεώνει τις ρυθμίσεις τους.
- **build_config()**: Ειδική μέθοδος της κλάσης kivy.app.App η οποία εκτελείται πριν την αρχικοποίηση της εφαρμογής και δημιουργεί default ρυθμίσεις για κάθε ένα από τα εγκατεστημένα Widgets.
- **build_settings()**: Ειδική μέθοδος της κλάσης kivy.app.App η οποία εκτελείται όταν θέλουμε να δείξουμε τις ρυθμίσεις της εφαρμογής. Διαβάζει από κάθε εγκατεστημένο Widget τις ρυθμίσεις τους (οι οποίες βρίσκονται σε ένα αρχείο "settings.json" και τις προσθέτει στο μενού ρυθμίσεων.

4.3.2 MainPage



Σχήμα 4.4: UML της κλάσης MainPage

Περιγραφή Κλάσης

Η κλάση αυτή κληρονομεί από την κλάση ScreenManager²⁷ που περιέχεται στο πακέτο `kivy.uix.screenmanager`. Η κλάση αυτή είναι το πρωτεύον Widget που επιστρέφει η εφαρμογή και κάτω από αυτήν μπαίνουν όλες οι εφαρμογές που εγκαθιστά ο χρήστης (σαν καινούργια Screens).

Χαρακτηριστικά Κλάσης

- `_welcome`: Μεταβλητή που χρησιμοποιείται για να αποθηκεύσει ένα μήνυμα που καλωσορίζει τον χρήστη.

Μέθοδοι Κλάσης

- `__init__()`: Η συνάρτηση δόμησης της κλάσης η οποία καλή την συνάρτηση δόμησης της κλάσης `kivy.uix.screenmanager.ScreenManager` μέσω του `super` και αρχικοποιεί τη μεταβλητή `_welcome` να πάρει μία τυχαία επιλογή από μια λίστα μηνυμάτων. Τέλος καλεί την εσωτερική συνάρτηση `installer` προκειμένου να εγκαταστήσει όλες τις εφαρμογές που βρίσκονται στον φάκελο `/widgets` σαν ξεχωριστά Screens.
- `add_widget(screen: kivy.uix.screenmanager.Screen)`: Ειδική συνάρτηση της κλάσης `kivy.uix.screenmanager.ScreenManager` η οποία προσθέτει ένα καινούργιο Widget στα παιδιά του τρέχοντος Widget. Στην υπερφορτωμένη της έκδοση, επιτρέπεται η πρόσθεση μόνο ενός Widget τύπου `kivy.uix.screenmanager.Screen` για περισσότερη ασφάλεια.

²⁷<https://kivy.org/doc/stable/api-kivy.uix.screenmanager.html#kivy.uix.screenmanager.ScreenManager>

- `installer()`: Συνάρτηση η οποία για κάθε ένα από τα παιδιά της `MainPage` καλεί την συνάρτηση `install` την οποία πρέπει να ορίσουν οι εξωτερικές εφαρμογές προκειμένου να μπορούν να εγκατασταθούν από τον καθρέφτη.

4.3.3 Controller

Controller
Attributes
<ul style="list-style-type: none"> • <code>config: configparser.ConfigParser</code> • <code>_launch_phrase: string</code> • <code>_close_phrase: string</code> • <code>_s: Speech</code> • <code>_gui: SmartMirrorApp</code> • <code>_widgets: dictionary</code> • <code>_bot: Bot</code> • <code>_action: Action</code> • <code>_running: threading.Event</code> • <code>daemon: boolean</code>
Methods
<ul style="list-style-type: none"> • <code>__init__()</code> • <code>_check_launch_phrase(text: string)</code> • <code>_check_close_phrase(text: string)</code> • <code>_check_close(text: string)</code> • <code>pause()</code> • <code>resume()</code> • <code>run()</code> • <code>_authenticate_mode()</code> • <code>_command_mode()</code>

Σχήμα 4.5: UML της κλάσης Controller

Περιγραφή Κλάσης

Η κλάση αυτή κληρονομεί από την κλάση `threading.Thread`²⁸. Ο controller είναι ένα ανεξάρτητο thread από το main thread στο οποίο τρέχει η γραφική εφαρμογή και είναι υπεύθυνος για τον έλεγχο των λειτουργιών του συστήματος. Μέσα από αυτήν την κλάση ελέγχεται η ροή του προγράμματος, επιλέγονται οι ενέργειες που θα εκτελέσει ο καθρέφτης και ανανεώνεται το γραφικό περιβάλλον με βάση τις εντολές του χρήστη.

Χαρακτηριστικά Κλάσης

- `config`: Αντικείμενο τύπου `configparser.ConfigParser` για την ανάγνωση ρυθμίσεων.
- `_launch_phrase`: Μεταβλητή για την αποθήκευση της φράσης ενεργοποίησης του καθρέφτη.
- `_close_phrase`: Μεταβλητή για την αποθήκευση της φράσης απενεργοποίησης του καθρέφτη.
- `_s`: Αντικείμενο τύπου `Speech` το οποίο χρησιμοποιείται για την αναγνώριση της φωνής του χρήστη.
- `_gui`: Αντικείμενο τύπου `SmartMirrorApp` το οποίο χρησιμοποιείται για να γίνεται αλληλεπίδραση μεταξύ του Controller και του γραφικού περιβάλλοντος.
- `_widgets`: `Dictionary` το οποίο αποθηκεύει όλα τα Widgets του γραφικού περιβάλοντος με τα ids τους ως κλειδιά.
- `_bot`: Αντικείμενο τύπου `Bot` το οποίο χρησιμοποιείται για την αλληλεπίδραση με το Wit.ai.
- `_action`: Αντικείμενο τύπου `Action` το οποίο χρησιμοποιείται ως βοηθημα για την εκτέλεση της εντολής του χρήστη.
- `_running`: Μεταβλητή τύπου `threading.Event` η οποία χρησιμοποιείται για τον έλεγχο του Thread του Controller.
- `daemon`: Μεταβλητή που καθορίζει αν το Thread θα εκτελεστεί στο προσκήνιο ή στο παρασκήνιο.

Μέθοδοι Κλάσης

- `__init__()`: Η συνάρτηση δόμησης της κλάσης διαβάζει από τις ρυθμίσεις τις φράσεις ενεργοποίησης/απενεργοποίησης, αρχικοποιεί τα αντικείμενα των βοηθητικών κλάσεων που θα χρησιμοποιηθούν για τον έλεγχο του καθρέφτη και τέλος ξεκινάει ένα Thread στο παρασκήνιο στο οποίο θα γίνεται ο λογικός έλεγχος της ροής του προγράμματος.
- `_check_launch_phrase(text: String)`: Βοηθητική συνάρτηση όπου επιστρέφει True αν η μεταβλητή `text` ταυτίζεται με την φράση έναρξης και False σε αντίθετη περίπτωση.

²⁸<https://docs.python.org/3/library/threading.html#threading.Thread>

- `_check_close_phrase(text: String)`: Βοηθητική συνάρτηση όπου επιστρέφει `True` αν η μεταβλητή `text` ταυτίζεται με την φράση απενεργοποίησης και `False` σε αντίθετη περίπτωση.
- `_check_close(text: String)`: Βοηθητική συνάρτηση που ελέγχει εαν η μεταβλητή `text` είναι ίση με `"quit"` ή `"exit"` προκειμένου να διακόψει τη λειτουργία του καθρέφτη.
- `pause()`: Βοηθητική συνάρτηση η οποία σταματάει το Thread του Controller.
- `resume()`: Βοηθητική συνάρτηση η οποία ενεργοποιεί το Thread του Controller.
- `run()`: Η συνάρτηση αυτή ενεργοποιεί το Thread του Controller και καλεί τις συναρτήσεις `_authenticate_mode()` και `_command_mode()` η οποίες ελέγχουν την κατάσταση του καθρέφτη.
- `_authenticate_mode()`: Στη συγκεκριμένη κατάσταση ο καθρέφτης είναι αδρανής και περιμένει να ακούσει την φράση ενεργοποίησης προκειμένου να δώσει πρόσβαση στον χρήστη και να ακολουθεί τις εντολές του. Με αυτόν τον τρόπο ο καθρέφτης θα ακούει τις εντολές του χρήστη μόνο όταν αυτός το θέλει.
- `_command_mode()`: Στη συγκεκριμένη κατάσταση γίνεται ο έλεγχος της ροής του καθρέφτη όπου γίνεται επανάληπτικά η ακοή της εντολής του χρήστη, η αλληλεπίδραση με το Wit.ai για την αναγνώριση της πρόθεσης του χρήστη και η εκτέλεση της εντολής αν αυτή είναι αποδεκτή. Η επανάληψη συνεχίζεται έως ότου ο χρήστης πει την φράση απενεργοποίησης ή μία εκ των εντολών `"quit"` και `"exit"` οι οποίες απενεργοποιούν τον καθρέφτη.

4.3.4 Speech

Speech
Attributes
<ul style="list-style-type: none"> • <code>_r: speech_recognition.Recognizer</code> • <code>_audio: speech_recognition.AudioData</code> • <code>_text: string</code> • <code>_label: kivy.uix.label.Label</code>
Methods
<ul style="list-style-type: none"> • <code>__init__(label: kivy.uix.label.Label)</code> • <code>get_text()</code> • <code>_listen_for_audio()</code> • <code>_speech_to_text()</code> • <code>listen()</code> • <code>speak_back(text: string)</code>

Σχήμα 4.6: UML της κλάσης Speech

Περιγραφή Κλάσης

Η κλάση αυτή υλοποιεί όλες τις μεθόδους και την διεπαφή για την ανάγνωση και ομιλία ήχου. Μέσω αυτής της κλάσης ο καθρέφτης μπορεί να ακούει τον χρήστη από το μικρόφωνο ανά τακτά διαστήματα και να μετατρέπει τις εντολές του σε κείμενο το οποίο θα χρησιμοποιηθεί αργότερα για την αναγνώριση της πρόθεσης. Τέλος, μέσω αυτής της κλάσης ο καθρέφτης δύναται να μιλήσει στον χρήστη, λέγοντάς του αποτελέσματα ενεργειών ή πληροφορίες.

Χαρακτηριστικά Κλάσης

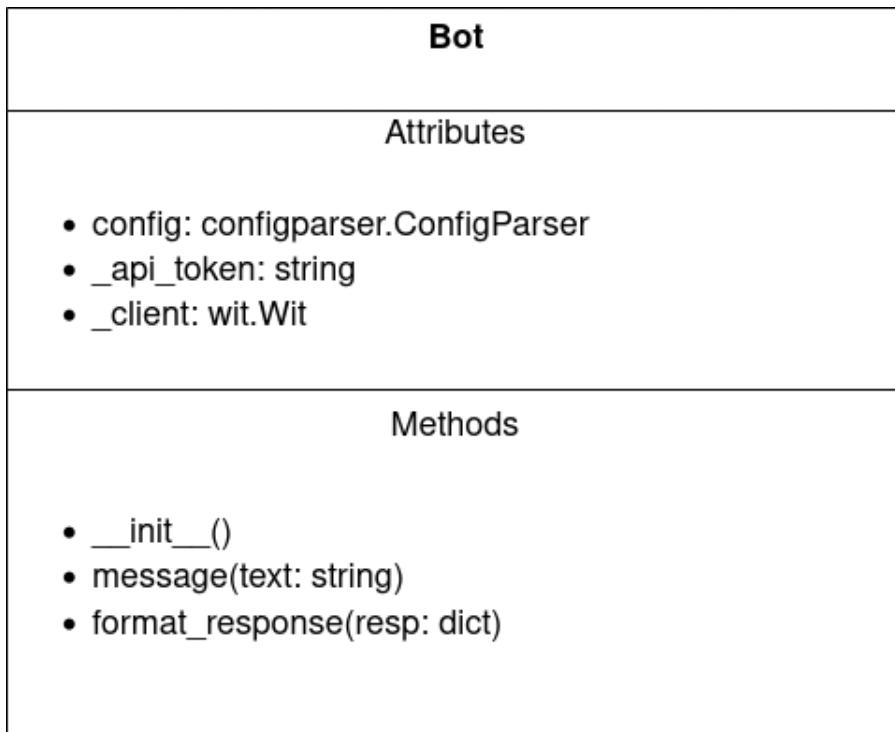
- `_r`: Αντικείμενο τύπου `speech_recognition.Recognizer` μέσω του οποίο γίνεται η ανάγνωση της φωνής και η αναγνώρισή της σε κείμενο.
- `_audio`: Αντικείμενο τύπου `speech_recognition.AudioData` το οποίο αποθηκεύει τα δεδομένα του ήχου τα οποία χρησιμοποιούνται για την αναγνώριση και μετατροπή σε κείμενο.
- `_text`: Μεταβλητή στην οποία αποθηκεύεται σε γραπτή μορφή η εντολή του χρήστη μετά την αναγνώριση.

- `_label`: Αντικείμενο τύπου `kivy.uix.Label` το οποίο χρησιμοποιείται προκειμένου να εμφανίζεται στην οθόνη του καθρέφτη πότε το σύστημα ακούει εντολές από τον χρήστη.

Μέθοδοι Κλάσης

- `__init__(kivy.uix.Label)`: Συνάρτηση δόμησης της κλάσης η οποία αρχικοποιεί τις μεταβλητές που χρησιμποιεί η κλάση και δέχεται ως είσοδο το Label στο οποίο θα υποδηλώνεται πότε ο καθρέφτης ακούει για εντολές.
- `get_text()`: Βοηθητική συνάρτηση η οποία επιστρέφει την τιμή της μεταβλητής `_text`.
- `_listen_for_audio()`: Βοηθητική συνάρτηση η οποία ακούει από το μικρόφωνο τις εντολές του χρήστη. Η λειτουργία ανάγνωσης του ήχου από το μικρόφωνο του συστήματος γίνεται με την αρχικοποίηση ενός αντικειμένου τύπου `speech_recognition.Microphone`. Τα δεδομένα ήχου αποθηκεύονται στο αντικείμενο `_audio` ενώ γίνεται και εγγραφή στο Label του καθρέφτη για την ενημέρωση του χρήστη ότι το σύστημα περιμένει εντολή.
- `_speech_to_text()`: Συνάρτηση που καλεί την μέθοδο `recognize_google()` του αντικειμένου `_r` η οποία αλληλεπιδρά με το API της Google για την μετατροπή των δεδομένων ήχου σε κείμενο. Σε περίπτωση που δεν γίνεται αναγνώριση το κείμενο τίθεται κενό.
- `listen()`: Συνάρτηση η οποία καλείται από τον Controller και καλεί τις 2 συναρτήσεις για ανάγνωση του ήχου και μετατροπή σε κείμενο.
- `speak_back(text: string)`: Συνάρτηση η οποία καλείται εξωτερικά για την ομιλία του καθρέφτη προς τον χρήστη. Μέσω της βιβλιοθήκης gTTS μετατρέπεται το κείμενο εισόδου σε ομιλία, αποθηκεύεται ως ένα προσωρινό αρχείο .mp3 και αναπαράγεται μέσω της βιβλιοθήκης pydub. Τέλος, το προσωρινό αρχείο διαγράφεται.

4.3.5 Bot



Σχήμα 4.7: UML της κλάσης Bot

Περιγραφή Κλάσης

Η κλάση αυτή αποτελεί την διεπαφή για την αλληλεπίδραση με το Wit.ai και την εξαγωγή της πρόθεσης της εντολής του χρήστη αλλά και των οντοτήτων οι οποίες θα δοθούν ως ορίσματα κατά την εκτέλεση των εντολών.

Χαρακτηριστικά Κλάσης

- **config:** Αντικείμενο τύπου `configparser.ConfigParser` για την ανάγνωση ρυθμίσεων.
- **_api_token:** Μεταβλητή η οποία αποθηκεύει το `api_key` που χρησιμοποιείται για να αποκτήσει η εφαρμογή πρόσβαση στο Wit.ai.
- **_client:** Αντικείμενο τύπου `wit.Wit` μέσω του οποίου θα γίνεται η αλληλεπίδραση με το Wit.ai.

Μέθοδοι Κλάσης

- **`__init__()`:** Συνάρτηση δόμησης της κλάσης η οποία διαβάζει το κλειδί από τις ρυθμίσεις και αρχικοποιεί ένα αντικείμενο τύπου `wit.Wit`.
- **`message(text: string)`:** Η συνάρτηση που καλείται από τον Controller, στέλνει την εντολή σε μορφή κειμένου στο Wit.ai και επιστρέφει την πρόθεση και τις οντότητες που εξάγονται. Σε περίπτωση που δεν αναγνωριστεί η εντολή γίνεται κατάλληλος έλεγχος σφάλματος.

- `format_response(resp: dict)`: Βοηθητική συνάρτηση η οποία δέχεται ως είσοδο το αποτέλεσμα του Wit.ai και μορφοποιεί την πρόθεση και τις οντότητες στην επιθυμητή μορφή ενός dictionary.

4.3.6 Action

Action
<p>Attributes</p> <ul style="list-style-type: none"> • <code>_gui: SmartMirrorApp</code> • <code>_s: speech.Speech</code> • <code>_widgets: list</code> • <code>_commands: dict</code> • <code>_current_screen: kivy.uix.screenmanager.Screen</code>
<p>Methods</p> <ul style="list-style-type: none"> • <code>__init__(gui: SmartMirrorApp)</code> • <code>_update_commands()</code> • <code>_get_screen(name: string)</code> • <code>_extract_widgets(screen: kivy.uix.screenmanager.Screen)</code> • <code>_extract_functions()</code> • <code>perform(resp: dict)</code> • <code>_config_change(conf: list)</code>

Σχήμα 4.8: UML της κλάσης Action

Περιγραφή Κλάσης

Η κλάση αυτή αποτελεί την διεπαφή για την εκτέλεση των εντολών του χρήστη και είναι στην ουσία μια επέκταση του Controller. Μέσω της συγκεκριμένης κλάσης, γίνεται η μετατροπή της πρόθεσης που παίρνει το σύστημα από το Wit.ai σε εκτέλεση της κατάλληλης πράξης που υποδηλώνεται από την πρόθεση (αν αυτή υπάρχει και είναι προσβάσιμη).

Χαρακτηριστικά Κλάσης

- `_gui`: Αντικείμενο τύπου `SmartMirrorApp` το οποίο χρησιμοποιείται για να γίνεται αλληλεπίδραση μεταξύ του `Controller` και του γραφικού περιβάλλοντος.
- `_s`: Αντικείμενο τύπου `Speech` το οποίο χρησιμοποιείται για την ομιλία του καθρέφτη προς τον χρήστη.
- `_widgets`: Μία λίστα που αποθηκεύει όλα τα `Widgets` που βρίσκονται στην τωρινή οθόνη και το αντικείμενο `_gui`.
- `_commands`: Ένα `dictionary` που αποθηκεύει τις δυνατές εντολές που εξάγει κάθε `Widget` ως κλειδιά μαζί με τις αντίστοιχες συναρτήσεις που θα κληθούν για την κάθε εντολή ως τιμές.
- `_current_screen`: Αντικείμενο τύπου `kivy.uix.screenmanager.Screen` το οποίο έχει την τιμή της οθόνης στην οποία βρίσκεται κάθε στιγμή ο καθρέφτης.

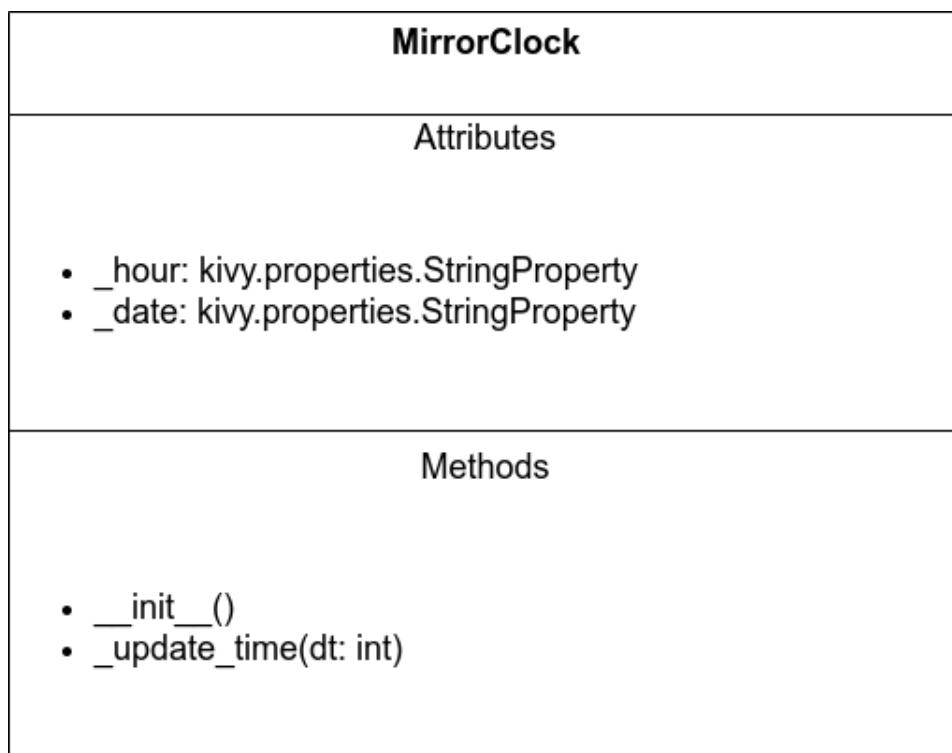
Μέθοδοι Κλάσης

- `__init__(gui: SmartMirrorApp)`: Συνάρτηση δόμησης της κλάσης η οποία δέχεται ως είσοδο ένα αντικείμενο τύπου `SmartMirrorApp` για τον έλεγχο της γραφικής εφαρμογής, αρχικοποιεί ένα αντικείμενο τύπου `Speech` για την διαχείριση της φωνής του καθρέφτη και τέλος καλεί την βοηθητική συνάρτηση `_update_commands` για να αρχικοποιηθούν οι διαθέσιμες εντολές στην τωρινή κατάσταση του καθρέφτη.
- `_update_commands()`: Βοηθητική συνάρτηση η οποία καθορίζει τις διαθέσιμες εντολές που μπορεί να εκτελέσει ο καθρέφτης. Αρχικά εξάγεται η τωρινή οθόνη, στην συνέχεια εξάγονται όλα τα `Widgets` που είναι διαθέσιμα στην οθόνη και διαθέτουν την μέθοδο `subscribe` ενώ πάντα μετά προστίθεται το αντικείμενο της εφαρμογής προκειμένου να είναι διαθέσιμες οι κάποιες βασικές εντολές, ενώ τέλος αποθηκεύονται οι συναρτήσεις που εκθέτουν τα παραπάνω `Widgets` μέσω της συνάρτησης `subscribe`.
- `_get_screen(name: string)`: Βοηθητική συνάρτηση η οποία δέχεται το όνομα μιας οθόνης και επιστρέφει ένα αντικείμενο `authtic`.
- `_extract_widgets(screen: kivy.uix.screenmanager.Screen)`: Βοηθητική συνάρτηση η οποία δέχεται ένα αντικείμενο τύπου `Screen` και επιστρέφει μια λίστα με όλα τα `Widgets` που βρίσκονται στην οθόνη και διαθέτουν συνάρτηση `subscribe`.
- `_extract_functions()`: Βοηθητική συνάρτηση που ελέγχει όλα τα διαθέσιμα `widgets` και διαβάζει τις εντολές που εκθέτουν μέσω της συνάρτησης `subscribe`. Οι εντολές έχουν ως κλειδί την πρόθεση που θα έχει η εντολή και το αντικείμενο της συνάρτησης που θα κληθεί ως τιμή.
- `perform(resp: dict)`: Συνάρτηση που δέχεται ως είσοδο ένα `dictionary` με την πρόθεση και τα ορίσματα από το `Wit` και εκτελεί την εντολή. Η πρόθεση ελέγχεται αν βρίσκεται στα κλειδιά της μεταβλητής `_commands` και εκτελεί την συνάρτηση στην οποία δείχνει το συγκεκριμένο κλειδί. Εαν η συνάρτηση επιστρέφει μία από τις τιμές "config", "update" ή "speech" τότε ο καθρέφτης

ανανεώνει τις ρυθμίσεις ή ανανεώνει τις διαθέσιμες εντολές (συνήθως όταν γίνεται αλλαγή οθόνης) ή ο καθρέφτης μιλάει στον χρήστη.

- `_config_change(conf: list)`: Βοηθητική συνάρτηση η οποία καλείται όταν κάποια εντολή του χρήστη θα αλλάξει τις ρυθμίσεις του καθρέφτη. Δέχεται μια ως είσοδο μια λίστα η οποία καθορίζει τις νέες ρυθμίσεις.

4.3.7 MirrorClock



Σχήμα 4.9: UML της κλάσης MirrorClock

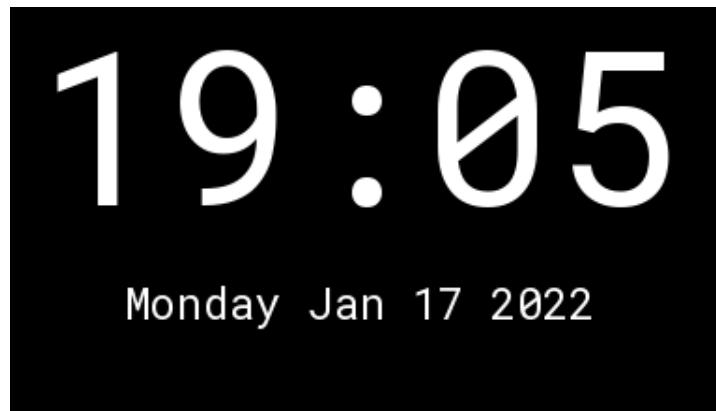
Περιγραφή Κλάσης

Η κλάση αυτή κληρονομεί από την κλάση Widget²⁹ που περιέχεται στο πακέτο `kivy.uix.widget` και υλοποιεί ένα ρολόι και ημερολόγιο στην αρχική οθόνη του καθρέφτη. Στο Σχήμα 4.10 φαίνεται γραφικά το συγκεκριμένο Widget.

Χαρακτηριστικά Κλάσης

- `_hour`: Μεταβλητή τύπου `kivy.properties.StringProperty` η οποία αποθηκεύει την ώρα σε μορφή "HH:MM".
- `_date`: Μεταβλητή τύπου `kivy.properties.StringProperty` η οποία αποθηκεύει την ημερομηνία σε μορφή "Μέρα εβδομάδας, Μήνας, Μέρα του μήνα, Χρόνος".

²⁹<https://kivy.org/doc/stable/api-kivy.uix.widget.html>



Σχήμα 4.10: Γραφική αναπαράσταση του Widget MirrorClock

Μέθοδοι Κλάσης

- `__init__()`: Συνάρτηση δόμησης της κλάσης η οποία καλή την συνάρτηση δόμησης της κλάσης `kivy.uix.widget.Widget` μέσω του `super` και καλεί την συνάρτηση `schedule_interval`³⁰ της κλάσης `kivy.clock.Clock` με χρόνο 1 sec η οποία καλεί με την σειρά της περιοδικά την συνάρτηση για την ανανέωση του χρόνου. Με αυτόν τον τρόπο η ώρα ανανεώνεται κάθε δευτερόλεπτο.
- `_update_time(dt: int)`: Συνάρτηση η οποία διαβάζει την ώρα μέσω της standard βιβλιοθήκης `datetime`³¹ της Python και θέτει τις δύο μεταβλητές `_hour` και `_date` στις κατάλληλες τιμές.

³⁰https://kivy.org/doc/stable/api-kivy.clock.html#kivy.clock.CyClockBase.schedule_interval

³¹<https://docs.python.org/3/library/datetime.html>

4.3.8 Weather

Weather	
Attributes	
<ul style="list-style-type: none">• <code>_temperature: kivy.properties.StringProperty</code>• <code>_icon: kivy.properties.StringProperty</code>• <code>_description: kivy.properties.StringProperty</code>• <code>_api_key: kivy.properties.ConfigParserProperty</code>• <code>_city_id: kivy.properties.ConfigParserProperty</code>• <code>_city_name: kivy.properties.ConfigParserProperty</code>• <code>_update_interval: kivy.properties.ConfigParserProperty</code>• <code>_config: kivy.config.ConfigParser</code>• <code>_lon: int</code>• <code>_lat: int</code>	
Methods	
<ul style="list-style-type: none">• <code>__init__()</code>• <code>on_kv_post(dt: int)</code>• <code>update_config()</code>• <code>_get_weather(dt)</code>• <code>_diff_of_dates(s_date: string)</code>• <code>_diff_of_hours(self, date: string)</code>• <code>request_day(datetime: string)</code>• <code>request_hour(datetime: string)</code>• <code>request_location(location: string)</code>• <code>subscribe()</code>	

Σχήμα 4.11: UML της κλάσης Weather

Περιγραφή Κλάσης

Η κλάση αυτή κληρονομεί από την κλάση Widget³² που περιέχεται στο πακέτο `kivy.uix.widget` και υλοποιεί ένα παράθυρο που δείχνει τον καιρό στην αρχική οθόνη του καθρέφτη αλλά και διάφορες συναρτήσεις για την πληροφόρηση σχετικά με τις καιρικές συνθήκες. Στο Σχήμα 4.12 φαίνεται γραφικά το συγκεκριμένο Widget.

³²<https://kivy.org/doc/stable/api-kivy.uix.widget.html>



Σχήμα 4.12: Γραφική αναπαράσταση του Widget Weather

Χαρακτηριστικά Κλάσης

- `_temperature`: Μεταβλητή που αποθηκεύει την τιμή της θερμοκρασίας για να εμφανιστεί στο Widget.
- `_icon`: Μεταβλητή που αποθηκεύει το URL της εικόνας που αντιστοιχεί στον καιρό.
- `_description`: Μεταβλητή που αποθηκεύει μια πρόταση η οποία περιγράφει τον καιρό.
- `_api_key`: Μεταβλητή τύπου `kivy.properties.ConfigParserProperty`³³ η οποία διαβάζει από τις ρυθμίσει το κλειδί πρόσβασης για την αλληλεπίδραση με το OpenWeather³⁴ API.
- `_city_id`: Μεταβλητή τύπου `kivy.properties.ConfigParserProperty` η οποία διαβάζει το id της πόλης που επιθυμεί ο χρήστης να ενημερωθεί για τον καιρό από τις ρυθμίσεις.
- `_city_name`: Μεταβλητή τύπου `kivy.properties.ConfigParserProperty` η οποία διαβάζει το όνομα της πόλης που επιθυμεί ο χρήστης να ενημερωθεί για τον καιρό από τις ρυθμίσεις.
- `_update_interval`: Μεταβλητή τύπου `kivy.properties.ConfigParserProperty` η οποία διαβάζει από τις ρυθμίσεις τη χρονική περίοδο στην οποία επιθυμεί ο χρήστης να γίνεται ανανέωση του καιρού.
- `_update_interval`: Μεταβλητή τύπου `kivy.config.ConfigParser` η οποία χρησιμοποιείται για την ανάγνωση των ρυθμίσεων του καθρέφτη.
- `_lon`: Μεταβλητή τύπου `int` που αποθηκεύει το γεωγραφικό μήκος της πόλης για την οποία επιθυμεί ο χρήστης να ενημερωθεί για τον καιρό.
- `_lat`: Μεταβλητή τύπου `int` που αποθηκεύει το γεωγραφικό πλάτος της πόλης για την οποία επιθυμεί ο χρήστης να ενημερωθεί για τον καιρό.

³³<https://kivy.org/doc/stable/api-kivy.properties.html#kivy.properties.ConfigParserProperty>

³⁴<https://openweathermap.org/>

Μέθοδοι Κλάσης

- `__init__()`: Συνάρτηση δόμησης της κλάσης η οποία αρχικοποιεί το Widget.
- `on_kivy_post(dt: int)`: Ειδική συνάρτηση η οποία εκτελείται όταν το συγκεκριμένο Widget εμφανιστεί. Καλεί την συνάρτηση `update_config` και αρχικοποιεί ένα ρολόι το οποία καλεί την συνάρτηση `_get_weather` περιοδικά ανάλογα την τιμή της μεταβλητής `_update_interval`.
- `update_config()`: Συνάρτηση η οποία διαβάζει τις ρυθμίσεις και ανανεώνει κατάλληλα τις μεταβλητές μέλη της κλάσης και καλεί τις συνάρτησεις `_city_to_coord` και `_get_weather`.
- `_get_weather(dt: int)`: Συνάρτηση η οποία αλληλεπιδρά με το API της OpenWeather και ανανεώνει τα στοιχεία του Widget ώστε να δείχνουν την τελευταία ανάγνωση του καιρού.
- `_city_to_coord()`: Συνάρτηση η οποία διαβάζει ένα αρχείο με τις τοποθεσίες κάθε πόλης και επιστρέφει την τοποθεσία της πόλης που είναι αποθηκευμένη στην μεταβλητή `_city_name`.
- `_diff_of_dates(s_date: string)`: Συνάρτηση η οποία δέχεται μια ημερομηνία και υπολογίζει την διαφορά σε μέρες από την σημερινή.
- `_diff_of_hours(date: string)`: Συνάρτηση η οποία δέχεται μια ημερομηνία και υπολογίζει την διαφορά σε ώρες από την τωρινή ώρα.
- `request_date(datetime: string)`: Συνάρτηση η οποία δέχεται μια ημερομηνία ως όρισμα και λέει στον χρήστη τον καιρό για την εισαγώμενη μέρα.
- `request_hour(datetime: string)`: Συνάρτηση η οποία δέχεται μια ημερομηνία ως όρισμα και λέει στον χρήστη τον καιρό που θα κάνει την ζητούμενη ώρα.
- `request_location(location: string)`: Συνάρτηση η οποία δέχεται μια τοποθεσία, ενημερώνει τον χρήστη για τον καιρό στην συγκεκριμένη τοποθεσία και ενημερώνει τις ρυθμίσεις του καθρέφτει να δείχνει πλέον την εισαγώμενη τοποθεσία.
- `subscribe()`: Συνάρτηση η οποία εκθέτει στον Controller τις εντολές που δέχεται το Widget και ποιες συναρτήσεις θα εκτελεστούν.

5

Υλοποίηση Εκτίμησης Πόζας

Η εφαρμογή της εκτίμησης πόζας, ονομαζόμενη *Exercisor*, του έξυπνου καθρέφτη αποτελεί μία εφαρμογή εξατομικευμένης άσκησης και ευεξίας. Πρωταρχικός σκοπός της είναι να αποτελέσει έναν καθημερινό αρωγό και παρακινητή του χρήστη στην εκτέλεση ασκήσεων οποιασδήποτε μορφής. Κατά την λειτουργία παιχνιδιού η εφαρμογή αναπαράγει μία επιλεγμένη από τον χρήστη άσκηση αναφοράς ενώ ταυτόχρονα γίνεται εκτίμηση της πόζας του. Τοιουτοτρόπως, προβάλλονται παράλληλα η άσκηση αναφοράς και η εκτίμηση πόζας του χρήστη, απεικονίζοντας την απόκλιση των δύο με χρήση διορθωτικών βελών. Με το πέρας κάθε επανάληψης της άσκησης υπολογίζεται μία βαθμολογία με βάση την απόκλιση της εκτέλεσης του χρήστη από την άσκηση αναφοράς.

Εκτός από την λειτουργία παιχνιδιού, η εφαρμογή διαθέτει και την λειτουργία επιμελητή ασκήσεων αναφοράς. Στην λειτουργία επιμελητή ο χρήστης μπορεί να εισάγει ένα βίντεο άσκησης, με βάση το οποίο γίνεται εκτίμηση πόζας και τα δεδομένα της εκτίμησης αποθηκεύονται για χρήση ως δεδομένων άσκησης αναφοράς. Επιπλέον, μέσω του επιμελητή ο χρήστης μπορεί να αναπαράγει τις ασκήσεις αναφοράς και να τις επεξεργαστεί ώστε να μειώσει τα σφάλματα εκτίμησης πόζας ή να τις εξατομικεύσει στις ανάγκες του.

Στη συνέχεια του κεφαλαίου, παρουσιάζονται αναλυτικά οι απαιτήσεις συστήματος της εφαρμογής *Exercisor* του έξυπνου καθρέφτη, αποσαφηνίζοντας τις λειτουργίες του και τα ποιοτικά του χαρακτηριστικά. Έπειτα, παρουσιάζονται οι κλάσεις και οι συναρτήσεις που υλοποιούν τις αναφερθείσες λειτουργίες.

5.1 ΑΠΑΙΤΗΣΕΙΣ ΣΥΣΤΗΜΑΤΟΣ

ΑΑ-1

Το σύστημα πρέπει να μπορεί να εκτιμάει την ανθρώπινη πόζα.

Περιγραφή: Το σύστημα πρέπει να κάνει προβλέψεις για την θέση των σημείων κλειδιών του ανθρώπινου σώματος έχοντας ως είσοδο τις εικόνες καρέ από την κάμερα του χρήστη ή από βίντεο εισόδου.

User Priority (5/5): Η απαίτηση εκτίμησης πόζας είναι απαραίτητη για τον χρήστη καθώς όλες οι λειτουργίες χρησιμες για αυτόν βασίζονται στην δυνατότητα εκτίμησης πόζας.

Technical Priority (5/5): Η απαίτηση εκτίμησης πόζας είναι απαραίτητη για την εφαρμογή καθώς αποτελεί τον ακρογωνιαίο λίθο της λειτουργίας της.

ΑΑ-2

Ο χρήστης πρέπει να μπορεί να εισάγει ορθές ασκήσεις.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να ανεβάσει ένα βίντεο με μία επανάληψη άσκησης η οποία χρησιμοποιείται ως ορθή άσκηση αναφοράς. Τότε, η εφαρμογή εκτιμάει την ανθρώπινη πόζα από το βίντεο και αποθηκεύει τα εξαγόμενα δεδομένα για χρήση κατά τη λειτουργία παιχνιδιού.

User Priority (3/5): Η απαίτηση αυτή δεν είναι απαραίτητη για τον χρήστη, καθώς ικανοποιεί χρήστες που επιθυμούν την εξατομίκευση των ασκήσεων τους. Ο απλός χρήστης της εφαρμογής μπορεί να χρησιμοποιεί τα ήδη διαθέσιμα δεδομένα ορθών ασκήσεων, χωρίς να εισάγει ποτέ δικά του.

Technical Priority (5/5): Η απαίτηση της εισαγωγής ορθών ασκήσεων είναι άκρως απαραίτητη για την λειτουργία της εφαρμογής καθώς τα δεδομένα ορθής άσκησης χρησιμοποιούνται ως αναφορά για την λειτουργία παιχνιδιού της εφαρμογής.

ΑΑ-3

Το σύστημα πρέπει να μπορεί να συγκρίνει την εκτέλεση άσκησης του χρήστη με κάποια επιλεγμένη άσκηση αναφοράς.

Περιγραφή: Το σύστημα πρέπει να μπορεί να εκτιμάει την πόζα του χρήστη σε κάθε καρέ και να την συγκρίνει με την πόζα που θα έπρεπε να έχει για αυτό το καρέ με βάση την επιλεγμένη άσκηση αναφοράς.

User Priority (5/5): Η απαίτηση σύγκρισης της πόζας του χρήστη με την πόζα αναφοράς είναι απαραίτητη για τον χρήστη καθώς η διαδικασία αυτή βρίσκεται στο επίκεντρο της λειτουργίας παιχνιδιού η οποία αποτελεί τον βασικό κόμβο σταθμό για τον χρήστη.

Technical Priority (4/5): Η απαίτηση σύγκρισης πόζας είναι απαραίτητη για την λειτουργία παιχνιδιού και εξ ακολούθως άκρως σημαντική για την εφαρμογή.

ΛΑ-4

Το σύστημα πρέπει να προβάλει ταυτόχρονα τις εκτιμήσεις πόζας του χρήστη και της άσκησης αναφοράς.

Περιγραφή: Το σύστημα πρέπει σε κάθε καρέ να εμφανίζει στην οθόνη ταυτόχρονα τις εκτιμήσεις πόζας του χρήστη και της άσκησης αναφοράς.

User Priority (5/5): Η απαίτηση ταυτόχρονης προβολής είναι απαραίτητη για τον χρήστη καθώς με αυτό τον τρόπο μπορεί να δει τη πόζα αναφοράς που θα έπρεπε να έχει και την πόζα που έχει στην πραγματικότητα.

Technical Priority (2/5): Η απαίτηση αυτή δεν είναι απαραίτητη για την λειτουργία της εφαρμογής, άλλα αποτελεί προϋπόθεση άλλων απαιτήσεων, σημαντικών για την εκπλήρωση των στόχων της εφαρμογής.

ΛΑ-5

Το σύστημα πρέπει να παρέχει οπτική ανατροφοδότηση στον χρήστη κατά την εκτέλεση της άσκησης σχετικά με τον τρόπο εκτέλεσής της.

Περιγραφή: Το σύστημα πρέπει να προβάλει στην οθόνη δυναμικά διορθωτικά βέλη, τα οποία ξεκινάνε από την προβαλλόμενη εκτίμηση πόζας του χρήστη και δείχνουν προς την πόζα αναφοράς. Έτσι, υποδεικνύεται σε κάθε καρέ η απόκλιση της πόζας του χρήστη από την πόζα αναφοράς.

User Priority (5/5): Η απαίτηση αυτή είναι απαραίτητη για τον χρήστη καθώς αποτελεί σημαντική κατευθυντήρια δύναμη, παρέχοντας ξεκάθαρη και άμεση ανατροφοδότηση για την πιο σωστή εκτέλεση της άσκησης αλλά και λειτουργώντας ως παρακινητής του χρήστη κατά την λειτουργία του παιχνιδιού.

Technical Priority (1/5): Η απαίτηση αυτή δεν επηρεάζει με κανέναν τρόπο την λειτουργία της εφαρμογής.

ΚΕΦΑΛΑΙΟ 5. ΥΛΟΠΟΙΗΣΗ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

ΛΑ-6

Το σύστημα πρέπει να μετράει τον αριθμό επαναλήψεων που έγιναν και να τον προβάλει.

Περιγραφή: Το σύστημα πρέπει να προβάλει έναν μετρητή όπου φαίνεται ο αριθμός των επαναλήψεων που έχουν ολοκληρωθεί και που απομένουν για την συγκεκριμένη άσκηση. Με το πέρας της τελευταίας επανάληψης η άσκηση τερματίζεται.

User Priority (5/5): Η απαίτηση αυτή είναι άκρως σημαντική για τον χρήστη καθώς πρέπει να ξέρει σε πιο στάδιο εκτέλεσης της άσκησης βρίσκεται, λειτουργώντας επίσης ως παρακινητής για την ολοκλήρωση της άσκησης.

Technical Priority (4/5): Η απαίτηση αυτή είναι σημαντική για την λειτουργία της εφαρμογής καθώς με την μέτρηση των επαναλήψεων που έχουν γίνει είναι δυνατή η λήξη της εκάστοτε άσκησης.

ΛΑ-7

Ο χρήστης πρέπει να μπορεί να αναπαράγει τις ασκήσεις αναφοράς και να ελέγχει την αναπαραγωγή.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να δει τα διαδοχικά καρέ της άσκησης αναφοράς. Επίσης, πρέπει να μπορεί να πατήσει παύση και συνέχεια της άσκησης καθώς και να διαλέξει το σημείο αναπαραγωγής με χρήση ενός ολισθητή.

User Priority (3/5): Ο απλός χρήστης μπορεί έτσι να μελετήσει τον τρόπο εκτέλεσης της εκάστοτε άσκησης. Στην περίπτωση εισαγωγής άσκησης αναφοράς η απαίτηση χρησιμεύει προς επιβεβαίωση των εκτιμώμενων δεδομένων αναφοράς.

Technical Priority (2/5): Η δυνατότητα αναπαραγωγής των ασκήσεων αναφοράς δεν είναι απαραίτητη για την λειτουργία της εφαρμογής.

ΛΑ-8

Ο χρήστης πρέπει να μπορεί να επεξεργάζεται τα δεδομένα των ασκήσεων αναφοράς.

Περιγραφή: Τα δεδομένα των ασκήσεων αναφοράς ενδέχεται να περιέχουν σφάλματα εκτίμησης. Κατά συνέπεια, ο χρήστης πρέπει να μπορεί να επεξεργάζεται τις ασκήσεις αναφοράς ώστε να ελαχιστοποιήσει το σφάλμα. Για κάθε σημείο κλειδί του ανθρώπινου σώματος και για κάθε άξονα περιστροφής ο χρήστης μπορεί να ορίσει μία σταθερή τιμή ή ένα περιορισμένο εύρος για τις μοίρες περιστροφής του σημείου κλειδιού γύρω από τον εκάστοτε άξονα.

5.1. ΑΠΑΙΤΗΣΕΙΣ ΣΥΣΤΗΜΑΤΟΣ

User Priority (2/5): Η απαίτηση της επεξεργασίας των ορθών ασκήσεων δεν είναι απαραίτητη για τον χρήστη, καθώς η εφαρμογή λειτουργεί ως αρωγός στην παρακίνηση του και η απόλυτη ακρίβεια δεν είναι ρεαλιστικά εφικτή.

Technical Priority (2/5): Η απαίτηση αυτή δεν είναι απαραίτητη για την σωστή λειτουργία της εφαρμογής, εφόσον τα βίντεο ορθών ασκήσεων είναι καλής ποιότητας ώστε οι εκτιμήσεις ασκήσεων αναφοράς να έχουν ελάχιστα σφάλματα.

ΛΑ-9

Ο χρήστης πρέπει να μπορεί να αλλάξει το χρώμα του προβαλλόμενου ανθρώπινου μοντέλου.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να αλλάξει το χρώμα του ανθρώπινου μοντέλου ώστε να ικανοποιεί την αισθητική του.

User Priority (3/5): Η απαίτηση της αλλαγής χρώματος είναι σημαντική για τον χρήστη καθώς η ικανοποίηση της αισθητικής του είναι απαραίτητη για την παρακίνηση του.

Technical Priority (1/5): Η απαίτηση αυτή δεν επηρεάζει με κανέναν τρόπο τη λειτουργία της εφαρμογής.

5.1.1 Μη Λειτουργικές Απαιτήσεις

ΜΛΑ-1

Το σύστημα πρέπει να προβάλει το ανθρώπινο μοντέλο με τουλάχιστον 30 καρέ το δευτερόλεπτο.

Περιγραφή: Το σύστημα, στην λειτουργία παιχνιδιού, πρέπει να εκτιμάει την ανθρώπινη πόζα, να κάνει την απαραίτητη επεξεργασία των δεδομένων και να τα προβάλει αρκετά γρήγορα ώστε να επιτυγχάνει 30 καρέ το δευτερόλεπτο, δηλαδή ο συνολικός χρόνος επεξεργασίας του καρέ εισόδου εώς την προβολή του ανθρώπινου μοντέλου να είναι λιγότερο από 35ms.

User Priority (5/5): Η απαίτηση αυτή είναι άκρως σημαντική για τον χρήστη καθώς η ομαλή διαδοχή των καρέ στο μάτι είναι απαραίτητη για την ικανοποιητική χρήση της εφαρμογής.

Technical Priority (5/5): Η απαίτηση αυτή είναι πολύ σημαντική για το σύστημα, καθώς καθορίζει τα τεχνικά χαρακτηριστικά και την δομή του, την επιλογή μοντέλων για την εκτίμηση πόζας και τις απαιτήσεις των αλγορίθμων απεικόνισης των

ΚΕΦΑΛΑΙΟ 5. ΥΛΟΠΟΙΗΣΗ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

δεδομένων.

ΜΛΑ-2

Το σύστημα πρέπει να μπορεί να προσαρτηθεί στο λειτουργικό του καθρέφτη.

Περιγραφή: Το σύστημα πρέπει να μπορεί να προσαρτηθεί εύκολα στο λειτουργικό σύστημα.

User Priority (5/5): Η απαίτηση αυτή είναι άκρως σημαντική για τον χρήστη, καθώς η διεπαφή του με το σύστημα γίνεται μέσω του λειτουργικού του καθρέφτη.

Technical Priority (5/5): Η απαίτηση αυτή είναι άκρως σημαντική για το σύστημα, καθώς η εφαρμογή λειτουργεί στα πλαίσια του λειτουργικού του καθρέφτη.

ΜΛΑ-3

Το σύστημα πρέπει να παρέχει στο λειτουργικό του καθρέφτη τις απαραίτητες φωνητικές εντολές για την λειτουργία του.

Περιγραφή: Το σύστημα πρέπει να εξάγει μία διεπαφή προς το λειτουργικό του καθρέφτη, όπου καθορίζονται οι απαιτούμενες φωνητικές εντολές και οι λειτουργίες που επιτελούν.

User Priority (5/5): Η απαίτηση αυτή είναι άκρως σημαντική για τον χρήστη, καθώς έτσι επιτυγχάνεται η αλληλεπίδρασή του με το σύστημα με την χρήση φωνής.

Technical Priority (5/5): Η απαίτηση αυτή είναι άκρως σημαντική για το σύστημα, καθώς με αυτόν τον τρόπο γίνεται ο έλεγχος των λειτουργιών του συστήματος.

ΜΛΑ-4

Το σύστημα πρέπει να είναι αρθρωτό.

Περιγραφή: Οι αλγόριθμοι που χρησιμοποιεί το σύστημα, όπως ο αλγόριθμος εκτίμησης πόζας και ο αλγόριθμος απεικόνισης του ανθρώπινου μοντέλου, πρέπει να μπορούν να χρησιμοποιηθούν αυτόνομα. Με αυτόν τον τρόπο, η αξιοποίηση τους για ανάπτυξη επιπλέον εφαρμογών για τον καθρέφτη διευκολύνεται σε μεγάλο βαθμό.

User Priority (2/5): Η απαίτηση αυτή δεν επηρεάζει τον χρήστη κατά την λειτουργία της εφαρμογής, αλλά αυξάνει την ευκολία ανάπτυξης αντίστοιχων εφαρμογών, οι οποίες επαυξάνουν τις δυνατότητες του έξυπνου καθρέφτη.

5.2. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΛΑΣΕΩΝ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

Technical Priority (4/5): Η απαίτηση αυτή είναι πολύ σημαντική για το σύστημα, καθώς καθιστά την λειτουργία του συστήματος πιο εύρωστη, την ανάπτυξη του πιο εύκολη αλλά και προωθεί την επεκτασιμότητα των εφαρμογών του καθρέφτη.

ΜΛΑ-5

Το σύστημα πρέπει να είναι διαισθητικά απλό και εύχρονο.

Περιγραφή: Το σύστημα πρέπει να είναι απλό στην χρήση του και να παρέχει ενημερωτικό κείμενο σχετικά με την κατάσταση του.

User Priority (5/5): Η απαίτηση αυτή είναι άκρως σημαντική για τον χρήστη, καθώς η εύχρονη λειτουργία της εφαρμογής εξωραΐζει την εμπειρία του χρήστη με το σύστημα.

Technical Priority (3/5): Η απαίτηση αυτή δεν επηρεάζει την λειτουργία του συστήματος, εντούτοις η απλοϊκότητα των λειτουργιών συνεπάγεται και απλοϊκότητα υλοποίησης, βελτιώνοντας την ποιότητα και την ευρωστία του συστήματος.

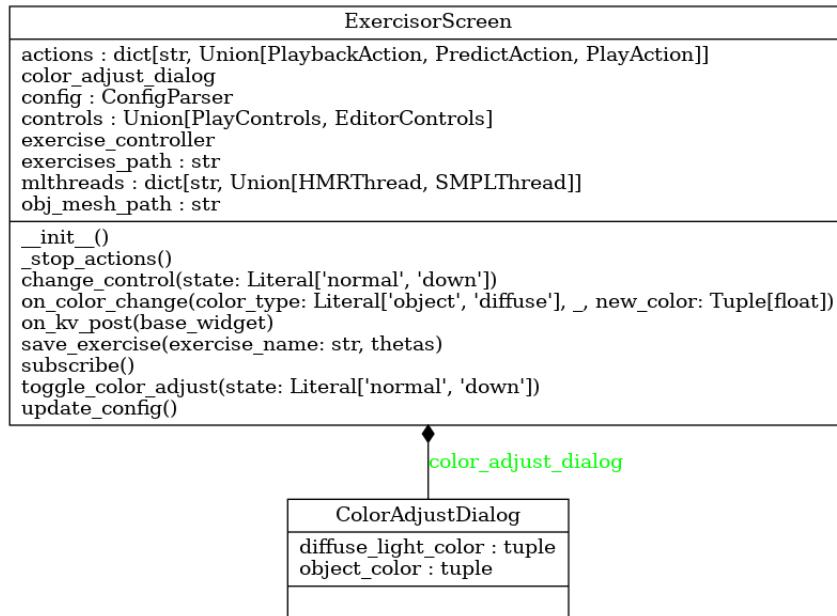
5.2 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΛΑΣΕΩΝ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

Για την εφαρμογή *Exercisor* χρησιμοποιήθηκε η βιβλιοθήκη Kivy, ακρογωνιαίος λίθος της οποίας είναι το μοντέλο Model-View-Controller, όπως περιγράφηκε στην ενότητα 4.1. Επεξηγηματικά, το επίπεδο View επιτυγχάνεται με το Kivy χρησιμοποιώντας την γλώσσα KV³⁵, σε ειδικά αρχεία κατάληξης .kv, τα οποία χρησιμοποιούνται για να καθορίσουν τα γραφικά στοιχεία, την εμφάνιση τους και την διάταξη τους στην οθόνη. Ταυτόχρονα, οι κλάσεις ορισμένες με την γλώσσα KV είναι συνδεδεμένες προαιρετικά με τον αντίστοιχο ορισμό τους στην γλώσσα Python, η λογική του οποίου υλοποιεί το επίπεδο Controller.

Στο παρών κεφάλαιο θα αναλυθούν οι κλάσεις και οι σχέσεις μεταξύ τους με την γλώσσα UML, αναφέροντας σχεδιαστικά πρότυπα που χρησιμοποιήθηκαν. Σημειώνεται ότι δεν θα επεκταθούμε στον τρόπο λειτουργίας των μοντέλων μηχανικής μάθησης HMR και SMPL, καθώς από την μία ο κώδικας δεν αναπτύχθηκε από εμάς, και από την άλλη περιγράφεται εκτενώς η λειτουργία τους στα κεφάλαια 3.5.4 και 3.4.1 αντίστοιχα. Επίσης, δεν θα σχολιάσουμε τον τρόπο παραγωγής γραφικών με την γλώσσα KV παρά μόνο τους αντίστοιχους ελεγκτές που χρήζουν σχολιασμό.

³⁵<https://kivy.org/doc/stable/guide/lang.html>

5.2.1 ExercisorScreen



Σχήμα 5.1: UML διάγραμμα της κλάσης `ExercisorScreen`

Περιγραφή Κλάσης Η κλάση αυτή κληρονομεί από την κλάση `ScreenManager`³⁶ και είναι η βασική κλάση του `Exercisor`.

Κατά την αρχικοποίηση της, στην μέθοδο `__init__` φορτώνονται τα αρχεία `.kv` και με το πέρας της διαδικασίας εκτελείται η συνάρτηση `on_kv_post()`. Τότε, γίνεται η αρχικοποίηση του ελεγκτή λειτουργίας παιχνιδιού, `controls`, κλάση τύπου `AbstractControls` 5.2.3, των διαθέσιμων ενεργειών, `actions` κλάσεις τύπου `AbstractAction` 5.2.4, οι οποίες με την σειρά τους αρχικοποιούν τα threads των μοντέλων βαθιάς μηχανικής μάθησης, `mlthreads` κλάσεις τύπου `MLThread` 5.2.5. Επιπλέον, αρχικοποιείται ο `exercise_controller`, κλάση τύπου `ExerciseController` 5.2.2, ο οποίος είναι ο ελεγκτής των διαθέσιμων ασκήσεων. Ένα εποπτικό UML διάγραμμα της αρχιτεκτονικής φαίνεται στο Σχήμα 5.2.

Χαρακτηριστικά κλάσης³⁷

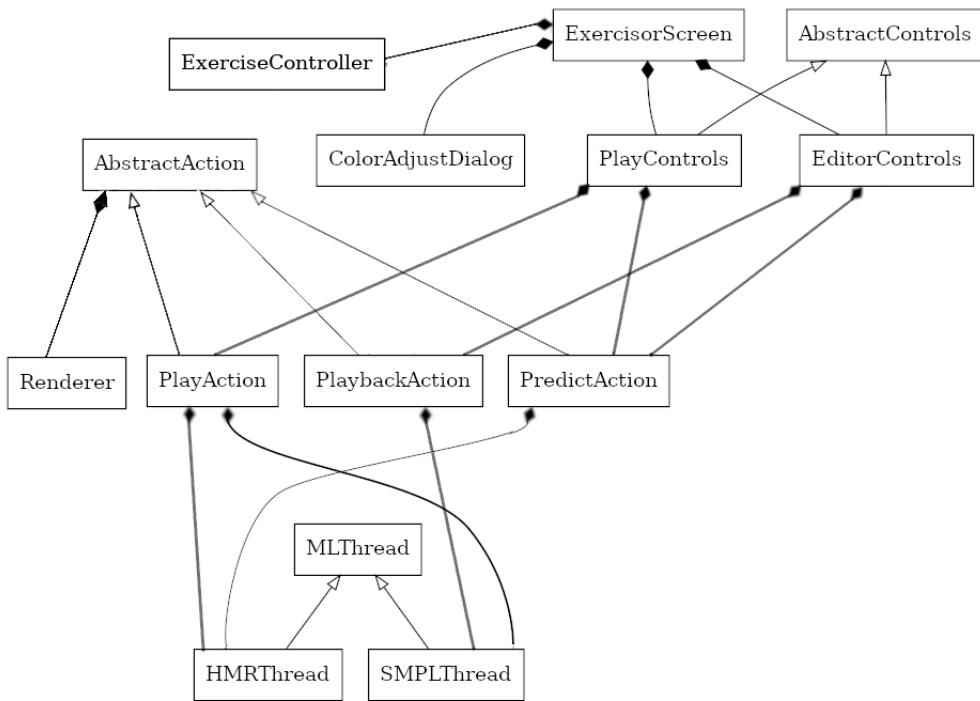
- **config:** Μεταβλητή τύπου `kivy.properties.ConfigParserProperty`³⁸ που επιτρέπει την αλλαγή του `path` των προ-εκπαιδευμένων μοντέλων, του φακέλου με τα δεδομένα ασκήσεων και του αρχικού φακέλου με τα βίντεο για είσοδο ασκήσεων αναφοράς.

³⁶<https://kivy.org/doc/stable/api-kivy.ux.screenmanager.html#kivy.ux.screenmanager.ScreenManager>

³⁷Σημειώνεται ότι στις επεξηγήσεις των χαρακτηριστικών και των μεθόδων παραλείπονται αυτές που έχουν ήδη αναφερθεί ή που είναι προφανής η χρήση τους. Επίσης, όταν παραλείπεται μέθοδος δόμησης `__init__()` που παίρνει ορίσματα, θεωρείται ότι το μόνο που κάνει είναι να αναθέσει τα αντίστοιχα ορίσματα σε χαρακτηριστικά της κλάσης.

³⁸<https://kivy.org/doc/stable/api-kivy.properties.html#kivy.properties.ConfigParserProperty>

5.2. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΛΑΣΕΩΝ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ



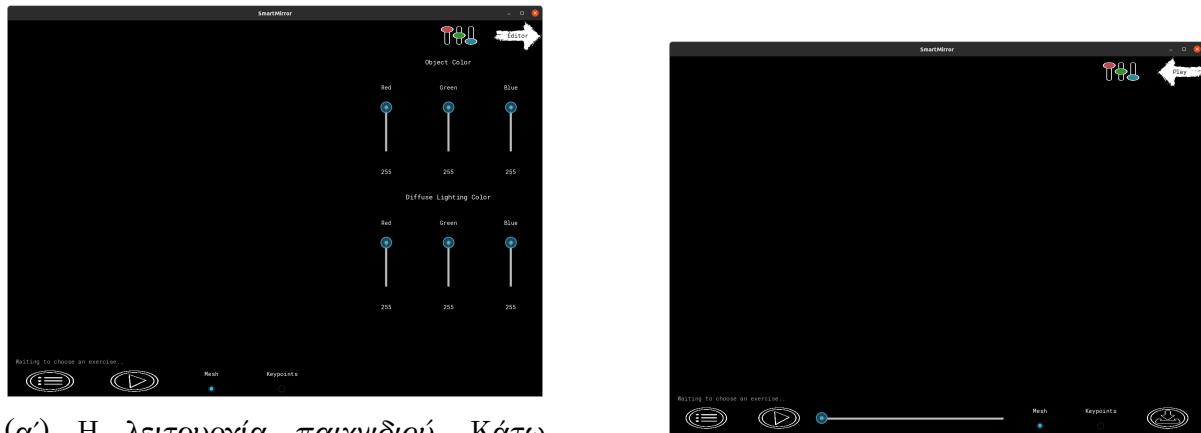
Σχήμα 5.2: Εποπτικό UML διάγραμμα σχέσεων του `ExercisorScreen`.

- `exercises_path`: Το path του φακέλου όπου βρίσκονται αποθηκευμένα τα δεδομένα ασκήσεων αναφοράς.
- `object_mesh_path`: Το path όπου βρίσκεται ένα αρχείο πλέγματος κατάληξης `.obj` που μπορεί να χρησιμοποιηθεί για αποσφαλμάτωση.

Μέθοδοι Κλάσης

- `change_control(state)`: Αλλάζει τα κουμπιά στο κάτω μέρος της οθόνης από λειτουργία παιχνιδιού σε λειτουργία επιμελητή και ανάποδα, ανάλογα με την τιμή του `state`, Σχήμα 5.3.
- `toggle_color_adjust(state)`: Εμφανίζει ή σβήνει το γραφικό της κλάσης `ColorAdjustDialog`, ανάλογα με την τιμή του `state`, Σχήμα 5.3α'. Μέσω αυτού του στοιχείου, εκτελείται η μέθοδος `on_color_change(color_type, new_color)` η οποία θέτει το RGB χρώμα `new_color` στο χρώμα του `ambient` φωτισμού ή του χρώματος των απεικονιζόμενων πλεγμάτων ανάλογα με το `color_type`.
- `subscribe()`: Συνάρτηση η οποία εκθέτει στον Controller τις εντολές που δέχεται το Widget και ποιες συναρτήσεις θα εκτελεστούν.
- `update_config()`: Συνάρτηση η οποία διαβάζει τις ρυθμίσεις και ανανεώνει κατάλληλα τις μεταβλητές της κλάσης.
- `save_exercise(exercise_name, thetas)`: Αποθηκεύει με όνομα `exercise_name` τα δεδομένα ασκησης αναφοράς `thetas` που είναι ένας πίνακας $N \times 82$, όπου N ο αριθμός των καρέ του βίντεο και 82 οι παράμετροι του μοντέλου SMPL.

ΚΕΦΑΛΑΙΟ 5. ΥΛΟΠΟΙΗΣΗ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

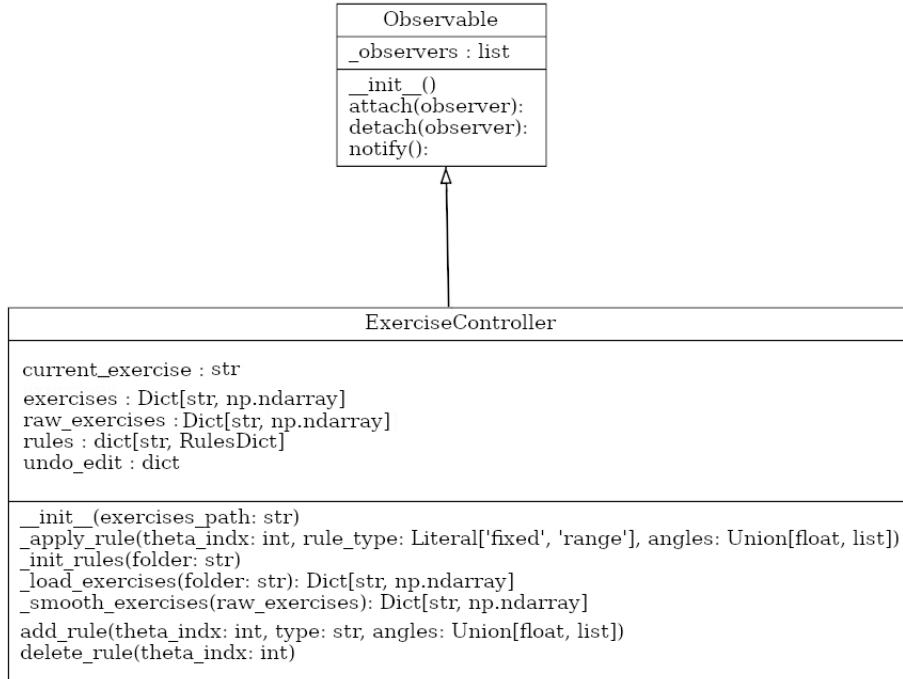


(α') Η λειτουργία παιχνιδιού. Κάτω φαίνεται το widget **PlayControls** ενώ στα δεξιά φαίνεται το widget **ColorAdjustDialog**.

(β') Η λειτουργία επιμελητή. Κάτω φαίνεται το widget **EditorControls**.

Σχήμα 5.3: Οι βασικές λειτουργίες του *Exercisor*

5.2.2 ExerciseController



Σχήμα 5.4: UML διάγραμμα της κλάσης **ExerciseController**

Περιγραφή Κλάσης Η κλάση αυτή είναι υπεύθυνη για την διαχείριση των διαθέσιμων ασκήσεων και την επεξεργασία τους.

Κληρονομεί από την κλάση **Observable** η οποία παρέχει μεθόδους για την υλο-

ποίηση του σχεδιαστικού προτύπου *Observer Pattern*³⁹. Αναλυτικότερα, αντικείμενα που ενδιαφέρονται για τις διαθέσιμες ασκήσεις πρέπει να υλοποιούν μία μέθοδο `update()` και να καλούνε την μέθοδο `attach` με όρισμα τον εαυτό τους, η οποία τις βάζει στην λίστα `_observers`. Έτσι, όταν οι ασκήσεις, `exercises`, ανανεώνονται, καλείται η μέθοδος `notify()` η οποία καλεί την `update()` όλων των εγγεγραμμένων αντικειμένων. Αναφορικά, τα αντικείμενα που εγγράφονται στις ασκήσεις είναι οι ελεγκτές τύπου `PlayControls` και `EditorControls`.

Χαρακτηριστικά κλάσης

- `current_exercise`: Το όνομα της τωρινής άσκησης. Όλες οι επεξεργασίες γίνονται πάνω στην τωρινή άσκηση.
- `exercises`: Ένα dictionary με key το όνομα της άσκησης και value έναν πίνακα \times 82, όπου N ο αριθμός των καρέ της άσκησης και 82 οι παράμετροι SMPL. Τα δεδομένα αυτών των ασκήσεων είναι μετά από την οποιαδήποτε επεξεργασία.
- `raw_exercises`: Οι ασκήσεις όπως αναφέρθηκε παραπάνω αλλά με τα δεδομένα έτσι όπως είναι αποθηκευμένα, πριν από επεξεργασία.
- `rules`: Περιλαμβάνει κανόνες επεξεργασίας των ασκήσεων. Τα keys είναι το όνομα της άσκησης, ενώ το `RulesDict` είναι ένα dictionary με key έναν δείκτη που ορίζει μία παράμετρο από τις 82 και values ένα tuple της μορφής (`rule_type`, `angles`), όπου καθορίζεται αν ο κανόνας επεξεργασίας είναι `fixed` ή `range` και οι σταθερές γωνίες περιορισμού.
- `undo_edit`: Dictionary αντίστοιχης μορφής με το `rules` που κρατάει όμως τις παραμέτρους SMPL για κάθε καρέ πριν την επεξεργασία.

Μέθοδοι Κλάσης

- `__init__(exercises_path)`: Διαβάζει τα δεδομένα των αποθηκευμένων ασκήσεων από το path `exercise_path` με χρήση της `_load_exercises`, αρχικοποιεί τους κανόνες με την `_init_rules()` και εφαρμόζει ομαλοποίηση με την `_smooth_exercises`.
- `_apply_rule(theta_idx, rule_type, angles)`: Επεξεργάζεται την παράμετρο με δείκτη `theta_idx` της τωρινής άσκησης, εφαρμόζοντας έναν κανόνα τύπου `fixed` ή `range` περιορίζοντας τις γωνίες που μπορεί να πάρει.
- `_init_rules(folder)`: Αρχικοποιεί το dictionary με τους κανόνες επεξεργασίας με βάση τους αποθηκευμένους.
- `_load_exercises(folder)`: Διαβάζει τον φάκελο με τις αποθηκευμένες ασκήσεις τις μετατρέπει σε ένα dictionary και το επιστρέφει.
- `_smooth_exercises(self, raw_exercises)`: Εφαρμόζει έναν αλγόριθμο κινούμενου μέσου⁴⁰ με παράθυρο 10 καρέ στις παραμέτρους SMPL για κάθε άσκηση. Αυτό γίνεται για να ομαλοποιηθούν οι προβλέψεις από καρέ σε καρέ και να μην 'τρέμει' η απεικόνιση του ανθρώπινου πλέγματος.

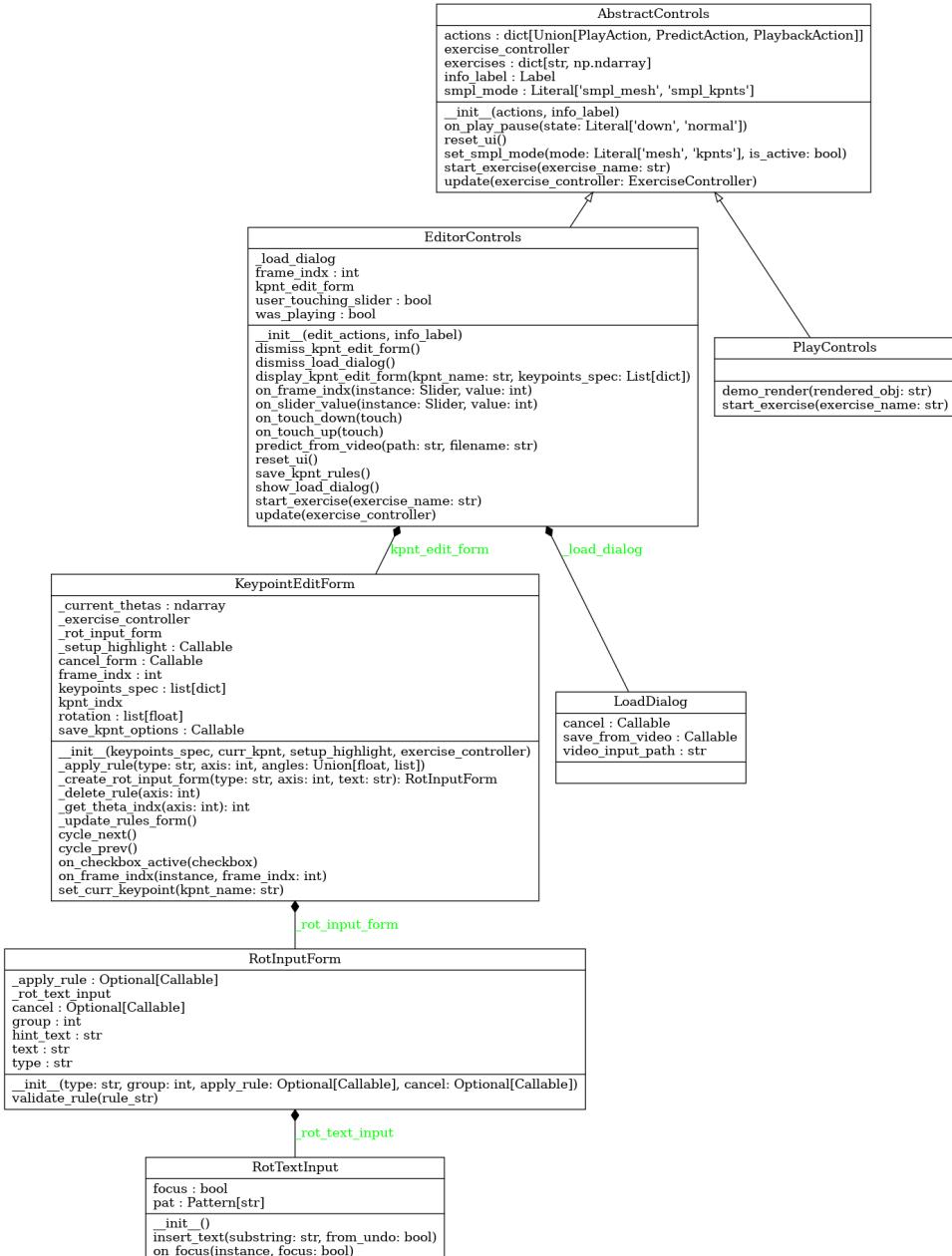
³⁹https://en.wikipedia.org/wiki/Observer_pattern

⁴⁰https://en.wikipedia.org/wiki/Moving_average

ΚΕΦΑΛΑΙΟ 5. ΥΛΟΠΟΙΗΣΗ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

- `add_rule(theta_idx, type, angles)`: Βάζει τον κανόνα στα `rules` και καλεί την `_apply_rule`.
- `delete_rule(theta_idx)`: Αφαιρεί τον κανόνα από τα `rules` και επαναφέρει τις παραμέτρους ανά καρέ από το `undo_edit`.

5.2.3 Controls



Σχήμα 5.5: UML διάγραμμα των κλάσεων τύπου `AbstractControls` και των σχέσεων τους.

5.2. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΛΑΣΕΩΝ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

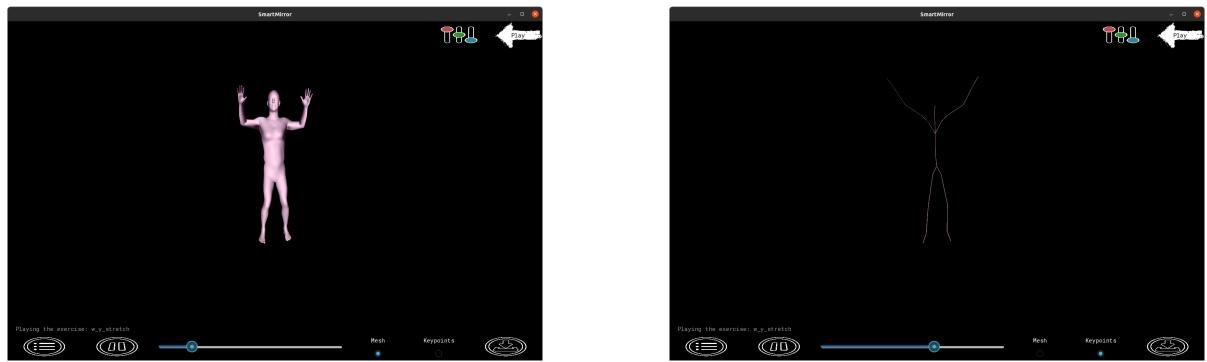
Περιγραφή Κλάσεων Οι κλάσεις αυτές είναι υπεύθυνες για την διαχείριση των διαθέσιμων ενεργειών, actions, στην εκάστοτε λειτουργία. Όπως φαίνεται στο Σχήμα 5.2, η κλάση PlayControls διαχειρίζεται τα actions τύπου PlayAction και PredictAction, ενώ η EditorControls τα actions PlaybackAction και PredictAction. Η γραφική τους απεικόνιση φαίνεται στο Σχήμα 5.6.⁴¹



Σχήμα 5.6: Η γραφική αναπαράσταση των κλάσεων τύπου AbstractControls.

Χαρακτηριστικά κλάσης Τα χαρακτηριστικά actions, exercise_controller και exercises έχουν ήδη σχολιαστεί.

- **info_label:** Το widget κειμένου πληροφορίας που φαίνεται πάνω από τα κουμπιά στα αριστερά της οθόνης στο Σχήμα 5.6.
- **smpl_mode:** Ο τρόπος προβολής του ανθρώπου. Μπορεί να είναι είτε 'smpl_mesh' είτε 'smpl_kpnts' για την απεικόνιση του πλέγματος ή του σκελετού αντίστοιχα, όπως φαίνεται στο Σχήμα 5.7.



Σχήμα 5.7: Οι τρόποι απεικόνισης του ανθρώπου.

Μέθοδοι Κλάσης

- **__init__(actions, info_label):** Αποθηκεύει τα χαρακτηριστικά actions και info_label και θέτει την προεπιλεγμένη τιμή του smpl_mode σε 'smpl_mesh'.
- **on_play_pause(state):** Όταν το state είναι 'normal' κάνει παύση όλα τα actions. Όταν είναι 'down' συνεχίζει τα actions που ήταν ενεργοποιημένα. Αν δεν υπήρχε κανένα ενεργοποιημένο action τότε η κατάσταση του κουμπιού δεν αλλάζει.

⁴¹Η εφαρμογή Exercisor, όπως αναφέρθηκε εξάγει τις εντολές αναγκαίες για την έναρξη του παιχνιδιού. Πιο περίπλοκες ενέργειες όμως, όπως οι λειτουργίες του επιμελητή, απαιτούν την χρήση ποντικιού και πληκτρολογίου. Ιδανικά οι λειτουργίες του επιμελητή δεν πρέπει να γίνονται πάνω στον καθρέφτη αλλά σε εξωτερικό σύστημα.

ΚΕΦΑΛΑΙΟ 5. ΥΛΟΠΟΙΗΣΗ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

- `reset_ui()`: Επαναφέρει την γραφική διεπαφή των κουμπιών στην αρχική τους κατάσταση και κλείνει ενδεχομένως ανοιχτούς διαλόγους.
- `set_smpl_mode(mode, is_active)`: Όταν το `is_active` είναι αληθές, αλλάζει τον τρόπο απεικόνισης του ανθρώπου σε πλέγμα ή σκελετό ανάλογα με την τιμή του `mode`.
- `start_exercise(exercise_name)`: Αρχίζει την αναπαραγωγή της άσκησης με όνομα `exercise_name`, θέτοντας την τιμή του `current_exercise` του αντικειμένου `ExerciseController`. Σταματάει το `PredictAction` και ξεκινάει το `PlayAction` ή το `PlaybackAction` αν είμαστε στην κλάση `PlayControls` ή `EditorControls` αντίστοιχα.
- `update(exercise_controller)`: Αναβαθμίζει τα δεδομένα των διαθέσιμων ασκήσεων. Καλείται από το αντικείμενο `Exercise Controller` όταν αλλάζουν τα δεδομένα.

PlayControls

Τα κουμπιά της κλάσης `PlayControls` από τα αριστερά προς τα δεξιά, όπως φαίνονται στο Σχήμα 5.6α' είναι

- Κουμπί επιλογής ασκήσεως αναφοράς για την έναρξη παιχνιδιού.
- Κουμπί παύσης ή συνέχισης της αναπαραγόμενης άσκησης.
- Διακόπτης επιλογής του τρόπου απεικόνισης του ανθρώπινου μοντέλου άσκησης αναφοράς.

Μέθοδοι Κλάσης

- `demo_render(rendered_obj)`: Μέθοδος για αποσφαλμάτωση του απεικονιστή κλάσης `Renderer` 5.2.6. Η τιμή του `rendered_obj` μπορεί να είναι 'smpl' ή 'monkey' όπου στην πρώτη περίπτωση γίνεται απεικόνιση του ανθρώπου από την εκτίμηση πόζας χωρίς καμία επιπλέον λειτουργία, ενώ στην δεύτερη περίπτωση γίνεται απεικόνιση ενός πλέγματος διαβασμένο από αρχείο κατάληξης `.obj`.

EditorControls

Περιγραφή Κλάσης Η κλάση `EditorControls` έχει όλα τα κουμπιά που έχει και η `PlayControls`, όπως φαίνεται στο Σχήμα 5.6β' αλλα επιπλέον περιλαμβάνει:

- Έναν ολισθητή ο οποίος δείχνει το τωρινό καρέ της άσκησης και μέσω αυτού μπορεί να ελεγχθεί η αναπαραγωγή της άσκησης.
- Ένα κουμπί με το οποίο ανοίγει ένας διάλογος επιλογής αρχείου βίντεο, μέσω του οποίου γίνεται η εκτίμηση πόζας και η αποθήκευση των δεδομένων ως ασκήσεως αναφοράς.

5.2. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΛΑΣΕΩΝ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

- Με μονό κλικ κατά την αναπαραγωγή ασκήσεως αναφοράς ανοίγει ένας διάλογος δίνοντας την δυνατότητα επεξεργασίας της περιστροφής οποιασδήποτε άρθρωσης - σημείου κλειδιού, όπως φαίνεται στο Σχήμα 5.8β'

Χαρακτηριστικά κλάσης

- `frame_indx`: Ο δείκτης του τωρινού καρέ. Ελέγχει την τιμή στην οποία βρίσκεται ο ολισθητής, που είναι κλάσης `kivy.ux.slider.Slider`⁴².
- `user_touching_slider`: Εάν ο χρήστης ακουμπάει αυτή την στιγμή των ολισθητή.
- `was_playing`: Εάν η αναπαραγωγή της άσκησης ήταν σε κατάσταση παύσης ή όχι όταν ξεκίνησε το κλικ, `on_touch_down()`.

Μέθοδοι Κλάσης

- `dismiss_keypoint_edit_form()`: Κλείνει τον διάλογο κλάσης `KeypointEditForm`.
- `dismiss_load_dialog()`: Κλείνει τον διάλογο επιλογής βίντεο.
- `display_kpnt_edit_form(kpnt_name, keypoints_spec)`: Ανοίγει τον διάλογο `KeypointEditForm` στο σημείο κλειδί `kpnt_name`. Επίσης, δίνει στον διάλογο όλες τις πληροφορίες για τα διαθέσιμα σημεία κλειδιά `keypoints_spec`.
- `on_frame_indx(instance, value)`: Καλείται όταν αλλάζει το `frame_indx` θέτοντας το `frame_indx` του διαλόγου `KeypointEditForm` αν υπάρχει.
- `on_slider_value(instance, value)`: Καλείται όταν αλλάζει το `frame_indx` λόγω του κλικ του χρήστη. Θέτει το τωρινό καρέ της αναπαραγόμενης άσκησης στην τιμή `value`.
- `on_touch_down(touch)`: Όταν ο χρήστης κάνει κλικ αν είναι μέσα στον χώρο του ολισθητή θέτει την `true` στο `user_touching_slider` και κρατάει την κατάσταση παύσης ή όχι του `action` στο `was_playing`.
- `on_touch_up(touch)`: Συνεχίζει την αναπαραγωγή από την κατάσταση που ήταν πριν το `on_touch_down()`.
- `predict_from_video(path, filename)`: Εκτελείται όταν ο χρήστης διαλέξει ένα βίντεο από τον διάλογο `LoadDialog`. Αρχικά καλεί την `dismiss_load_dialog()` και στην συνέχεια ξεκινάει το `action PredictAction`, σταματώντας ενδεχομένως το `PlaybackAction`, με είσοδο το αρχείο, αν υπάρχει, στο `path/filename`. Έπειτα, γίνεται η εκτίμηση πόζας και τα δεδομένα του βίντεο αποθηκεύονται ως δεδομένα άσκησης αναφοράς.
- `save_kpnt_rules()`: Αποθηκεύει τους κανόνες που τέθηκαν από το `KeypointEditForm` και κλείνει τον διάλογο καλώντας την `dismiss_kpnt_edit_form()`. Σημειώνεται ότι η αποθήκευση στον δίσκο των κανόνων δεν έχει υλοποιηθεί, αλλά αποθηκεύονται μόνο στην μνήμη.
- `show_load_dialog()`: Ανοίγει τον διάλογο `LoadDialog`.

⁴²<https://kivy.org/doc/stable/api-kivy.ux.slider.html>

ΚΕΦΑΛΑΙΟ 5. ΥΛΟΠΟΙΗΣΗ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ



(α') Γραφική αναπαράσταση του LoadDialog, ο διάλογος επιλογής βίντεο προς εισαγωγή ως ασκήσεως αναφοράς.

(β') Γραφική αναπαράσταση του EditKeypointForm, ο διάλογος επεξεργασίας περιστροφής σημείου κλειδιού. Στο ροζ ορθογώνιο φαίνεται η κλάση RotInputForm ενώ στο κίτρινο η RotInputText.

Σχήμα 5.8: Οι διαθέσιμοι διάλογοι της κλάσης EditorControls.

KeypointEditForm

Για χάρη συντομίας δεν θα αναλύσουμε διεξοδικά τις μεθόδους και μεταβλητές της κλάσης καθώς είναι κυρίως γραφικές κλάσεις με λειτουργικότητα που αφορά μόνο την εμφάνιση και έλεγχο των checkboxes και των φορμών εισόδου γωνιών περιστροφής για την διαχείριση των κανόνων.

Περιγραφή Κλάσης

Η κλάση KeypointEditForm μαζί με τις RotInputForm και RotTextInput φαίνονται στο Σχήμα 5.8β'. Ανάλογα με το διαλεγμένο σημείο κλειδί καλείται η μέθοδος `_setup_highlight()` για την εμφάνιση της τιρκουάζ περιοχής πάνω στο ανθρώπινο πλέγμα. Η επιλογή του σημείου κλειδιού γίνεται με κλικ στο πλέγμα, όπου επιλέγεται το κοντινότερο σημείο, ή με τα βελάκια *Previous* και *Next* που καλούνται τις μεθόδους `cycle_prev()` και `cycle_next()` για να πάνε στο προηγούμενο ή επόμενο σημείο κλειδί αντίστοιχα.

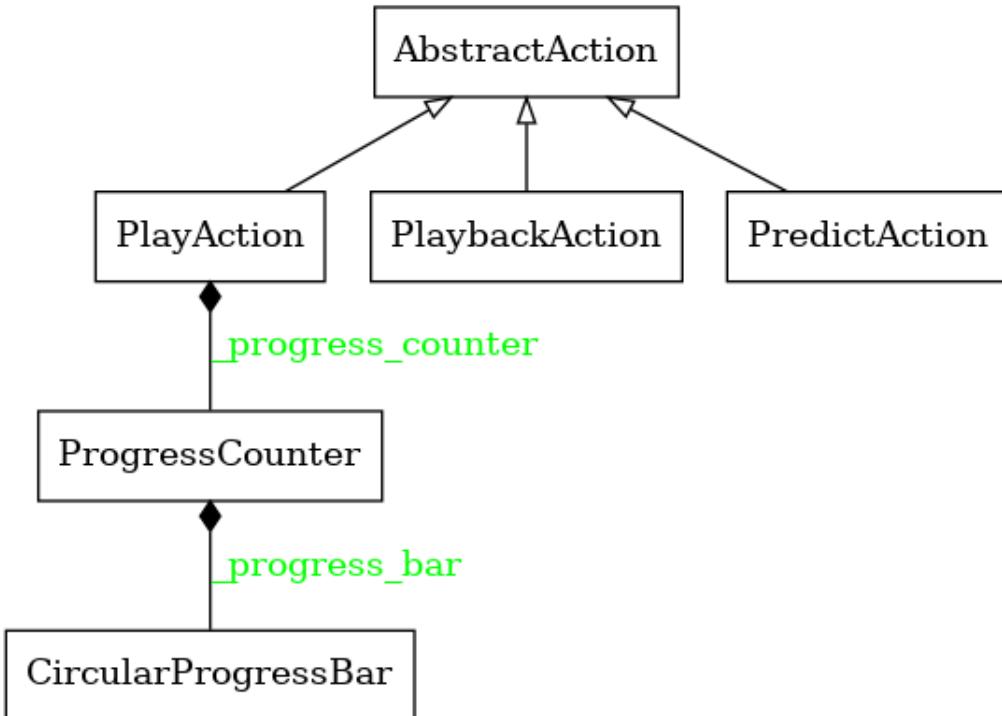
Στην φόρμα, φαίνονται οι περιστροφές σε μοίρες ανά καρέ, γύρω κάθε άξονα για το επιλεγμένο σημείο κλειδί. Για την εισαγωγή ενός κανόνα, αρκεί ο χρήστης να κάνει κλικ στο checkbox στην στήλη του άξονα και στην γραμμή ανάλογα με τον κανόνα που επιθυμεί. Τότε, το checkbox μετατρέπεται σε ένα πεδίο εισόδου κειμένου, όπου μπορεί να πληκτρολογήσει γωνίες και να θέσει τον κανόνα, είτε να πατήσει το X και να σβήσει τον κανόνα.

Στην περίπτωση εισόδου κανόνα `fixed` ο χρήστης μπορεί να πληκτρολογήσει μόνο έναν αριθμό, που είναι η σταθερή γωνία περιστροφής γύρω από τον άξονα για αυτό το σημείο κλειδί. Αντίθετα, στην περίπτωση κανόνα `range`, ο χρήστης μπορεί να βάλει 2 γωνίες με την μορφή `angle1:angle2`, όπου καθορίζεται το διάστημα μοιρών μέσα

5.2. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΛΑΣΕΩΝ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

στο οποίο μπορεί να κινηθεί το σημείο κλειδί.

5.2.4 Actions



Σχήμα 5.9: Εποπτικό UML διάγραμμα των κλάσεων τύπου **AbstractActions** και των σχέσεων τους.

Οι κλάσεις τύπου **AbstractAction** είναι οι ελεγχτές των διαφορετικών λειτουργιών που απαιτούν μοντέλα μηχανικής μάθησης. Το κάθε action είναι υπεύθυνο για την διαχείριση των μοντέλων που είναι αναγκαία για την διεκπεραίωση της λειτουργίας του. Δέχεται τα δεδομένα των εκτιμήσεων και περνάνε τα δεδομένα στις κλάσεις **Renderer**, [5.2.6](#) για την απεικόνιση τους.

Αναλυτικότερα, οι διαθέσιμες ενέργειες, **actions**, είναι η απλή εκτίμηση πόζας, **PredictAction**, η αναπαραγωγή αποθηκευμένης άσκησης αναφοράς, **PlaybackAction**, και η λειτουργία παιχνιδιού, **PlayAction**.

AbstractAction

exercisor.actions.AbstractAction
controls : Union[PlayControls, EditorControls]
paused : bool
renderer
running : bool
<code>_init_(renderer: Renderer)</code>
<code>init_rendererers(smpl_mode: Literal['smpl_mesh', 'smpl_kpnts', 'error_vectors', 'monkey'])</code>
<code>initialize(smpl_mode: Literal['smpl_mesh', 'smpl_kpnts', 'error_vectors', 'monkey'])</code>
<code>pause()</code>
<code>render_mesh(renderer: Renderer, new_vertices, new_kpnts)</code>
<code>reset_rendererers()</code>
<code>resume()</code>
<code>stop()</code>

Σχήμα 5.10: UML διάγραμμα της κλάσης AbstractAction.

Περιγραφή Κλάσης Το κάθε action χαρακτηρίζεται από 2 καταστάσεις, τις *running* και *paused*. Η κατάσταση *running* δηλώνει αν το συγκεκριμένο action είναι το ενεργό, ενώ η κατάσταση *paused* αν το action είναι σε παύση, δηλαδή δεν γίνεται κάποια εκτίμηση αυτή την στιγμή.

Χαρακτηριστικά κλάσης

- **controls:** Αναφορά σε αντικείμενο τύπου PlayControls ή EditorControls.
- **renderer:** Αντικείμενο κλάσης Renderer το οποίο λειτουργεί ως ο κύριος απεικονιστής των εκτιμήσεων που παράγονται από τα μοντέλα μηχανικής μάθησης.
- **paused:** Μεταβλητή που δηλώνει αν το action βρίσκεται στην κατάσταση *paused* ή όχι.
- **running:** Μεταβλητή που δηλώνει αν το action βρίσκεται στην κατάσταση *running* ή όχι.

Μέθοδοι κλάσης

- **init_rendererers(smpl_mode):** Αρχικοποιεί τον renderer ορίζοντας το αντικείμενο που θα απεικονιστεί.
- **initialize(smpl_mode):** Ενεργοποιεί την ενέργεια. Αρχικά, καλεί την μέθοδο `stop()`, καλεί την `init_rendererers()` και θέτει την `running` σε true και τρέχει την `resume()`.
- **pause():** Θέτει την `paused` σε true και αν υπάρχει η `controls` βάζει το *Play* *Pause* κουμπί στην κατάσταση παύσης.
- **render_mesh(renderer, new_vertices, new_kpnts):** Θέτει τις κορυφές του απεικονιστή renderer στις τιμές είτε των `new_vertices` είτε των `new_kpnts`, ανάλογα με το ποιο υπάρχει.

- **reset_renderers()**: Καλεί την συνάρτηση **reset_scene()** όλων των αντικειμένων **Renderer** που υπάρχουν στην οθόνη.
- **resume()**: Θέτει την **paused** σε **false** και αν υπάρχει η **controls** βάζει το *Play*
Pause κουμπί στην κατάσταση συνέχειας.
- **stop()**: Θέτει την **running** σε **false** καλεί την **reset_renderers()** και αν υπάρχει η **controls** καλεί την μέθοδο της **reset_ui()**.

PredictAction

exercisor.actions.PredictAction
hmr_thread
<pre>_init_(hmr_thread: HMRThread, renderer: Renderer) initialize(smpl_mode: Literal['smpl_mesh', 'smpl_kpnts'], source: str, save_exercise: bool) pause() render_mesh(new_vertices, new_kpnts) resume() stop()</pre>

Σχήμα 5.11: UML διάγραμμα της κλάσης PredictAction.

Περιγραφή Κλάσης Η κλάση **PredictAction** διαχειρίζεται ένα αντικείμενο κλάσης **HMRThread** για την διεκπεραίωση της απλής εκτίμησης πόζας και απεικόνισης των αποτελεσμάτων. Η ενέργεια αυτή χρησιμοποιείται για την εκτίμηση των δεδομένων ασκήσεως αναφοράς από ένα βίντεο.

Χαρακτηριστικά κλάσης

- **hmr_thread**: Αναφορά σε αντικείμενο κλάσης **HMRThread**, [5.2.5](#).

Μέθοδοι Κλάσης

Οι μέθοδοι **pause()** και **stop()** σταματάνε την λειτουργία του **HMRThread** ενώ η **resume()** την συνεχίζει.

- **initialize(smpl_mode, source, save_exercise)**: Καλεί την μέθοδο **initialize()** της **AbstractAction** και θέτει την συνάρτηση εξόδου του **HMRThread** να είναι η μέθοδος **render_mesh()**. Επίσης, θέτει την πιγγή εισόδου βίντεο του **HMRThread** που μπορεί να είναι είτε *"cam"*⁴³, όπου γίνεται εκτίμηση από την κάμερα του χρήστη, είτε το path από ένα αρχείο βίντεο. Αν η **save_exercise** είναι **true**, τότε όταν τελειώσει η αναπαραγωγή αποθηκεύονται όλα τα εκτιμώμενα δεδομένα.
- **render_mesh(new_vertices, new_kpnts)**: Καλεί την συνάρτηση **render_mesh** της **AbstractAction** με όρισμα **renderer** το χαρακτηριστικό **renderer** του αντικειμένου.

⁴³Η λειτουργία αυτή χρησιμοποιήθηκε για αποσφαλμάτωση.

PlaybackAction

exercisor.actions.PlaybackAction
<pre>exercise keypoints_spec : List[dict] smpl_thread __init__(smpl_thread: SMPLThread, renderer: Renderer, keypoints_spec: List[Dict]) initialize(smpl_mode: str, exercise) pause() render_mesh(new_vertices, new_kpnts, frame: Dict) resume() seek(time_point: Union[int, float], fmt: Literal['frame', 'duration']) single_click_handle(touch_pos: Tuple[float, float], closest_kpnt: str) stop()</pre>

Σχήμα 5.12: UML διάγραμμα της κλάσης PlaybackAction.

Περιγραφή Κλάσης Η κλάση `PlaybackAction` διαχειρίζεται ένα αντικείμενο κλάσης `SMPLThread` για την μετατροπή των παραμέτρων SMPL σε κορυφές πλέγματος ανθρώπου και σκελετού προς απεικόνιση. Η ενέργεια αυτή χρησιμοποιείται για την αναπαραγωγή μιας αποθηκευμένης άσκησης αναφοράς στην λειτουργία επιμελητή.

Χαρακτηριστικά κλάσης

- `exercise`: Ο πίνακας $\times 82$ με τις 82 παραμέτρους SMPL για κάθε καρέ από τα N.
- `keypoints_spec`: Τα χαρακτηριστικά των διαθέσιμων σημείων κλειδιών από το μοντέλο SMPL.
- `smpl_thread`: Αναφορά σε αντικείμενο κλάσης `SMPLThread`, [5.2.5](#).

Μέθοδοι Κλάσης

Οι μέθοδοι `pause()` και `stop()` σταματάνε την λειτουργία του `SMPLThread` ενώ η `resume()` την συνεχίζει.

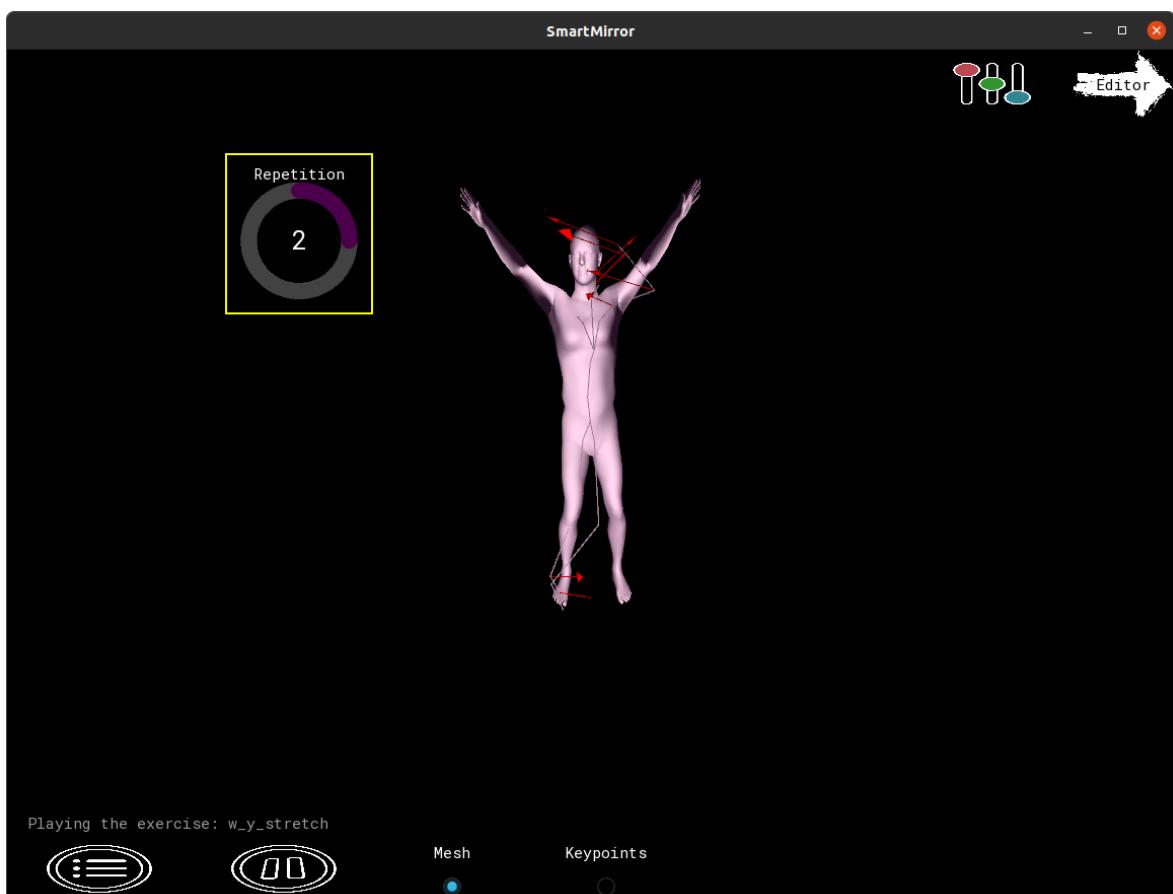
- `__init__(smpl_thread, renderer, keypoints_spec)`: Θέτει στον `renderer` να εκτελεί την μέθοδο `single_click_handle` όταν γίνεται ένα κλικ.
- `initialize(smpl_mode, exercise)`: Καλεί την `initialize` της `AbstractControl`, θέτει την μέθοδο εξόδου του `SMPLThread` να είναι η `render_mesh()`, θέτοντας επίσης τα δεδομένα της άσκησης που θα γίνει αναπαραγωγή.
- `render_mesh(new_vertices, new_kpnts, frame)`: Καλεί την συνάρτηση `render_mesh` της `AbstractAction` με όρισμα `renderer` το χαρακτηριστικό του αντικειμένου. Επίσης, θέτει το `frame_indx` του `EditorControls` σύμφωνα με το `frame`⁴⁴ ώστε να αντικατοπτρίζει το καρέ που μόλις απεικονίστηκε.

⁴⁴Το `frame` περιλαμβάνει πληροφορίες για το καρέ όπως τον δείκτη και την χρονική στιγμή που λήφθηκε.

5.2. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΛΑΣΕΩΝ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

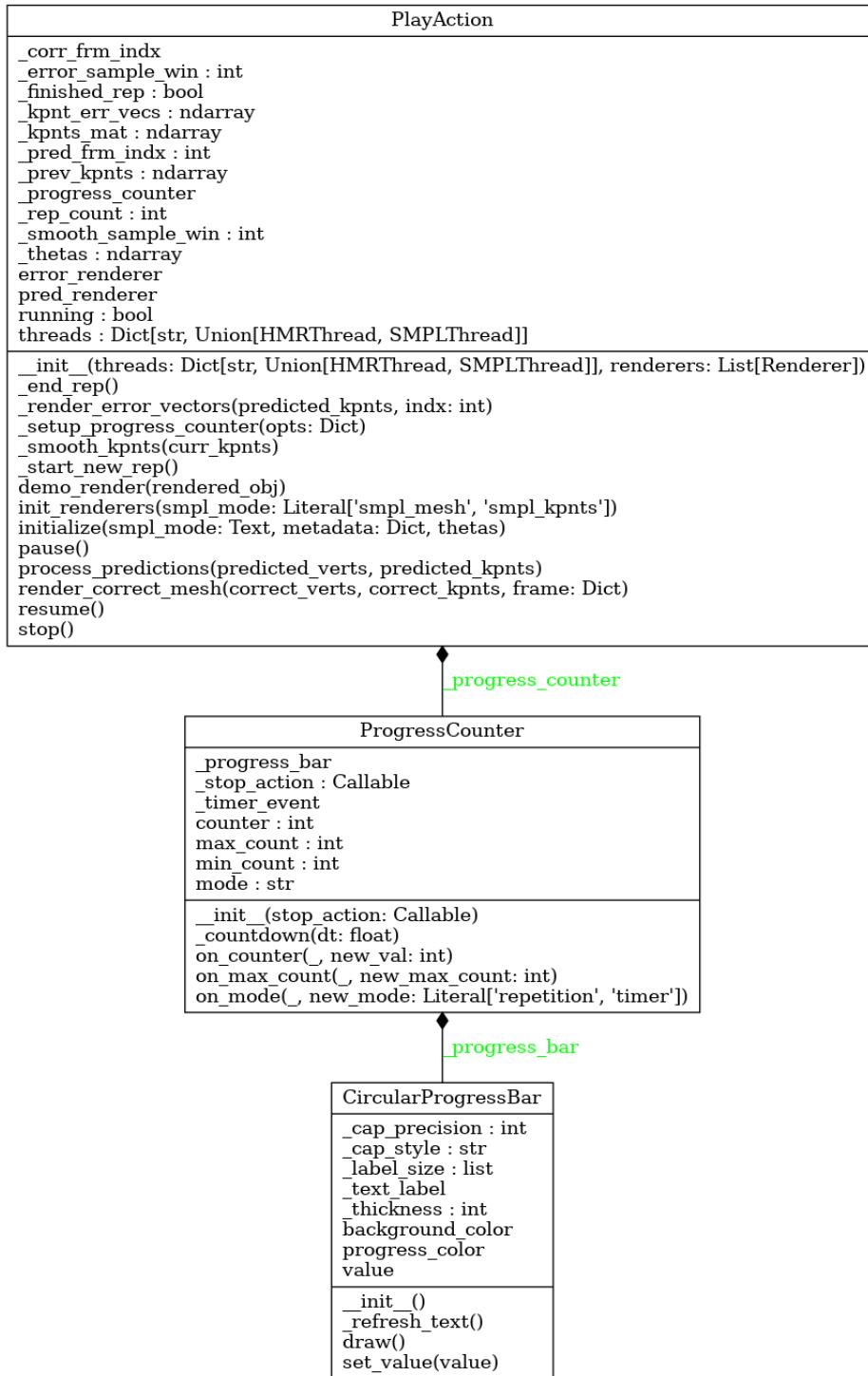
- `seek(time_point, fmt)`: Μέθοδος μέσω της οποίας ελέγχεται το σημείο καρέ της αναπαραγωγής. Αν το `fmt` είναι `"frame"` τότε θέτει το `frame_idx` του `SMPLThread` και του `EditorControls` σε αυτόν τον αριθμό. Αν το `fmt` είναι `"duration"` τότε η αναπαραγωγή πηγαίνει πίσω ή μπροστά τόσα δευτερόλεπτα (ανάλογα με το πρόσημο του `time_point`).
- `single_click_handle(touch_pos, closest_kpnt)`: Εκτελεί την μέθοδο `display_kpnt_edit_form` του `EditorControls` με όρισμα το όνομα του πιο κοντινού σημείου κλειδιού `closest_kpnt`.

PlayAction



Σχήμα 5.13: Γραφική απεικόνιση της λειτουργίας παιχνιδιού. Το ανθρώπινο πλέγμα είναι η απεικόνιση της άσκησης αναφοράς ενώ με τον σκελετό απεικονίζεται η εκτίμηση πόζας του χρήστη. Στο κίτρινο ορθογώνιο φαίνεται το widget `ProgressCounter`.

ΚΕΦΑΛΑΙΟ 5. ΥΛΟΠΟΙΗΣΗ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ



Σχήμα 5.14: UML διάγραμμα της κλάσης `PlayAction`.

Περιγραφή Κλάσης

Η ενέργεια `PlayAction` ελέγχει την κυρίως λειτουργία παιχνιδιού της εφαρμογής. Χρησιμοποιεί ταυτόχρονα τα `threads HMRThread` και `SMPLThread` για την εκτίμηση πόζας του χρήστη από την κάμερα και την αναπαραγωγή της άσκησης

αναφοράς αντίστοιχα. Το κάθε thread απεικονίζει τα αποτελέσματα στον δικό του renderer. Ένας τρίτος renderer χρησιμοποιείται για την απεικόνιση διορθωτικών βελών ή διανυσμάτων σφάλματος, δείχνοντας από την πόζα του χρήστη στην πόζα αναφοράς για κάθε σημείο κλειδί. Τα διορθωτικά βέλη και οι εκτιμήσεις του μοντέλου HMR χρησιμοποιούν έναν αλγόριθμο κινούμενου μέσου όρου στα τελευταία καρέ για να αντισταθμίσουν ενδεχόμενες ακραίες εκτιμήσεις και να εξομαλύνουν την απεικόνιση. Η λειτουργία παιχνιδιού φαίνεται στο Σχήμα 5.14.

Τέλος, ένα επιπλέον widget, κλάσης ProgressCounter, λειτουργεί ως μετρητής του αριθμού των επαναλήψεων ή ως αντίστροφος χρονομετρητής που καθορίζεται από το χαρακτηριστικό του mode. Τα max_count και min_count είναι τα όρια του μετρητή. Όταν φτάσει σε κάποιο από τα όρια ανάλογα με την λειτουργία του εκτελείται η _stop_action που του έχει δοθεί στην αρχικοποίηση για να σταματήσει η ενέργεια. Η κλάση CircularProgressBar υλοποιεί την γραφική διεπαφή του μετρητή.

Χαρακτηριστικά κλάσης

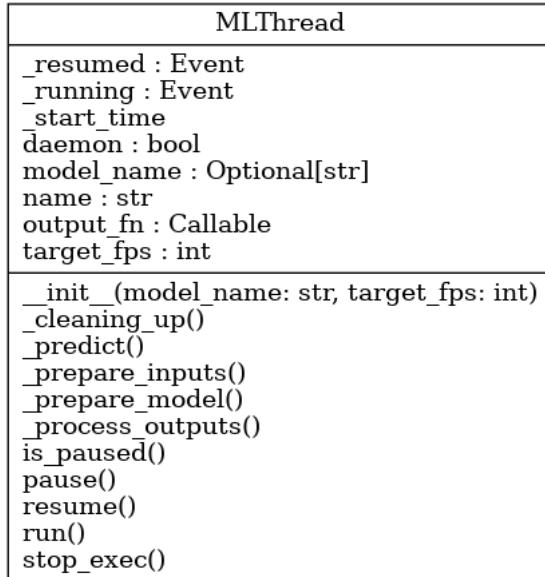
- threads: Αναφορές σε αντικείμενα κλάσης HMRThread και SMPLThread.
- _corr_frm_idx: Ο δείκτης καρέ που βρίσκεται το SMPLThread.
- _error_sample_win: Το παράθυρο του κινούμενου μέσου όρου σε αριθμό καρέ για την ομαλοποίηση των διανυσμάτων σφάλματος (error vectors).
- _finished_rep: Μεταβλητή boolean που δηλώνει πότε η αναπαραγωγή από το SMPLThread φτάνει στο τέλος της.
- _kpnt_err_vecs: Πίνακας διαστάσεων $N \times 24 \times 3$ όπου αποθηκεύονται οι 3 συντεταγμένες των διανυσμάτων σφάλματος των 24 σημείων κλειδιών για τα N καρέ της άσκησης.
- _kpnts_mat: Πίνακας $\times 24 \times 3$ που αποθηκεύονται από το SMPLThread οι σωστές 3 συντεταγμένες των 24 σημείων κλειδιών για τα N καρέ.
- _pred_frm_idx: Ο αριθμός των εκτιμήσεων του HMRThread που έχουν γίνει.
- _prev_kpnts: Πίνακας διαστάσεων ($_smooth_sample_window \times 24 \times 3$) που αποθηκεύονται οι 3 συντεταγμένες των 24 σημείων κλειδιών για την εφαρμογή του κινούμενου μέσου όρου.
- _progress_counter: Αντικείμενο της κλάσης ProgressCounter.
- _rep_count: Μετρητής του αριθμού επαναλήψεων.
- _smooth_sample_win: Το παράθυρο του κινούμενου μέσου όρου σε αριθμό καρέ για την ομαλοποίηση των εκτιμήσεων του HMRThread.
- _thetas: Τα δεδομένα της άσκησης αναφοράς, πίνακας $\times 82$ των 82 παραμέτρων SMPL για τα N καρέ.

Μέθοδοι Κλάσης

- _end_rep(): Ύπολογίζει το συνολικό μέσο τετραγωνικό σφάλμα της επανάληψης και εκτελεί την μέθοδο play_animation του renderer.

- `_render_error_vectors(predicted_kpnts, indx)`: Ύπολογίζει τον κινούμενο μέσο όρο των διανυσμάτων σφάλματος στον δείκτη `indx` και απεικονίζει τα διανύσματα που έχουν μέτρο μεγαλύτερο από μία τιμή.
- `_setup_progress_counter(opts)`: Αρχικοποιεί τον μετρητή `ProgressCounter` στην λειτουργία μετρητή ή αντίστροφου χρονομέτρου.
- `_smooth_kpnts(curr_kpnts)`: Εφαρμόζει τον κινούμενο μέσο όρο στις εκτιμήσεις `curr_kpnts`.
- `_start_new_rep()`: Επαναφέρει στις αρχικές τιμές τα χαρακτηριστικά `_pred_frm_indx`, `_finished_rep`, `_kpnts_mat`, `_kpnt_err_vecs`, αυξάνει το `_rep_count` κατά ένα και καλεί την `reset_scene` του `error_renderer`.
- `init_renderers(smpl_mode)`: Θέτει τον `smpl_renderer` στον τρόπο απεικόνισης που ορίζεται από το `smpl_mode` και τον `pred_renderer` στον αντίθετο.
- `initialize(smpl_mode, metadata, thetas)`: Θέτει τις μεθόδους εξόδου των `SMPLThread` και `HMRThread` στις μεθόδους `render_correct_mesh()`, `process_predictions()` αντίστοιχα. Θέτει την άσκηση αναφοράς `_thetas` στο `SMPLThread` το `_rep_count` στο 0, αρχικοποιεί τον `ProgressCounter` καλώντας την `_setup_progress_counter()` και καλεί την `_start_new_rep()`.
- `process_predictions(predicted_verts, predicted_kpnts)`: Καλεί την `_smooth_kpnts` στις εκτιμήσεις, υπολογίζει τα διανύσματα σφάλματος και τα αποθηκεύει, αυξάνει την `_pred_frm_indx`, απεικονίζει τα αποτελέσματα καλώντας την `render_mesh()` του `AbstractAction` και ελέγχει την `_finished_rep`, όπου αν τελείωσε η επανάληψη για να εκτελέσει τις `_end_rep()` και `_start_new_rep()`.
- `render_correct_mesh(correct_verts, correct_kpnts, frame)`: Αποθηκεύει τις σωστές εκτιμήσεις για το συγκεκριμένο καρέ, απεικονίζει τα αποτελέσματα καλώντας την `render_mesh()` του `AbstractAction` και ελέγχει αν το καρέ είναι το τελευταίο οπότε θέτει την `_finished_rep` σε `true`.

5.2.5 MLThreads



Σχήμα 5.15: UML διάγραμμα της κλάσης MLThread.

Περιγραφή Κλάσης

Η κλάση `MLThread` είναι μία αφαιρετική κλάση διαχείρισης ενός `thread` μοντέλου μηχανικής μάθησης. Κληρονομώντας από την `threading.Thread`⁴⁵ παρέχει απαραίτητες μεθόδους για την έναρξη, σταματημό ή παύση του `thread`. Με την αρχικοποίηση του `thread` μπαίνει στον βρόγχο λειτουργίας του στην μέθοδο `run()` η οποία σε μία επανάληψη, εκτελεί μία εκτίμηση του μοντέλου. Επιπλέον, η κλάση `MLThread` υλοποιεί το σχεδιαστικό πρότυπο *Template design pattern*⁴⁶ κατά το οποίο ορίζονται η αλληλουχία των βημάτων του αλγορίθμου και στην συνέχεια οι κλάσεις που την κληρονομούν είναι υπεύθυνες για την υλοποίηση αυτών καθ' αυτών των βημάτων.

Χαρακτηριστικά κλάσης

- `_resumed: threading.Event`⁴⁷ που καθορίζει αν ο βρόγχος λειτουργίας του `thread` λειτουργεί ή είναι σε παύση.
- `_running: threading.Event` το οποίο αρχικοποιείται με `set` και όσο παραμένει ο βρόγχος λειτουργίας συνεχίζεται. Όταν γίνει `clear` το `thread` τερματίζει την μέθοδο `run` του.

⁴⁵<https://docs.python.org/3/library/threading.html#threading.Thread>

⁴⁶https://en.wikipedia.org/wiki/Template_method_pattern

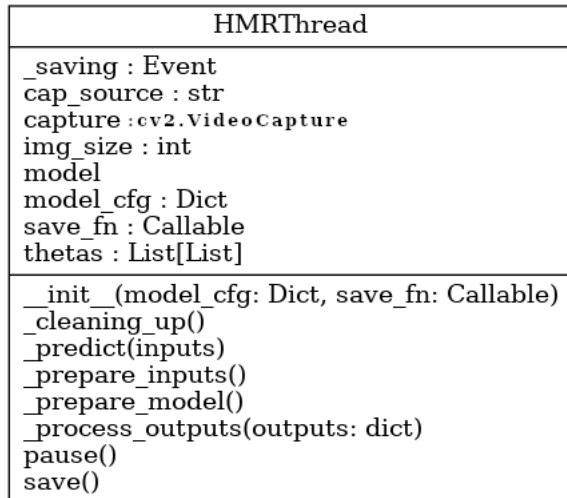
⁴⁷<https://docs.python.org/3/library/threading.html#event-objects>. Συνοπτικά, μπορούμε να φανταστούμε την κλάση αυτή ως μια `boolean` μεταβλητή, την οποία μπορεί το `thread` να περιμένει μέχρι να γίνει `true`.

- `_start_time`: Η χρονική στιγμή στην αρχή της επανάληψης του βρόγχου λειτουργίας.
- `daemon`: Μεταβλητή που καθορίζει αν το thread τρέχει σαν daemon. Ορίζεται ως `true` ώστε όταν τερματίσει το main thread να τερματίσει και αυτό.
- `model_name`: Το όνομα του μοντέλου μηχανικής μάθησης.
- `name`: Το όνομα του thread.
- `output_fn`: Η μέθοδος που καλείται με όρισμα τα αποτελέσματα του μοντέλου.
- `target_fps`: Οι μέγιστες προβλέψεις ανά δευτερόλεπτο που επιτρέπεται να λειτουργήσει το μοντέλο.

Μέθοδοι Κλάσης Οι μέθοδοι `_prepare_models`, `_prepare_inputs`, `_predict`, `_process_outputs` και `_cleaning_up` είναι αφηρημένες (abstract) και πρέπει να υλοποιηθούν από τους κληρονόμους της κλάσης `MLThread`.

- `__init__(model_name, target_fps)`: Καθορίζει το όνομα του thread με βάση το όνομα του μοντέλου `model_name`. Επίσης, καλεί την μέθοδο `start()` της κλάσης `threading.Thread`, η οποία ξεκινάει το thread, τρέχοντας την μέθοδο `run()`. Το thread ξεκινάει την λειτουργία του στην αρχικοποίηση του ώστε να αρχίσουν να φορτώνουν απευθείας τα μοντέλα μηχανικής μάθησης, ξεκινώντας όμως σε κατάσταση παύσης.
- `is_paused()`: Επιστρέφει `true` αν το thread βρίσκεται σε κατάσταση παύσης και `false` διαφορετικά.
- `pause()`: Κάνει `clear` το `_resumed`, οπότε το thread κάνει παύση στην εκτέλεση του βρόγχου λειτουργίας του.
- `resume()`: Κάνει `set` το `_resumed`, οπότε το thread συνεχίζει την εκτέλεση του βρόγχου λειτουργίας.
- `stop_exec()`: Κάνει `clear` το `_running`, οπότε το thread τερματίζει τον βρόγχο λειτουργίας του.
- `run()`: Αρχικά, φορτώνει το μοντέλο μηχανικής μάθησης εκτελώντας την μέθοδο `_prepare_model()`. Έπειτα, ξεκινάει ο κύριος βρόγχος λειτουργίας του thread που τρέχει όσο το `_running` είναι `set`. Στην αρχή του βρόγχου, αναμένεται μέχρι το `_resumed` να γίνει `set`. Τότε, θέτεται το `_start_time`, εποιμάζονται οι είσοδοι του μοντέλου, εκτελώντας την `_prepare_inputs()`. Οι είσοδοι δίνονται ως όρισμα στην `_predict()` και στη συνέχεια τα αποτελέσματα στην `_process_outputs`. Τέλος, αν ο χρόνος που έγινε η εκτίμηση είναι πιο γρήγορος από τον μέγιστο χρόνο ώστε να μην ξεπερνιούνται τα `_target_fps`, το thread κοιμάται, εκτελώντας την `time.sleep()`. Με το πέρας του βρόγχου λειτουργίας, εκτελείται η `_cleaning_up()` για να κλείσουν ενδεχόμενα ανοιχτά resources.

HMRThread



Σχήμα 5.16: UML διάγραμμα της κλάσης HMRThread.

Περιγραφή Κλάσης

Η κλάση `HMRThread` είναι το thread που ελέγχει το μοντέλο μηχανικής μάθησης HMR, διεκπεραιώνοντας έτσι τις εκτιμήσεις πόζας. Σε κάθε επανάληψη του βρόγχου λειτουργίας του δέχεται ως είσοδο ένα καρέ από την κάμερα του χρήστη ή από ένα βίντεο και παράγει τις 85 παραμέτρους SMPL, τις 3D συντεταγμένες των 6980 κορυφών πλέγματος και 24 σημείων κλειδιών του ανθρώπινου σώματος.

Χαρακτηριστικά κλάσης

- `_saving`: `threading.Event` που καθορίζει αν θα αποθηκευτούν οι 82⁴⁸ παράμετροι SMPL, για χρήση ως δεδομένων άσκησης αναφοράς, ή όχι.
- `cap_source`: Καθορίζει την πηγή λήψης των καρέ. Παίρνει την τιμή `"cam"`, οπότε η πηγή είναι η κάμερα του χρήστη, ενώ διαφορετικά παίρνει την τιμή του path σε ένα αρχείο βίντεο.
- `capture`: Με βάση την τιμή του `cap_source` αρχικοποιείται η `capture`, κλάσης `VideoCapture`⁴⁹ της OpenCV, μέσω της οποίας γίνεται η λήψη του καρέ.
- `img_size`: Η διάσταση της τετραγωνικής εικόνας που δέχεται το μοντέλο. Στην συγκεκριμένη υλοποίηση αυτή είναι 224×224 .
- `model`: Το μοντέλο HMR, κλάσης `HMR`, αφού φορτωθεί στην μνήμη.
- `model_config`: Ένα namespace με μεταβλητές που καθορίζουν τον τρόπο λειτουργίας του μοντέλου HMR. Περιλαμβάνει τις διαστάσεις τις εικόνας, τον

⁴⁸Οι 3 παράμετροι που παραλείπονται κατά την αποθήκευση είναι οι 3 συντεταγμένες της θέσης της κάμερας

⁴⁹https://docs.opencv.org/3.4/d8/dfe/classcv_1_1VideoCapture.html

αριθμό σταδίων του επαναληπτικού παλινδρομητή (όπως περιγράφηκε στην 3.5.4), τα paths των αρχείων των προ-εκπαιδευμένων μοντέλων και τον αριθμό των σημείων κλειδιών.

- **save_fn:** Η συνάρτηση που πρέπει να κληθεί για την αποθήκευση των παραμέτρων.
- **thetas:** Οι 82 παράμετροι SMPL που πρέπει να αποθηκευτούν για κάθε καρέ.

Μέθοδοι Κλάσης

- **_prepare_model():** Αρχικοποιεί το `session`⁵⁰ και το `graph`⁵¹ του Tensorflow, φορτώνει το μοντέλο και αποθηκεύει χρησιμοποιώντας την συνάρτηση `summary.FileWriter` του Tensorflow τον γράφο του μοντέλου, για οπτικοποίηση με το Tensorboard.
- **_prepare_inputs():** Λαμβάνει ένα καρέ από την πηγή, καλώντας την συνάρτηση `read()` του `VideoCapture`. Έπειτα, το προεπεξεργάζεται, σμικρύνοντας την εικόνα στις διαστάσεις `image_size`, διατηρώντας όμως τις αναλογίες διαστάσεων. Επιστρέφει την εικόνα καρέ και μία boolean δηλώντας αν ήταν εφικτή η απόκτηση του καρέ. Για την αποσφαλμάτωση, χρησιμοποιείται η συνάρτηση `imshow` της OpenCV για την προβολή των εικόνων καρέ σε ένα άλλο παράθυρο.
- **_predict(inputs):** Αν η εικόνα είχε ληφθεί με επιτυχία, καλεί την μέθοδο `predict()` του μοντέλου HMR για την εκτίμηση πόζας και επιστρέφει τις συντεταγμένες των κορυφών και των σημείων κλειδιών. Αν η `_saving` είναι `set` τότε όσο έρχονται επιτυχώς καρέ, αποθηκεύονται στην `thetas`, ενώ όταν δεν έρθει εικόνα γίνεται παύση του `thread`, καλώντας την `pause()` και επιστρέφονται τα `thetas`.
- **_process_outputs(outputs):** Αν υπάρχουν τα `thetas` στο όρισμα `outputs` τότε απλά καλείται η `save_fn()` για αποθήκευση των παραμέτρων. Διαφορετικά, καλείται η `output_fn()` με όρισμα τις κορυφές και τα σημεία κλειδιά.
- **_cleaning_up:** Απελευθερώνει το `capture`, δηλαδή την κάμερα ή το αρχείο βίντεο, και κλείνει όλα τα παράθυρα που άνοιξαν μέσω του OpenCV.
- **save():** Επαναφέρει τα `thetas` σε κενή λίστα και κάνει `set` την `_saving`.

⁵⁰https://www.tensorflow.org/api_docs/python/tf/compat/v1/Session

⁵¹https://www.tensorflow.org/api_docs/python/tf/Graph

SMPLThread

SMPLThread
<code>_joint_type : Literal['cocoplus', 'lsp'] _smpl_model_path : str exercise : ndarray frame_index : int</code>
<code>_init_(smpl_model_path: str, joint_type: str) _cleaning_up() _predict(thetas) _prepare_inputs() _prepare_model() _process_outputs(outputs)</code>

Σχήμα 5.17: UML διάγραμμα της κλάσης SMPLThread.

Περιγραφή Κλάσης

Η κλάση SMPLThread είναι το thread που ελέγχει το μοντέλο μηχανικής μάθησης SMPL, διεκπεραιώνοντας την μετατροπή των 82 παραμέτρων SMPL στις 3D συντεταγμένες των 6980 κορυφών πλέγματος και των 24 σημείων κλειδιών του ανθρώπινου σώματος, όπως αναλύθηκαν στο 3.4.1. Τοιουτοτρόπως, σε κάθε επανάληψη του βρόγχου λειτουργίας του SMPLThread γίνεται ακριβώς αυτή η μετατροπή.

Χαρακτηριστικά κλάσης

- `_joint_type`: Ανάλογα με την τιμή του επιστρέφονται τα 19 σημεία κλειδιά ορισμένα με την σειρά του προτύπου coco⁵² ή τα 14 του lsp⁵³ dataset. Εντούτοις, στην εφαρμογή χρησιμοποιούνται τα 24 σημεία κλειδιά με την σειρά ορισμένη από το SMPL⁵⁴.
- `smpl_model_path`: Το path του προ-εκπαιδευμένου μοντέλου SMPL.
- `exercise`: Οι 82 παράμετροι SMPL για κάθε καρέ της άσκησης προς αναπαραγωγή.
- `frame_index`: Ο δείκτης καρέ που καθορίζει το σημείο της αναπαραγωγής. Μεταβάλλοντας αυτό το χαρακτηριστικό ελέγχεται η αναπαραγωγή.

Μέθοδοι Κλάσης

- `_prepare_model()`: Αρχικοποιεί το `session` και το `graph` του Tensorflow και χτίζει τον υπολογιστικό γράφο του μοντέλου. Προς επίτευξη αυτού του σκοπού, ορίζονται οι 82 παράμετροι ως tensors εισόδου και οι κορυφές και τα σημεία κλειδιά ως tensors εξόδου και καλείται η συνάρτηση μετατροπής του SMPL.

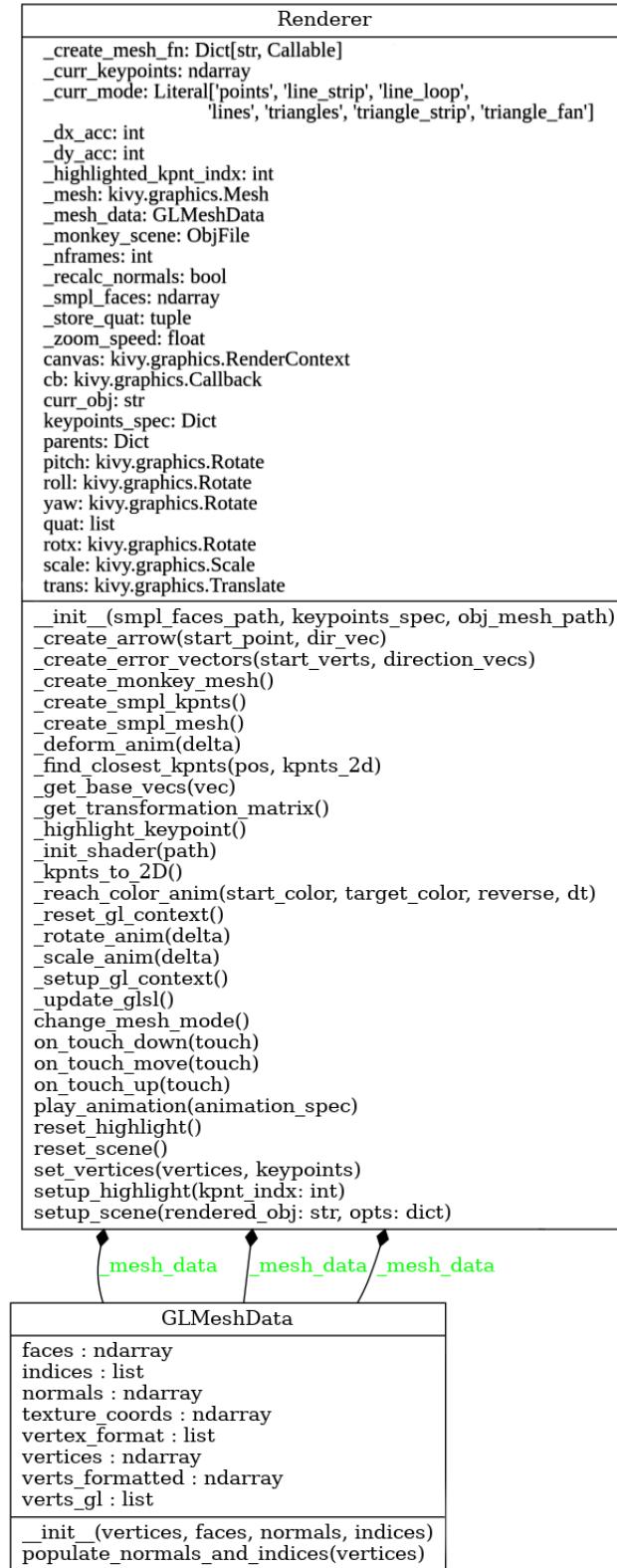
⁵²<https://cocodataset.org/>

⁵³https://dbcollection.readthedocs.io/en/latest/datasets/leeds_sports_pose_extended.html

⁵⁴Για το πλήρες κινηματικό δέντρο, σελίδα 24 του pdf <https://files.is.tue.mpg.de/black/talks/SMPL-made-simple-FAQs.pdf>

- `_prepare_inputs()`: Λαμβάνει από την `exercise` τις παραμέτρους SMPL ανάλογα με το τωρινό `frame_index` και τις επιστρέψει.
- `_predict(inputs)`: Καλείται η συνάρτηση `run()` του `session` ώστε να γίνει η εκτίμηση των κορυφών και των σημείων κλειδιών, τα οποία επιστρέφονται.
- `_process_outputs(outputs)`: Καλείται η συνάρτηση `output_fn()` με ορίσματα τις εξόδους του SMPL καθώς και πληροφορίες για το καρέ, δηλαδή τον δείκτη `frame_index` και την στιγμή λήψης του καρέ, `_start_time`.
- `_cleaning_up`: Δεν υλοποιεί κάποια λειτουργία.

5.2.6 Renderer



Σχήμα 5.18: UML διάγραμμα της κλάσης `Renderer` και των σχέσεων της.

Περιγραφή Κλάσης

Η κλάση `Renderer` είναι η κλάση που ελέγχει την απεικόνιση του ανθρώπινου πλέγματος, του σκελετού αλλά και των διορθωτικών βελών. Επιπλέον, παρέχει τη δυνατότητα αναπαραγωγής απλών εφέ πάνω στο πλέγμα, περιστροφής του πλέγματος με μονό κρατημένο κλικ, τη μεγέθυνση ή σμίκρυνση του πλέγματος με χρήση της ροδέλας του ποντικιού και κατά την λειτουργία επιμελητή την αλλαγή του χρώματος στην περιοχή ενός σημείου κλειδιού με μονό κλικ ώστε να καθιστά εμφανές το διαλεγμένο σημείο κλειδί προς επεξεργασία (θα αναφερόμαστε σε αυτή την λειτουργία ως *highlight*).

Έτσι, ο `Renderer` για τον καθορισμό των δεδομένων των αντικειμένων στην σκηνή χρησιμοποιεί την κλάση `GLMeshData` και για την απεικόνιση αυτών χρησιμοποιείται το χαμηλότερου επιπέδου API της *OpenGL ES 2.0* μέσω των bindings στην `Kivy` και συγκεκριμένα του πακέτου `kivy.graphics`⁵⁵.

Πιο συγκεκριμένα, οι κορυφές αποθηκεύονται στην κλάση `GLMeshData` η οποία μέσω του χαρακτηριστικού `faces`, ενός πίνακα $F \times 3$, καθορίζει ποιες 3 κορυφές σχηματίζουν κάθε ένα από τα F τρίγωνα που απαρτίζουν το ανθρώπινο πλέγμα. Ταυτόχρονα, μέσω της συνάρτησης `populate_normals_and_indices()` υπολογίζει τα `normals` για κάθε κορυφή. Τέλος, η *OpenGL* για την είσοδο των κορυφών απαιτεί όλες τις κορυφές, `vertices`, τα `normals` τους, `normals`, και τις UV συντεταγμένες για καθορισμό `texture`, `texture_coords`, να διατάσσονται σε μία μονοδιάστατη λίστα `verts_gl` με συγκεκριμένη σειρά στην οποία αναφέρεται η λίστα `indices` για τον καθορισμό των τριγώνων.

Χαρακτηριστικά κλάσης

- `_create_mesh_fn`: Ένα dictionary με keys το όνομα ενός αντικειμένου και values την μέθοδο της κλάσης η οποία το κατασκευάζει.
- `_curr_keypoints`: Τα σημεία κλειδιά σε ένα συγκεκριμένο καρέ. Χρησιμοποιείται για να ορίζεται το κέντρο του *highlight* δυναμικά καθώς αλλάζει η πόζα του ανθρώπου.
- `_curr_mode`: Ο τρόπος με τον οποίο βάφονται οι κορυφές. Η επιλογή `triangles` είναι ο συνηθισμένος τρόπος όπου βάφεται η εξωτερική όψη του κάθε τριγώνου.
- `_dx_acc & _dy_acc`: Συσσωρευτές μετακίνησης του ποντικιού στους άξονες της οθόνης `x` και `y` αντίστοιχα από την στιγμή που έγινε το πρώτο κλικ. Χρησιμοποιούνται για την ομαλή περιστροφή του αντικειμένου.
- `_highlighted_kpnt_indx`: Ο δείκτης του επιλεγμένου σημείου κλειδιού κατά την επεξεργασία του στην λειτουργία επιμελητή.
- `_mesh`: Το πλέγμα του αντικειμένου της σκηνής, κλάσης `kivy.graphics.Mesh`⁵⁶.

⁵⁵<https://kivy.org/doc/stable/api-kivy.graphics.html>

⁵⁶<https://kivy.org/doc/stable/api-kivy.graphics.html#kivy.graphics.Mesh>

- **_mesh_data:** Τα δεδομένα κορυφών, normals και δεικτών (indices) που δίνονται σαν όρισμα στην κατασκευή του `_mesh`.
- **_monkey_scene:** Το αντικείμενο τύπου `.obj` προς απεικόνιση για αποσφαλμάτωση.
- **_nframes:** Ο αριθμός των καρέ που έχουν απεικονιστεί από την αρχικοποίηση αυτής της σκηνής.
- **_recalc_normals:** Καθορίζει αν πρέπει να υπολογιστούν τα normals. Ο υπολογισμός των normals γίνεται μόνο όταν αρχικοποιηθεί η σκηνή. Αυτό δημιουργεί προβλήματα στον τρόπο που φωτίζεται το ανθρώπινο πλέγμα, αλλά ο υπολογισμός γίνεται στην CPU απαιτώντας υπολογιστική ισχύ και μειώνοντας τον αριθμό καρέ ανά δευτερόλεπτο.
- **_smpl_faces:** Πίνακας $F \times 3$ όπου καθορίζονται οι 3 κορυφές των F τριγώνων που απαρτίζουν το ανθρώπινο πλέγμα.
- **_store_quat:** Το αρχικό τετραδόνιο της περιστροφής του αντικειμένου. Χρησιμοποιείται για την ομαλή περιστροφή στο διαρκές κλικ.
- **_zoom_speed:** Η ταχύτητα με την οποία γίνεται μεγέθυνση ή συμίχρυνση του αντικειμένου με την ροδέλα.
- **canvas:** Ο καμβάς πάνω στον οποίο γίνεται η απεικόνιση.
- **cb:** Μέθοδος που καλείται όταν γίνεται η διαδικασία της παραγωγής γραφικών πάνω στον καμβά.
- **curr_obj:** Το όνομα του αντικειμένου που απεικονίζει ο Renderer.
- **keypoints_spec:** Ο προσδιορισμός των χαρακτηριστικών των σημείων κλειδιών.
- **parents:** Το κινηματικό δέντρο των σημείων κλειδιών του μοντέλου SMPL.
- **roll & pitch & yaw:** Η περιστροφή του αντικειμένου γύρω από τους άξονες σχετικά με τον προσανατολισμό του.
- **quat:** Το τετραδόνιο της περιστροφής του αντικειμένου καθώς περιστρέφεται με το διαρκές κλικ. Χρησιμοποιείται για την ομαλή περιστροφή του αντικειμένου.
- **rotx:** Η περιστροφή του αντικειμένου γύρω από τον άξονα x.
- **scale:** Η μεγέθυνση ή συμίχρυνση του αντικειμένου.
- **trans:** Η μετατόπιση του αντικειμένου.

Μέθοδοι Κλάσης

- **`__init__(smpl_faces_path, keypoints_spec, obj_mesh_path):`** Υπολογίζεται το κινηματικό δέντρο, το `parents`, με βάση το `keypoint_spec`, αρχικοποιείται ο καμβάς `canvas`, φορτώνονται οι `shaders` με κλήση της `_init_shader()`, μηδενίζονται τα χαρακτηριστικά `_store_quat`, `_dx_acc`, `_dy_acc`, `_nframes`, `curr_obj` και παίρνει την τιμή `triangles` η `_curr_mode`.

- `_create_arrow(start_point, dir_vec)`: Δημιουργεί ένα πλέγμα βέλους που ξεκινάει από το σημείο `start_point` και δείχνει προς την κατεύθυνση του διανύσματος `dir_vec`.
- `_create_error_vectors(start_verts, direction_vecs)`: Για κάθε σημείο στη λίστα `start_verts` και κατεύθυνση του `start_verts`, δημιουργεί ένα βέλος. Όλα τα βέλη υφίστανται μία περιστροφή 180 μοιρών γύρω από τον άξονα x, με χοήση του `rotx`, επειδή οι άξονες x και y του SMPL και του καμβά είναι αντιστραμμένη. Επίσης, μετατοπίζει τα βέλη κατά τον άξονα z, με χοήση του `trans`, ώστε να φαίνονται πιο έντονα και καθαρά στην απεικόνιση.
- `create_monkey_mesh() & _create_smpl_mesh() & _create_smpl_kpnts()`: Οι μέθοδοι δημιουργίας των αντικειμένων 'monkey', 'smpl mesh' και 'smpl skeleton' αντίστοιχα. Προσαρμόζουν τα δεδομένα κορυφών στην επιθυμητή μορφή με χοήση της κλάσης `GLMeshData`. Οι μέθοδοι του μοντέλου SMPL εφαρμόζουν επίσης μία περιστροφή 180 μοιρών γύρω από τον άξονα x.
- `_deform_anim(delta)`: Εφαρμόζει μία τυχαία μικρή μετατόπιση όλων των κορυφών του πλέγματος προς τυχαίες διευθύνσεις.
- `_find_closest_kpnts(pos, kpnts_2d)`: Βρίσκει το πιο κοντινό σημείο κλειδί στην θέση της οθόνης `pos` από την λίστα με τις 2D συντεταγμένες όλων των σημείων κλειδιών `kpnts_2d`, και επιστρέφει το όνομα του.
- `_get_base_vecs(vec)`: Με δεδομένο ένα διάνυσμα `vec` βρίσκει άλλα δύο διανύσματα με τα οποία δημιουργούνε μία βάση ορθοκανονικού συστήματος, και τα επιστρέφει.
- `_get_transformation_matrix()`: Υπολογίζει τον πίνακα γραμμικού μετασχηματισμού βάση όλων των περιστροφών, μετατοπίσεων και μεγεθύνσεων που έχουν γίνει.
- `_highlight_keypoint()`: Αν υπάρχει η `_highlighted_kpnt_indx`, δηλαδή κάποιο σημείο κλειδί είναι επιλεγμένο, τότε θέτει τις μεταβλητές κέντρου σφαίρας `canvas['sphere_center']` στο σημείο αυτό και ακτίνας `canvas['sphere_radius']`, που καθορίζεται από το `keypoints_spec`, τις οποίες μεταβλητές διαβάζει ο shader. Έτσι, στον shader σχηματίζεται μία σφαίρα και όσες κορυφές βρίσκονται εντός της χρωματίζονται διαφορετικά δημιουργώντας το highlight εφέ.
- `_kpnts_to_2D()`: Με βάση των πίνακα Μετασχηματισμού-Κάμερας (modelview) και των πίνακα Προβολής (projection) καθώς και των πίνακα μετασχηματισμών (transformation) υπολογίζονται η 2D συντεταγμένες των σημείων κλειδιών σε αυτό το καρέ απεικόνισης.
- `_reach_color_anim(start_color, target_color, reverse, dt)`: Παίζει ένα εφέ στο απεικονιζόμενο αντικείμενο. Το αντικείμενο ξεκινάει από το RGB χρώμα `start_color`, φτάνει με γραμμική παρεμβολή στο χρώμα `target_color`, και αν η `reverse` είναι `true` επιστρέφει ξανά στο αρχικό χρώμα με πιο αργό ρυθμό.

5.2. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΛΑΣΕΩΝ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

- `_setup_gl_context()` & `_reset_gl_context()`: Μέθοδοι που χρησιμοποιούνται ως `kivy.graphics.Callback` για να θέσουν και να επαναφέρουν αντίστοιχα το περιβάλλον δημιουργίας γραφικών στον καμβά.
- `_update_glsl()`: Θέτει το επίπεδο αποκοπής καθορίζοντας τον πίνακα προβολής.
- `_rotate_anim(delta)`: Παίζει ένα εφέ όπου το αντικείμενο περιστρέφεται συνεχόμενα γύρω από τον κατακόρυφο άξονα y.
- `_scale_anim(delta)`: Παίζει ένα εφέ όπου το αντικείμενο μεγεθύνεται και σμικρύνεται ταυτόχρονα στους άξονες x και y ακολουθώντας δύο ημιτονοειδής συναρτήσεις με διαφορά φάσης 180 μοίρες.
- `on_touch_down(touch)`: Αν υπάρχει απεικονιζόμενο πλέγμα και το κλικ έγινε στην περιοχή του widget, τότε αν το γεγονός προκλήθηκε από την ροδέλα σμικρύνεται ή μεγεθύνεται το αντικείμενο. Διαφορετικά, μηδενίζονται οι συσσωρευτές `_dx_acc` και `_dy_acc`, αποθηκεύεται το αρχικό τετραδόνιο περιστροφής στην `_store_quat` και δεσμεύεται το γεγονός του κλικ από το widget.
- `on_touch_move(touch)`: Αν το κλικ είναι δεσμευμένο από το widget του Renderer, τότε αυξάνονται οι `_dx_acc` και `_dy_acc` ανάλογα με την μετατόπιση του κλικ, μετατρέπονται οι μετατοπίσεις σε τετραδόνιο το οποίο πολλαπλασιάζεται με το αρχικό `_store_quat` παράγοντας το τετραδόνιο περιστροφής `quat`. Τέλος, το `quat` μετατρέπεται πάλι σε γωνίες και έτσι καθορίζεται η περιστροφή στα `roll`, `pitch` και `yaw`.
- `on_touch_up(touch)`: Όταν απελευθερώθει το κλικ, αν δεν είχε κουνηθεί καθόλου και δεν ήταν η ροδέλα, τότε διαχειρίζεται το μονό κλικ, υπολογίζοντας το κοντινότερο σημείο κλειδί, με χρήστη των `_kpnts_to_2D()` και `_find_closest_kpnts()`, και εκτελώντας την `single_click_handle()` αν έχει ορισθεί από το action που διαχειρίζεται αυτόν τον Renderer.
- `play_animation(animation_spec)`: Με βάση το όνομα ενός εφέ, `animation_spec`, παίζει το αντίστοιχο εφέ.
- `reset_highlight()`: Αν υπάρχει το `_highlighted_kpnt_indx`, το διαγράφει και θέτει την ακτίνα της σφαίρας, `canvas['sphere_radius']`, στο 0.
- `reset_scene()`: Καθαρίζει τον καμβά από όλα τα πλέγματα αντικειμένων και μηδενίζει την `_nframes`.
- `set_vertices(vertices, keypoints)`: Αν το απεικονιζόμενο αντικείμενο είναι το `mpl_mesh` ή το `mpl_kpnts` θέτει τις κορυφές του `GLMeshData` να είναι τα `vertices` ή `keypoints` αντίστοιχα. Επίσης, αυξάνει τον μετρητή `nframes`, κατά ένα και αφού θέσει τα καινούργια σημεία κλειδιά στην `_curr_keypoints` καλεί την `_highlight_keypoint()`.
- `setup_scene(rendered_obj, opts)`: Θέτει την `_recalc_normals` σε true, προετοιμάζει τον καμβά για απεικόνιση με OpenGL, καλώντας τις `_setup_gl_context` και στο πέρας της απεικόνισης την `_reset_gl_context`, θέτει τις αρχικές

ΚΕΦΑΛΑΙΟ 5. ΥΛΟΠΟΙΗΣΗ ΕΚΤΙΜΗΣΗΣ ΠΟΖΑΣ

τιμές των περιστροφών, μετατοπίσεων και μεγεθύνσεων, καλεί κάποια συνάρτηση δημιουργίας πλέγματος αντικειμένου με βάση το όνομα του αντικειμένου προς απεικόνιση `rendered_object` και την αντιστοιχία `_create_mesh_fn`.

6

Συμπεράσματα και Μελλοντικές Επεκτάσεις

6.1 ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ

Η παρούσα παράγραφος περιγράφει την διαδικασία που πρέπει να ακολουθηθεί για να αναπτύξει κάποιος μια εφαρμογή η οποία θα προσαρτηθεί πάνω στον καθρέφτη. Η οργάνωση του συστήματος έχει γίνει με την χρήση των κλάσεων kivy.uix.screenmanager.ScreenManager και kivy.uix.screenmanager.Screen. Τα σημεία που πρέπει να προσέξει κάποιος για να προσαρτήσει την δική του εφαρμογή είναι στην δομή των φακέλων του κώδικα, στην ορθή ανάγνωση του Widget από το λειτουργικό του καθρέφτη, οι σωστή δομή των ρυθμίσεων της προσαρτούμενης εφαρμογής και τέλος οι φωνητικές εντολές που θα είναι διαθέσιμες για τον χρήστη.

6.1.1 Δομή Φακέλων

Η δομή που έχει ο κώδικας του καθρέφτη φαίνεται στο παρακάτω δέντρο.

```
SmartMirror
├── smartmirror.py
├── smartmirror.kv
└── mirror_settings.py
└── modules/
    └── action.py
```

```

basedir.py
bot.py
controller.py
speech.py
widgets/
└── clock/
└── excisor/
└── weather/

```

Οποιαδήποτε καινούργια εφαρμογή αναπτύσσει ο χρήστης πρέπει να μπει κάτω από τον φάκελο widgets με την παρακάτω δομή.

```

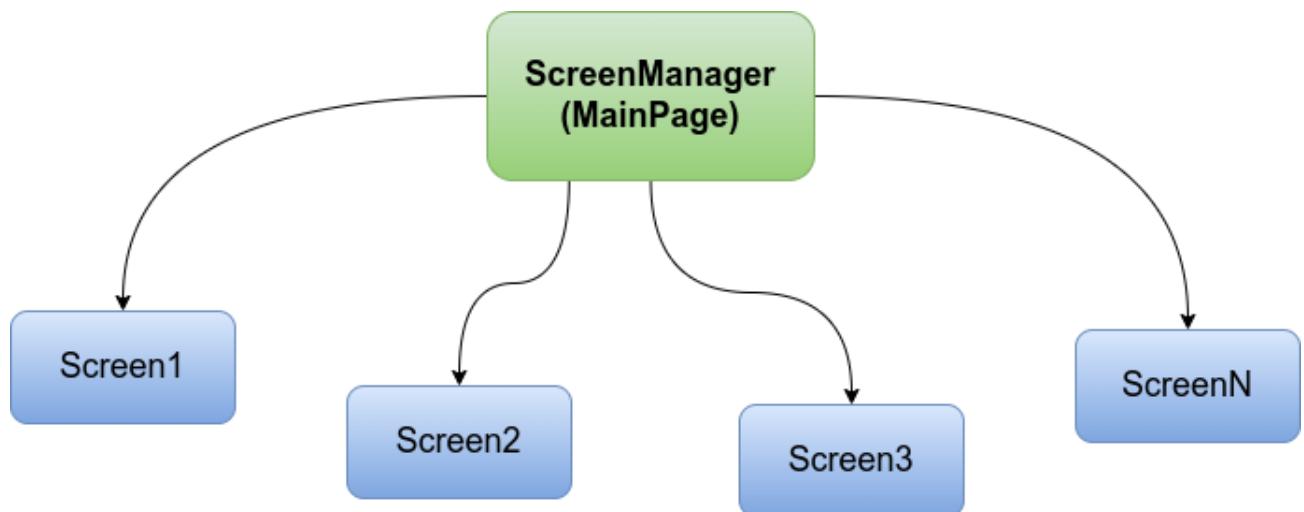
widgets/
└── <widget_name>/
    ├── <widget_name>.py
    ├── <widget_name>.kv
    └── settings.json

```

Με την παραπάνω οργάνωση το λειτουργικό είναι σε θέση να βρει αυτόματα και να προσαρτήσει την εφαρμογή.

6.1.2 Εγκατάσταση Οθόνης

Η γενική οργάνωση του καθρέφτη όσον αφορά την λογική των εξωτερικών εφαρμογών φαίνεται στο Σχήμα 6.1. Η κλάση MainPage είναι ένας ScreenManager ο οποίος διαβάζει όλες τις οθόνες που βρίσκονται στον φάκελο widgets/ και τις προσαρτίζει σε αυτόν.



Σχήμα 6.1: Αναπαράσταση δομής του καθρέφτη

Προκειμένου να γίνει η προσάρτηση, κάθε widget πρέπει να ορίσει μία συνάρ-

τηση `install(manager)` η οποία καλεί την εντολή `manager.add_widget(<Screen>)`. Κατά την εκκίνηση της εφαρμογής η κλάση `MainPage` καλεί τις συναρτήσεις `install` του κάθε `widget` και έτσι προσθέτει τις συγκεκριμένες οθόνες.

6.1.3 Ρυθμίσεις

Για την ανάγνωση των ρυθμίσεων της εφαρμογής απαιτείται η δημιουργία ενός αρχείου με όνομα `settings.json` το οποίο θα έχει 2 κλειδιά: το `"default_json"` και το `"settings_json"`. Στο πρώτο κλειδί θα ανατεθεί ένα αντικείμενο που ορίζει τις προκαθορισμένες τιμές που θα έχουν οι ρυθμίσεις, ενώ το δεύτερο κλειδί περιέχει μία λίστα με όλες τις ρυθμίσεις, τα ονόματά τους, τον τύπο τους αλλά και την προκαθορισμένη τιμή τους. Για την καλύτερη κατανόηση παρατίθεται παρακάτω ένα παράδειγμα του αρχείου ρυθμίσεων του Widget Weather.

```
{
    "default_json": {
        "api_key": "API_KEY",
        "city_id": "CITY_ID",
        "city_name": "CITY_NAME",
        "update_interval": 1800
    },
    "settings_json": [
        {
            "type": "title",
            "title": "Weather"
        },
        {
            "type": "string",
            "title": "API KEY",
            "desc": "Openweathermap API key",
            "section": "weather",
            "key": "api_key"
        },
        {
            "type": "string",
            "title": "City Id",
            "desc": "Openweathermap City's id",
            "section": "weather",
            "key": "city_id"
        },
        {
            "type": "string",
            "title": "City Name",
            "desc": "Openweathermap City's name",
            "section": "weather",
            "key": "city_name"
        },
        {
            "type": "numeric",
            "title": "Update Interval",
            "desc": "How often to update the weather in seconds",
            "section": "weather",
            "key": "update_interval"
        }
    ]
}
```

```

        "key": "update_interval"
    }
}

```

Τέλος, η εφαρμογή πρέπει να ορίσει και μία συνάρτηση με όνομα ”update_config” η οποία θα εκτελείται κάθε φορά που γίνεται ανανέωση των ρυθμίσεων προκειμένου να ενημερωθούν οι τιμές των μεταβλητών του Widget. Σε αυτήν την περίπτωση κανείς μπορεί να ανατρέξει στο αρχείο ”weather.py” του Widget Weather για ένα παράδειγμα ανάγνωσης ρυθμίσεων μέσω του Kivy και υλοποίηση της συνάρτησης ”update_config”.

6.1.4 Φωνητικές Εντολές

Για τον καθορισμό των διαθέσιμων εντολών που παρέχει η εφαρμογή απαιτείται η δημιουργία μιας συνάρτησης με όνομα ”subscribe” η οποία επιστρέφει ένα dictionary το οποίο κάνει map τις προθέσεις σε συναρτήσεις που θα κληθούν. Ένα παράδειγμα των διαθέσιμων εντολών του Widget Weather φαίνεται παρακάτω

```

{
    "request_location": self.request_location,
    "request_hour": self.request_hour,
    "request_day": self.request_day
}

```

Το παραπάνω dictionary υποδεικνύει τρεις εντολές, ανάλογα με την πρόθεση που θα αναγνωρίσει το Wit.ai

Ο χρήστης που αναπτύσσει την εφαρμογή είναι υπεύθυνος να εκπαιδεύσει το Wit.ai με τα δικά του δεδομένα στις δικές τους εντολές και επομένως Θα χρειαστεί να συνδεθεί στην πλατφόρμα στο <https://wit.ai>. Ένα βασικό πακέτο εντολών είναι διαθέσιμο στον φάκελο assets/ το οποίο θα μπορεί να εισαχθεί απευθείας στο wit και από εκεί να επεκταθεί ανάλογα τις ανάγκες του χρήστη.

6.2 ΣΥΜΠΕΡΑΣΜΑΤΑ

Η παρούσα διπλωματική εργασία υλοποιεί ένα ελαφρύ, αρθρωτό λειτουργικό σύστημα για χρήση σε έξυπνους καθρέφτες με στόχο την εύκολη ανάπτυξη εφαρμογών που αφορούν τον τομέα της υγείας. Το γραφικό περιβάλλον του λειτουργικού είναι σχεδιασμένο με τρόπο που επαυξάνει την ανακλαστική επιφάνεια του κλασσικού καθρέφτη χαρίζοντας την δυνατότητα στον χρήστη για ανάγνωση πληροφορίων ενώ ταυτόχρονα παραμένει ορατό το είδωλό του. Επίσης, η αλληλεπίδραση μεταξύ χρήστη και συστήματος επιτυγχάνεται μέσω φωνητικών εντολών, οι οποίες αναγνωρίζονται από τον έξυπνο καθρέφτη που με τη σειρά του εκτελεί κατάλληλες

6.3. ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

επιθυμητές ενέργειες. Τέλος, ιδιαίτερη μνεία έχει δοθεί στην επεκτασιμότητα του συστήματος προκειμένου να μπορεί ο καθένας να αναπτύξει χωρίς μεγάλη δυσκολία την δική του εφαρμογή πάνω στον έξυπνο καθρέφτη.

Ταυτόχρονα, αναπτύχθηκε και ενσωματώθηκε μία εφαρμογή για την εκτίμηση της ορθότητας μιας άσκησης. Ο χρήστης θα μπορεί να αθλείται μπροστά στον έξυπνο καθρέφτη και να λαμβάνει ανατροφοδότηση σχετικά με την απόδοσή του κατά την εκτέλεση της άσκησης. Οι ασκήσεις αναφοράς βάση των οποίων ελέγχεται η ορθότητα της εκτελούμενης άσκησης εισάγονται από τον χρήστη σε μορφή βίντεο μιας ορθής επανάληψης, ενώ παράλληλα δίνεται η δυνατότητα επεξεργασίας τους. Κατά αυτόν τον τρόπο είναι εφικτή η διεύρυνση των διαθέσιμων ασκήσεων προς αναπαραγωγή και η εξατομίκευση τους στις ανάγκες του κάθε χρήστη.

Το λογισμικό δοκιμάστηκε να εκτελεστεί πάνω στο Raspberry Pi 4 Model B⁵⁷, κάτι που δεν πέτυχε, αφού εξαιτίας των απαιτήσεων της εφαρμογής για τον έλεγχο της άσκησης η οποία χρησιμοποιεί νευρωνικά δίκτυα δεν ήταν εφικτή η ομαλή λειτουργία του συστήματος.

6.3 ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Μια λύση στο πρόβλημα επιδόσεων του Raspberry Pi θα ήταν η ενσωμάτωση του λογισμικού στο Nvidia Jetson Nano⁵⁸ το οποίο έχει αυξημένες δυνατότητες, λόγω της κάρτας γραφικών, για εκτέλεση αλγορίθμων μηχανικής μάθησης. Επίσης, ο κώδικας του Controller του λειτουργικού συστήματος χρήζει βελτίωσης για καλύτερο έλεγχο των εγκατεστημένων εφαρμογών, όπως παραλληλοποίηση των εργασιών μέσω threads, αλλά και μεγαλύτερη εξοικονόμηση ενέργειας σε περιόδους αδράνειας του έξυπνου καθρέφτη.

Όσον αφορά την εφαρμογή εκτίμησης πόζας, προτείνονται δύο τομείς για βελτίωση ή επέκταση. Από την μία πλευρά, η απεικόνιση του μοντέλου του χρήστη μπορεί να ωραιοποιηθεί ικανοποιώντας καλύτερα τις αισθητικές απαιτήσεις του. Προς επίτευξη αυτού του σκοπού μπορούν να αξιοποιηθούν οι δυνατότητες της OpenGL με σκοπό την ομορφότερη απεικόνιση του πλέγματος του ανθρώπινου μοντέλου ή να ενσωματωθούν μοντέλα μηχανικής μάθησης όπως το pix2surf⁵⁹ εξατομικεύοντας την πρόσοψη του χρήστη. Από την άλλη πλευρά, η λειτουργία επεξεργασίας των δεδομένων των ασκήσεων αναφοράς θα μπορούσε να λειτουργήσει ως ένα framework σχολιασμού δεδομένων για εκπαίδευση αντίστοιχων μοντέλων μηχανικής μάθησης.

Επιπλέον, οι δυνατότητες του καθρέφτη μπορούν να επαυξηθούν μελλοντικά με προσάρτηση νέων εφαρμογών που αφορούν την υγεία, αφού ο έξυπνος καθρέφτης είναι αρκετά βολικότερος για την λήψη βιομετρικών μετρήσεων σε σχέση με την

⁵⁷<https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>

⁵⁸<https://developer.nvidia.com/embedded/jetson-nano>

⁵⁹<https://github.com/aymenmir1/pix2surf>

ΚΕΦΑΛΑΙΟ 6. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

επίσκεψη σε μια κλινική ή ένα νοσοκομείο. Στο [32] παρουσιάζονται αρκετοί τομείς της υγείας στους οποίους ο έξυπνος καθρέφτης μπορεί να βοηθήσει τους ασθενείς. Η αναγνώριση συναισθημάτων, η εκτίμηση ρίσκου για καρδιαγγειακά νοσήματα, η λήψη μετρικών σχετικά με το αίμα όπως το ζάχαρο και η μέτρηση καρδιακών παλμών αποτελούν ενδεικτικές εφαρμογές για μελλοντική έρευνα και ανάπτυξη.

Εντούτοις, για την προσάρτηση νέων εφαρμογών θα ήταν ιδιαιτέρως βολική η ύπαρξη ενός κεντρικού διακομιστή ο οποίος θα συμβάλλει στην εύρωστη και εύχρηστη λειτουργία του έξυπνου καθρέφτη. Έτσι, θα γίνει εφικτή η αυτοματοποίηση εγκατάστασης και αναβάθμισης εξωτερικών εφαρμογών και η λήψη δεδομένων που συμπληρώνουν τη λειτουργία τους. Για παράδειγμα, στην υπάρχουσα εφαρμογή εκτίμησης πόζας θα δίνεται η δυνατότητα σε έναν ειδικό ιατρό ή φυσιοθεραπευτή να παρέχει απευθείας τις ασκήσεις αναφοράς.

Κλείνοντας, παρά την δεδομένη αξία που μπορεί να προσδώσει ο έξυπνος καθρέφτης στους τομείς της υγείας και της ευεξίας, εξακολουθούν να χρειάζονται ακόμη αρκετές προσπάθειες για την ευρεία υιοθέτησή του. Η ανάπτυξη ενός ενοποιημένου λειτουργικού συστήματος στο οποίο θα μπορούν όλοι να έχουν πρόσβαση, η συνεχής πρόοδος των αλγορίθμων μηχανικής μάθησης που θα επιτρέπουν την λήψη αξιόπιστων αποτελεσμάτων αλλά και η βελτιστοποίηση της αναγνώρισης φωνής για την ευκολότερη αλληλεπίδραση μεταξύ του καθρέφτη είναι κάποιοι από τους παράγοντες που θα βοηθήσουν στην διάδοση του έξυπνου καθρέφτη τόσο στις κλινικές όσο και στα σπίτια.

Βιβλιογραφία

- [1] Boehm. “*Software Engineering*“. IEEE Transactions on Computers, C-25(12): 1226–1241, 1976.
- [2] Abiy Biru Chebudie, Roberto Minerva, and Domenico Rotondi. “*Towards a definition of the Internet of Things (IoT)*“. PhD thesis, 08 2014.
- [3] Pallavi Sethi and Smruti R. Sarangi. “*Internet of Things: Architectures, Protocols, and Applications*“. JECE, 2017, January 2017. ISSN 2090-0147.
- [4] William Stallings. “*Operating Systems: Internals and Design Principles*“. Pearson, 9 edition, 2018.
- [5] Behrouz Forouzan and Firouz Mosharraf. “*Foundations of computer science*“. Cengage Learning EMEA, 2 edition, 2008.
- [6] Noam Nisan and Shimon Schocken. “*The Elements of Computing Systems: Building a Modern Computer from First Principles*“. The MIT Press, 2 edition, 2021.
- [7] Mark Segal and Kurt Akeley. “*The OpenGL Graphics System: A Specification (Version 4.6 (Core Profile) - October 22, 2019)*“, 2021. URL <https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf>.
- [8] Hans van Vliet. “*Software Engineering: Principles and Practice*“. Wiley, 2007.
- [9] Roger S. Pressman and Bruce R. Maxim. “*Software Engineering: A practitioner’s approach*“. McGraw-Hill Education, 8 edition, 2014. ISBN 978-0-07-802212-8.
- [10] Linda G. Shapiro and George C. Stockman. “*Computer Vision*“. Prentice Hall, 2001. ISBN 0-13-030796-3.
- [11] Riza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. “*DensePose: Dense Human Pose Estimation in the Wild*“. In “*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*“, June 2018.
- [12] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. “*SMPL: A Skinned Multi-Person Linear Model*“. ACM Transactions on Graphics, (Proc. SIGGRAPH Asia), 34(6):248:1–248:16, October 2015.

- [13] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G. Derpanis, and Kostas Daniilidis. “*Coarse-To-Fine Volumetric Prediction for Single-Image 3D Human Pose*“. In “*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*“, July 2017.
- [14] Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang, and Yichen Wei. “*Deep Kinematic Pose Regression*“, 2016.
- [15] Xiao Sun, Jiaxiang Shang, Shuang Liang, and Yichen Wei. “*Compositional Human Pose Regression*“. In “*Proceedings of the IEEE International Conference on Computer Vision (ICCV)*“, October 2017.
- [16] Diogo C. Luvizon, Hedi Tabia, and David Picard. “*Human pose regression by combining indirect part detection and contextual information*“. *Computers & Graphics*, 85:15–22, 2019. ISSN 0097-8493.
- [17] Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast. “*3D Human Pose Estimation with 2D Marginal Heatmaps*“. arXiv preprint arXiv:1806.01484, 2018.
- [18] Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast. “*Numerical Coordinate Regression with Convolutional Neural Networks*“. arXiv preprint arXiv:1801.07372, 2018.
- [19] Julieta Martinez, Rayat Hossain, Javier Romero, and James J. Little. “*A Simple yet Effective Baseline for 3D Human Pose Estimation*“. In “*Proceedings of the IEEE International Conference on Computer Vision (ICCV)*“, October 2017.
- [20] Denis Tome, Chris Russell, and Lourdes Agapito. “*Lifting From the Deep: Convolutional 3D Pose Estimation From a Single Image*“. In “*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*“, July 2017.
- [21] Georgios Pavlakos, Xiaowei Zhou, and Kostas Daniilidis. “*Ordinal Depth Supervision for 3D Human Pose Estimation*“. In “*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*“, June 2018.
- [22] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. “*Keep It SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image*“. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, “*Computer Vision – ECCV 2016*“, pages 561–578, Cham, 2016. Springer International Publishing.
- [23] Nikos Kolotouros, Georgios Pavlakos, Michael J. Black, and Kostas Daniilidis. “*Learning to Reconstruct 3D Human Pose and Shape via Model-Fitting in the Loop*“. In “*Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*“, October 2019.
- [24] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. “*End-to-End Recovery of Human Shape and Pose*“. In “*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*“, June 2018.

- [25] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. “*Learning to Estimate 3D Human Pose and Shape From a Single Color Image*“. In “*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*“, June 2018.
- [26] “*Kivy Documentation*“. URL <https://kivy.org/doc/stable/>.
- [27] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J. Black, and Peter V. Gehler. “*Unite the People: Closing the Loop Between 3D and 2D Human Representations*“. CoRR, abs/1701.02468, 2017.
- [28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “*Identity Mappings in Deep Residual Networks*“, 2016.
- [29] Abdul Majeed and Ibtisam Rauf. “*MVC Architecture: A Detailed Insight to the Modern Web Applications Development*“. 2018.
- [30] P. Bourque and R.E. Fairley. “*Guide to the Software Engineering Body of Knowledge*“. 2014.
- [31] Alireza Souri, Mohammad Sharifloo, and Monire Norouzi. “*Formalizing class diagram in UML*“. 07 2011.
- [32] Riccardo Miotto, Matteo Danieletto, Jerome Scelza, and Brian Kidd. “*Reflecting health: smart mirrors for personalized medicine*“. npj Digital Medicine, 1, 12 2018.

Παράρτημα

A' ΟΡΙΣΜΟΙ

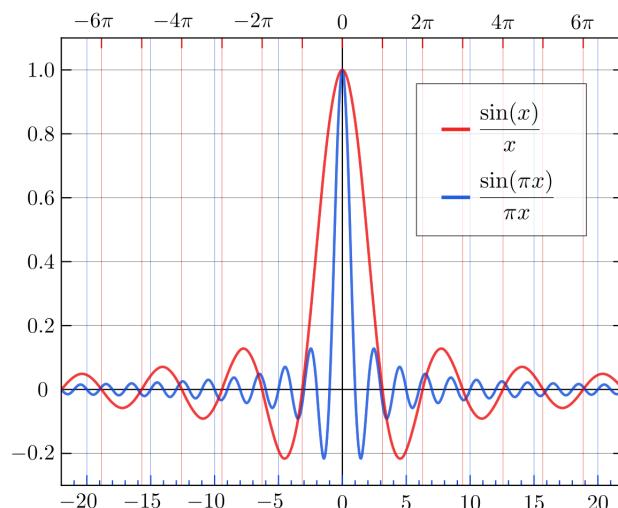
A.1 Συναρτήσεις ενεργοποίησης

Συνάρτηση argmax

Δεδομένων ενός τυχαίου συνόλου X , ενός συνόλου Y και μιας συνάρτησης $f : X \rightarrow Y$, το μέγιστο όρισμα, argmax, πάνω σε ένα υποσύνολο $S \subseteq X$ ορίζεται ως

$$\operatorname{argmax}_S f := \operatorname{argmax}_{x \in S} f(x) := \{x \in S \mid f(s) \leq f(x) \forall s \in S\}$$

Με άλλα λόγια, η συνάρτηση argmax είναι το σύνολο των σημείων x για τα οποία η $f(x)$ παίρνει την μέγιστη τιμή της.



Σχήμα A'. 1: Για παράδειγμα, οι συναρτήσεις sinc της γραφικής παράστασης έχουν και οι δύο $\operatorname{argmax} = 0$ επειδή έχουν στο $x = 0$ μέγιστο το 1.

Συνάρτηση soft-argmax

Η συνάρτηση soft-argmax $f : \mathbb{R}^K \rightarrow [0, 1]^K$ ορίζεται όταν $\|K\| > 1$ ως

$$\text{soft_argmax} = \sigma(\mathbf{x})_i := \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \text{ για } i = 1, \dots, K \text{ και } \mathbf{x} = (x_1, \dots, x_K) \in \mathbb{R}^K$$

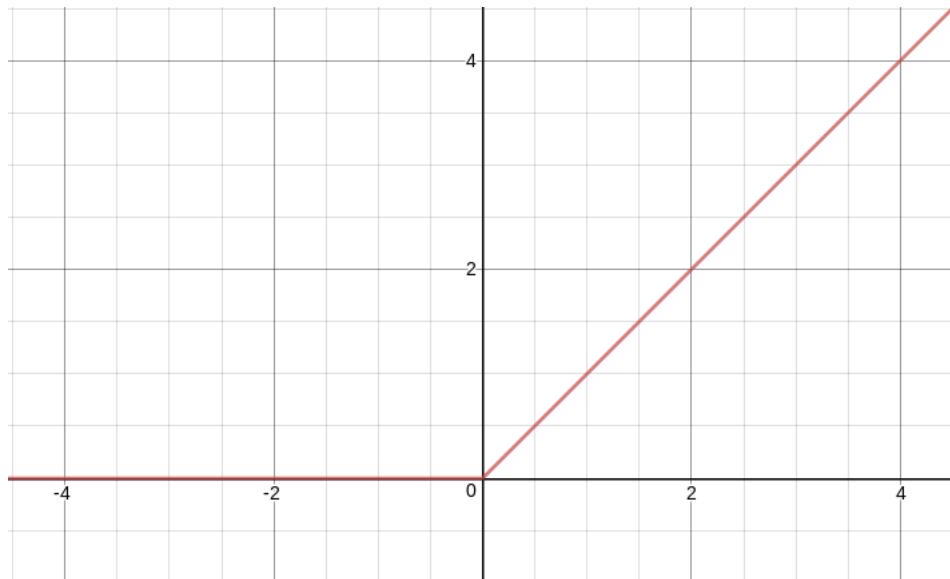
Ουσιαστικά, η συνάρτηση soft-argmax εφαρμόζει την κανονική εκθετική συνάρτηση σε κάθε στοιχείο x_i του διανύσματος \mathbf{x} και κανονικοποιεί τις τιμές διαιρώντας με το άθροισμα όλων των εκθετικών. Η κανονικοποίηση αυτή εξασφαλίζει ότι το άθροισμα των στοιχείων του διανύσματος εξόδου $\sigma(\mathbf{x})$ είναι 1.

Συνάρτηση ReLU

Η συνάρτηση ReLU $f : \mathbb{R} \rightarrow \mathbb{R}^+$ ορίζεται ως

$$f(x) = \max(0, x)$$

Δηλαδή, η συνάρτηση έχει ως έξοδο το όρισμα της όταν $x > 0$, ενώ όταν $x \leq 0$ η έξοδος είναι 0.



Σχήμα Α'. 2: Η συνάρτηση ReLU

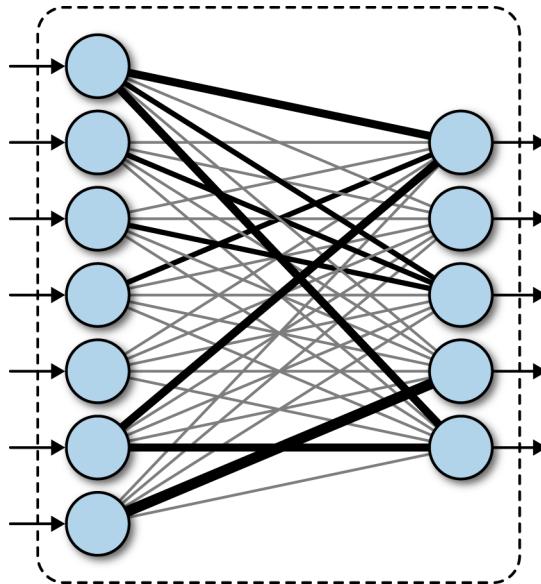
A.2 Επίπεδα νευρωνικών δικτύων

Επίπεδο Fully Connected

Έστω το διάνυσμα εισόδου $\mathbf{x} \in \mathbb{R}^N$ ενός πλήρως συνδεδεμένου επίπεδου (fully connected layer) με N κόμβους εισόδου. Τότε, η έξοδος του i κόμβου, $y_i \in \mathbb{R}$, θα δίνεται από

$$y_i(\mathbf{x}) = f\left(\sum_{j=1}^N w_{ij}x_j + w_{i0}\right)$$

Ουσιαστικά, κάθε κόμβος εφαρμόζει έναν γραμμικό μετασχηματισμό στο διάνυσμα εισόδου μέσω του γινομένου του με τον πίνακα βαρών. Στο άθροισμα προστίθεται ένας σταθερός όρος ανά κόμβο το bias w_{i0} . Τέλος, στο αποτέλεσμα εφαρμόζεται ένας ενδεχομένως μη-γραμμικός μετασχηματισμός μέσω της συνάρτησης ενεργοποίησης f .



Σχήμα A'. 3: Επίπεδο Fully Connected. Στο σχήμα φαίνεται ένα πλήρως συνδεδεμένο επίπεδο όπου όλοι οι κόμβοι εισόδου συνδέονται με όλους του κόμβους εξόδου. Οι ακμές με ποιο έντονο χρώμα υποδηλώνουν μεγαλύτερος βάρος μεταξύ του κόμβου εισόδου και εξόδου.

Επίπεδο Batch Normalization

Η κανονικοποίηση παρτίδας (batch normalization) είναι μία μέθοδος στα νευρωνικά δίκτυα για την παραγωγή πιο γρήγορων και σταθερών αποτελεσμάτων. Η

κανονικοποίηση παρτίδας στην είσοδο του κάθε επιπέδου γίνεται με την προσαρμογή της μέσης τιμής και της διακύμανσης των δεδομένων της παρτίδας, κατά την διαδικασία της εκπαίδευσης.

Έστω ότι B είναι μία παρτίδα μεγέθους m δεδομένων. Τότε, η εμπειρική μέση τιμή και διακύμανση της B θα είναι:

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i \text{ και } \sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$$

Για ένα επίπεδο του δικτύου με είσοδο διαστάσεων d , $\mathbf{x} = (x^{(1)}, \dots, x^{(d)})$, κάθε διάσταση της εισόδου κανονικοποιείται ξεχωριστά,

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)2} + \epsilon}}, \text{ όπου } k \in [1, d] \text{ και } i \in [1, m]$$

Η $x_i^{(k)}$ και $\sigma_B^{(k)2}$ είναι η ανά διάσταση μέση τιμή και διακύμανση αντίστοιχα. Το ϵ στον παρανομαστή είναι μία αυθαίρετη μικρή σταθερά για αριθμητική σταθερότητα.

Η τελική κανονικοποιημένη είσοδος $\hat{x}_i^{(k)}$ έχει μηδενική μέση τιμή και μοναδιαία διακύμανση, αν δεν λάβουμε υπόψη το ϵ . Για να αποκατασταθεί η αναπαραστατική ικανότητα του δικτύου, ένα βήμα μετασχηματισμού ακολουθεί

$$y_i^k = \gamma^{(k)} \hat{x}_i^{(k)} + \beta^{(k)}$$

όπου οι παράμετροι $\gamma^{(k)}$ και $\beta^{(k)}$ μαθαίνονται κατά την εκπαίδευση.

Επίσημα, η διαδικασία που υλοποιεί την κανονικοποίηση παρτίδας είναι ο μετασχηματισμός $BN_{\gamma^{(k)}, \beta^{(k)}} : x_{1\dots m}^{(k)} \rightarrow y_{1\dots m}^{(k)}$, ονομαζόμενος Batch Normalizing transform. Η έξοδος του μετασχηματισμού $y^{(k)} = BN_{\gamma^{(k)}, \beta^{(k)}}(x^{(k)})$ περνάει στα επόμενα επίπεδα του δικτύου, ενώ η κανονικοποιημένη είσοδος $\hat{x}_i^{(k)}$ παραμένει εσωτερικά του επιπέδου κανονικοποίησης παρτίδας.

Επίπεδο Dropout

Το επίπεδο Dropout των νευρωνικών δικτύων είναι μία μέθοδος ελάττωσης της υπερπροσαρμογής (overfitting) στα δεδομένα. Αυτό πετυχαίνεται παραλείποντας κόμβους είτε εμφανής είτε κρυμμένους κατά την διαδικασία της εκπαίδευσης.

Για παράδειγμα σε ένα απλό πλήρως συνδεδεμένο γραμμικό επίπεδο η έξοδος περιγράφεται από

$$y_i = \sum_j w_{ij}x_j \text{ ή σε μορφή διανυσμάτων } \mathbf{y} = \mathbf{W}\mathbf{x}$$

όπου y_i η έξοδος του κόμβου i , w_{ij} τα βάρη πριν από το επίπεδο dropout και x_j η είσοδος από τον κόμβο j .

Η έξοδος του επιπέδου dropout ορίζεται ως

$$\hat{w}_j = \begin{cases} w_j, & \text{με } P(c) \\ 0, & \text{διαφορετικά} \end{cases}$$

όπου $P(c)$ η πιθανότητα να παραμείνει η γραμμή του πίνακα βαρών.

Με άλλα λόγια, το επίπεδο dropout μηδενίζει τυχαία κάποιους κόμβους του προηγούμενου επιπέδου. Η διαδικασία που το πετυχαίνει αυτό, θέτοντας τα βάρη ίσα με το μηδέν, αφαιρώντας εξολοκλήρου τον κόμβο ή με κάποιον άλλον τρόπο, δεν επηρεάζει το τελικό αποτέλεσμα.

Επίπεδο Convolution

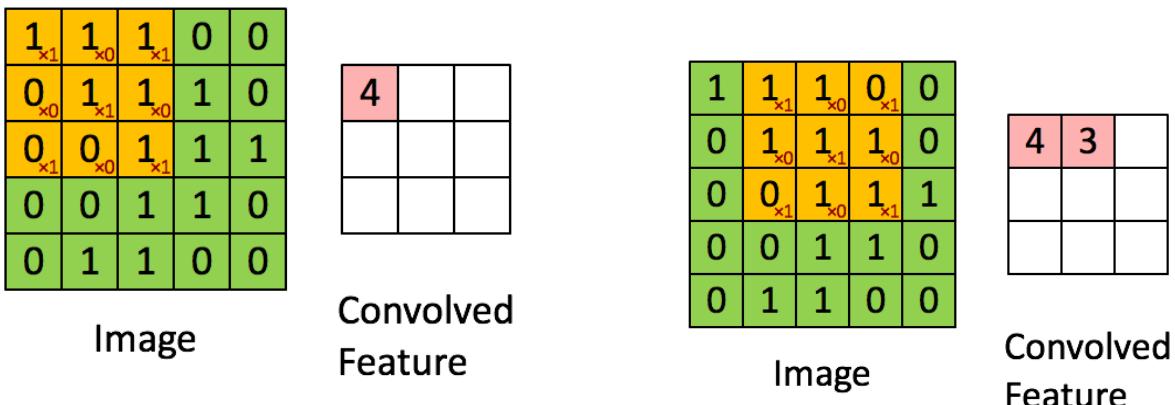
Το convolution επίπεδο χρησιμοποιείται κατά κόρων σε διεργασίες υπολογιστικής όρασης καθώς είναι άκρως αποδοτικό στην επεξεργασία εικόνων και στην εξαγωγή χαρακτηριστικών από αυτές. Η επεξεργασία γίνεται χωρικά πάνω στην εικόνα συλλαμβάνοντας έτσι αλληλεξαρτήσεις των πίξελ. Επιπλέον, απαιτεί μικρό αριθμό παραμέτρων και τα βάρη είναι επαναχρησιμοποιήσιμα βελτιώνοντας σημαντικά την εκπαίδευση.

Στο συνελικτικό επίπεδο (convolution layer) ένα φίλτρο ή πυρήνας περνάει διαδοχικά πάνω από τα 2D δεδομένα εισόδου, όπως φαίνεται στο Σχήμα A. 4, πολλαπλασιάζοντας ανά στοιχείο τα στοιχεία του πίνακα στην συγκεκριμένη θέση και αθροίζοντας τα. Με αυτόν τον τρόπο, η αρχική εικόνα ή χάρτης χαρακτηριστικών μετατρέπεται σε έναν διαφορετικό χάρτη χαρακτηριστικών μικρότερων διαστάσεων, καθιστώντας πιο εύκολη και γρήγορη την επεξεργασία του.

Φυσικά, κάποια χαρακτηριστικά της εικόνας χάνονται σε αυτή την διαδικασία. Εντούτοις, τα κύρια χαρακτηριστικά που είναι απαραίτητα για την πρόβλεψη του δικτύου διατηρούνται καθώς το δίκτυο κατά την εκπαίδευση έχει βελτιστοποιήσει τις παραμέτρους των απαιτούμενων φίλτρων για την διεξαγωγή της πρόβλεψης.

Επίπεδο Pooling

Αντίστοιχα με το convolution επίπεδο, ο σκοπός του επιπέδου pooling είναι να μειώσει το χωρικό μέγεθος του χάρτη χαρακτηριστικών. Τοιουτοτρόπως, μειώνοντας τις διαστάσεις επιτυγχάνεται μείωση της υπολογιστικής ισχύς που απαιτείται



(α') Η θέση του πυρήνα φαίνεται με κίτρινο χρώμα και με κόκκινη γραμματοσειρά οι τιμές του σε κάθε στοιχείο. Για τον υπολογισμό του χάρτη χαρακτηριστικών γίνεται πολλαπλασιασμός ανά στοιχείο και άθροιση.

(β') Στη συνέχεια, ο πυρήνας κινείται κατά μια θέση δεξιά. Θα συνεχίσει να κινείται πάνω από τα δεδομένα εισόδου μέχρι να σαρώσει όλη την περιοχή.

Σχήμα Α'. 4: Αριθμητικό παράδειγμα συνελικτικού επιπέδου

για την επεξεργασία των δεδομένων. Επιπλέον, είναι χρήσιμο στην εξαγωγή των επικρατέστερων χαρακτηριστικών που είναι αμετάβλητα κατά την περιστροφή και μετακίνηση, βελτιώνοντας έτσι την απόδοση της εκπαίδευσης.

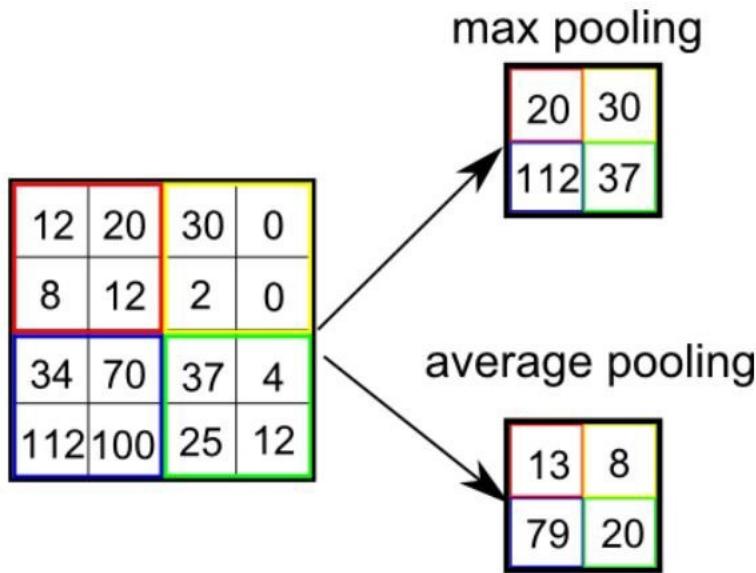
Υπάρχουν δύο είδη επιπέδων pooling, το max pooling και το average pooling, όπως φαίνεται στο Σχήμα Α'. 5. Στην περίπτωση του max pooling, η έξοδος σε κάθε θέση του πυρήνα είναι το μέγιστο, ενώ στο average pooling η έξοδος σε κάθε θέση είναι ο μέσος όρος των στοιχείων που καλύπτονται από τον πυρήνα.

A.3 Γιπόλοιπα

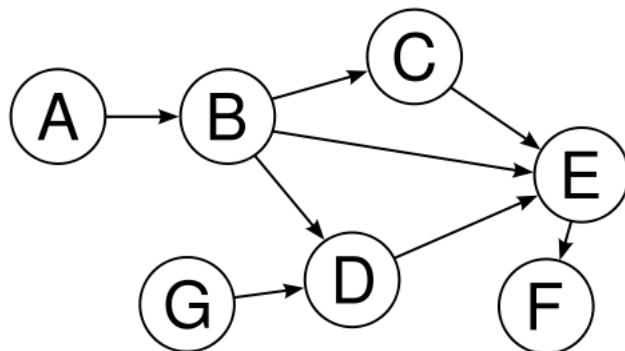
Κατευθυνόμενος Άκυκλος Γράφος

Ένας γράφος ονομάζεται κατευθυνόμενος άκυκλος γράφος αν είναι κατευθυνόμενος και δεν περιέχει κύκλους. Με άλλα λόγια, αποτελείται από ακμές και κόμβους, με κάθε ακμή να κατευθύνεται από έναν κόμβο σε κάποιον άλλο, ενώ ταυτόχρονα ακολουθώντας τις κατευθύνσεις δεν δημιουργείται ποτέ ένας κλειστός βρόγχος. Δηλαδή για κανένα κόμβο δεν υπάρχει μονοπάτι (πέρα από το τετριμμένο) που να ξεκινά από αυτόν και να καταλήγει σ' αυτόν.

Ένας κατευθυνόμενος γράφος είναι άκυκλος αν και μόνο αν μπορεί να ταξινομηθεί τοπολογικά, τοποθετώντας τους κόμβους σε μία γραμμική διάταξη όντας σύμφωνη με τις κατευθύνσεις των ακμών, όπως φαίνεται στο παράδειγμα του Σχήματος Α'. 6.



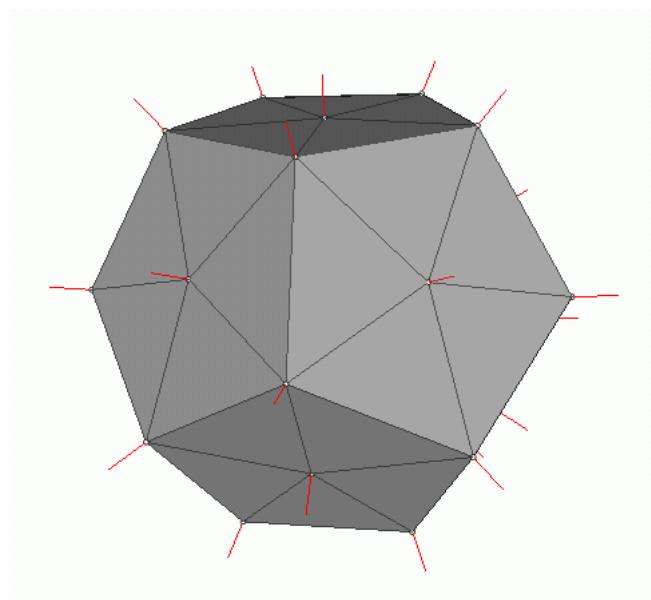
Σχήμα Α'. 5: Τύποι επιπέδων pooling. Στο σχήμα φαίνεται η έξοδος των επιπέδων max pooling (πάνω δεξιά) και average pooling (κάτω δεξιά) όταν τα δεδομένα εισόδου είναι ο πίνακας στα αριστερά. Σημειώνεται ότι ο πυρήνας και στις δύο περιπτώσεις κινείται πάνω στα δεδομένα με βήμα 2.



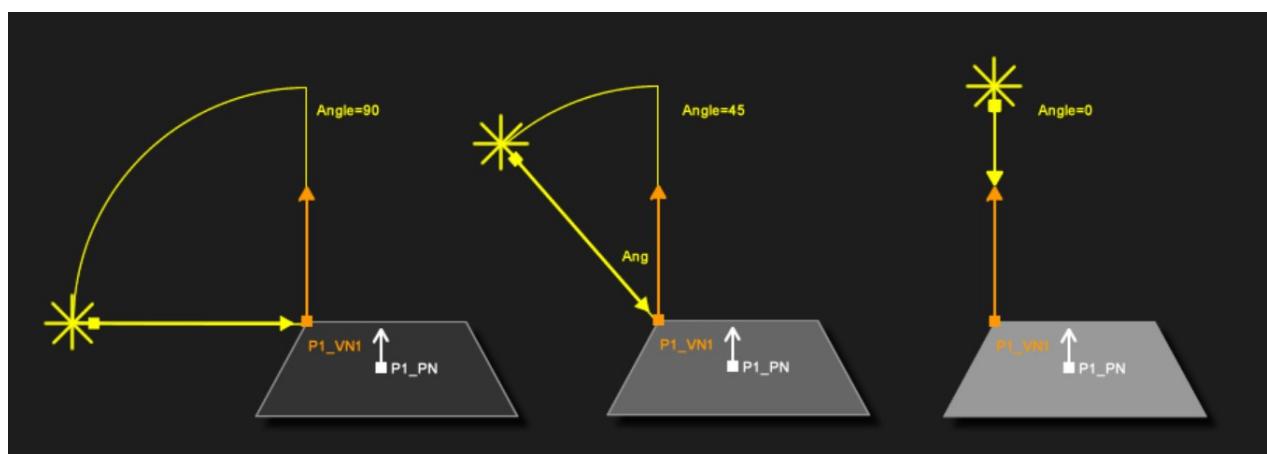
Σχήμα Α'. 6: Παράδειγμα κατευθυνόμενου άκυκλου γράφου

Κατακόρυφο διάνυσμα κορυφής

Στη γεωμετρία των γραφικών υπολογιστών, το κανονικό διάνυσμα κορυφής (*vertex normal*) σε μια κορυφή ενός πολυέδρου είναι το ευκλείδειο διάνυσμα κατεύθυνσης, προοριζόμενο ως αντικατάσταση του γεωμετρικού διανύσματος κάθετο στην επιφάνεια. Συνήθως, υπολογίζεται ως ο κανονικοποιημένος μέσος όρος των κάθετων διανυσμάτων επιφάνειας των πλευρών που περιέχουν την κορυφή, όπως φαίνεται στο Σχήμα Α'. 7. Τα vertex normals χρησιμοποιούνται για τον υπολογισμό της αντανάκλασης του φωτός όπως φαίνεται στο Σχήμα Α'. 8.



Σχήμα A'. 7: Τα κάθετα διανύσματα ενός δωδεκαέδρου πλέγματος σε κάθε κορυφή των τριγώνων που το απαρτίζουν.



Σχήμα A'. 8: Καθώς η γωνία μεταξύ του διανύσματος φωτός και του vertex normal τείνει προς το μηδέν ο φωτισμός που δέχεται η κορυφή αυξάνεται στο μέγιστο. Αυτή η αρχή είναι η βάση όλων των υπολογισμών τρισδιάστατης σκίασης και δείχνει την σημαντικότητα των vertex normals.