ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΜΣ: «Μηχανική Λογισμικού για Διαδικτυακές & Φορητές Εφαρμογές»

«Ανάπτυξη τεχνικών ανίχνευσης βλέμματος»

Επιβλέπων καθηγητής: **Σάββας Κ. Ηλίας**

Αδάμου-Τζανή Μαρίνα

mariadam4@uth.gr

Α.Μ. 7419001

Οκτώβριος 2021

Περιεχόμενα

Πίνακας Εικόνων

1. Περίληψη

Οι εφαρμογές ανίχνευσης βλέμματος χρησιμοποιούνται πλέον από ένα πλήθος τομέων, όπως είναι το marketing, η ψυχολογία και η ασφάλεια, όμως εμείς θα επικεντρωθούμε στην ανίχνευση βλέμματος για την βελτίωση ανάπτυξης και σχεδιασμού ιστοσελίδων. Σε αυτόν τον τομέα υπάρχουν νέες τεχνολογίες που συνεχώς αναπτύσσονται, όπως για παράδειγμα οι νέες κάμερες υψηλής ακρίβειας, καθώς και οι νέες βιβλιοθήκες γλωσσών προγραμματισμού που ασχολούνται όλο και περισσότερο με την «Όραση Υπολογιστών (Computer Vision)», δηλαδή της ικανότητας ενός υπολογιστή να «αντιλαμβάνεται» ψηφιακές εικόνες.

Σκοπός της παρούσας εργασίας είναι να δημιουργηθεί ένα πρόγραμμα ανίχνευσης βλέμματος χρησιμοποιώντας την γλώσσα προγραμματισμού Python, το οποίο χρησιμοποιώντας μία web camera ελάχιστου κόστους, να ανιχνεύει το βλέμμα του χρήστη στην οθόνη καθώς παρατηρεί για λίγα δευτερόλεπτα μία σειρά από ιστοσελίδες, και να εμφανίζει τα αποτελέσματα. Αυτά θα συγκριθούν με αντίστοιχα αποτελέσματα που θα παραχθούν από επαγγελματική υπέρυθρη κάμερα ανίχνευσης βλέμματος υψηλού κόστους, με σκοπό να αποδείξουμε ότι η ανίχνευση βλέμματος δεν χρειάζεται πλέον να εξαρτάται από ακριβά εξαρτήματα και τεχνολογίες, και ότι μπορεί να είναι εύκολα προσβάσιμη και αξιοποιήσιμη από όλους, ανεξαρτήτως κόστους.

Λέξεις κλειδιά: Τεχνικές, Ανίχνευση βλέμματος, Ανάπτυξη λογισμικού, Computer Vision, Σχεδιασμός Ιστοσελίδων

1. Abstract

Eye-tracking applications are now used by a number of scientific sectors, such as marketing, psychology and security/privacy, but we will focus on using eye-tracking in order to improve web development and design. In this area there are new technologies that are constantly evolving, such as new high-precision cameras, as well as novel programming libraries that are focusing in "Computer Vision", meaning the ability of a computer to “understand” digital images.

The purpose of this thesis is to create an eye-tracking program using Python programming language, which, using a low-cost web camera, detects the users’ gaze on the screen, while several images of websites are presented in front of them for a few seconds. Lastly, it displays the results, which will be compared to similar results generated by a professional high-cost infrared eye-tracking camera, to demonstrate that eye-tracking no longer has to depend on expensive components and technologies, and that it can be easily accessible and usable by anyone, regardless of the cost.

Keywords: Techniques, Eye Detection, Software Development, Computer Vision, Website Design

2. Εισαγωγή

Το ανθρώπινο βλέμμα προσελκύεται από διαφορετικά ερεθίσματα σε μία οθόνη, ανάλογα με το περιεχόμενο και τον τρόπο παρουσίασής του. Για παράδειγμα, οι περισσότεροι άνθρωποι αφιερώνουν πολύ περισσότερο χρόνο στο να κοιτούν μία εικόνα ενός προϊόντος, σε σχέση με το κείμενο που περιέχει τα χαρακτηριστικά του. Αυτό ονομάζεται επιλεκτική όραση, και το αντίθετο αυτής είναι η οπτική εξερεύνηση.

Οπτική εξερεύνηση λέγεται η διαδικασία που κάνει το μάτι σε μία ιστοσελίδα, όταν ο χρήστης δεν ξέρει τι να περιμένει και προσπαθεί να ελέγξει γρήγορα όσα περισσότερα μπορεί σε αυτό. Σε αυτή την περίπτωση, η προσοχή του χρήστη μπορεί να εστιαστεί στα πιο αναπάντεχα σημεία. Αυτά τα σημεία είναι το επίκεντρο της έρευνας που σχετίζεται με την ανάπτυξη τεχνολογιών ανίχνευσης βλέμματος.

Υπάρχουν πολλοί τομείς επιστημών που επωφελούνται από την έρευνα στην ανίχνευση βλέμματος, κυρίως για σκοπούς βελτίωσης αποδοτικότητας και παροχής «ανατροφοδότησης (feedback)». Οι πιο βασικοί είναι οι εξής:

Ψυχολογία

Από το 1970 ακόμα, πριν από την ευρεία χρήση υπολογιστών, μελετητές έκαναν ψυχολογικές δοκιμές στους ασθενείς τους χρησιμοποιώντας ταινίες και τηλεοπτικές σειρές. Έλεγχαν την αλλαγή στην προσοχή τους κατά την διάρκεια των διαφορετικών σκηνών, βγάζοντας ανάλογα συμπεράσματα. Πλέον έχουν αναπτυχθεί εξελιγμένες μέθοδοι[1] για χρήση ανίχνευσης βλέμματος για πρώιμη διάγνωση ψυχολογικών ιδιαιτεροτήτων.

Marketing & Πωλήσεις

Κατά την διάρκεια σχεδιασμού διαφημίσεων[2] στα σημερινά social media, ο σκοπός του δημιουργού είναι να τραβά την προσοχή του χρήστη σε μία σελίδα κατάμεστη με πληροφορίες, και να τον κάνει να την επιλέξει. Με την ανίχνευση βλέμματος είναι εύκολο να μετρηθεί ο χρόνος που αφιέρωσαν οι χρήστες να κοιτούν την διαφήμισή αυτή, και ποιο από τα στοιχεία που παρουσιάζει τους προσέλκυσε περισσότερο την προσοχή.

Έλεγχο απόδοσης υπαλλήλων

Μεγάλες εταιρίες χρησιμοποιούν τεχνολογίες ανίχνευσης βλέμματος με σκοπό να βελτιώσουν την απόδοση και να ανακαλύπτουν πιθανά λάθη που γίνονται από τους υπαλλήλους[3]. Αυτή η μέθοδος τους επιτρέπει να αναγνωρίζουν σε ποια σημεία της πώλησης αντιμετωπίζουν προβλήματα οι πελάτες και ίσως σταματούν εντελώς την διαδικασία αγοράς του προϊόντος ή της υπηρεσίας. Αυτό επιτρέπει στην ομάδα να βελτιώσει την επικοινωνία μεταξύ τους, να δημιουργήσουν συστήματα φιλικά προς τον χρήστη και να αλλάξουν παρωχημένες διαδικασίες που ίσως να μην ταιριάζουν πια στο προφίλ της επιχείρησης.

Ασφάλεια & Ιδιωτικότητα

Σύμφωνα με έρευνες[4], τα τελευταία 20 χρόνια δημιουργούνται συνέχεια μέθοδοι με τις οποίες η ανίχνευση βλέμματος ενσωματώνεται σε εφαρμογές ασφαλείας, όπως για παράδειγμα σε εφαρμογές αυθεντικοποίησης (χρήση κίνησης βλέμματος για άνοιγμα οθόνης κινητού), προστασίας ιδιωτικότητας (σβήσιμο οθόνης όταν ανιχνεύεται βλέμμα πίσω από τον χρήστη) και ελέγχου βλέμματος κατά την διάρκεια κρίσιμων παραβιάσεων ασφάλειας (κατά την διάρκεια επιθέσεων phishing κτλ)

Τέλος, η ανίχνευση βλέμματος είναι πολύ χρήσιμη στην ανάπτυξη και τον σχεδιασμό ιστοσελίδων[5], και αυτό είναι και το θέμα στο οποίο θα επικεντρωθούμε και να αναλύσουμε παρακάτω.

Τεχνολογίες ανίχνευσης βλέμματος στην ανάπτυξη & σχεδιασμό ιστοσελίδων

Η ανίχνευση βλέμματος ενός χρήστη δεν επιτρέπει μόνο να μάθουμε που και πότε έστρεψε την προσοχή του, αλλά και πόση ώρα αφιέρωσε σε συγκεκριμένα στοιχεία της οθόνης. Μας αποκαλύπτει, λοιπόν, πληροφορίες σχετικά με την «οπτική» και «γνωσιακή (cognitive)[6]» επεξεργασία που κάνει ο χρήστης καθώς του παρουσιάζεται μία ιστοσελίδα.

Οι μέθοδοι ανίχνευσης βλέμματος στην ανάπτυξη και τον σχεδιασμό ιστοσελίδων είναι πολύτιμες για την ανάλυση της συμπεριφοράς του χρήστη, και την παρουσίαση πιθανών προβλημάτων στον σχεδιασμό της ιστοσελίδας ή της εφαρμογής.

Για παράδειγμα, ο έλεγχος των κινήσεων της ίριδας ενός χρήστη που παρακολουθεί ένα βίντεο, μπορεί να καταδείξει το επίπεδο του ενδιαφέροντος του χρήστη αυτού στο περιεχόμενο του βίντεο. Ακόμα, η ανίχνευση βλέμματος μπορεί να υποδείξει αν οι χρήστες είναι ενοχλημένοι από ένα σχεδιαστικό στοιχείο της ιστοσελίδας (όπως τα αναδυόμενα παράθυρα σχετικά με τα cookies).

Οι βασικές Μέθοδοι Ανίχνευσης Βλέμματος με εξειδίκευση στον σχεδιασμό και την ανάπτυξη ιστοσελίδων είναι οι εξής:

Heatmaps

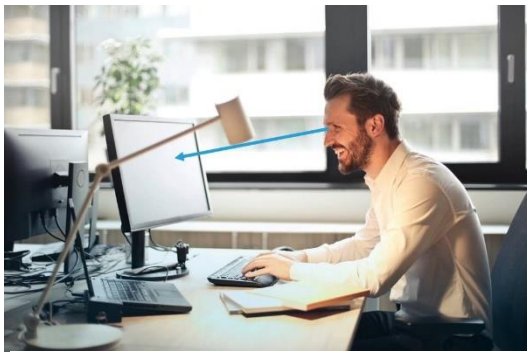
Τα heatmaps (ή gaze plots) είναι η πιο δημοφιλής μέθοδος εντοπισμού των σημείων προσέλκυσης προσοχής του χρήστη σε μία ιστοσελίδα. Υποδεικνύουν τα σημεία στα οποία ο χρήστης αφιέρωσε τον περισσότερο χρόνο, καθώς και τα σημεία που αγνόησε εντελώς. Χρησιμοποιώντας τα κατά την διάρκεια δοκιμών χρηστικότητας (usability testing), μπορούμε να μετατρέψουμε μία ιστοσελίδα πιο φιλική προς τον χρήστη, να τονίσουμε τα βασικά σημεία της ιστοσελίδας και να δώσουμε έμφαση σε αυτά που ήταν δύσκολο να τα προσέξει.

Εικόνα 1 Heatmaps (https://blog.saleslayer.com/hs-fs/hubfs/A_Blog/2020-01-January/heatmap.jpg?width=900&name=heatmap.jpg)

Click maps

Τα click maps καθορίζουμε ποιο αντικείμενο στην οθόνη έλαβε τα περισσότερα clicks μέσα σε συγκεκριμένη χρονική περίοδο. Με αυτά, εντοπίζονται οι υπερσύνδεσμοι που δεν οδηγούν σε χρήσιμο περιεχόμενο[7] (για παράδειγμα, αν επέλεξαν το λάθος υπερσύνδεσμο ή αν δεν διάβασαν τίποτα από το κείμενο που συνοδεύει τον συγκεκριμένο υπερσύνδεσμο)

Για την ανίχνευση του βλέμματος χρησιμοποιούνται οι εξής δύο τεχνολογίες:



Εικόνα 2 Ανίχνευση βλέμματος με απλή κάμερα Η/Υ https://learnopencv.com/wp-content/uploads/2019/11/screen-based-gaze-tracking-768x509.jpg

Ανιχνευτές βλέμματος βασισμένοι σε κάμερα και οθόνη Η/Υ

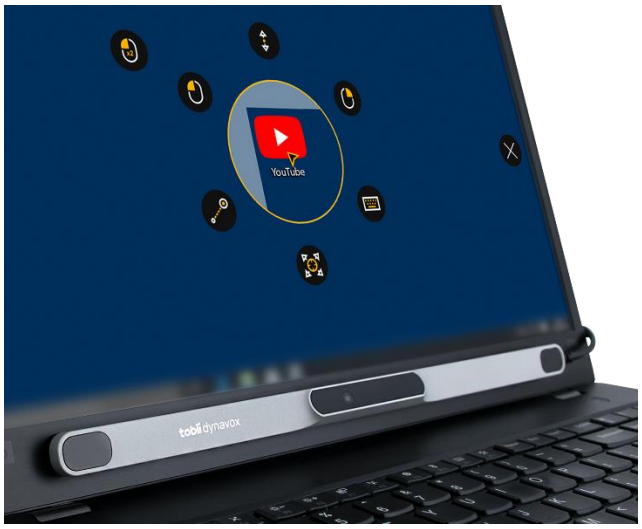
Στους συγκεκριμένους ανιχνευτές, ο σκοπός είναι να ανακαλύψουμε πού κοιτάει ο χρήστης σε μία οθόνη, εντοπίζοντας τις συντεταγμένες (x,y) πάνω σε αυτή. Αυτό μπορεί να συμβεί με την χρήση εξωτερικών επαγγελματικών καμερών ανίχνευσης βλέμματος, ή με τις ήδη ενσωματωμένες ή κοινές κάμερες εμπορίου.

Wearable ανιχνευστές βλέμματος

Στην περίπτωση των wearable ανιχνευτών βλέμματος, απαιτείται η χρήση ειδικών γυαλιών με ενσωματωμένη κάμερα στο κέντρο τους, και σκοπός τους είναι να βρουν την συντεταγμένες (x,y) στο πλαίσιο των γυαλιών, στο σημείο ακριβώς που κοιτά ο χρήστης.

Εικόνα 3 Wearable eye tracker (https://www.tobiipro.com/imagevault/publishedmedia/8clvxud7x9j3fqte76kj/TobiiPro_Glasses_2_Eye_Tracker_side_2_1.jpg)

Σε αυτή την εργασία θα επικεντρωθούμε στην πρώτη κατηγορία, δηλαδή στους ανιχνευτές βλέμματος βασισμένους σε κάμερα και οθόνη Η/Υ, και συγκεκριμένα θα αναπτύξουμε εφαρμογή χρησιμοποιώντας μία απλή ενσωματωμένη κάμερα σε laptop ελάχιστου κόστους, και στη συνέχεια θα συγκρίνουμε τα αποτελέσματα με αποτελέσματα που παράχθηκαν από μία επαγγελματική υπέρυθρη κάμερα υψηλού κόστους (Tobii Dynavox PCEye 5).



Εικόνα 4 Η υπέρυθρη κάμερα υψηλού κόστους που χρησιμοποιήσαμε για σύγκριση στο πρόγραμμα μας – Tobii Dynavox PCEye 5 – (https://i.shgcdn.com/c76fc3ff-44eb-484e-979e-a860b4dd6f91/-/format/auto/-/preview/3000x3000/-/quality/lighter/)

Η εφαρμογή αυτή θα αναπτυχθεί με την χρήση γλωσσών προγραμματισμού που έχουν πληθώρα βιβλιοθηκών σχετικές με την «Όραση Υπολογιστών (Computer Vision)», όπως είναι

η python, η julia κτλ, δηλαδή της ικανότητας ενός υπολογιστή να «αντιλαμβάνεται» ψηφιακές εικόνες. Με αυτή την εφαρμογή σκοπεύουμε να αποδείξουμε ότι η ανίχνευση βλέμματος δεν απαιτεί εξειδικευμένη τεχνολογία υψηλού κόστους ώστε να είναι αποτελεσματική, και ότι τα οφέλη της ανίχνευσης βλέμματος, τα οποία προαναφέραμε, πρέπει να είναι εύκολα προσβάσιμα και αξιοποιήσιμα από όλους.

Στην συνέχεια θα αναλύσουμε την γλώσσα προγραμματισμού που χρησιμοποιήσαμε, καθώς και τις βιβλιοθήκες της. Στην συνέχεια, θα αναλύσουμε τον τρόπο που αναπτύξαμε την εφαρμογή, και, τέλος, θα εξετάσουμε τα αποτελέσματά της.

3. Η γλώσσα προγραμματισμού Python

Η Python[8] είναι μία υψηλού επιπέδου γλώσσα προγραμματισμού[9]. Είναι δυναμική γλώσσα προγραμματισμού και υποστηρίζει συλλογή απορριμμάτων (garbage collection ή GC).

Δημιουργήθηκε από τον Ολλανδό Guido van Rossum στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI) το 1989 και κυκλοφόρησε για πρώτη φορά το 1991.

Εικόνα 5: Logo της Python έως 2006 (https://en.wikipedia.org/wiki/History_of_Python)

Οι διερμηνευτές της Python είναι διαθέσιμοι για εγκατάσταση σε πολλά λειτουργικά συστήματα, επιτρέποντας στην Python την εκτέλεση κώδικα σε ευρεία γκάμα συστημάτων. Χρησιμοποιώντας εργαλεία τρίτων, όπως το Py2exe ή το Pyinstaller, ο κώδικας της Python μπορεί να πακεταριστεί σε αυτόνομα εκτελέσιμα προγράμματα για μερικά από τα πιο δημοφιλή λειτουργικά συστήματα, επιτρέποντας τη διανομή του βασισμένου σε Python λογισμικού για χρήση σε αυτά τα περιβάλλοντα χωρίς να απαιτείται εγκατάσταση του διερμηνευτή της Python.

Εικόνα 6: Logo της Python από το 2006 έως σήμερα (https://en.wikipedia.org/wiki/History_of_Python)

Η Python αναπτύσσεται ως ανοιχτό λογισμικό (open source) και η διαχείρισή της γίνεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation. Ο κώδικας διανέμεται με την άδεια Python Software Foundation License η οποία είναι συμβατή με την GPL. Το όνομα της γλώσσας προέρχεται από την ομάδα των Άγγλων κωμικών Monty Python και δεν έχει καμιά σχέση με το φίδι πύθωνα, παρότι το λογότυπό της παραπέμπει σε κάτι τέτοιο.

Ο κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της. Το συντακτικό της επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα από ότι θα ήταν δυνατόν σε γλώσσες όπως η C++ ή η Java. Διακρίνεται λόγω του ότι έχει πολλές βιβλιοθήκες που διευκολύνουν ιδιαίτερα αρκετές συνηθισμένες εργασίες και για την ταχύτητα εκμάθησης της.

Η πληθώρα βιβλιοθηκών της Python ήταν η αιτία που επιλέχθηκε για την ανάπτυξη του προγράμματος στα πλαίσια της παρούσας διπλωματικής. Παρακάτω θα αναλύσουμε κάποιες από τις κυριότερες βιβλιοθήκες της python και στη συνέχεια θα αναφερθούμε στις βιβλιοθήκες που χρησιμοποιήθηκαν για το πρόγραμμα μας.

3.1 Βασικές Βιβλιοθήκες Python

Οι βιβλιοθήκες[10] της Python χρησιμοποιούνται από εκατομμύρια προγραμματιστές και αναλυτές δεδομένων[11] σε όλο τον κόσμο. Χρησιμοποιούνται για την πρόβλεψη αποτελεσμάτων, για την αυτοματοποίηση διαδικασιών κ.ά. και χωρίζονται στις εξής βασικές κατηγορίες:

Standard Βιβλιοθήκη Python

Περιέχει πάνω από 200 modules και είναι προεγκατεστημένα με την βασική έκδοση της Python. Είναι γραμμένη στην γλώσσα προγραμματισμού C, είναι εκτεταμένη[12] προσφέροντας μία ευρεία λίστα επιλογών, η οποία στην πλειοψηφία της αυξάνει την φορητότητα των προγραμμάτων που είναι γραμμένα σε python.

Εξόρυξη Δεδομένων (Data Mining)

Βιβλιοθήκες όπως η **Scrapy**[13] και η **Beautiful Soup**[14] δίνουν την δυνατότητα συλλογής δομημένων δεδομένων από το διαδίκτυο (web crawling) τα οποία μπορούν να χρησιμοποιηθούν σε μοντέλα μηχανικής μάθησης της Python. Χρησιμοποιούν την τεχνολογία API των ιστοσελίδων και δίνουν δυνατότητα διαφοροποίησης μορφής (format) δεδομένων, ώστε τα δεδομένα να μπορούν να επεξεργαστούν από προγράμματα με διάφορα μοντέλα σχεδιασμού.

Επεξεργασία Δεδομένων

Βιβλιοθήκες όπως η **NumPy**[15] επιτρέπουν στον χρήστη την χρήση πινάκων και επεξεργασία αυτών. Με αυτό τον τρόπο οι μαθηματικές πράξεις και η διανυσματοποίηση (vectorization) των πινάκων αυξάνει την ταχύτητα εκτέλεσης και γενικά την αποδοτικότητα των προγραμμάτων. Επίσης η **Pandas**[16] είναι μία δημοφιλής βιβλιοθήκη για δημιουργία και επεξεργασία δομών δεδομένων.

Μία ακόμα χρήσιμη βιβλιοθήκη είναι η **SciPy**[17] η οποία περιλαμβάνει modules για γραμμική άλγεβρα, βελτιστοποίηση και στατιστικές αναλύσεις. Η κύρια λειτουργία της χτίστηκε πάνω στην NumPy, και έχει χρήση σε σχεδόν όλους τους επιστημονικούς τομείς προγραμματιστικών projects (πληροφορικής, μαθηματικών, μηχανικής).

Νευρωνικά Δίκτυα & Μοντελοποίηση

Βιβλιοθήκες όπως η **Keras**[18] είναι ιδανική για κατασκευή Νευρωνικών Δικτύων και μοντελοποίηση τους. Είναι εύκολη στην χρήση και παρέχει στους προγραμματιστές μεγάλο βαθμό επεκτασιμότητας.

Μηχανική Μάθηση, Deep Learning

Βιβλιοθήκες όπως η **PyTorch**[19] και **TensorFlow**[20] έχουν γίνει δημοφιλείς λόγω της δυνατότητας τους να εκτελούν προβλήματα μηχανικής μάθησης και deep learning, όπως αναγνώριση φωνής, αντικειμένων και πολλών άλλων, ενώ δεν περιορίζει τον χρήστη σε ένα είδος δεδομένων, αλλά επιτρέπει την χρήση πολλαπλών data sets. Χρησιμοποιούνται επίσης για την δημιουργία υπολογιστικών γραφημάτων κ.ά.

Η βιβλιοθήκη **XGBoost**[21] επιτρέπει στον προγραμματιστή να εφαρμόσει αλγορίθμους μηχανικής μάθησης, είναι ευέλικτη και πολύ αποδοτική. Προσφέρει διάφορες λύσεις σε προβλήματα data science, και μπορεί ο ίδιος κώδικας να τρέξει σε μεγάλα κατανεμημένα περιβάλλοντα, όπως το Hadoop, SGE και MPI.

Οπτικοποίηση δεδομένων

Η δημιουργία διαγραμμάτων, γραφημάτων και heatmaps είναι απαραίτητη για την οπτικοποίηση και παρουσίαση αποτελεσμάτων των προγραμμάτων. Για αυτό το λόγο, χρησιμοποιούνται βιβλιοθήκες όπως η **Matplotlib**[22], η **Seaborn**[23], η **Plotly**[24] και η **Pydot**[25]. Επεκτείνονται συνεχώς με νέα γραφικά και λειτουργίες για καλύτερη απεικόνιση και διαδραστικότητα.

Κβαντικός Προγραμματισμός

Η Python ενδείκνυται για κβαντικό προγραμματισμό, και με την χρήση βιβλιοθηκών όπως η **QuTiP**[26], **Qiskit**[27] και **Cirq**[28] επιτυγχάνεται η απεικόνιση των κβαντικών συστημάτων με φιλικό προς τον χρήστη και αποδοτικό τρόπο.

Γραφικά και εικόνες

Βιβλιοθήκες όπως η **Pyglet**[29] και η **PyGame**[30] δίνουν την δυνατότητα δημιουργίας εύκολων προγραμμάτων με γραφικά, ήχο και εικόνες με απλό τρόπο, και είναι ιδανικές για δημιουργία παιχνιδιών. Για χρήση αποκλειστικά εικόνων, η Pillow είναι η κατάλληλη βιβλιοθήκη.

3.2 Βιβλιοθήκες Python που χρησιμοποιήθηκαν στο πρόγραμμα ανίχνευσης βλέμματος

Είναι πλέον προφανές ότι οι βιβλιοθήκες της Python δίνουν απεριόριστες δυνατότητες στην δημιουργία προγραμμάτων. Για τον λόγο αυτό, το πρόγραμμα που αναπτύξαμε χρησιμοποίησε αρκετές βιβλιοθήκες, εκ των οποίων η κάθε μία ικανοποιούσε και μία διαφορετική ανάγκη του προγράμματος. Στην συνέχεια θα παρουσιάσουμε τις βιβλιοθήκες αυτές, και τους λόγους που χρησιμοποιήθηκαν.

Το πρόγραμμα που δημιουργήθηκε, όπως έχουμε ήδη αναφέρει, στοχεύει στην αναγνώριση βλέμματος (gaze recognition). Για να επιτευχθεί, έπρεπε να εγκατασταθούν εξειδικευμένες βιβλιοθήκες, οι οποίες συνδυαστικά θα επέτρεπαν σε μία απλή κάμερα laptop να αναγνωρίζει την ίριδα του ματιού και να εντοπίζει την θέση της στην οθόνη. Οι βιβλιοθήκες αυτές είναι οι εξής:

OpenCV

Η βιβλιοθήκη **OpenCV**[31] (παλιότερα cv2) είναι βιβλιοθήκη που υποστηρίζει την «Όραση Υπολογιστικών Συστημάτων (Computer Vision)» σε πραγματικό χρόνο. Πολύ ευρεία βιβλιοθήκη με εφαρμογές σε όλο το φάσμα της αναγνώρισης κινήσεων, προσώπου και αντικειμένων, με πολύ καλή εφαρμογή στην ρομποτική και την μηχανική μάθηση.

Σημαντικότατη βιβλιοθήκη, χωρίς την οποία δεν θα υπήρχε η συγκεκριμένη εργασία, αφού οι εναλλακτικές είναι λίγες και όχι εξίσου ανεπτυγμένες και πολύπλευρες. Στην περίπτωση μας, χρησιμοποιήθηκε αρχικά για την αναγνώριση των ματιών στο πρόσωπο που βρίσκεται μπροστά στην κάμερα, και αργότερα για τον εντοπισμό της ίριδας στην εικόνα.

NumPy

Όπως προαναφέρθηκε, η **NumPy** είναι μία βασική βιβλιοθήκη για την δημιουργία και επεξεργασία πινάκων. Την χρησιμοποιήσαμε κατά την επεξεργασία της εικόνας του ματιού και της ίριδας, και αργότερα για τον συσχετισμό θέσης ίριδας και θέσης βλέμματος στην οθόνη του υπολογιστή.

Cmake

Βασική cross-platform βιβλιοθήκη, η **Cmake**[32] βοηθάει στην αυτοματοποίηση της εγκατάστασης λογισμικού χωρίς την συμβολή compiler. Έπρεπε να εγκατασταθεί για να προχωρήσουμε στην εγκατάσταση της **Dlib**.

Dlib

Η **Dlib**[33] είναι βιβλιοθήκη που περιέχει αλγορίθμους μηχανικής μάθησης και εργαλεία για λύση περίπλοκων προβλημάτων λογισμικού. Έχει εφαρμογή σε επεξεργασία εικόνας, γραφικά μοντέλα, threading κ.ά. Στο πρόγραμμα χρησιμοποιήθηκε ο «frontal face detector» που εμπεριέχει, για την αναγνώριση των σημείων του προσώπου, ώστε τελικά να καταφέρουμε να απομονώσουμε τα μάτια για να προχωρήσουμε στην αναγνώριση βλέμματος.

Collections (defaultdict)

Η **Collections**[34] είναι ουσιαστικά ένα module που περιέχει διαφορετικά είδη container αντικειμένων (όπως είναι τα προεγκατεστημένα tuple, list και dictionary). Εμείς χρησιμοποιήσαμε συγκεκριμένα το container defaultdict, και χρησιμοποιήθηκε για να γεμίσουμε μία σειρά από dictionaries με default τιμές.

Math (sqrt)

Πολύ σημαντική βιβλιοθήκη είναι και η **Math**[35], η οποία περιέχει τις περισσότερες μαθηματικές συναρτήσεις, και εμείς χρησιμοποιήσαμε την sqrt, δηλαδή την ύψωση στο τετράγωνο, για τον υπολογισμό απόστασης μεταξύ θέσης ματιού και θέσης βλέμματος στην οθόνη.

Matplotlib (pyplot και patches.circle)

Όπως προαναφέρθηκε, η **Matplotlib** είναι μία από τις βασικότερες βιβλιοθήκες για δημιουργία διαγραμμάτων, γραφημάτων, και γενικά για την οπτικοποίηση και παρουσίαση αποτελεσμάτων των προγραμμάτων. Εμείς χρησιμοποιήσαμε το **Pyplot**[36] από την βιβλιοθήκη Matplotlib, στην τελευταία φάση του προγράμματος, για την ανάγνωση εικόνων τις οποίες θα εμφάνιζε το πρόγραμμα στον χρήστη. Χρησιμοποιήθηκε επίσης το **Patches** και συγκεκριμένα το **Circle**[37] για την δημιουργία κύκλων στα σημεία της εικόνας που κοίταξε ο χρήστης.

4. Ανάπτυξη εφαρμογής

Η εφαρμογή αναπτύχθηκε χρησιμοποιώντας γλώσσα προγραμματισμού Python 3.7, στο περιβάλλον ανάπτυξης (IDE) Pycharm[38], στην έκδοση 2021.1.3 και μπορείτε να την βρείτε στο παράρτημα της εργασίας, καθώς και να την κατεβάσετε στο παρακάτω link:

https://www.dropbox.com/sh/gjcr7z2uarn3hzr/AACr7RMcmcaTE3a_ltRuiaLxa?dl=0

Ο χρήστης εισάγει μία σειρά από εικόνες ιστοσελίδων, χρησιμοποιεί την ενσωματωμένη κάμερα του υπολογιστή, και ανιχνεύει τα σημεία της εικόνας στα οποία κοίταξε ο χρήστης. Με αυτό τον τρόπο μπορεί ο σχεδιαστής της ιστοσελίδας να αναγνωρίσει τα βασικά σημεία προσοχής του χρήστη της ιστοσελίδας, και να αξιοποιήσει αυτή την πληροφορία προς όφελός του.

Η ανάπτυξη της εφαρμογής χωρίζεται στα εξής στάδια τα οποία θα αναλύσουμε:

- Ανίχνευση ματιού & ανίχνευση ίριδας χρήστη
- Εφαρμογή calibration για συσχέτιση θέσης ματιού & θέσης βλέμματος στην οθόνη
- Παρουσίαση εικόνων στον χρήστη
- Εμφάνιση αποτελέσματος εφαρμογής

Παρακάτω θα αναφερθούμε στις φάσεις αυτές, στις τεχνολογίες που χρησιμοποιήσαμε για να επιτύχουμε το επιθυμητό αποτέλεσμα, καθώς και στιγμιότυπα από την εφαρμογή κατά τις δοκιμές.

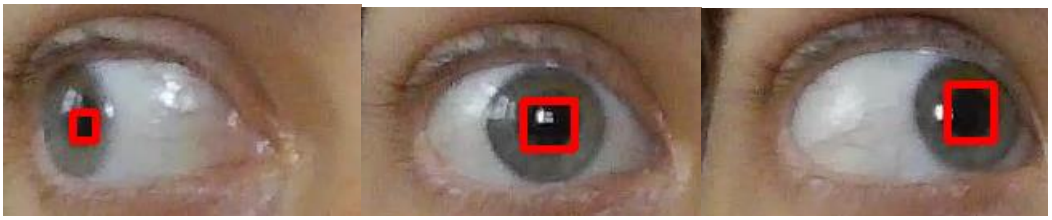
4.1 Ανίχνευση ματιού & ανίχνευση ίριδας χρήστη

Ξεκινώντας με την ανάπτυξη της εφαρμογής, ήταν αναγκαία η αναγνώριση του ματιού του χρήστη. Ξεκινώντας από ένα προϋπάρχον βίντεο που περιέχει ένα μάτι του οποίου η ίριδα αλλάζει συνεχώς κατευθύνσεις, ο στόχος ήταν να μπορέσει το πρόγραμμα να εντοπίσει το μάτι.

Κάνοντας χρήση του δεδομένου ότι η ίριδα του ματιού είναι το σκουρότερο στρογγυλό τμήμα του ματιού, χρησιμοποιούμε τη βιβλιοθήκη αναγνώρισης εικόνων και προσώπου **OpenCV** (την αναπτύξαμε περισσότερο στο προηγούμενο κεφάλαιο), ώστε να την ανιχνεύσουμε.

Η μέθοδος που χρησιμοποιήθηκε είναι να μετατρέψουμε την έγχρωμη εικόνα σε **ασπρόμαυρη**, και αυτό επειδή στοχεύουμε στην ανίχνευση του σκουρότερου σημείου της εικόνας. Χρησιμοποιούμε επίσης την μέθοδο **GaussianBlur**[39], μέθοδος που χρησιμοποιείται από δεκάδες προγράμματα μετατροπής εικόνων όπως το Photoshop[40], που εξομαλύνει (θολώνει) την εικόνα χρησιμοποιώντας τον Γκαουσιανό αλγόριθμο[41]. Αυτό το χρησιμοποιούμε γιατί μία τέτοια μέθοδος αφαιρεί τον θόρυβο από την εικόνα και επιτυγχάνεται ο εντοπισμός του ματιού. Το αποτέλεσμα του προγράμματος φαίνεται παρακάτω, σχηματίζοντας ένα κόκκινο πλαίσιο γύρω από την ίριδα του ματιού.

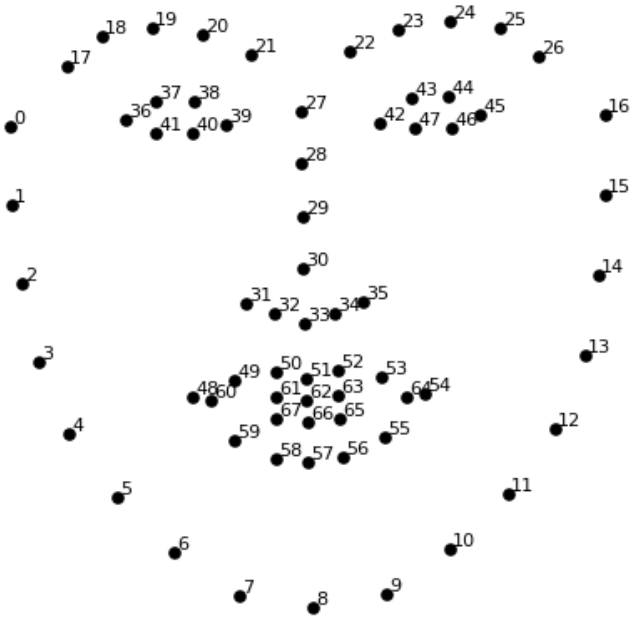
Εικόνα 7 Ίδια εικόνα ασπρόμαυρη, έγχρωμη και gaussian blur



Εικόνα 8 Εντοπισμός ίριδας σε διαφορετικές κατευθύνσεις

Επόμενο στάδιο είναι η χρήση της κάμερας για real-time αναγνώριση του ματιού. Η μέθοδος που χρησιμοποιείται είναι η ίδια με την προηγούμενη, με την διαφορά ότι πλέον πρέπει το πρόγραμμα να ανιχνεύει το μάτι σε οποιαδήποτε θέση και να βρίσκεται μπροστά στην κάμερα.

Αυτό επιτυγχάνεται με την χρήση της βιβλιοθήκης Dlib (η οποία αναλύθηκε στο προηγούμενο κεφάλαιο), και συγκεκριμένα του predictor που περιέχει, τον οποίο εκπαιδεύουμε εισάγοντας εικόνες στις οποίες συσχετίζουμε τα σημεία του προσώπου με νούμερα, όπως φαίνεται στην εικόνα που ακολουθεί.



Εικόνα 9 Σημεία προσώπου για predictor Dlib (https://www.researchgate.net/figure/The-68-specific-human-face-landmarks_fig4_331769278)

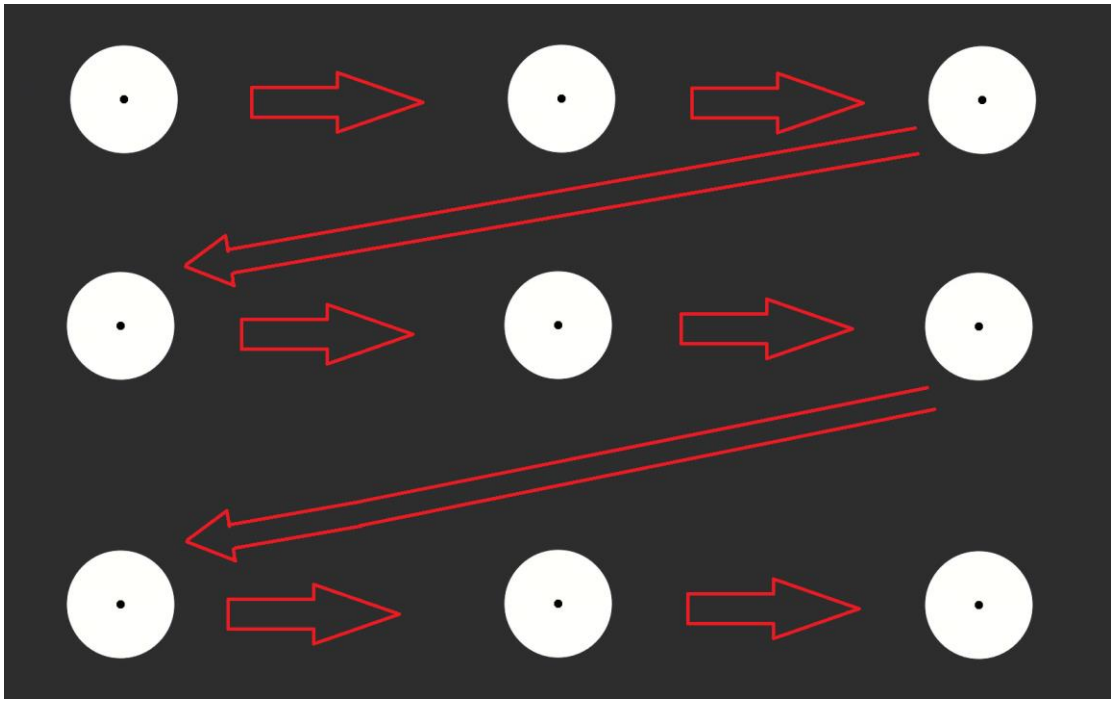
Το αποτέλεσμα φαίνεται στις εικόνες που ακολουθούν. Χρησιμοποιώντας τα σημεία του ματιού, το πρόγραμμα εντοπίζει το μάτι (πρώτη εικόνα παρακάτω), με μαθηματικούς υπολογισμούς εντοπίζεται η ίριδα (δεύτερη εικόνα) και απομονώνοντας μόνο το μάτι, και χρησιμοποιώντας τις μεθόδους που αναλύσαμε νωρίτερα, εντοπίζεται το κέντρο της ίριδας (τρίτη εικόνα).

Εικόνα 10 Ανίχνευση ματιού και ίριδας από το πρόγραμμα

4.2 Εφαρμογή calibration για συσχέτιση θέσης ματιού & θέσης βλέμματος στην οθόνη

Πλέον έχουμε εντοπίσει την θέση του ματιού στην εικόνα της κάμερας, και τώρα σκοπός είναι να ταυτοποιήσουμε πού κοιτάει κάθε στιγμή ο χρήστης. Αυτό θα το καταφέρουμε με το calibration, το οποίο καταφέρνει την συσχέτιση της θέσης ματιού και της θέσης του βλέμματος στην οθόνη[42].

Σύμφωνα με μελέτες[43], το calibration πρέπει να έχει τουλάχιστον 4 σημεία στην οθόνη, με την επιτυχία του να αυξάνεται καθώς αυξάνονται τα σημεία. Με σκοπό να αυξήσουμε το ποσοστό επιτυχίας του, χωρίς όμως να επέλθει κόπωση στον χρήστη, επιλέξαμε 9 σημεία για το calibration.



Εικόνα 11 Calibration με 9 σημεία

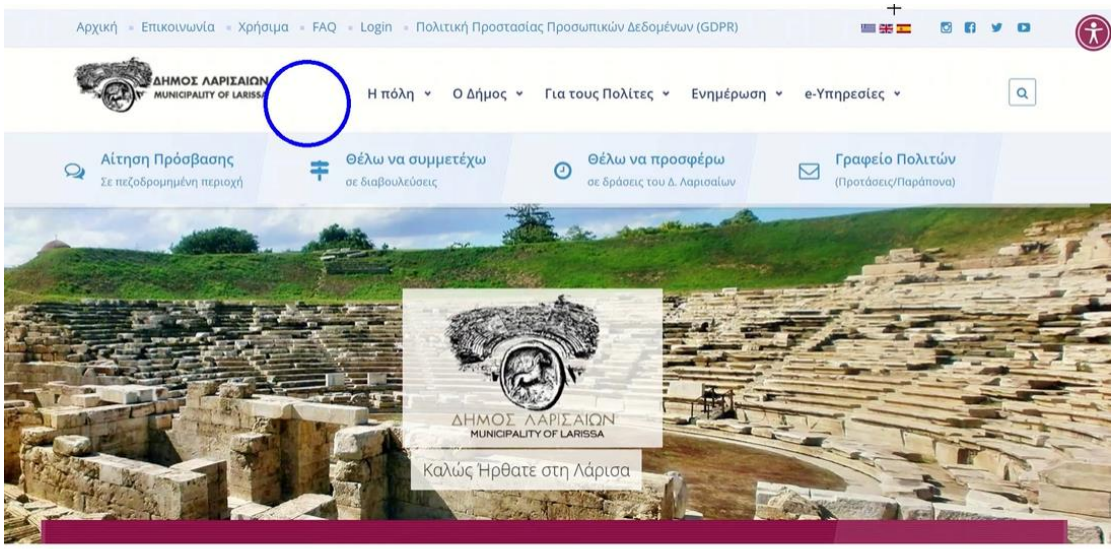
Ξεκινήσαμε την διαδικασία του calibration[44] ως εξής: Καθώς το πρόγραμμα ανιχνεύει τις κινήσεις των ματιών, μία λευκή κουκκίδα κινείται στην οθόνη. Ο χρήστης ακολουθεί την κίνηση της λευκής κουκκίδας, και η θέση του ματιού του συσχετίζεται με την θέση που βρίσκεται η κουκκίδα πάνω στην οθόνη.

Με αυτό τον τρόπο έχουμε εξασφαλίσει ότι γνωρίζουμε πότε κοιτάει ο χρήστης αυτά τα 9 σημεία της οθόνης. Χωρίζοντας την οθόνη σε ισόποσες μεταξύ αυτών αποστάσεις, βρίσκουμε και τις συντεταγμένες του ματιού σε όλα τα άλλα σημεία της οθόνης. Οι συντεταγμένες αυτές καταγράφονται σε ένα txt αρχείο, και θα χρησιμοποιηθούν στην επόμενη φάση της εφαρμογής.

4.3 Παρουσίαση εικόνων στον χρήστη

Σε αυτή τη φάση του προγράμματος παρουσιάζεται στον χρήστη μία σειρά από εικόνες, οι οποίες εμφανίζονται για λίγα δευτερόλεπτα, και ο χρήστης μπορεί να παρατηρήσει λίγα στοιχεία της κάθε εικόνας, μέχρι να εμφανιστεί η επόμενη. Το βλέμμα του χρήστη την δεδομένη στιγμή, υποδεικνύεται από έναν μπλε κύκλο, όπως φαίνεται στις επόμενες εικόνες.

Εικόνα 12 Η ιστοσελίδα του Δήμου Θεσσαλονίκης εμφανίζεται στον χρήστη (μπλε κύκλος=θέση βλέμματος)



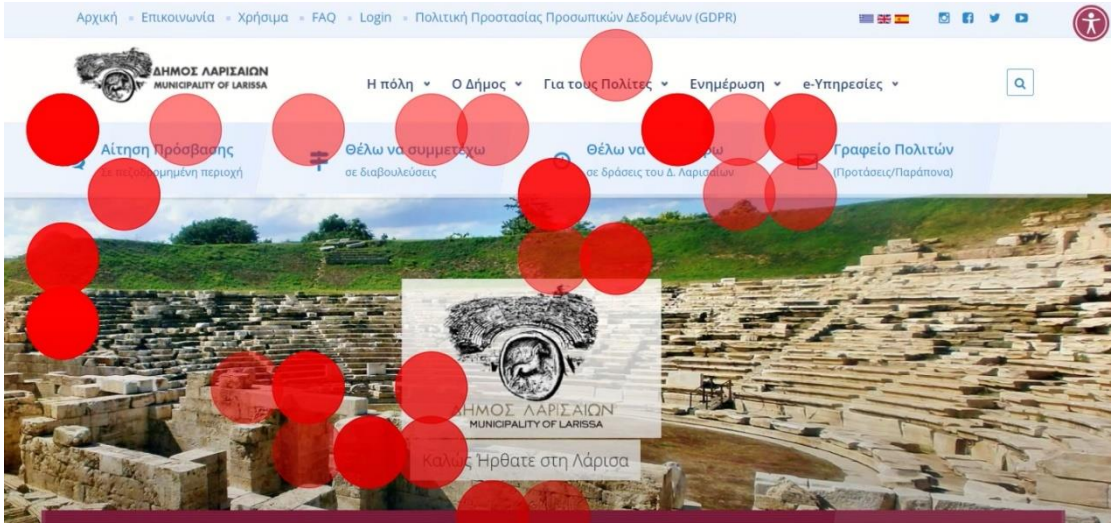
Εικόνα 13 Η ιστοσελίδα του Δήμου Λαρισαίων εμφανίζεται στον χρήστη (μπλε κύκλος=θέση βλέμματος)

Έγινε επιλογή εικόνων ιστοσελίδων, καθώς είναι αποδεδειγμένο[45] ότι η ανάπτυξη εφαρμογών που ελέγχουν το βλέμμα του χρήστη σε μία ιστοσελίδα, μπορούν να βελτιώσουν τον σχεδιασμό της. Με γνώμονα αυτό, χρησιμοποιήσαμε ιστοσελίδες Δήμων, ώστε να εντοπίσουμε αδυναμίες και ευκαιρίες βελτίωσης στην ανάπτυξη των δημοτικών ιστοσελίδων, για την καλύτερη εξυπηρέτηση του πολίτη και βελτίωση της εικόνας των Δήμων στους δημότες τους.

4.4 Εμφάνιση αποτελέσματος εφαρμογής

Σαν τελικό αποτέλεσμα, η εφαρμογή παράγει τις εικόνες που εμφανίστηκαν στον χρήστη, σε συνδυασμό με τα σημεία στα οποία κοίταξε ο χρήστης κατά την διάρκεια της σύντομης παρουσίασής τους μπροστά του. Ένα δείγμα αυτών εμφανίζεται παρακάτω, και η πλήρης παρουσίασή τους θα γίνει στο επόμενο κεφάλαιο.

Εικόνα 14 Η ιστοσελίδα του Δήμου Θεσσαλονίκης, και τα σημεία που κοίταξε ο χρήστης



Εικόνα 15 Η ιστοσελίδα του Δήμου Λαρισαίων, και τα σημεία που κοίταξε ο χρήστης

Τα σημεία που κοίταξε ο χρήστης πάνω από μία φορά, έχουν πιο έντονο κόκκινο χρώμα, ώστε να φαίνεται πού έδωσε μεγαλύτερη έμφαση.

Στο επόμενο κεφάλαιο θα γίνει λεπτομερής παρουσίαση και ανάλυση των αποτελεσμάτων, καθώς και σύγκριση με την υφιστάμενη τεχνολογία ανίχνευσης βλέμματος.

5. Αποτελέσματα & Καινοτομία Διπλωματικής Εργασίας - Σύγκριση με Υφιστάμενη Κατάσταση

Στόχος της εργασίας αυτής ήταν να δημιουργηθεί ένα πρόγραμμα που, χρησιμοποιώντας χαμηλού κόστους τεχνολογίες, ανιχνεύει κινήσεις της ίριδας του ματιού, και συλλέγει δεδομένα σχετικά με το ενδιαφέρον του χρήστη σε σχέση με τις εικόνες που του παρουσιάζονται.

Στο συγκεκριμένο πείραμα χρησιμοποιήθηκαν ιστοσελίδες ελληνικών Δήμων, ώστε να εντοπίσουμε αδυναμίες και ευκαιρίες βελτίωσης στην ανάπτυξη των δημοτικών ιστοσελίδων, για την καλύτερη εξυπηρέτηση του πολίτη και βελτίωση της εικόνας των Δήμων στους δημότες τους.

Σε αυτό το σημείο πρέπει να τονίσουμε ότι, ενώ υπάρχουν στην αγορά κάμερες αναγνώρισης βλέμματος υπέρυθρης τεχνολογίας με υψηλό κόστος, για την ανάπτυξη του προγράμματος χρησιμοποιήσαμε απλή web camera ελάχιστου κόστους, η οποία είναι τοποθετημένη στο επάνω μέρος της οθόνης, και αναγνωρίζει σε χιλιοστά του δευτερολέπτου την θέση του βλέμματος του χρήστη. Για τους σκοπούς της σύγκρισης με τεχνολογίες που υπάρχουν στο εμπόριο, κατά την παρουσίαση και την ανάλυση των αποτελεσμάτων, θα παρουσιάζουμε και αντίστοιχα αποτελέσματα χρησιμοποιώντας επαγγελματική υπέρυθρη κάμερα υψηλού κόστους (Tobii Dynavox PCEye 5), σε συνδυασμό με το πρόγραμμα ανίχνευσης βλέμματος της.

Ουσιαστικά θα μελετηθεί η συσχέτιση μεταξύ των σημείων που είναι τοποθετημένα τα μέρη της ιστοσελίδας, και του ποσοστού προσοχής που έλαβαν αυτά τα μέρη από τον χρήστη. Τα σημεία της ιστοσελίδας που θα μελετηθούν είναι τα εξής:

- Logo ιστοσελίδας
- Κεφαλίδα (Header)
- Οριζόντιο ή/και πλευρικό Μενού
- Slideshow
- Σώμα ιστοσελίδας

Θα γίνει λοιπόν έλεγχος χρηστικότητας των συγκεκριμένων ιστοσελίδων (website usability testing) και θα εξάγουμε τα συμπεράσματά μας, βασιζόμενοι στα δεδομένα που εξάγαμε από το πρόγραμμα.

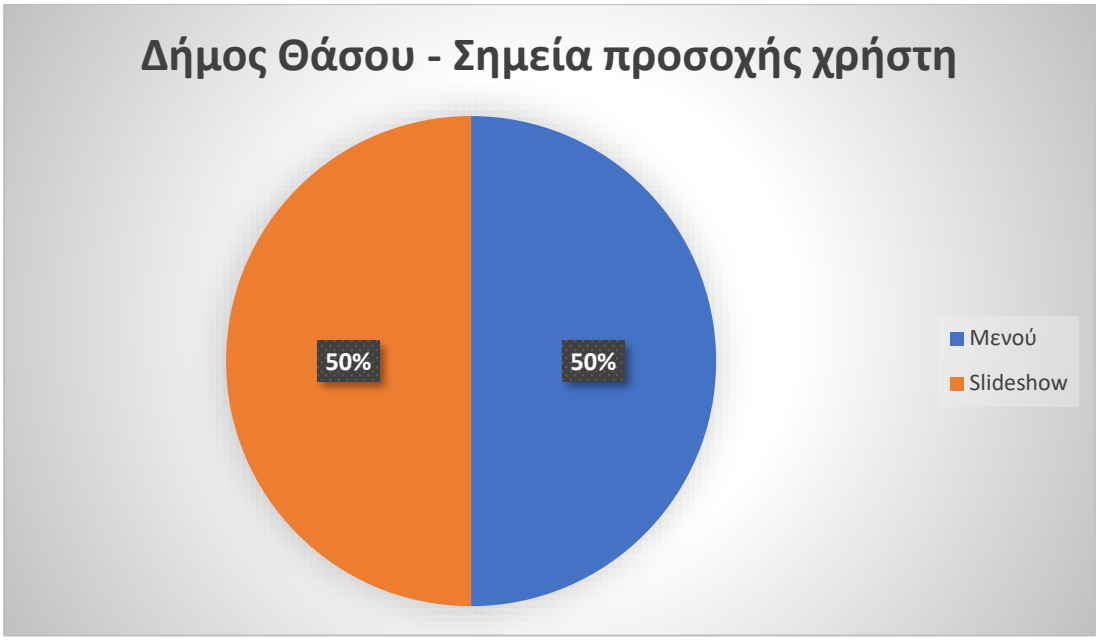
Βάσει των παρατηρήσεων μας μπορούμε να βγάλουμε ασφαλή συμπεράσματα μόνο στο κατά πόσον ήταν εύκολα εντοπίσιμα τα μέρη της ιστοσελίδας μέσα στα ελάχιστα δευτερόλεπτα που εμφανίστηκε η ιστοσελίδα στον χρήστη, δίνοντας μας έτσι πληροφορίες για τον σχεδιασμό που απαιτείται, όταν πρόκειται για ευκολία χρήσης και επιτάχυνση των διαδικασιών. Μετά από κάθε αποτέλεσμα λοιπόν, θα εμφανίζεται στατιστική ανάλυση με το ποσοστό της προσοχής του χρήστη που δεσμεύτηκε από κάθε ένα από τα σημεία της ιστοσελίδας που προαναφέρθηκε, δίνοντας έτσι πληροφορίες για τις διαφορές της εμπειρίας του χρήστη εξαιτίας της διαφορετικής σχεδιαστικής μεθόδου της ιστοσελίδας.

5.1 Ιστοσελίδα Δήμου Θάσου

Στην περίπτωση του Δήμου Θάσου, επέλεξαν το μενού τους να βρίσκεται στο πάνω μέρος της ιστοσελίδας, και να υπάρχει slideshow στην υπόλοιπη σελίδα. Ο χρήστης επικεντρώνεται στην εικόνα αλλά εντοπίζει, χωρίς ιδιαίτερη δυσκολία, και το μενού.

Εικόνα 16 Ιστοσελίδα Δήμου Θάσου

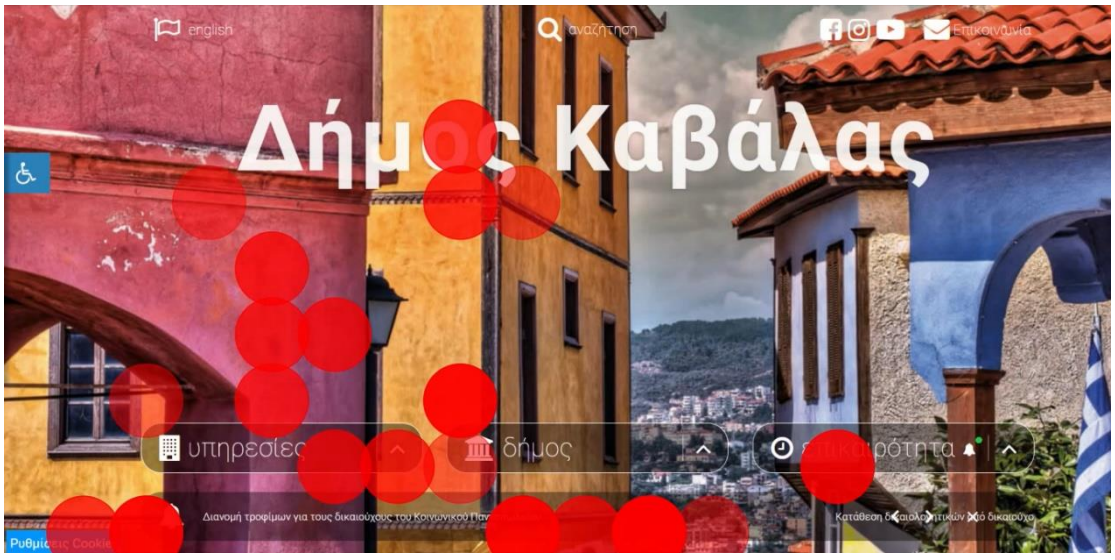
Ακολουθεί το γράφημα που δείχνει το ποσοστό της προσοχής του χρήστη που έλαβε το κάθε σημείο της ιστοσελίδας.



Εικόνα 17 Δήμος Θάσου - Σημεία προσοχής χρήστη

5.2 Ιστοσελίδα Δήμου Καβάλας

Στον Δήμο Καβάλας, σε αντίθεση με τον Δήμο Θάσου, ολόκληρη η σελίδα αποτελείται από εικόνες της πόλης, που αλλάζουν σταδιακά, το οποίο μπορεί να φέρει μία μερική απόσπαση προσοχής στον χρήστη. Το μενού επίσης βρίσκεται στο κάτω μέρος της οθόνης, και έχει χωριστεί σε 3 βασικές κατηγορίες, το οποίο είναι πιο αποδοτικό[46] σε περιπτώσεις περίπλοκων ιστοσελίδων όπως είναι οι δημοτικές.



Εικόνα 18 Ιστοσελίδα Δήμου Καβάλας

Ακολουθεί το γράφημα που δείχνει το ποσοστό της προσοχής του χρήστη που έλαβε το κάθε σημείο της ιστοσελίδας.

Εικόνα 19 Δήμος Καβάλας - Σημεία προσοχής χρήστη

5.3 Ιστοσελίδα Δήμου Αθηναίων

Η ιστοσελίδα του Δήμου Αθηναίων έχει την εξής ιδιομορφία: Το μενού έχει ίδιο χρώμα με το σώμα της ιστοσελίδας, κάνοντας το αισθητικά ευχάριστο, χάνοντας όμως την ικανότητα να ξεχωρίζει στον χρήστη. Αντίθετα, το βλέμμα του χρήστη επικεντρώνεται στο slideshow που δείχνει τα νέα και δράσεις του Δήμου, μονοπωλώντας το ενδιαφέρον του.

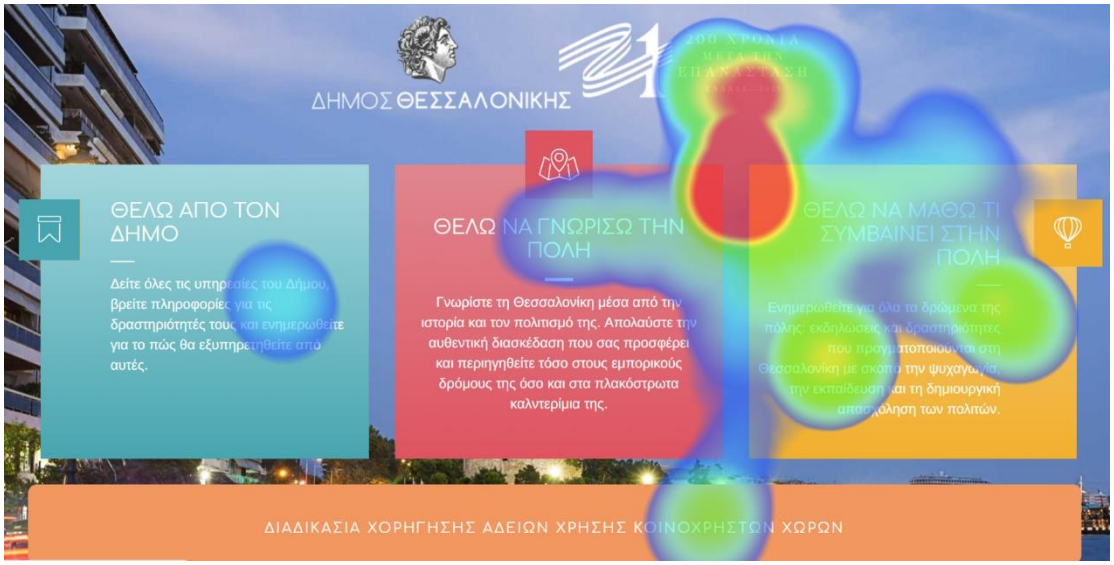
Εικόνα 20 Ιστοσελίδα Δήμου Αθηναίων

Ακολουθεί το γράφημα που δείχνει το ποσοστό της προσοχής του χρήστη που έλαβε το κάθε σημείο της ιστοσελίδας.

Εικόνα 21 Δήμος Αθηναίων - Σημεία προσοχής χρήστη

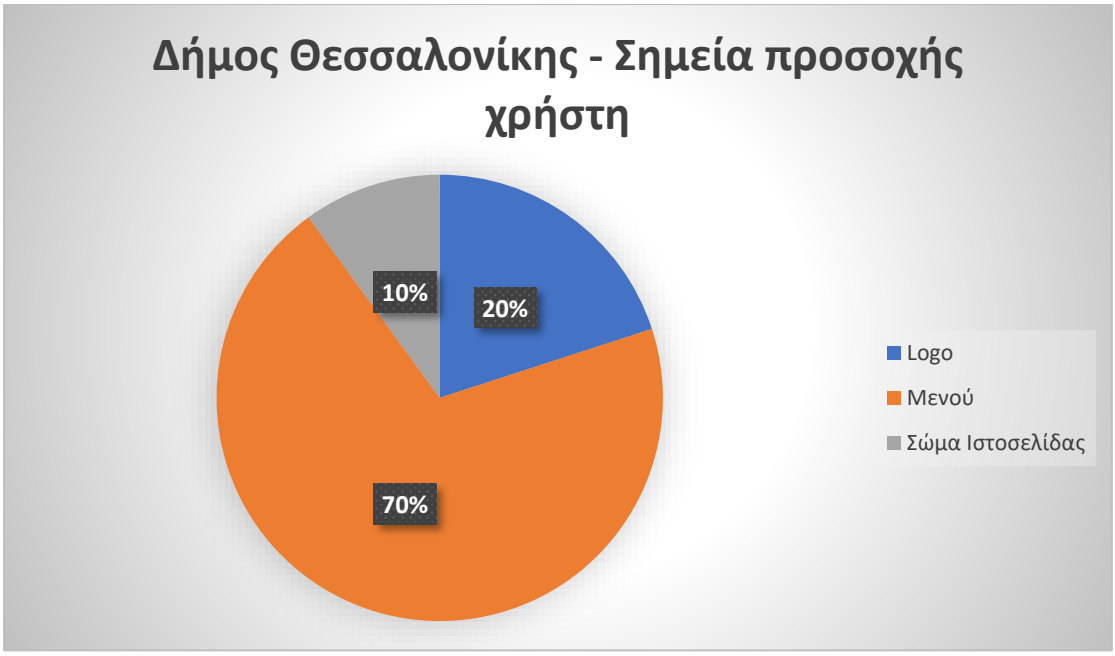
5.4 Ιστοσελίδα Δήμου Θεσσαλονίκης

Στον Δήμο Θεσσαλονίκης, το logo της είναι αναλογικά μεγαλύτερο σε σχέση με τις προηγούμενες ιστοσελίδες, και το μενού καλύπτει το μεγαλύτερο κομμάτι της οθόνης, καθώς περιέχει και εκτεταμένες πληροφορίες για το τι εμπεριέχει η κάθε κατηγορία. Παρ' όλο λοιπόν που ακολούθησαν την ίδια τακτική που ακολουθεί και ο Δήμος Καβάλας και μείωσαν το μενού σε τρεις μόνο κατηγορίες, έχει αυξήσει το μέγεθος του σε σημείο που δεν μπορεί να εισαχθεί κανένα άλλο στοιχείο στην αρχική σελίδα της ιστοσελίδας (slideshow, Νέα κτλ).



Εικόνα 22 Ιστοσελίδα Δήμου Θεσσαλονίκης

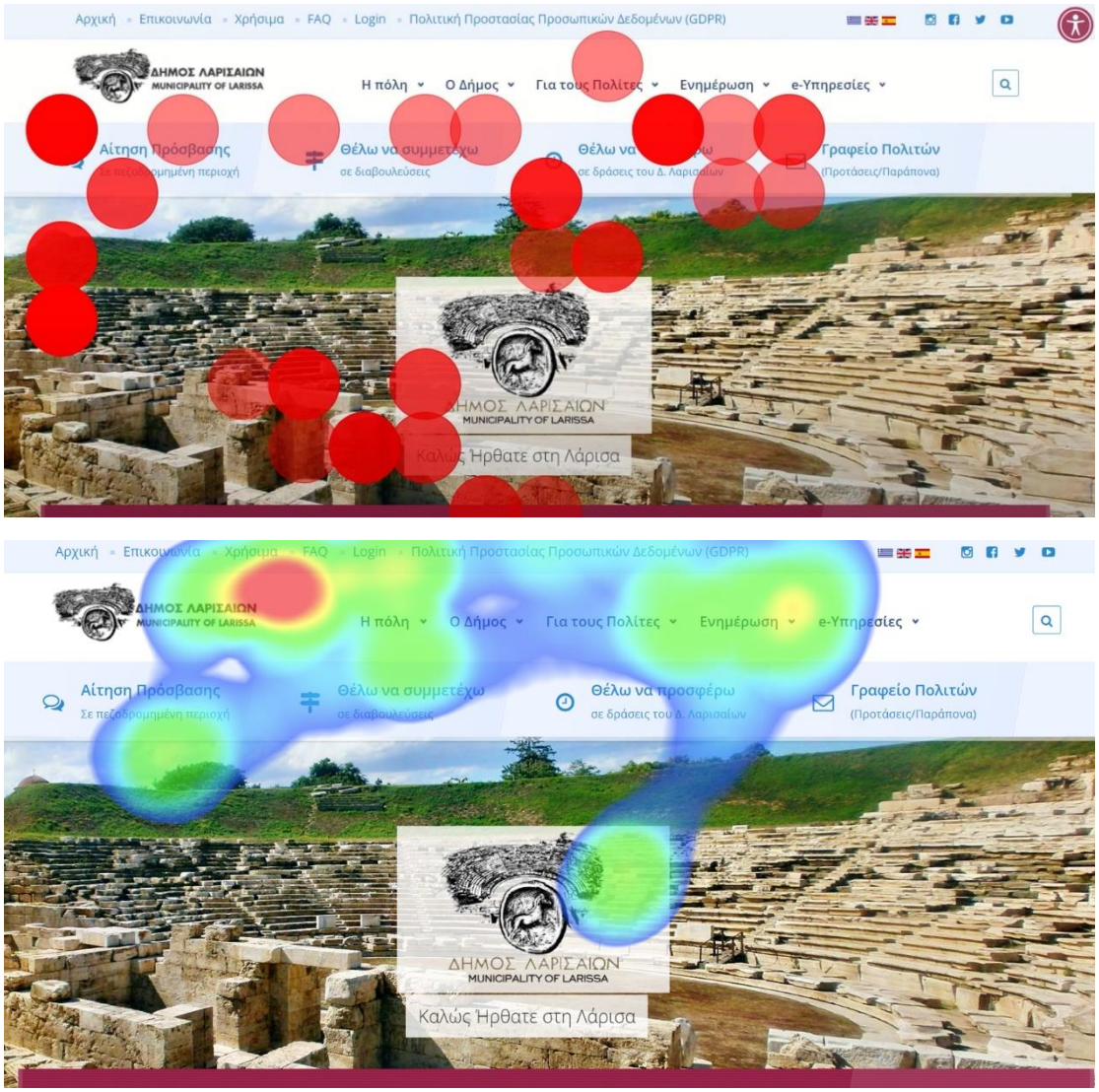
Ακολουθεί το γράφημα που δείχνει το ποσοστό της προσοχής του χρήστη που έλαβε το κάθε σημείο της ιστοσελίδας.



Εικόνα 23 Δήμος Θεσσαλονίκης - Σημεία προσοχής χρήστη

5.5 Ιστοσελίδα Δήμου Λαρισαίων

Ο Δήμος Λαρισαίων επέλεξε διπλή σειρά μενού που καλύπτουν το 40% της αρχικής οθόνης, και slideshow που καλύπτει το υπόλοιπο, το οποίο δείχνει γνωστά μνημεία της πόλης, κάτι που είναι ήδη αναγνωρίσιμο, τις περισσότερες φορές, στον χρήστη, οπότε και επικεντρώνεται στα μενού, ώστε εντοπίζει γρήγορα αυτό που ψάχνει.



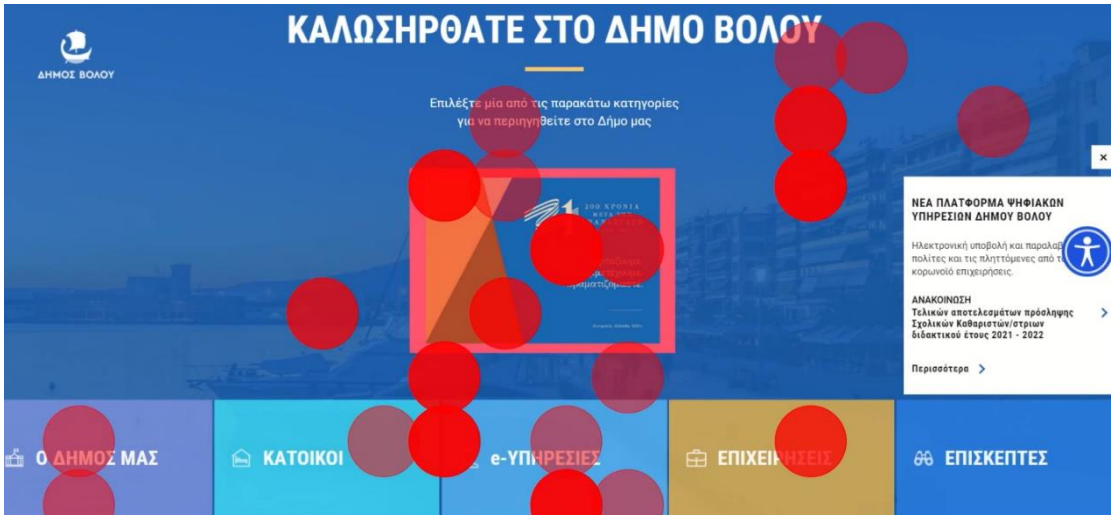
Εικόνα 24 Ιστοσελίδα Δήμου Λαρισαίων

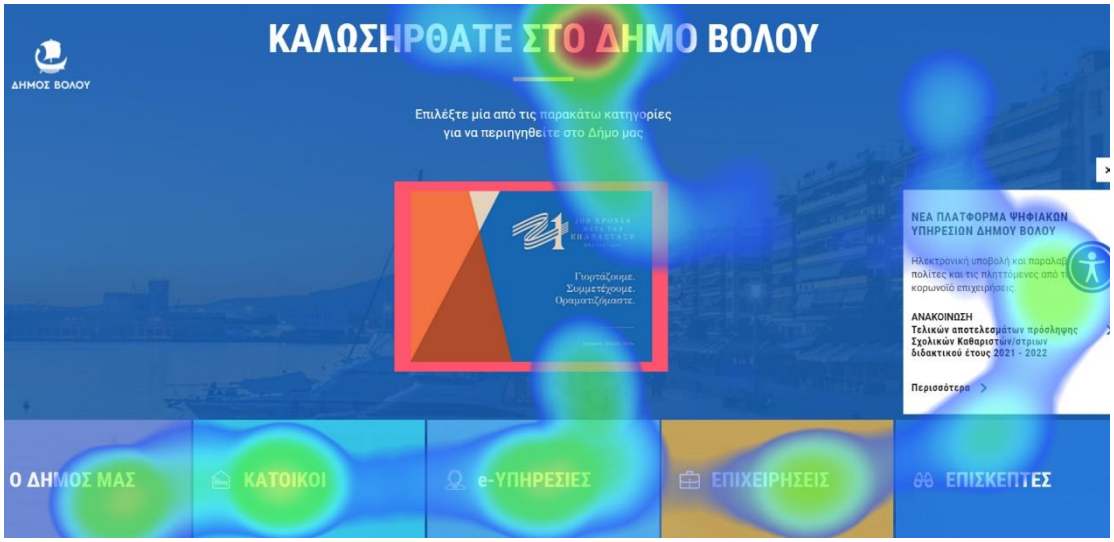
Ακολουθεί το γράφημα που δείχνει το ποσοστό της προσοχής του χρήστη που έλαβε το κάθε σημείο της ιστοσελίδας.

Εικόνα 25 Δήμος Λαρισαίων - Σημεία προσοχής χρήστη

5.6 Ιστοσελίδα Δήμου Βόλου

Η ιστοσελίδα του Δήμου Βόλου είναι ομοιόμορφη χρωματικά, με μόνα σημεία που ξεχωρίζουν να είναι το μενού, που βρίσκεται στο κάτω μέρος της σελίδας, και μία μικρή εικόνα στο κέντρο της. Ο χρήστης φαίνεται ότι εντοπίζει αμέσως, απερίσπαστος, το σημείο που αναζητά, χωρίς ιδιαίτερη δυσκολία.





Εικόνα 26 Ιστοσελίδα Δήμου Βόλου

Ακολουθεί το γράφημα που δείχνει το ποσοστό της προσοχής του χρήστη που έλαβε το κάθε σημείο της ιστοσελίδας.

Εικόνα 27 Δήμος Βόλου - Σημεία προσοχής χρήστη

6. Συμπεράσματα

Παρ' όλο που υπάρχει ακόμα μεγάλο περιθώριο εξέλιξης της «Όρασης Υπολογιστών (Computer Vision)», είναι αλήθεια ότι έχουν γίνει πολλά βήματα για την βελτίωση της, και πλέον δεν είναι απαραίτητο να βασιζόμαστε σε ακριβό τεχνολογικό εξοπλισμό για την αξιοποίηση της.

Σκοπός της εργασίας ήταν η δημιουργία προγράμματος που, χρησιμοποιώντας μία απλή web κάμερα, ανιχνεύει την κίνηση της ίριδας του χρήστη. Επιλέξαμε την προβολή ιστοσελίδων στον χρήστη, ώστε να βγάλουμε συμπέρασμα για την χρηστικότητα τους, ενώ μπορεί να χρησιμοποιηθεί σε πολλούς τομείς, όπως σε θέματα ασφάλειας, σε μελέτες διάσπασης προσοχής, στον έλεγχο αποδοτικότητας υπαλλήλων[47] κτλ.

Μετά από την παρουσίαση έξι ιστοσελίδων ελληνικών Δήμων στον χρήστη, μπορούμε να συμπεράνουμε ότι ο σχεδιασμός των ιστοσελίδων επηρεάζει σε μέγιστο βαθμό την εμπειρία του χρήστη, καθώς κάποιες ιστοσελίδες επικεντρώνονται στην παρουσίαση των δράσεων του Δήμου, άλλοι στην παρουσίαση των τουριστικών τους μνημείων, και άλλοι στην κατηγοριοποίηση και προβολή των ενοτήτων της ιστοσελίδας.

Είναι επίσης εμφανές ότι σε ανάλογα projects που σχετίζονται με ανίχνευση βλέμματος, η επιλογή της python για τον προγραμματισμό τους είναι μία ασφαλής επιλογή, μιας και η πληθώρα βιβλιοθηκών σχετικές με το θέμα είναι πολύ χρήσιμη και επιτρέπει στον προγραμματιστή να πραγματοποιήσει σχεδόν τα πάντα.

Μελλοντικά, η εφαρμογή αυτή θα επεκταθεί πέρα από την απλή ανίχνευση του βλέμματος, ώστε να επιτρέπει στον χρήστη να αλληλεπιδρά με αντικείμενα στην οθόνη ενός ηλεκτρονικού υπολογιστή, χρησιμοποιώντας μόνο το βλέμμα του αντί για τη χρήση ποντικιού ή πληκτρολογίου[48]. Με αυτό τον τρόπο θα διευκολύνεται η επικοινωνία ανθρώπων με κινητικές ιδιαιτερότητες και η εν γένει συμμετοχή τους σε κοινωνικές δραστηριότητες[49].

7. Βιβλιογραφία

1. Brunyé, T.T.,Drew, T.,Weaver, D.L.*, «A review of eye tracking for understanding and improving diagnostic interpretation.»,* Cogn. Research 4,7, 2019,doi: https://doi.org/10.1186/s41235-019-0159-2
2. Białowąs, Sylwester & Szyszka, Adrianna, *«Eye-tracking in Marketing Research»,* 2019, doi: 10.12657/9788379862771-6.
3. Harper, Allen V.R.*, «Eye Tracking and Performance Evaluation: Automatic Detection of User Outcomes»,* CUNY Academic Works, 2015, doi: https://academicworks.cuny.edu/gc_etds/964
4. Katsini C., Abdrabou Y., E. Raptis G., Khamis M., and Alt F., «*The Role of Eye Gaze in Security and Privacy Applications: Survey and Future HCI Research Directions*», Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, 1–21, 2020, doi:https://doi.org/10.1145/3313831.3376840
5. Sharif B., Shaffer T., *«The Use of Eye Tracking in Software Development.»,* In: Schmorrow D.D., Fidopiastis C.M., Foundations of Augmented Cognition. AC 2015. Lecture Notes in Computer Science, vol 9183. Springer, Cham., 2015, doi: https://doi.org/10.1007/978-3-319-20816-9_77
6. Wikipedia, *«Γνωσιακή επιστήμη»*, url: https://el.wikipedia.org/wiki/Γνωσιακή_επιστήμη , Οκτώβριος 2021
7. Walters B., Shaffer T., Sharif B., and Kagdi H., *«Capturing software traceability links from developers' eye gazes»,* Proceedings of the 22nd International Conference on Program Comprehension (ICPC 2014). Association for Computing Machinery, New York, NY, USA, 201–204, 2014, doi: https://doi.org/10.1145/2597008.2597795
8. Python, *«Python Official Website»*, url:https://www.python.org/ , Οκτώβριος 2021
9. Wikipedia, *«Γλώσσα Προγραμματισμού Python»*, url:https://el.wikipedia.org/wiki/Python, Οκτώβριος 2021
10. Data Flair, *«Python Libraries – Python Standard Library & List of Important Libraries»*, url:https://data-flair.training/blogs/python-libraries/, Οκτώβριος 2021
11. Dataquest, *«15 Python Libraries for Data Science»*, url:https://www.dataquest.io/blog/15-python-libraries-for-data-science/, Οκτώβριος 2021
12. Python, *«The Python Standard Library»*, url:https://docs.python.org/3/library/, Οκτώβριος 2021
13. Scrapy, *«Scrapy Library»*, url:https://scrapy.org/ , Οκτώβριος 2021
14. Crummy, *«BeautifulSoup Library»*, url:https://www.crummy.com/software/BeautifulSoup/, Οκτώβριος 2021
15. NumPy, *«NumPy Library»*, url:https://numpy.org/, Οκτώβριος 2021
16. Pydata, *«Pandas Library»*, url:https://pandas.pydata.org/, Οκτώβριος 2021
17. SciPy, *«SciPy Library»*, url:https://www.scipy.org/, Οκτώβριος 2021
18. Keras, *«Keras Library»*, url:https://keras.io/, Οκτώβριος 2021
19. PyTorch, *«PytorchLibrary»*, url:https://pytorch.org/, Οκτώβριος 2021
20. Tensorflow, *«TensorFlow Library»*, url:https://www.tensorflow.org/, Οκτώβριος 2021
21. XGBoost, *«XGBoost Library»*, url:https://xgboost.ai/, Οκτώβριος 2021
22. Matplotlib, *«Matplotlib»*, url:https://matplotlib.org/, Οκτώβριος 2021
23. Pydata, *«Seaborn Library»*, url:https://seaborn.pydata.org/, Οκτώβριος 2021

24. Plotly, *«Plotly Library»*, url:https://plotly.com/, Οκτώβριος 2021
25. PyPi, *«Pydot Library»*, url:https://pypi.org/project/pydot/, Οκτώβριος 2021
26. QuTip, *«QuTip Library»*, url:https://qutip.org/, Οκτώβριος 2021
27. QisKIT, *«QisKIT Library»*, url:https://qiskit.org/, Οκτώβριος 2021
28. Google's QuantumAI, *«CirQ Library»*, url:https://quantumai.google/cirq, Οκτώβριος 2021
29. Pyglet, *«Pyglet Library»*, url:http://pyglet.org/, Οκτώβριος 2021
30. Pygame, *«Pygame Library»*, url:https://www.pygame.org/, Οκτώβριος 2021
31. OpenCV, *«OpenCV Library»*, https://opencv.org/, Οκτώβριος 2021
32. Cmake, *«Cmake Library»*, url: https://cmake.org/, Οκτώβριος 2021
33. Dlib, *«Dlib Library»*, url:http://dlib.net/, Οκτώβριος 2021
34. Python Docs, *«Collections Library»,* https://docs.python.org/3/library/collections.html, Οκτώβριος 2021
35. Python Docs, *«Math Library»,* https://docs.python.org/3/library/math.html, Οκτώβριος 2021
36. Matplotlib, *«Pyplotlib of Matplotlib Library»,* https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html, Οκτώβριος 2021
37. Matplotlib, *«Patches.Circle of Matplotlib Library»,* https://matplotlib.org/stable/api/_as_gen/matplotlib.patches.Circle.html, Οκτώβριος 2021
38. Jetbrains, *«Pycharm»,* https://www.jetbrains.com/pycharm/, Οκτώβριος 2021
39. Wikipedia, *«Gaussian Blur»,* https://en.wikipedia.org/wiki/Gaussian_blur, Οκτώβριος 2021
40. Adobe, *«Photoshop»,* https://www.adobe.com/gr_en/products/photoshop.html, Οκτώβριος 2021
41. Wikipedia, *«Gaussian Elimination»,* https://en.wikipedia.org/wiki/Gaussian_elimination, Οκτώβριος 2021
42. Kasprowski P., Harężlak K., Stasch M.*, «Guidelines for the Eye Tracker Calibration Using Points of Regard»,* In: Piętka E., Kawa J., Wieclawek W., Information Technologies in Biomedicine, Volume 4. Advances in Intelligent Systems and Computing, vol 284. Springer, Cham, 2014, https://doi.org/10.1007/978-3-319-06596-0_21
43. Nyström, M., Andersson, R., Holmqvist, K. et al., *«The influence of calibration method and eye physiology on eyetracking data quality»*, Behav Res 45, 272–288, 2013, https://doi.org/10.3758/s13428-012-0247-4
44. Pasarica, Alexandru & Andruseac, Gladiola & Adochiei, Ioana & Rotariu, Cristi & Costin, Hariton & Adochiei, Felix, *«Remote Control of an Autonomous Robotic Platform Based on Eye Tracking»*, Advances in Electrical and Computing Engineering. 16. 95-100, 2016, doi:10.4316/AECE.2016.04015.
45. Weichbroth, Paweł & Redlarski, Krzysztof & Garnik, Igor., «*Eye-tracking Web Usability Research*», Annals of computer science and information systems, 2016, Doi:8. 10.15439/2016F127.
46. Goldreich, David and Halaburda, Hanna, «*When Smaller Menus are Better: Variability in Menu-Setting Ability»*, Management Science 59, no. 11: 2518–2535, 2013, http://dx.doi.org/10.2139/ssrn.2023956
47. Walters B., Shaffer T., Sharif B., and Kagdi H., *«Capturing software traceability links from developers' eye gazes»*, Proceedings of the 22nd International Conference on

Program Comprehension (ICPC 2014). Association for Computing Machinery, New York, NY, USA, 201–204, 2014, doi: https://doi.org/10.1145/2597008.2597795

48. Borgestig M., Al Khatib I., Masayko S. & Hemmingsson H., *«The Impact of Eye-gaze Controlled Computer on Communication and Functional Independence in Children and Young People with Complex Needs – A Multicenter Intervention Study»*, Developmental Neurorehabilitation, 2021, doi: 10.1080/17518423.2021.1903603
49. Hemmingsson, H., & Borgestig, M., «*Usability of Eye-Gaze Controlled Computers in Sweden: A Total Population Survey*», International journal of environmental research and public health, 17(5), 1639, 2020, doi: https://doi.org/10.3390/ijerph17051639

8. Παράρτημα – Κώδικας εφαρμογής

```
import cv2 as cv
import numpy as np
import dlib
from collections import defaultdict
from math import sqrt
import matplotlib.pyplot as plt
from matplotlib.patches import Circle

bigAverageArray = [] # array with estimations of all eye positions across
screen
screenDimArray = [] # array with screen dimensions
imgTime = []

dx = defaultdict(list)
dy = defaultdict(list)
averagex = defaultdict(list)
averagey = defaultdict(list)
pointx=defaultdict(list)
pointy=defaultdict(list)
circleCoord=defaultdict(list)

cap = cv.VideoCapture(0)

detector = dlib.get_frontal_face_detector()
predictor =
dlib.shape_predictor("Resources/shape_predictor_68_face_landmarks.dat")

f = open("CoordinationFile.txt", "w")

#VIDEO
fileName="Resources/video3.mp4"
slomo_frame = 20
capVideo = cv.VideoCapture(fileName)       # load the video
timestamps = [capVideo.get(cv.CAP_PROP_POS_MSEC)]

def midpoint(p1,p2):
    return int((p1.x + p2.x)/2), int ((p1.y + p2.y)/2)

def Average(lst):
    return sum(lst) / len(lst)

def distance(p1x,p1y,p2x,p2y):
    return sqrt( ((int(p1x)-int(p2x))**2)+((int(p1y)-int(p2y))**2) )

while(capVideo.isOpened()): # play the video by reading frame by frame
    _,frame = cap.read()
    #VIDEO
    ret, frameVideo = capVideo.read()
    if ret == True:
        #VIDEO
        cv.imshow('frameVideo', frameVideo)  # show the video
        timestamps.append(capVideo.get(cv.CAP_PROP_POS_MSEC))


        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)

        faces = detector(gray)
        # getting dimensions of image
        dim = gray.shape
        # creating mask
        mask = np.zeros(dim, dtype=np.uint8)


        for face in faces:
            landmarks = predictor(gray, face)
```

```
            eyePoints = []

            for n in range (36, 42):
                point = (landmarks.part(n).x, landmarks.part(n).y)
                eyePoints.append(point)

            # converting eyePoints into Numpy arrays.
            PollyPoints = np.array(eyePoints, dtype=np.int32)
            # Filling the Eyes portion with WHITE color.
            cv.fillPoly(mask, [PollyPoints], 255)

            # Writing gray image where color is White  in the mask using
Bitwise and operator.
            eyeImage = cv.bitwise_and(gray, gray, mask=mask)

            # getting the max and min points of eye inorder to crop the eyes
from Eye image .

            maxX = (max(eyePoints, key=lambda item: item[0]))[0]
            minX = (min(eyePoints, key=lambda item: item[0]))[0]
            maxY = (max(eyePoints, key=lambda item: item[1]))[1]
            minY = (min(eyePoints, key=lambda item: item[1]))[1]

            # other then eye area will black, making it white
            eyeImage[mask == 0] = 255

            # cropping the eye form eyeImage.
            cropedEye = eyeImage[minY:maxY, minX:maxX]

            # getting width and height of cropedEye
            height, width = cropedEye.shape

            divPart = int(width / 3)

            #  applying the threshold to the eye .
            ret, thresholdEye = cv.threshold(cropedEye, 45, 255,
cv.THRESH_BINARY_INV)

            contours, _ = cv.findContours(thresholdEye, cv.RETR_TREE,
cv.CHAIN_APPROX_SIMPLE)
            contours = sorted(contours, key=lambda x: cv.contourArea(x),
reverse=True)
            for cnt in contours:
                (x, y, w, h) = cv.boundingRect(cnt)
                iris_center=(int((minX+x)+w/2),int((minY+y)+h/2))
                cv.circle(frame, iris_center, 1, (0,0,255), 2)
                coord=(x+w/2,y+h/2)
                #print(coord)
                f.write(str(coord)+" "+
str(capVideo.get(cv.CAP_PROP_POS_MSEC))+"\n")
                second=int(capVideo.get(cv.CAP_PROP_POS_MSEC)/1000)
                dx[second].append(float(x+w/2))
                dy[second].append(float(y+h/2))
                break


        cv.imshow("Frame", frame)


        key = cv.waitKey(1)
        if key == 27:
            break
    else:
        break
f.close()
cap.release()
capVideo.release()
cv.destroyAllWindows()
```

```
average_file = open("average_file.txt", "w")
current = 0
while current < len(dx):
    averagex[current] = Average(dx[current])
    averagey[current] = Average(dy[current])
    average_file.write(str(current) + " " + str(averagex[current]) + " " +
str(averagey[current]) + "\n")
    current+=1
#average = Average(lst)
average_file.close()
if current <35 : print("Calibration incomplete. Please retry")
#point 1
pointx[1]=(float(averagex[3])+float(averagex[4]))/2
pointy[1]=(float(averagey[3])+float(averagey[4]))/2
circleCoord[1] = (135,108)
#point 2
pointx[2]=(float(averagex[6])+float(averagex[7])+float(averagex[8]))/3
pointy[2]=(float(averagey[6])+float(averagey[7])+float(averagey[8]))/3
circleCoord[2] = (960,108)
#point 3
pointx[3]=(float(averagex[10])+float(averagex[11]))/2
pointy[3]=(float(averagey[10])+float(averagey[11]))/2
circleCoord[3] = (1786,108)
#point 4
pointx[4]=(float(averagex[14])+float(averagex[15]))/2
pointy[4]=(float(averagey[14])+float(averagey[15]))/2
circleCoord[4] = (135,540)
#point 5
pointx[5]=(float(averagex[18])+float(averagex[19]))/2
pointy[5]=(float(averagey[18])+float(averagey[19]))/2
circleCoord[5] = (960,540)
#point 6
pointx[6]=(float(averagex[21])+float(averagex[22]))/2
pointy[6]=(float(averagey[21])+float(averagey[22]))/2
circleCoord[6] = (1786,540)
#point 7
pointx[7]=(float(averagex[25])+float(averagex[26]))/2
pointy[7]=(float(averagey[25])+float(averagey[26]))/2
circleCoord[7] = (135,971)
#point 8
pointx[8]=(float(averagex[29])+float(averagex[30]))/2
pointy[8]=(float(averagey[29])+float(averagey[30]))/2
circleCoord[8] = (960,971)
#point 9
pointx[9]=(float(averagex[33])+float(averagex[34]))/2
pointy[9]=(float(averagey[33])+float(averagey[34]))/2
circleCoord[9] = (1786,971)


#for i in range(1,10):
#    print(str(i)+ " " + str(pointx[i])+" "+str(pointy[i]))

rows, cols = (209, 2)
bigAverageArray = [[0 for i in range(cols)] for j in range(rows)]
screenDimArray = [[0 for i in range(cols)] for j in range(rows)]
#point 1
bigAverageArray[0][0] = pointx[1]
bigAverageArray[0][1] = pointy[1]
screenDimArray[0][0] = 135
screenDimArray[0][1] = 108
#point 2
bigAverageArray[9][0] = pointx[2]
bigAverageArray[9][1] = pointy[2]
screenDimArray[9][0] = 960
screenDimArray[9][1] = 108
#point 3
bigAverageArray[18][0] = pointx[3]
bigAverageArray[18][1] = pointy[3]
```

```
screenDimArray[18][0] = 1786
screenDimArray[18][1] = 108
#point 4
bigAverageArray[95][0] = pointx[4]
bigAverageArray[95][1] = pointy[4]
screenDimArray[95][0] = 135
screenDimArray[95][1] = 540
#point 5
bigAverageArray[104][0] = pointx[5]
bigAverageArray[104][1] = pointy[5]
screenDimArray[104][0] = 960
screenDimArray[104][1] = 540
#point 6
bigAverageArray[113][0] = pointx[6]
bigAverageArray[113][1] = pointy[6]
screenDimArray[113][0] = 1786
screenDimArray[113][1] = 540
#point 7
bigAverageArray[190][0] = pointx[7]
bigAverageArray[190][1] = pointy[7]
screenDimArray[190][0] = 135
screenDimArray[190][1] = 971
#point 8
bigAverageArray[199][0] = pointx[8]
bigAverageArray[199][1] = pointy[8]
screenDimArray[199][0] = 960
screenDimArray[199][1] = 971
#point 9
bigAverageArray[208][0] = pointx[9]
bigAverageArray[208][1] = pointy[9]
screenDimArray[208][0] = 1786
screenDimArray[208][1] = 971

#between points 1 and 2
absValueX12 = (pointx[2] - pointx[1]) / 8 #8 spaces between 1 and 2 --- if 1
bigger than 2, it will be negative, what we want!
absValueY12 = (pointy[2] - pointy[1]) / 8
absScreenX = (screenDimArray[9][0] - screenDimArray[0][0]) /8 # only once,
everything the same
absScreenY = (screenDimArray[9][1] - screenDimArray[0][1]) /8
for i in range(1,9):
    bigAverageArray[i][0] = bigAverageArray[i-1][0] + absValueX12
    bigAverageArray[i][1] = bigAverageArray[i-1][1] + absValueY12
    screenDimArray[i][0] = int(screenDimArray[i-1][0] + absScreenX)
    screenDimArray[i][1] = int(screenDimArray[i-1][1] + absScreenY)
#between points 2 and 3
absValueX23 = (pointx[3] - pointx[2]) / 8
absValueY23 = (pointy[3] - pointy[2]) / 8
for i in range(10,18):
    bigAverageArray[i][0] = bigAverageArray[i-1][0] + absValueX23
    bigAverageArray[i][1] = bigAverageArray[i-1][1] + absValueY23
    screenDimArray[i][0] = int(screenDimArray[i-1][0] + absScreenX)
    screenDimArray[i][1] = int(screenDimArray[i-1][1] + absScreenY)
# between points 4 and 5
absValueX45 = (pointx[5] - pointx[4]) / 8
absValueY45 = (pointy[5] - pointy[4]) / 8
for i in range(96, 104):
    bigAverageArray[i][0] = bigAverageArray[i - 1][0] + absValueX45
    bigAverageArray[i][1] = bigAverageArray[i - 1][1] + absValueY45
    screenDimArray[i][0] = int(screenDimArray[i-1][0] + absScreenX)
    screenDimArray[i][1] = int(screenDimArray[i-1][1] + absScreenY)
# between points 5 and 6
absValueX56 = (pointx[6] - pointx[5]) / 8
absValueY56 = (pointy[6] - pointy[5]) / 8
for i in range(105, 113):
    bigAverageArray[i][0] = bigAverageArray[i - 1][0] + absValueX56
    bigAverageArray[i][1] = bigAverageArray[i - 1][1] + absValueY56
    screenDimArray[i][0] = int(screenDimArray[i-1][0] + absScreenX)
```

```
    screenDimArray[i][1] = int(screenDimArray[i-1][1] + absScreenY)
# between points 7 and 8
absValueX78 = (pointx[8] - pointx[7]) / 8
absValueY78 = (pointy[8] - pointy[7]) / 8
for i in range(191, 199):
    bigAverageArray[i][0] = bigAverageArray[i - 1][0] + absValueX78
    bigAverageArray[i][1] = bigAverageArray[i - 1][1] + absValueY78
    screenDimArray[i][0] = int(screenDimArray[i-1][0] + absScreenX)
    screenDimArray[i][1] = int(screenDimArray[i-1][1] + absScreenY)
# between points 8 and 9
absValueX89 = (pointx[9] - pointx[8]) / 8
absValueY89 = (pointy[9] - pointy[8]) / 8
for i in range(200, 208):
    bigAverageArray[i][0] = bigAverageArray[i - 1][0] + absValueX89
    bigAverageArray[i][1] = bigAverageArray[i - 1][1] + absValueY89
    screenDimArray[i][0] = int(screenDimArray[i-1][0] + absScreenX)
    screenDimArray[i][1] = int(screenDimArray[i-1][1] + absScreenY)

#fill up everything vertically between points 1 and 6
absScreenX = (screenDimArray[95][0] - screenDimArray[0][0]) / 4 # just once,
everything the same
absScreenY = (screenDimArray[95][1] - screenDimArray[0][1]) / 4
j=95 # position of point 4
for i in range(0, 19): # positions of points 1 to 3
    absValueX = (bigAverageArray[j][0] - bigAverageArray[i][0]) / 4 #4
spaces between
    absValueY = (bigAverageArray[j][1] - bigAverageArray[i][1]) / 4
    for k in range(1,5): #4 spaces between horizontals
        bigAverageArray[k*19+i][0] = bigAverageArray[(k-1)*19+i][0] +
absValueX
        bigAverageArray[k*19+i][1] = bigAverageArray[(k-1)*19+i][1] +
absValueY
        screenDimArray[k*19+i][0] = int(screenDimArray[(k-1)*19+i][0] +
absScreenX)
        screenDimArray[k*19+i][1] = int(screenDimArray[(k-1)*19+i][1] +
absScreenY)
    j+=1

#fill up everything vertically between points 4 and 9
j=190 # position of point 7
for i in range(95, 114): # positions of points 4 to 6
    absValueX = (bigAverageArray[j][0] - bigAverageArray[i][0]) / 4 #4
spaces between
    absValueY = (bigAverageArray[j][1] - bigAverageArray[i][1]) / 4
    for k in range(1,5): #4 spaces between horizontals
        bigAverageArray[k*19+i][0] = bigAverageArray[(k-1)*19+i][0] +
absValueX
        bigAverageArray[k*19+i][1] = bigAverageArray[(k-1)*19+i][1] +
absValueY
        screenDimArray[k*19+i][0] = int(screenDimArray[(k-1)*19+i][0] +
absScreenX)
        screenDimArray[k*19+i][1] = int(screenDimArray[(k-1)*19+i][1] +
absScreenY)
    j+=1

#SECOND STEP OF PROGRAM

cap = cv.VideoCapture(0)

f = open("AfterFile.txt", "w")

#VIDEO
fileName="Resources/dimoi.mp4"
slomo_frame = 10
capVideo = cv.VideoCapture(fileName)       # load the video
timestamps = [capVideo.get(cv.CAP_PROP_POS_MSEC)]

while(capVideo.isOpened()): # play the video by reading frame by frame
```

```
    tempMin = 100000000
    _,frame = cap.read()
    #VIDEO
    ret, frameVideo = capVideo.read()
    if ret == True:
        #VIDEO
        timestamps.append(capVideo.get(cv.CAP_PROP_POS_MSEC))


        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)

        faces = detector(gray)
        # getting dimensions of image
        dim = gray.shape
        # creating mask
        mask = np.zeros(dim, dtype=np.uint8)


        for face in faces:
            landmarks = predictor(gray, face)
            eyePoints = []

            for n in range (36, 42):
                point = (landmarks.part(n).x, landmarks.part(n).y)
                eyePoints.append(point)

            # converting eyePoints into Numpy arrays.
            PollyPoints = np.array(eyePoints, dtype=np.int32)
            # Filling the Eyes portion with WHITE color.
            cv.fillPoly(mask, [PollyPoints], 255)

            # Writing gray image where color is White  in the mask using
Bitwise and operator.
            eyeImage = cv.bitwise_and(gray, gray, mask=mask)

            # getting the max and min points of eye inorder to crop the eyes
from Eye image .

            maxX = (max(eyePoints, key=lambda item: item[0]))[0]
            minX = (min(eyePoints, key=lambda item: item[0]))[0]
            maxY = (max(eyePoints, key=lambda item: item[1]))[1]
            minY = (min(eyePoints, key=lambda item: item[1]))[1]

            # other then eye area will black, making it white
            eyeImage[mask == 0] = 255

            # cropping the eye form eyeImage.
            cropedEye = eyeImage[minY:maxY, minX:maxX]

            # getting width and height of cropedEye
            height, width = cropedEye.shape

            divPart = int(width / 3)

            #  applying the threshold to the eye .
            ret, thresholdEye = cv.threshold(cropedEye, 45, 255,
cv.THRESH_BINARY_INV)

            contours, _ = cv.findContours(thresholdEye, cv.RETR_TREE,
cv.CHAIN_APPROX_SIMPLE)
            contours = sorted(contours, key=lambda x: cv.contourArea(x),
reverse=True)
            for cnt in contours:
                (x, y, w, h) = cv.boundingRect(cnt)
                iris_center=(int((minX+x)+w/2),int((minY+y)+h/2))
                cv.circle(frame, iris_center, 1, (0,0,255), 2)
                coord=(x+w/2,y+h/2)
                #cv.circle(frameVideo, (int(x+w/2),int(y+h/2)), 20, (0, 0,
```

```
0), -1)
                #print(coord)
                second=int(capVideo.get(cv.CAP_PROP_POS_MSEC)/1000)
                dx[second].append(float(x+w/2))
                dy[second].append(float(y+h/2))
                break


        cv.imshow("Frame", frame)
#         for i in range(1, 10):
#             dist = distance(int(x+w/2),int(y+h/2),pointx[i],pointy[i])
#             if dist < tempMin:
#                 tempMin = dist
#                 tempMinI = i
        for i in range(0, 209):
            dist =
distance(float(x+w/2),float(y+h/2),bigAverageArray[i][0],bigAverageArray[i][
1])
            if dist < tempMin:
                tempMin = dist
                tempMinI = i

        #print(str(tempMin) + " " + str(dist))
        #    print(str(i) + " " + str(pointx[i]) + " " + str(pointy[i]))
        cv.circle(frameVideo,
(screenDimArray[tempMinI][0],screenDimArray[tempMinI][1]), 70, (255, 0, 0),
5)
        cv.imshow('frameVideo', frameVideo)  # show the video
        f.write(str(screenDimArray[tempMinI][0]) + " " +
str(screenDimArray[tempMinI][1]) + " " +
str(capVideo.get(cv.CAP_PROP_POS_MSEC)) + "\n")

        key = cv.waitKey(1)
        if key == 27:
            break
    else:
        break
f.close()
cap.release()
capVideo.release()
cv.destroyAllWindows()


#THIRD STEP OF PROGRAM



# read coord file
with open('AfterFile.txt') as f:
    lines = [line.rstrip() for line in f]
#images
first_image_file="Resources/pics/dimos_thassou.png"
second_image_file="Resources/pics/dimos_kavalas.png"
third_image_file="Resources/pics/dimos_athinwn.png"
fourth_image_file="Resources/pics/dimos_thessalonikis.png"
fifth_image_file="Resources/pics/dimos_larisas.png"
sixth_image_file="Resources/pics/dimos_volou.png"

#image timestamps
firstImg = 2273
secondImg = 4353
thirdImg = 6497
fourthImg = 8609
fifthImg = 10593
sixthImg = 12367

#read images
img1 = plt.imread(first_image_file)
```

```
img2 = plt.imread(second_image_file)
img3 = plt.imread(third_image_file)
img4 = plt.imread(fourth_image_file)
img5 = plt.imread(fifth_image_file)
img6 = plt.imread(sixth_image_file)

# Create a figure. Equal aspect so circles look circular
fig, ax = plt.subplots(1)
ax.set_aspect('equal')
# Show the image
ax.imshow(img1)
for l in lines:
    # if first image
    if float(l.split()[2])<=firstImg:
        circ = Circle((float(l.split()[0]), float(l.split()[1])), 60,
alpha=0.5, color='red')
        ax.add_patch(circ)
    else:
        break
fig = plt.gcf()
fig.set_size_inches(18.5, 10.5)
fig.savefig('first.png', dpi=100)
# Show the image
ax.imshow(img1)
# Show the image
plt.show()

# Create a figure. Equal aspect so circles look circular
fig2, ax2 = plt.subplots(1)
ax2.set_aspect('equal')
# Show the image
ax2.imshow(img2)
for l in lines:
    # if second image
    if float(l.split()[2]) <= firstImg:
        continue

    if float(l.split()[2]) <= secondImg:
        circ2 = Circle((float(l.split()[0]), float(l.split()[1])), 60,
alpha=0.5, color='red')
        ax2.add_patch(circ2)
    else:
        break
fig2 = plt.gcf()
fig2.set_size_inches(18.5, 10.5)
fig2.savefig('second.png', dpi=100)
# Show the image
ax2.imshow(img2)
# Show the image
plt.show()

fig3, ax3 = plt.subplots(1)
ax3.set_aspect('equal')
ax3.imshow(img3)
for l in lines:
    # if third image
    if float(l.split()[2]) <= secondImg:
        continue
    if float(l.split()[2]) <= thirdImg:
        circ3 = Circle((float(l.split()[0]), float(l.split()[1])), 60,
alpha=0.5, color='red')
        ax3.add_patch(circ3)
    else:
        break
fig3 = plt.gcf()
fig3.set_size_inches(18.5, 10.5)
fig3.savefig('third.png', dpi=100)
ax3.imshow(img3)
```

```
plt.show()

fig4, ax4 = plt.subplots(1)
ax4.set_aspect('equal')
ax4.imshow(img4)
for l in lines:
    # if fourth image
    if float(l.split()[2]) <= thirdImg:
        continue
    if float(l.split()[2]) <= fourthImg:
        circ4 = Circle((float(l.split()[0]), float(l.split()[1])), 60,
alpha=0.5, color='red')
        ax4.add_patch(circ4)
    else:
        break
fig4 = plt.gcf()
fig4.set_size_inches(18.5, 10.5)
fig4.savefig('fourth.png', dpi=100)
ax4.imshow(img4)
plt.show()

fig5, ax5 = plt.subplots(1)
ax5.set_aspect('equal')
ax5.imshow(img5)
for l in lines:
    # if fifth image
    if float(l.split()[2]) <= fourthImg:
        continue
    if float(l.split()[2]) <= fifthImg:
        circ5 = Circle((float(l.split()[0]), float(l.split()[1])), 60,
alpha=0.5, color='red')
        ax5.add_patch(circ5)
    else:
        break
fig5 = plt.gcf()
fig5.set_size_inches(18.5, 10.5)
fig5.savefig('fifth.png', dpi=100)
ax5.imshow(img5)
plt.show()

fig6, ax6 = plt.subplots(1)
ax6.set_aspect('equal')
ax6.imshow(img6)
for l in lines:
    # if sixth image
    if float(l.split()[2]) <= fifthImg:
        continue
    if float(l.split()[2]) <= sixthImg:
        circ6 = Circle((float(l.split()[0]), float(l.split()[1])), 60,
alpha=0.5, color='red')
        ax6.add_patch(circ6)
    else:
        break
fig6 = plt.gcf()
fig6.set_size_inches(18.5, 10.5)
fig6.savefig('sixth.png', dpi=100)
ax5.imshow(img6)
plt.show()
```