Μεταπτυχιακό Πληροφορικής

Εργασία στο μάθημα « ΙΑΤΡΙΚΗ ΠΛΗΡΟΦΟΡΙΚΗ»

Εργασία του

Κορμάζου Ευάγγελου

(Α.Μ.:ΜΠΠΛ21036)

Επιβλέποντες:

Ευθύμιος Αλέπης,
Διονύσιος Σωτηρόπουλος

Αθήνα, Σεπτέμβριος 2023

Πίνακας περιεχομένων

Εισαγωγή

Η γραφική επεξεργασία εικόνας και η οπτικοποίηση αποτελούν σημαντικά πεδία της επιστήμης των υπολογιστών που έχουν επαναπροσδιορίσει τον τρόπο με τον οποίο αλληλεπιδρούμε με τα δεδομένα και τις εικόνες. Μια από τις βασικές τεχνικές που συμβάλλει στην αναπαράσταση και τον χειρισμό των γραφικών στον ψηφιακό χώρο είναι η χρήση των Bézier καμπύλων. Αυτές οι καμπύλες, προτείνονται από τους Pierre Bézier και Paul de Casteljau τη δεκαετία του 1960, καθώς αποτελούν έναν αποτελεσματικό τρόπο αναπαράστασης και επεξεργασίας γραφικών αντικειμένων και καμπυλών.

Οι Bézier καμπύλες είναι μαθηματικές καμπύλες που καθορίζονται από ένα σύνολο σημείων ελέγχου και χρησιμοποιούνται ευρέως σε εφαρμογές όπως η σχεδίαση γραφικών, η κατασκευή και επεξεργασία γραφικών αντικειμένων, η καμπυλοτεχνία, και η επεξεργασία εικόνων. Η μαγεία των Bézier καμπυλών βρίσκεται στην ικανότητά τους να περιγράφουν ποικίλα σχήματα, από απλές ευθείες γραμμές μέχρι πολύπλοκες καμπύλες και επιφάνειες, με βάση τη θέση των σημείων ελέγχου.

Στο πλαίσιο αυτής της εργασίας, θα εξερευνήσουμε τις Bézier καμπύλες, τη θεωρία που τις διέπει, τις εφαρμογές τους και τον τρόπο υλοποίησής τους σε γλώσσα προγραμματισμού Python. Θα διερευνήσουμε επίσης πώς μπορούν να χρησιμοποιηθούν για τη δημιουργία και την οπτικοποίηση γραφικών στον ψηφιακό χώρο. Τέλος, θα εξετάσουμε τις ποικίλες εφαρμογές των Bézier καμπυλών σε πρακτικά παραδείγματα, αναδεικνύοντας την σημασία τους στον τομέα της επεξεργασίας εικόνων και της γραφικής επεξεργασίας.

Αυτή η εργασία αναδεικνύει τον συναρπαστικό κόσμο των Bézier καμπυλών και τη σημασία τους στον τομέα της ψηφιακής επεξεργασίας και της αναπαράστασης γραφικών.

Κεφάλαιο 1: Θεωρητική Ανασκόπηση των Bézier Καμπυλών:

Το πρώτο κεφάλαιο αυτής της εργασίας αποτελεί μια θεωρητική ανασκόπηση των Bézier καμπυλών, παρέχοντας βασικές πληροφορίες σχετικά με την κατανόηση και τη χρήση τους. Αρχικά, εισάγουμε στον αναγνώστη στις βασικές έννοιες των Bézier καμπυλών, στη συνέχεια περιγράφουμε τη μαθηματική φύση των Bézier καμπυλών, και τελικά ανατρέχουμε στην ιστορία τους και στις εφαρμογές που έχουν στον κόσμο της γραφικής επεξεργασίας και της οπτικοποίησης.

1.1 Βασικές Έννοιες των Bézier Καμπυλών

Οι Bézier καμπύλες αποτελούν μια μαθηματική μέθοδο αναπαράστασης και δημιουργίας καμπυλών στον χώρο. Βασικές έννοιες περιλαμβάνουν:

- **Σημεία Ελέγχου (Control Points):** Τα σημεία ελέγχου είναι τα σημεία που καθορίζουν το σχήμα της καμπύλης. Κάθε σημείο έχει συντεταγμένες και επηρεάζει τη θέση και το σχήμα της Bézier καμπύλης.
- **Τάξη της Καμπύλης (Degree):** Η τάξη μιας Bézier καμπύλης αναφέρεται στον αριθμό των σημείων ελέγχου που χρησιμοποιούνται για την κατασκευή της καμπύλης. Οι κυβικές Bézier καμπύλες, για παράδειγμα, έχουν τάξη 4, δηλαδή 4 σημεία ελέγχου.

1.2 Μαθηματική Φύση των Bézier Καμπυλών

Οι Bézier καμπύλες περιγράφονται μαθηματικά από τη γραμμική συνδυαστική ιδιότητα, γνωστή και ως αλγόριθμος του De Casteljau. Ο αλγόριθμος αυτός χρησιμοποιεί τις συντεταγμένες των σημείων ελέγχου για να υπολογίσει τα σημεία της καμπύλης.

Οι παραμετρικές εξισώσεις χρησιμοποιούνται συνήθως για να εκφράσουν τις συντεταγμένες των σημείων που συνθέτουν ένα γεωμετρικό αντικείμενο, όπως μια καμπύλη ή μια επιφάνεια. Η γενική μορφή μιας παραμετρικής εξίσωσης σε δύο διαστάσεις φαίνεται στην εξίσωση (1):

$$\begin{cases} x = f(t) \\ y = g(t) \end{cases} \quad (1)$$

όπου $t \in [0, 1]$ είναι η παράμετρος και f (t), g(t) είναι οποιαδήποτε ρητή συνάρτηση του t. Για να ανακτήσουμε το σχήμα του αντικειμένου, μπορούμε να πάρουμε δείγμα t από το πεδίο της εξίσωσης και να λάβουμε και τα δύο x και y σύμφωνα με την εξίσωση (1). Διαφορετικά από τη χρήση μιας συνάρτησης μίας τιμής για τη μοντελοποίηση του σχήματος του αντικειμένου, μια

παραμετρική εξίσωση εκφράζει τις συντεταγμένες x και y του σχήματος του αντικειμένου ανεξάρτητα, που της επιτρέπει να μιμείται μια συνάρτηση πολλαπλών τιμών. Αυτή η ιδιότητα παρέχει μεγάλη ευελιξία σε σύγκριση με τη συνάρτηση μίας τιμής για την αναπαράσταση σχήματος.

Η καμπύλη Bézier είναι ένα σύνολο παραμετρικών καμπυλών που έχουν χρησιμοποιηθεί ευρέως σε πολλές πεδία ως αποτελεσματικό εργαλείο σχεδιασμού. Ο ρητός ορισμός μιας καμπύλης Bézier μπορεί να εκφραστεί ως εξίσωση (2):

$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathrm{P}_i, 0 \leq t \leq 1$$

όπου n ανα i είναι οι διωνυμικοί συντελεστές, P = (x , y) είναι το i-οστό σημείο ελέγχου του Bézier και n είναι ο βαθμός της καμπύλης Bézier. Μπορεί κανείς να κατασκευάσει μια καμπύλη Bézier ακολουθώντας τα εξής βήματα:

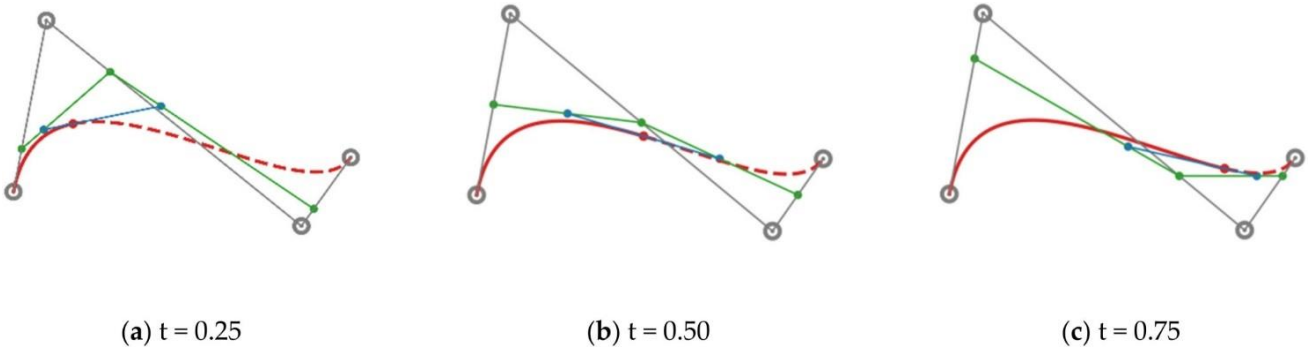
(1)Δημιουργήστε το πολύγωνο ελέγχου της καμπύλης Bézier συνδέοντας τα διαδοχικά σημεία ελέγχου.

(2)Εισάγετε ενδιάμεσα σημεία σε κάθε ευθύγραμμο τμήμα, με την αναλογία t : (1 - t).

(3)Αντιμετωπίστε τα ενδιάμεσα σημεία ως τα νέα σημεία ελέγχου και επαναλάβετε τα βήματα (1) και (2) μέχρι να καταλήξετε σε ένα μόνο σημείο.

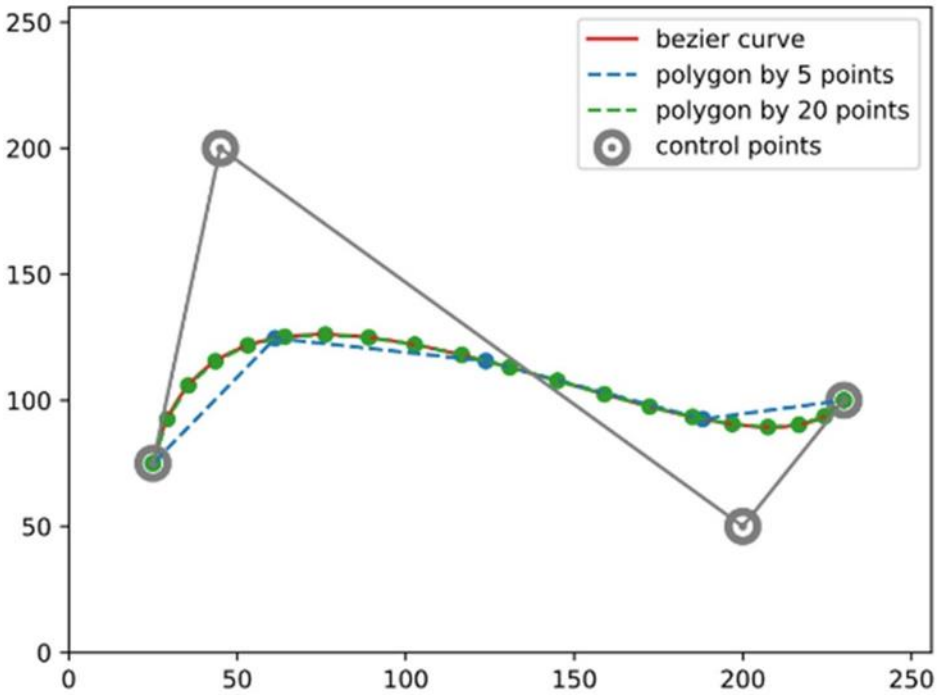
(4)Καθώς το t μεταβάλλεται από το 0 στο 1, η τροχιά αυτού του μοναδικού σημείου σχηματίζει την καμπύλη.

Το Σχήμα 1 δείχνει αυτή τη διαδικασία για την κατασκευή μιας κυβικής καμπύλης Bézier. Η χρήση μιας καμπύλης Bézier μπορεί να μειώσει τον αριθμό των παραμέτρων για την κωδικοποίηση του σχήματος. Όπως φαίνεται στο Σχήμα 2, παρόλο που η καμπύλη καθορίζεται από τέσσερα μόνο σημεία ελέγχου, μπορεί να εγγυηθεί την ποιότητα του σχήματος, δεδομένου ότι η ακρίβεια της αναπαράστασης της καμπύλης εξαρτάται πλήρως από το πόσο πυκνά δειγματοληπτείται το ts εντός του εύρους [0, 1], ενώ η αναπαράσταση πολυγώνου απαιτεί περισσότερες κορυφές για την επίτευξη παρόμοιας ακρίβειας.



(**a**) t = 0.25 (**b**) t = 0.50 (**c**) t = 0.75

Σχήμα 1. Η διαδικασία κατασκευής μιας κυβικής καμπύλης Bézier. Η τροχιά της κόκκινης κουκκίδας σχηματίζει την καμπύλη καθώς το t μεταβάλλεται από 0 έως 1.



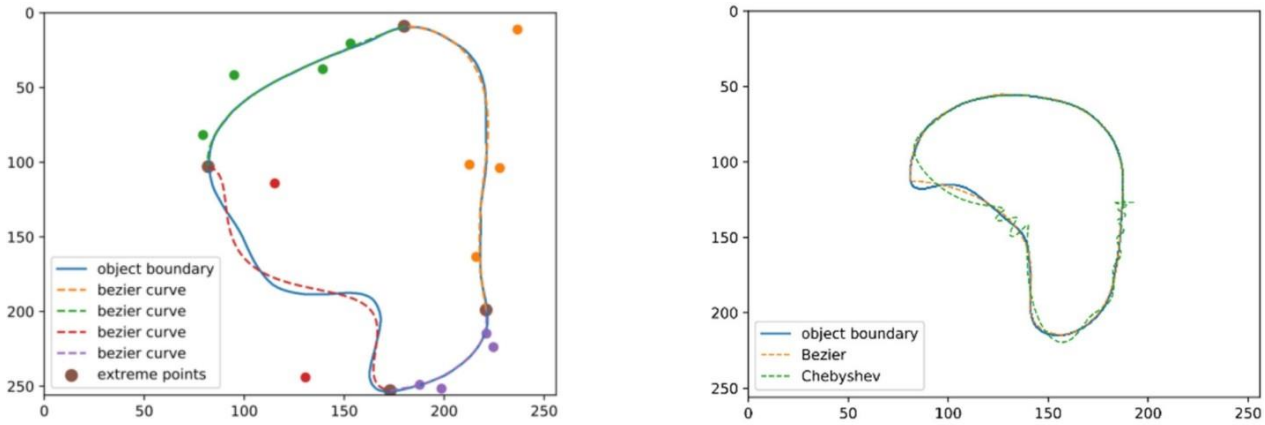
Σχήμα 2. Καμπύλη Bézier έναντι πολυγώνου. Η αναπαράσταση πολυγώνου απαιτεί περισσότερες κορυφές προκειμένου να περιγραφεί με ακρίβεια το σχήμα.

Σημειώστε ότι μπορεί κανείς να μοντελοποιήσει το σχήμα του αντικειμένου χρησιμοποιώντας επίσης μια καμπύλη Bézier με πολικές συντεταγμένες. Προκειμένου να διευκολύνουμε πολύ τη χειροκίνητη μεταγενέστερη βελτίωση του αποτελέσματος τμηματοποίησης, μοντελοποιούμε το σχήμα του αντικειμένου με καρτεσιανές συντεταγμένες.

Πραγματοποιήσαμε ένα πείραμα για να μελετήσουμε την ακρίβεια της παλινδρόμησης καμπύλης σε σχέση με τον βαθμό της καμπύλης Bézier και, σύμφωνα με τον Πίνακα 1, επιλέγουμε n = 5 για έναν συμβιβασμό μεταξύ χρηστικότητας και ακρίβειας τμηματοποίησης. Για να διατηρηθεί υψηλότερη ακρίβεια μετά τη μετατροπή του ορίου του αντικειμένου στην αναπαράσταση καμπύλης Bézier, προτείνουμε να παραμετροποιήσουμε το σχήμα του αντικειμένου με καμπύλες Bézier κατά κομμάτια. Συγκεκριμένα, πρώτα χωρίζουμε ολόκληρο το όριο του αντικειμένου με βάση τα τέσσερα ακραία σημεία του, δηλαδή το πάνω, το αριστερότερο, το κάτω και το δεξιότερο σημείο του αντικειμένου. Στη συνέχεια, για κάθε τμήμα του ορίου του αντικειμένου, το παραμετροποιούμε με μια καμπύλη Bézier. Στην εικόνα 3(α) δίνεται ένα παράδειγμα της αναπαράστασης της καμπύλης Bézier. Συγκρίνουμε επίσης την αναπαράσταση καμπύλης Bézier με την κωδικοποίηση σχήματος πολυωνύμου Chebyshev που προτείνεται στο [18]. Χρησιμοποιώντας το Σχήμα 3β, διαπιστώσαμε ότι η αναπαράσταση της καμπύλης Bézier μας μπορεί να αποφύγει την ταλάντωση στο τέλος του ορίου του αντικειμένου σε αντίθεση με την κωδικοποίηση πολυωνυμικού σχήματος Chebyshev.

Πίνακας 1. Βαθμός της καμπύλης Bézier. Ένας υψηλότερος βαθμός καμπύλης Bézier επιφέρει μεγαλύτερη ακρίβεια, ενώ ένας πολύ υψηλός βαθμός υποβαθμίζει την απόδοση.

Degree of Bézier Curve (n)	3	5	7	9
MIOU	0.753	0.755	0.749	0.736



(a) (b)

Σχήμα 3(α). Η αναπαράσταση της καμπύλης Bézier κατά τεμάχια, τα χρωματισμένα σημεία είναι τα σημεία ελέγχου κάθε αντίστοιχου τμήματος καμπύλης. (β) Η δική μας αναπαράσταση καμπύλης Bézier μπορεί να αποφύγει την ταλάντωση στο τέλος του ορίου του αντικειμένου σε αντίθεση με την κωδικοποίηση σχήματος με πολυώνυμα Chebyshev.

1.3 Ιστορία και Εφαρμογές των Bézier Καμπυλών

Οι Bézier καμπύλες πρωτοεμφανίστηκαν τη δεκαετία του 1960 από τον Γάλλο μηχανικό Pierre Bézier, ο οποίος τις χρησιμοποίησε στην αυτοκινητοβιομηχανία της Renault για τον σχεδιασμό αυτοκινήτων. Αργότερα, ο Paul de Casteljau ανέπτυξε τον αλγόριθμο που φέρει το όνομά του και που επιτρέπει τον υπολογισμό των Bézier καμπυλών.

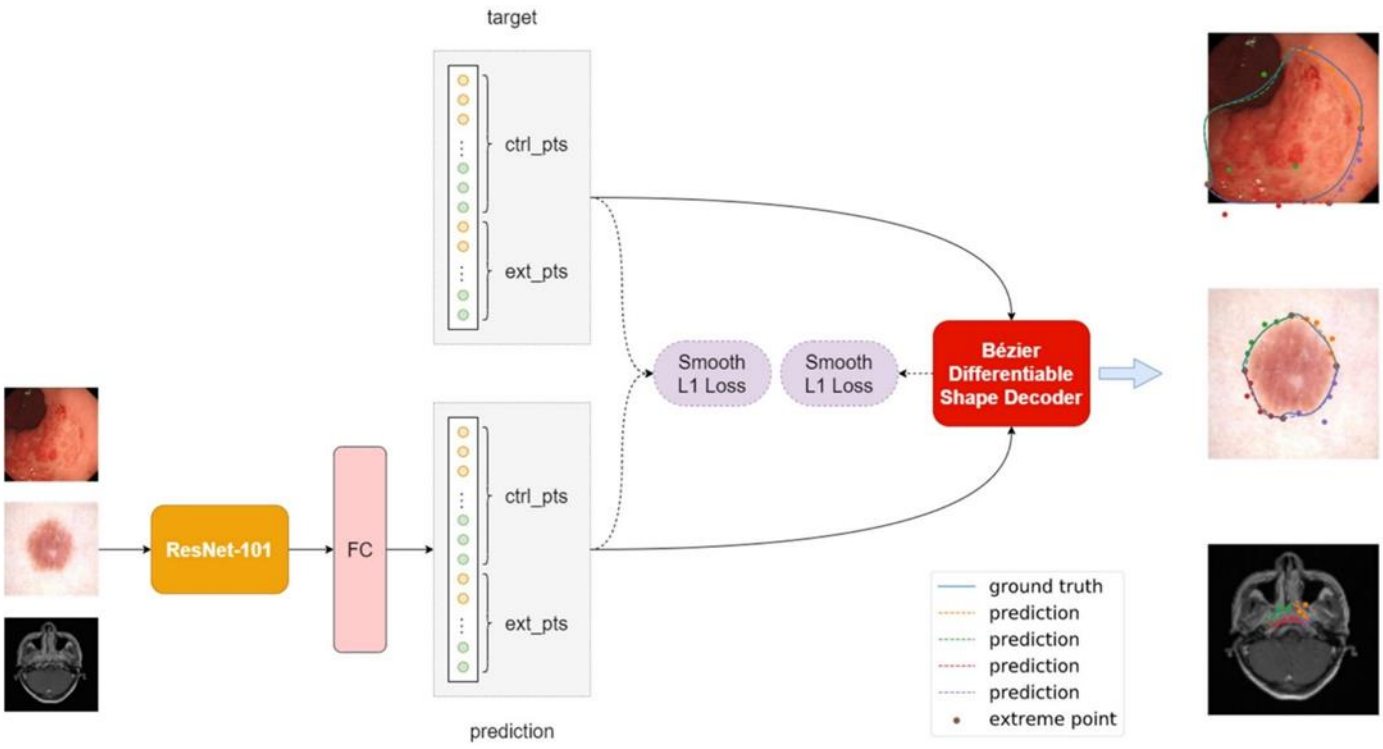
Οι Bézier καμπύλες έχουν ευρεία χρήση σε πολλούς τομείς, συμπεριλαμβανομένων:

- **Σχεδίαση Γραφικών:** Χρησιμοποιούνται για τη δημιουργία και την επεξεργασία γραφικών σχεδίων και σχεδιασμό.
- **Επεξεργασία Εικόνων:** Εφαρμόζονται στην επεξεργασία και τη μετατροπή εικόνων.
- **Καμπυλοτεχνία:** Χρησιμοποιούνται στην κατασκευή και τον χειρισμό των 3D μοντέλων.
- **Σχεδίαση Επιφανειών:** Επιτρέπουν την αναπαράσταση και την αλλαγή του σχήματος των επιφανειών σε προγράμματα σχεδίασης και προτυποποίησης.

Κεφάλαιο 2: Ιατρικό Μοντέλο

2.1 Μοντέλο αρχιτεκτονικής

Όπως φαίνεται στο Σχήμα 4, υιοθετούμε το ResNet-101 [21] ως τη ραχοκοκαλιά του μοντέλου μας. Δεδομένου ότι το μοντέλο μας δεν περιέχει στρώματα αναβάθμισης, αφαιρούμε απλώς το στρώμα ενεργοποίησης softmax που χρησιμοποιείται συνήθως σε εργασίες ταξινόμησης [22-24] και προσθέτουμε ένα επιπλέον πλήρως συνδεδεμένο στρώμα για να αντιστοιχίσουμε τα χαρακτηριστικά που εξάγονται από το δίκτυο κορμού στις προβλέψεις συντεταγμένων. Ο αριθμός των κόμβων εξόδου στο τελευταίο πλήρως συνδεδεμένο στρώμα αντιστοιχεί στους στόχους παλινδρόμησης. Για παράδειγμα, εάν απαιτούνται n_e ακραία σημεία και n_c σημεία ελέγχου, η έξοδος του θα ήταν το τελευταίο πλήρως συνδεδεμένο στρώμα πρέπει να αποτελείται από (ne + nc) × 2 κόμβους, χωρίς στρώματα ενεργοποίησης. Κατά τη διάρκεια της εκπαίδευσης, ο αποκωδικοποιητής διαφοροποιήσιμου σχήματος Bézier λαμβάνει το προβλεπόμενο σημείο ελέγχου συντεταγμένες και τις συντεταγμένες του σημείου ελέγχου του εδάφους ως είσοδο και εξάγει δύο σύνολα δειγματοληπτικών συντεταγμένων σημείων που ελέγχονται από τις δύο εισόδους του. Η απώλεια ανακατασκευής χτίζεται σε αυτά τα δύο σύνολα δειγματοληπτικών συντεταγμένων σημείου- η ελαχιστοποίηση αυτού του όρου απώλειας βοηθά έμμεσα στην εκμάθηση των συντεταγμένων σημείου ελέγχου. Σημειώστε ότι η ιδέα της παρούσας εργασίας μπορεί εύκολα να εφαρμοστεί στα κύρια πλαίσια ανίχνευσης αντικειμένων, τα οποία παρέχουν μια εναλλακτική λύση για την κατάτμηση παραδείγματος.



Σχήμα 4. Αρχιτεκτονική μοντέλου ext_pts και ctrl_pts είναι οι συντεταγμένες για τα neextreme σημεία και τα nc σημεία ελέγχου. Τα χρωματιστά σημεία που απεικονίζονται στις εικόνες είναι τα σημεία ελέγχου κάθε αντίστοιχου τμήματος καμπύλης. Τα χαρακτηριστικά της εικόνας που εξάγονται από το ResNet-101 διαβιβάζονται στο πλήρως συνδεδεμένο FC στρώμα για να προκύψουν (ne + nc) × 2 έξοδοι. Έτσι, οι προβλεπόμενες συντεταγμένες των ext_pts και ctrl_pts παλινδρομούν προς τους στόχους ext_pts και ctrl_pts υπό επίβλεψη με ομαλή απώλεια L1, που συμβολίζεται ως Lce. Η μονάδα αποκωδικοποιητή διαφορικού σχήματος Bézier ανακατασκευάζει N προβλεπόμενα οριακά σημεία και N οριακά σημεία βασικής αλήθειας σημεία. Lmatching είναι η ομαλή απόσταση L1

2.2 Δημιουργία ετικέτας εδάφους αλήθειας

Δεδομένης μιας μάσκας ενός αντικειμένου, εξάγουμε πρώτα το όριο του αντικειμένου από τη μάσκα. Στη συνέχεια, βρίσκουμε τέσσερα ακραία σημεία του αντικειμένου και τα χρησιμοποιούμε για να χωρίσουμε ολόκληρο το σχήμα σε τέσσερα μέρη. Εάν υπάρχουν πολλαπλά ακραία σημεία, όπως δύο ή περισσότερα κορυφαία σημεία, χρησιμοποιούμε το σημείο της αριστερής πάνω γωνίας, το οποίο είναι το ίδιο με το κάτω αριστερό, το κάτω δεξιό και το πάνω δεξιό για το πιο αριστερό, το κάτω και το πιο δεξιό ακραίο σημείο, αντίστοιχα. Για κάθε τμήμα του ορίου του αντικειμένου, το προσαρμόζουμε με μια καμπύλη Bézier πέμπτης τάξης. Το πρώτο και το τελευταίο σημείο ελέγχου της καμπύλης Bézier ορίζονται ως το σημείο αρχής και το σημείο τέλους του εν λόγω κομματιού του σχήματος, τα οποία είναι επίσης δύο διαδοχικά ακραία σημεία. Τα τέσσερα ενδιάμεσα σημεία ελέγχου της καμπύλης Bézier παραμένουν άγνωστα. Μετατρέπουμε την εξίσωση (2) σε μορφή πίνακα και τοποθετούμε τις συντεταγμένες του οριακού τμήματος του αντικειμένου στην εξίσωση, όπως φαίνεται στην εξίσωση (3):

$$\begin{bmatrix} C_1^1, & C_1^2, & C_1^3, & \cdots, & C_1^n \\ C_2^1, & C_2^2, & C_2^3, & \cdots, & C_1^n \\ C_3^1, & C_3^2, & C_3^3, & \cdots, & C_3^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_m^1, & C_m^2, & C_m^3, & \cdots, & C_m^n \end{bmatrix} \begin{bmatrix} x_1, & y_1 \\ x_2, & y_2 \\ x_3, & y_3 \\ \vdots & \vdots \\ x_n, & y_n \end{bmatrix} = \begin{bmatrix} \hat{x}_1, & \hat{y}_1 \\ \hat{x}_2, & \hat{y}_2 \\ \hat{x}_3, & \hat{y}_3 \\ \vdots & \vdots \\ \hat{x}_m, & \hat{y}_m \end{bmatrix} \quad (3)$$

στην οποία το m αντιπροσωπεύει το συνολικό αριθμό των σημείων του οριακού τμήματος του αντικειμένου, το n αντιπροσωπεύει το σύνολο των σημείων ελέγχου της καμπύλης Bézier.

$(\hat{x}_i, \hat{y}_i)$ είναι το i-οστό σημείο του οριακού τμήματος του αντικειμένου.

$$C_i^j = \binom{n}{j}(1-t_i)^{n-j}t_i^j, t_i \in [0,1]$$

Εμπνευσμένοι από τον αλγόριθμο του De Casteljau για την κατασκευή μιας καμπύλης Bézier, υποθέτουμε ότι

$t_i = (i-1)/(m-1)$ για το i-οστό σημείο του οριακού τμήματος του αντικειμένου. Δεδομένου ότι η πρώτη και η τελευταία σειρά της εξίσωσης (3) είναι εκφράσεις ταυτότητας

που αντιπροσωπεύουν τα ακραία σημεία και η πρώτη και η τελευταία στήλη είναι σταθερές τιμές, αφαιρούμε την πρώτη και την τελευταία γραμμή από την εξίσωση (3) και αφαιρούμε τις σταθερές τιμές και από τις δύο πλευρές της εξίσωσης (3). Για να απλουστεύσουμε τη συμβολική απόδοση, έστω τα εξής:

$$\mathbf{A} = \begin{bmatrix} C_2^2, & C_2^3, & C_2^4, & \cdots, & C_2^{n-1} \\ C_3^2, & C_3^3, & C_3^4, & \cdots, & C_3^{n-1} \\ C_4^2, & C_4^3, & C_4^4, & \cdots, & C_4^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{m-1}^2, & C_{m-1}^3, & C_{m-1}^4, & \cdots, & C_{m-1}^{n-1} \end{bmatrix},$$

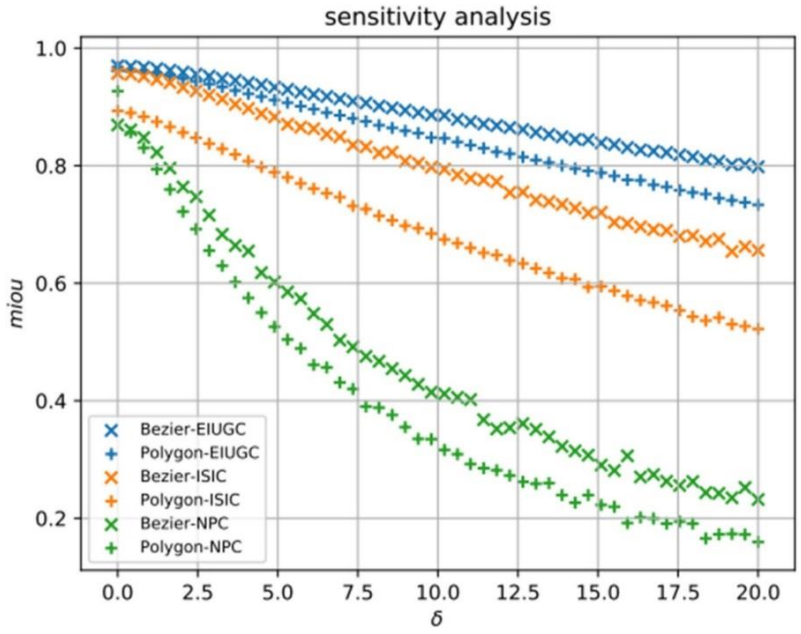
$$\mathbf{b} = \begin{bmatrix} \widehat{x_2} - x_1 C_2^1 - x_n C_2^n, & \widehat{y_2} - y_1 C_2^1 - y_n C_2^n \\ \widehat{x_3} - x_1 C_3^1 - x_n C_3^n, & \widehat{y_3} - y_1 C_3^1 - y_n C_3^n \\ \widehat{x_4} - x_1 C_4^1 - x_n C_4^n, & \widehat{y_4} - y_1 C_4^1 - y_n C_4^n \\ & \vdots \\ \vdots & \vdots \\ x_{m-1} - x_1 C_{m-1}^1 - x_n C_{m-1}^n, & y_{m-1} - y_1 C_{m-1}^1 - y_n C_{m-1}^n \end{bmatrix},$$

$$\mathbf{c} = \begin{bmatrix} x_2, & y_2 \\ x_3, & y_3 \\ x_4, & y_4 \\ \vdots & \vdots \\ x_{n-1}, & y_{n-1} \end{bmatrix}$$

Έτσι, Ac = b. Υπολογίζοντας το ψευδοαντίστροφο του A και πολλαπλασιάζοντάς το με τη δεξιά πλευρά της εξίσωσης, μπορούμε να λάβουμε τις συντεταγμένες των ενδιάμεσων σημείων ελέγχου $\mathbf{c} = \mathbf{A}^{-1}\mathbf{b}$. Τέλος, συνδέουμε τις συντεταγμένες των 4 ακραίων σημείων και των 16 ενδιάμεσων σημεία ελέγχου ως στόχο της παλινδρόμησης

$(n_e = 4,\ n_c = 16)$, με αποτέλεσμα ένα 40διάστατο διάνυσμα για κάθε εικόνα εισόδου. Παρόμοια με την ESE-Seg, διεξάγουμε επίσης μια ανάλυση ευαισθησίας για την αναπαράσταση της καμπύλης Bézier. Συγκεκριμένα, λαμβάνουμε τυχαία δείγματα κάποιων θορύβων από το N(0, δ) και τους προσθέτουμε στα σημεία ελέγχου και στα ακραία σημεία για να μιμηθούμε τη συμπεριφορά αβεβαιότητας του νευρωνικού δικτύου με συνελικτικό σύστημα. Συγκρίνουμε την αναπαράσταση της καμπύλης Bézier με την αναπαράσταση σχήματος πολυγώνου. Για να διατηρήσουμε την ίδια πολυπλοκότητα και για τις δύο αναπαραστάσεις σχήματος, επιλέγουμε ομοιόμορφα 20 σημεία κατά μήκος του ορίου του

αντικειμένου για την αναπαράσταση πολυγώνου. Το Σχήμα 5 δείχνει την ανθεκτικότητα της αναπαράστασης της καμπύλης Bézier. Όπως φαίνεται στο Σχήμα 5, η αναπαράσταση καμπύλης Bézier επιτυγχάνει πάντα υψηλότερο μέσο όρο διασταύρωσης-υπέρ-ένωσης (MIOU) από την αναπαράσταση πολυγώνου με τον ίδιο αριθμό σημείων. Μπορούμε να συμπεράνουμε ότι η αναπαράσταση καμπύλης Bézier είναι ανώτερη από την αναπαράσταση πολυγώνου, επειδή μας επιτρέπει να περιγράψουμε μια καμπύλη με μεγαλύτερη ακρίβεια. Η αναπαράσταση της καμπύλης Bézier είναι ισχυρή όταν χρησιμοποιείται στο σύνολο δεδομένων της ενδοσκόπησης εικόνων ανώτερου γαστρεντερικού καρκίνου (EIUGC) και στο σύνολο δεδομένων της διεθνούς συνεργασίας για την απεικόνιση δερματικών αλλοιώσεων (ISIC), παραμένοντας πάνω από 0,6 MIOU ακόμη και όταν εγχέεται θόρυβος δ = 20. Για την ίδια διαταραχή, το MIOU του συνόλου δεδομένων της μαγνητικής τομογραφίας ρινοφαρυγγικού καρκινώματος (NPCMRI) μειώθηκε ραγδαία, κυρίως λόγω του μικρότερου μέσου μεγέθους ROI σε σύγκριση με τα άλλα δύο σύνολα δεδομένων.



Σχήμα 5. Ανάλυση ευαισθησίας. Το αποτέλεσμα της ανάλυσης έδειξε ότι η αναπαράσταση της καμπύλης Bézier είναι πιο ανθεκτική από την αναπαράσταση πολυγώνου σε διαφορετικά επίπεδα θορύβου.

2.3. Διαφοροποιήσιμος αποκωδικοποιητής σχήματος Bézier

Παρόμοια με το FourierNet [25], επινοούμε έναν νέο αποκωδικοποιητή διαφοροποιήσιμου σχήματος που ονομάζεται αποκωδικοποιητής διαφοροποιήσιμου σχήματος Bézier, συντομογραφία BDSD. Το FourierNet χρησιμοποιεί τον αντίστροφο γρήγορο μετασχηματισμό Fourier ως αποκωδικοποιητή σχήματος για να μετατρέψει τους συντελεστές της σειράς Fourier σε σημεία περιγράμματος, ενώ εμείς εφαρμόζουμε την παραμετρική εξίσωση των καμπυλών Bézier για να αντιστοιχίσουμε τα σημεία ελέγχου σε σημεία περιγράμματος. Ακολουθώντας την εξίσωση (2), παίρνουμε τυχαία δείγματα N ts από το U[0, 1] και χρησιμοποιούμε τα ίδια ts για να ανακατασκευάσουμε N σημεία ορίου της βασικής αλήθειας και τα αντίστοιχα προβλεπόμενα σημεία ορίου. Σε όλα τα πειράματα, θέτουμε N = 72. Η μονάδα BDSD είναι πλήρως διαφοροποιήσιμη και επιτρέπει την αντίστροφη διάδοση της κλίσης μέσω του δικτύου, παρέχοντας έτσι πρόσθετη εποπτεία κατά τη φάση της εκπαίδευσης.

2.5. Απώλεια

Η συνολική συνάρτηση απωλειών για την εκπαίδευση του μοντέλου μας περιέχει δύο επιμέρους όρους απωλειών. Ο πρώτος, Lce , είναι η ομαλή απώλεια L1 για την παλινδρόμηση των σημείων ελέγχου και των ακραίων σημείων και ο δ ε ύ τ ε ρ ο ς, Lmatching, είναι μια άλλη ομαλή απώλεια L1 για την εκμάθηση της αντιστοίχισης σημείων,

η οποία μετρά τις διαφορές μεταξύ των εξόδων της μονάδας BDSD. Το Lce σταθμίζει εξίσου κάθε σημείο ελέγχου και το Lmatching αποδίδει σιωπηρά διαφορετικά βάρη στα σημεία ελέγχου.

σημεία. Η εξίσωση (4) δείχνει τη συνολική συνάρτηση απωλειών:

$$L = \lambda_{ce} L_{ce} + \lambda_{matching} L_{matching} \quad (4)$$

όπου λce και λmatching είναι υπερπαράμετροι εξισορρόπησης- θέτουμε και τις δύο ίσες με

1 σε όλα τα πειράματα.

Κεφάλαιο 3: Εφαρμογές των Bézier Καμπυλών:

3.1 Αλγόριθμοι κατάτμησης

Τα μοντέλα σημασιολογικής κατάτμησης ανά εικονοστοιχείο συνήθως βασίζονται σε πλήρως συνελικτικά δίκτυα (Fully Convolutional Networks - FCN). Αυτά τα μοντέλα έχουν επαναπροσδιορίσει τον τρόπο που εφαρμόζονται τα συνελικτικά δίκτυα για να αντιμετωπίσουν το πρόβλημα της σημασιολογικής κατάτμησης.

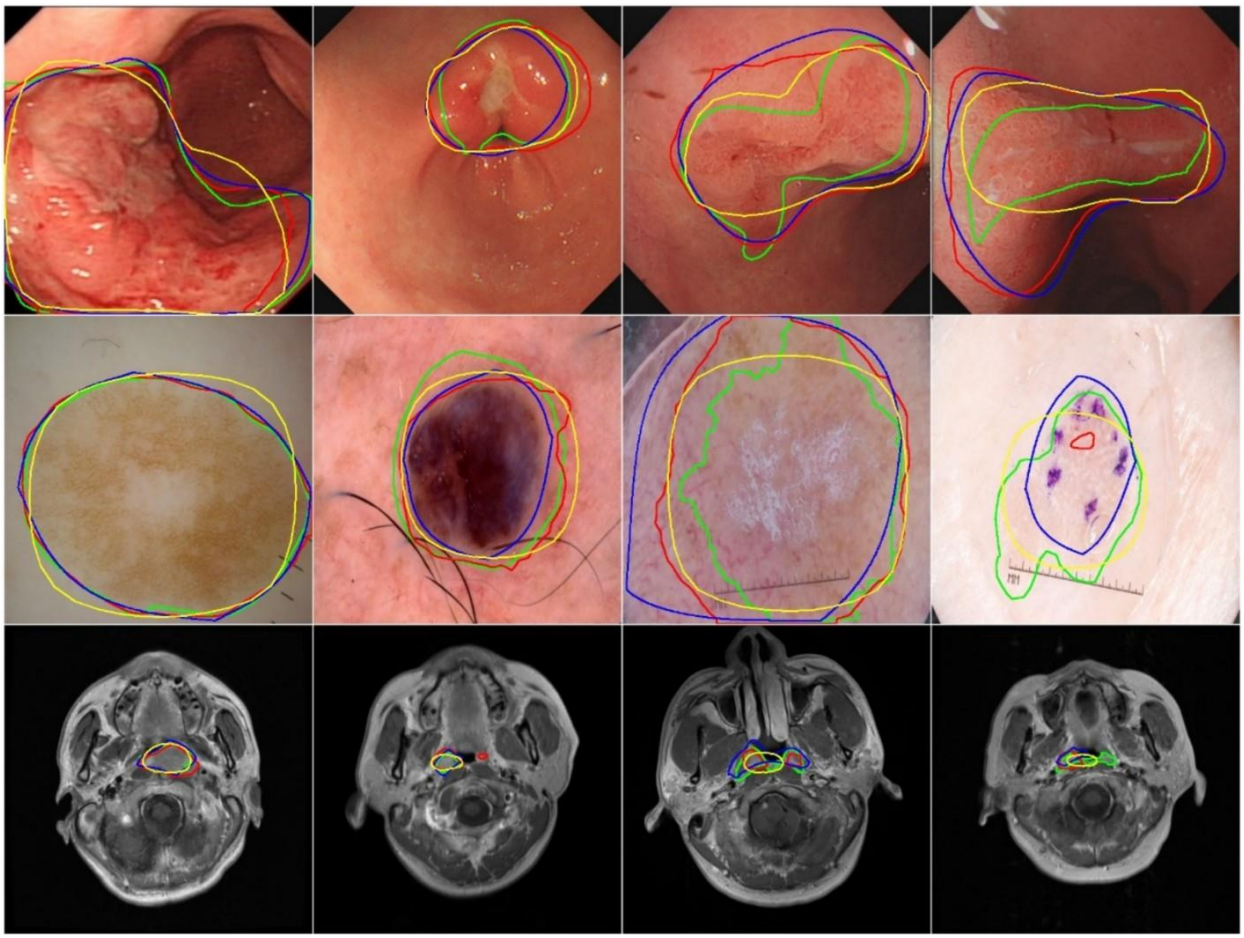
Ορισμένα από τα δημοφιλή μοντέλα σημασιολογικής κατάτμησης ανά εικονοστοιχείο που αναφέρατε είναι:

1. **U-Net:** Το U-Net είναι ένα αρχικό μοντέλο που χρησιμοποιείται για σημασιολογική κατάτμηση και έχει αρχιτεκτονική που περιλαμβάνει συνεκτικά επίπεδα για την αναγνώριση χαρακτηριστικών και αποσυνθέτει την εικόνα για τη δημιουργία του αποτελέσματος.
2. **PSPNet (Pyramid Scene Parsing Network):** Το PSPNet επεκτείνει τον χώρο προσαρμογής των συνελικτικών δικτύων χρησιμοποιώντας πυραμιδική δομή για την αναγνώριση σημασιολογικών χαρακτηριστικών σε διάφορες κλίμακες.
3. **BiSeNet (Bilateral Segmentation Network):** Το BiSeNet είναι μια αρχιτεκτονική που επικεντρώνεται στην επιτάχυνση της σημασιολογικής κατάτμησης και ενσωματώνει μηχανισμούς bilinear up-sampling για την αύξηση της ανάλυσης των αποτελεσμάτων.
4. **DeepLab v3+:** Το DeepLab v3+ είναι μια εξέλιξη του DeepLab μοντέλου και χρησιμοποιεί μια συνδυασμένη αρχιτεκτονική με επιπλέον χαρακτηριστικά, όπως atrous (atrous convolution) συνελικτικά επίπεδα και μηχανισμούς επιμέρους πολλαπλών κλίμακων.

Αυτά τα μοντέλα είναι σχεδιασμένα για να αντιμετωπίζουν το πρόβλημα της σημασιολογικής κατάτμησης με αποτελεσματικό τρόπο και έχουν εφαρμοστεί επιτυχώς σε πολλές εφαρμογές όπως η αναγνώριση αντικειμένων σε εικόνες, η ιατρική εικόνα, και η αυτόνομη οδήγηση, μεταξύ άλλων.

3.2 Παραμετρική Αναπαράσταση Σχημάτων: Ο BézierSeg ως Εναλλακτική Λύση στο DeepLab v3+ ResNet101

Όπως φαίνεται στο Σχήμα 7, το BézierSeg μπορεί πάντα να εξάγει ένα ομαλό περίγραμμα, ενώ το DeepLab v3+ ResNet101 εξάγει ένα τραχύ περίγραμμα και συχνά παράγει πολλαπλές διαχωρισμένες περιγράμματα. Το PolarMask δίνει πολυγωνικές εξόδους που εξασφαλίζουν περίπου την ομαλότητα. Ωστόσο, είναι δύσκολο να χειριστεί το σχήμα του αλτήρα, και έτσι το μοντέλο εξάγει ένα σχήμα βότσαλου στις περισσότερες από αυτές τις περιπτώσεις. Η χρήση της παραμετρικής αναπαράστασης σχήματος επιτρέπει στο BézierSeg να προβλέπει τα περιγράμματα των αντικειμένων καλύτερα από το PolarMask, καθιστώντας το BézierSeg λιγότερο ευαίσθητο στα τοπικά τεχνουργήματα. Συνοψίζοντας, το BézierSeg μπορεί να χρησιμοποιηθεί ως εναλλακτική λύση στο DeepLab v3+ ResNet101, ενώ έχει ταχύτερη ταχύτητα που είναι παρόμοια με την PolarMask.



Σχήμα 7. Σύγκριση ποιοτικών αποτελεσμάτων μεταξύ των BézierSeg, DeepLab v3+ ResNet101 και PolarMask σε τρία σύνολα δεδομένων. Η πρώτη σειρά, η δεύτερη σειρά και η τελευταία σειρά δείχνουν τα αποτελέσματα στο σύνολο δεδομένων EIUGC, στο σύνολο δεδομένων ISIC και στο σύνολο δεδομένων NPCMRI, αντίστοιχα. Πράσινο: ετικέτα βασικής αλήθειας- κόκκινο: DeepLab v3+ ResNet101- μπλε: BézierSeg- κίτρινο: PolarMask

Πειράματα και Οπτικοποίηση:

- Διεξάγετε πειράματα χρησιμοποιώντας τις Bézier καμπύλες σε πρακτικά παραδείγματα.

Οπτικοποιήστε τα αποτελέσματα των πειραμάτων

3.3 Αυτόματη Ταξινόμηση Εικόνων με Χρήση του MobileNet V2

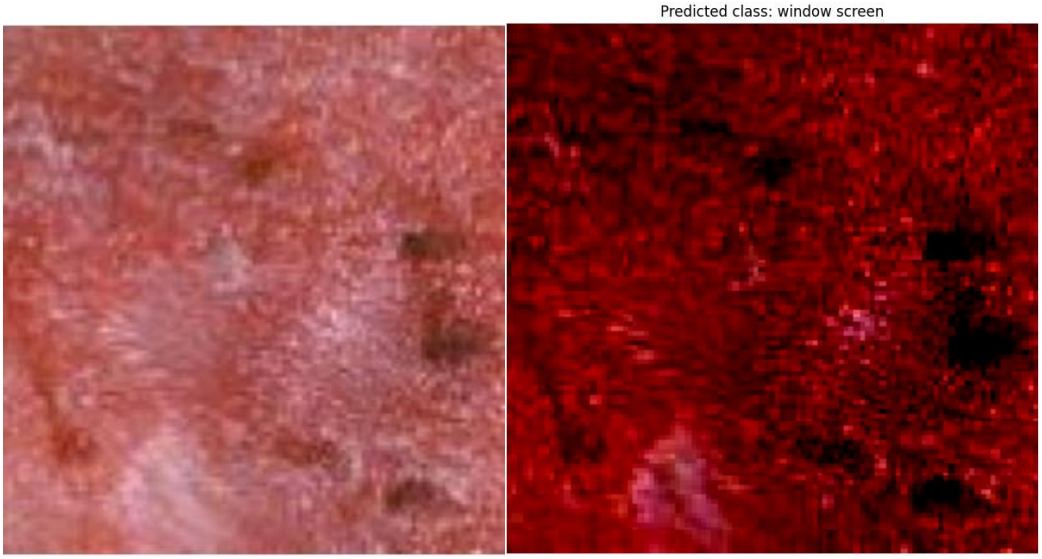
Ο παρακάτω κώδικας χρησιμοποιεί το TensorFlow και το TensorFlow Hub για να εκτελέσει τη διαδικασία ταξινόμησης μιας εικόνας χρησιμοποιώντας ένα προεκπαιδευμένο μοντέλο MobileNet V2. Αρχικά, δημιουργεί ένα σειριακό μοντέλο και ορίζει το σχήμα της εισόδου. Στη συνέχεια, φορτώνει μια εικόνα που θέλει να ταξινομήσει, εκτελεί προεπεξεργασία στην εικόνα, κάνει προβλέψεις με το MobileNet V2, αποκωδικοποιεί τα αποτελέσματα για να βρει την κατηγορία της εικόνας, και τέλος εμφανίζει την εικόνα και την προβλεπόμενη κατηγορία.

Κώδικας:

```
import tensorflow as tf
import tensorflow_hub as hub

# Create a Sequential model with the MobileNet V2 KerasLayer
m = tf.keras.Sequential([

hub.KerasLayer("https://tfhub.dev/google/imagenet/mobilenet_v2_130_224/classi
fication/5")
])

# Build the model with the desired input shape
m.build([None, 224, 224, 3])  # Batch input shape.

# Load an image that you want to classify
#image_path = "path/to/your/image.jpg"
image = tf.keras.utils.load_img(image_path, target_size=(224, 224))
image = tf.keras.utils.img_to_array(image)
image = tf.image.convert_image_dtype(image, tf.float32)
```

```
image = tf.expand_dims(image, axis=0)  # Add batch dimension

# Preprocess the image (normalize it)
image = tf.keras.applications.mobilenet_v2.preprocess_input(image)

# Make predictions
predictions = m.predict(image)

# Decode the predictions (assuming you want class labels)
imagenet_labels_path = tf.keras.utils.get_file("ImageNetLabels.txt",
"https://download.tensorflow.org/data/ImageNetLabels.txt")
imagenet_labels = []
with open(imagenet_labels_path) as f:
    imagenet_labels = f.read().splitlines()

predicted_class_index = tf.argmax(predictions, axis=1)
predicted_class_label = imagenet_labels[predicted_class_index[0]]

# Display the image and label
plt.figure(figsize=(8, 8))
plt.imshow(image[0])
plt.axis('off')
plt.title(f"Predicted class: {predicted_class_label}")
plt.show()
```

Παράδειγμα εικόνων που εμφανίζονται μετα την εκτέλεση του κώδικα:



Εικόνα 1. Παρατηρούμε εικόνα από ένα καρκίνο του δέρματος ,ο αλγόριθμους χρωματίζει με μαύρο τις προβληματικές περιοχές.

Κεφάλαιο 4: Υλοποίηση σε Python:

4.1 Δημιουργία και Οπτικοποίηση Bézier Καμπύλης σε Εικόνα με Python

Αυτός ο κώδικας εκτελεί μια σειρά εργασιών σε μια εικόνα χρησιμοποιώντας τη γλώσσα προγραμματισμού Python και διάφορες βιβλιοθήκες, σε τυχαία σημεία και χωρίς να εχει επιλεχθεί κάποια συγκεκριμένη περιοχη της εικόνας.
Αρχικά, φορτώνει τις απαραίτητες βιβλιοθήκες, όπως το PIL για την επεξεργασία εικόνων, το Matplotlib για την οπτικοποίηση, και το NumPy για την εργασία με πίνακες.

Στη συνέχεια, φορτώνει μια εικόνα από τον υπολογιστή και ορίζει μια λίστα σημείων ελέγχου στην εικόνα. Αυτά τα σημεία θα χρησιμοποιηθούν για να δημιουργηθεί μια Bézier καμπύλη.

Έπειτα, υλοποιεί τον αλγόριθμο De Casteljau για τον υπολογισμό της Bézier καμπύλης. Αυτός ο αλγόριθμος διαιρεί τα σημεία ελέγχου για να υπολογίσει τα σημεία της καμπύλης σε διάφορα σημεία.

Στη συνέχεια, ο κώδικας οπτικοποιεί τη Bézier καμπύλη πάνω στην εικόνα χρησιμοποιώντας το PIL. Σχεδιάζει τα σημεία ελέγχου ως κόκκινα σημεία και τα σημεία της Bézier καμπύλης ως μπλε σημεία.

Τέλος, εμφανίζει την εικόνα με τη Bézier καμπύλη χρησιμοποιώντας το Matplotlib και προσθέτει έναν τίτλο στο παράθυρο που εμφανίζει την εικόνα. Έτσι, ο κώδικας αυτός εκτελεί μια πλήρη διαδικασία για τη δημιουργία και οπτικοποίηση μιας Bézier καμπύλης πάνω σε μια εικόνα.

Κώδικας:

```
from PIL import Image, ImageDraw
import matplotlib
matplotlib.use('TkAgg')  # ή 'Qt5Agg' ανάλογα με το ποιο backend λειτουργεί σωστά στο σύστημά σας
import matplotlib.pyplot as plt
import numpy as np
import ssl
ssl._create_default_https_context = ssl._create_unverified_context

# Φόρτωση της εικόνας
image_path = r'C:\Users\Vagelis\Desktop\pythonProject\cancer1.jpg'
image = Image.open(image_path)

# Επιλογή σημείων ελέγχου (σημεία στην εικόνα)
control_points = [(100, 50), (200, 100), (300, 200), (400, 150)]
```

```
# Υλοποίηση της Bézier καμπύλης
def de_casteljau(control_points, t):
    if len(control_points) == 1:
        return control_points[0]

    new_points = []
    for i in range(len(control_points) - 1):
        new_point = tuple(
            (1 - t) * control_points[i][j] + t * control_points[i + 1][j] for
j in range(len(control_points[i])))
        new_points.append(new_point)

    return de_casteljau(new_points, t)

t_values = np.linspace(0, 1, 100)
curve_points = [de_casteljau(control_points, t) for t in t_values]

# Οπτικοποίηση της Bézier καμπύλης πάνω στην εικόνα
draw = ImageDraw.Draw(image)
draw.line(control_points, fill='red', width=2)
for point in control_points:
    draw.ellipse((point[0] - 3, point[1] - 3, point[0] + 3, point[1] + 3),
fill='red')

curve_x, curve_y = zip(*curve_points)
for x, y in zip(curve_x, curve_y):
    draw.point((x, y), fill='blue')

# Εμφάνιση της εικόνας με την Bézier καμπύλη σε ένα παράθυρο
fig1 = plt.figure()
fig1.canvas.manager.window.wm_geometry("+0+0")  # Set the window position to
display on the primary screen

plt.imshow(np.array(image))
plt.title('Καμπύλη Bézier στην Εικόνα')

plt.show()
```

Παράδειγμα εικόνων που εμφανίζονται μετά την εκτέλεση του κώδικα:

Εικόνα 2. Παρατηρούμε εικόνα από ένα καρκίνο του δέρματος ,ο αλγόριθμους εμφανίζει την εικόνα με τη Bézier καμπύλη

4.2 Ανάπτυξη Εφαρμογής Επεξεργασίας και Αποθήκευσης Εικόνων με Χρήση του Python και της Βιβλιοθήκης Tkinter

Ο παρακάτω κώδικας είναι ένα πρόγραμμα γραμμένο σε Python που χρησιμοποιεί την βιβλιοθήκη tkinter για τη δημιουργία ενός γραφικού περιβάλλοντος χρήστη (GUI) και την βιβλιοθήκη matplotlib για την εμφάνιση εικόνων και τη δυνατότητα επιλογής περιοχών σε αυτές τις εικόνες.

Στην αρχή, εισάγονται οι απαιτούμενες βιβλιοθήκες, όπως tkinter, matplotlib, PIL (Python Imaging Library) και numpy. Στη συνέχεια, δημιουργείται μια μεταβλητή save_counter για την αρίθμηση των αποθηκευμένων αρχείων.

Έπειτα, δημιουργείται μια συνάρτηση onselect που χειρίζεται την επιλογή περιοχής στην εικόνα και αποθηκεύει την επιλεγμένη περιοχή ως εικόνα. Η εικόνα αυτή εμφανίζεται ως εικονίδιο σε ένα δεύτερο subplot.

Στη συνέχεια, φορτώνεται μια αρχική εικόνα χρησιμοποιώντας την PIL. Δημιουργείται το κύριο παράθυρο της εφαρμογής GUI με τη χρήση της tkinter και δημιουργείται ένα Matplotlib figure και axis για την εμφάνιση της αρχικής εικόνας.

Ένα δεύτερο subplot δημιουργείται για την εμφάνιση της επιλεγμένης περιοχής (ικανότητα cropping), και δημιουργείται ένα RectangleSelector που επιτρέπει την επιλογή περιοχών στην αρχική εικόνα.

Τέλος, ξεκινά ο κύριος βρόχος της tkinter για την εκτέλεση της εφαρμογής. Κατά τη διάρκεια της εκτέλεσης, ο χρήστης μπορεί να επιλέξει μια περιοχή στην αρχική εικόνα, και αυτή η περιοχή θα εμφανίζεται ως εικονίδιο στο δεύτερο subplot. Η περιοχή αυτή αποθηκεύεται και ως ανεξάρτητή εικόνα στον υπολογιστή με αύξοντα αριθμό στο όνομα του αρχείου. Φορτώνεται η εικόνα που έχει αποθηκευτεί ως cropped_image_1.jpg στον υπολογιστή. Αυτός ο κώδικας δημιουργεί μια εφαρμογή GUI που επιτρέπει στον χρήστη να επιλέγει και να αποθηκεύει επιλεγμένες περιοχές από μια εικόνα.

Κώδικας:

```
import tkinter as tk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
from matplotlib.widgets import RectangleSelector
from PIL import Image
import numpy as np

# Initialize a counter for naming the saved files
save_counter = 0

# Function to handle the rectangle selection and save the cropped area
def onselect(eclick, erelease):
    global save_counter
    x1, y1 = int(eclick.xdata), int(eclick.ydata)
    x2, y2 = int(erelease.xdata), int(erelease.ydata)

    # Convert the PIL image to a NumPy array
    im_np = np.array(im)

    # Crop the selected area
```

```
    im_crop = im_np[y1:y2, x1:x2]

    # Display the cropped area as an icon
    ax_crop.clear()
    ax_crop.imshow(im_crop)
    ax_crop.axis('off')
    canvas.draw()

    # Save the cropped area as an image file
    save_counter += 1
    save_path = f"cropped_image_{save_counter}.jpg"
    cropped_image = Image.fromarray(im_crop)
    cropped_image.save(save_path)
    print(f"Saved as {save_path}")

# Load the initial image
initial_image_path = r'C:\Users\Vagelis\Desktop\pythonProject\cancer1.jpg'
im = Image.open(initial_image_path)

# Create the main GUI window using tkinter
root = tk.Tk()
root.title("Image Cropping Tool")

# Create a Matplotlib figure and axis for the initial image
fig = Figure(figsize=(8, 6))
ax = fig.add_subplot(121)
ax.imshow(im)
ax.axis('off')

# Create a subplot for the cropped area (icon)
ax_crop = fig.add_subplot(122)  # Adjust the position and size as needed
ax_crop.axis('off')

# Create a RectangleSelector to select the area
rs = RectangleSelector(ax, onselect)

# Create a Matplotlib canvas for embedding the plot in the tkinter window
canvas = FigureCanvasTkAgg(fig, master=root)
canvas_widget = canvas.get_tk_widget()
canvas_widget.pack()

# Start the tkinter main loop
root.mainloop()

# Φόρτωση της εικόνας
image_path = r'C:\Users\Vagelis\Desktop\pythonProject\cropped_image_1.jpg'
image = Image.open(image_path)
```

Παράδειγμα εικόνων που εμφανίζονται μετά την εκτέλεση του κώδικα:

Εικόνα . Παρατηρούμε εικόνα από ένα καρκίνο του δέρματος ,αυτός ο κώδικας δημιουργεί μια εφαρμογή GUI που επιτρέπει στον χρήστη να επιλέγει και να αποθηκεύει επιλεγμένες περιοχές από μια εικόνα.

4.3 Δημιουργία και Οπτικοποίηση Bézier Καμπύλης από Σημεία Ελέγχου σε Εικόνα

Ο παρακάτω κώδικας εκτελεί δύο βασικές λειτουργίες: την επιλογή σημείων ελέγχου σε μια εικόνα και την υλοποίηση και οπτικοποίηση μιας καμπύλης Bézier που προκύπτει από αυτά τα σημεία. Ας αναλύσουμε τον κώδικα σε παράγραφο:

Αρχικά, στον κώδικα ορίζονται τα σημεία ελέγχου (control points) στη μεταβλητή control_points. Αυτά τα σημεία καθορίζουν το σχήμα της καμπύλης Bézier. Στο παράδειγμα αυτό, έχουμε τέσσερα σημεία ελέγχου.

Στη συνέχεια, χρησιμοποιώντας τη συνάρτηση de_casteljau, υλοποιείται ο αλγόριθμος De Casteljau για τον υπολογισμό των σημείων της Bézier καμπύλης. Αυτή η συνάρτηση λαμβάνει

ως είσοδο τα σημεία ελέγχου και ένα παράμετρο t, και επιστρέφει το αντίστοιχο σημείο στην καμπύλη για την συγκεκριμένη τιμή του t.

Έπειτα, υπολογίζονται τα σημεία της Bézier καμπύλης για διάφορες τιμές του t, και τα αποθηκεύουμε στη λίστα curve_points. Στο επόμενο τμήμα του κώδικα, η Bézier καμπύλη οπτικοποιείται πάνω στην εικόνα χρησιμοποιώντας τη βιβλιοθήκη PIL. Τα σημεία ελέγχου απεικονίζονται σε κόκκινο και η Bézier καμπύλη σχεδιάζεται σε μπλε χρώμα. Επιπλέον, εμφανίζονται κύκλοι γύρω από τα σημεία ελέγχου για να τα διακρίνουμε εύκολα.

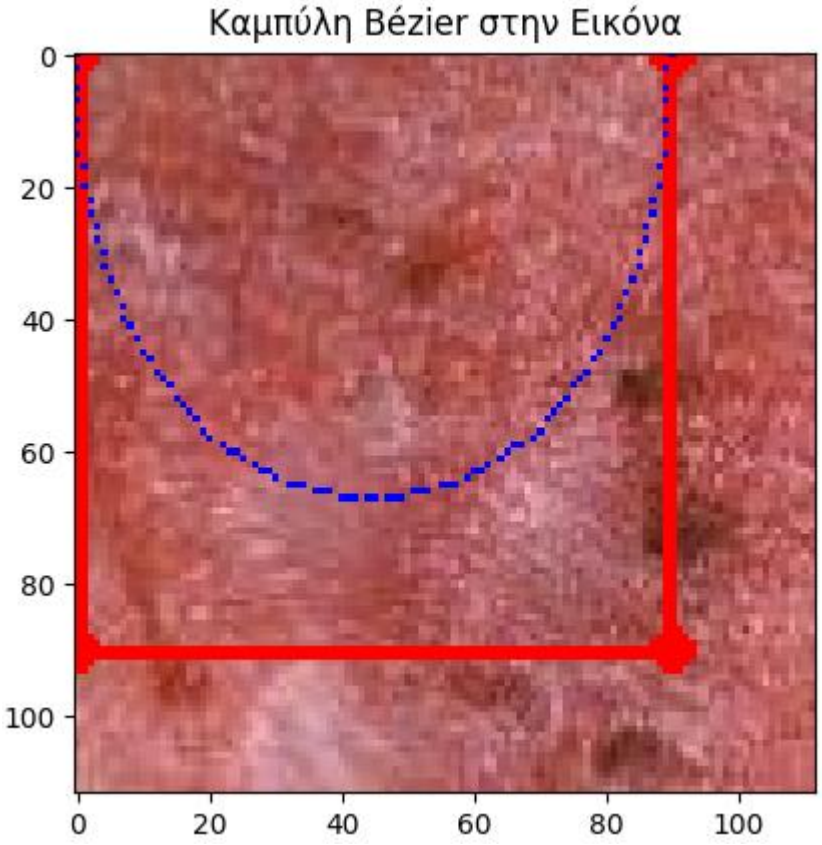
Τέλος, η εικόνα που περιλαμβάνει την Bézier καμπύλη εμφανίζεται σε ένα παράθυρο χρησιμοποιώντας τη βιβλιοθήκη Matplotlib. Το παράθυρο τοποθετείται στην αρχή της οθόνης για ορατότητα. Με αυτόν τον τρόπο, ο χρήστης μπορεί να ορίσει τα σημεία ελέγχου και να δει το αποτέλεσμα της Bézier καμπύλης στην εικόνα.

Κώδικας:

```
# Επιλογή σημείων ελέγχου (σημεία στην εικόνα)
control_points = [(0, 0), (0, 90), (90, 90), (90, 0)]

# Υλοποίηση της Bézier καμπύλης
def de_casteljau(control_points, t):
    if len(control_points) == 1:
        return control_points[0]

    new_points = []
    for i in range(len(control_points) - 1):
        new_point = tuple(
            (1 - t) * control_points[i][j] + t * control_points[i + 1][j] for
j in range(len(control_points[i])))
        new_points.append(new_point)

    return de_casteljau(new_points, t)

t_values = np.linspace(0, 1, 100)
curve_points = [de_casteljau(control_points, t) for t in t_values]

# Οπτικοποίηση της Bézier καμπύλης πάνω στην εικόνα
draw = ImageDraw.Draw(image)
draw.line(control_points, fill='red', width=2)
for point in control_points:
    draw.ellipse((point[0] - 3, point[1] - 3, point[0] + 3, point[1] + 3),
fill='red')

curve_x, curve_y = zip(*curve_points)
for x, y in zip(curve_x, curve_y):
    draw.point((x, y), fill='blue')

# Εμφάνιση της εικόνας με την Bézier καμπύλη σε ένα παράθυρο
```

```
fig1 = plt.figure()
fig1.canvas.manager.window.wm_geometry("+0+0")  # Set the window position to
display on the primary screen

plt.imshow(np.array(image))
plt.title('Καμπύλη Bézier στην Εικόνα')

plt.show()
```

Παράδειγμα εικόνων που εμφανίζονται μετά την εκτέλεση του κώδικα:

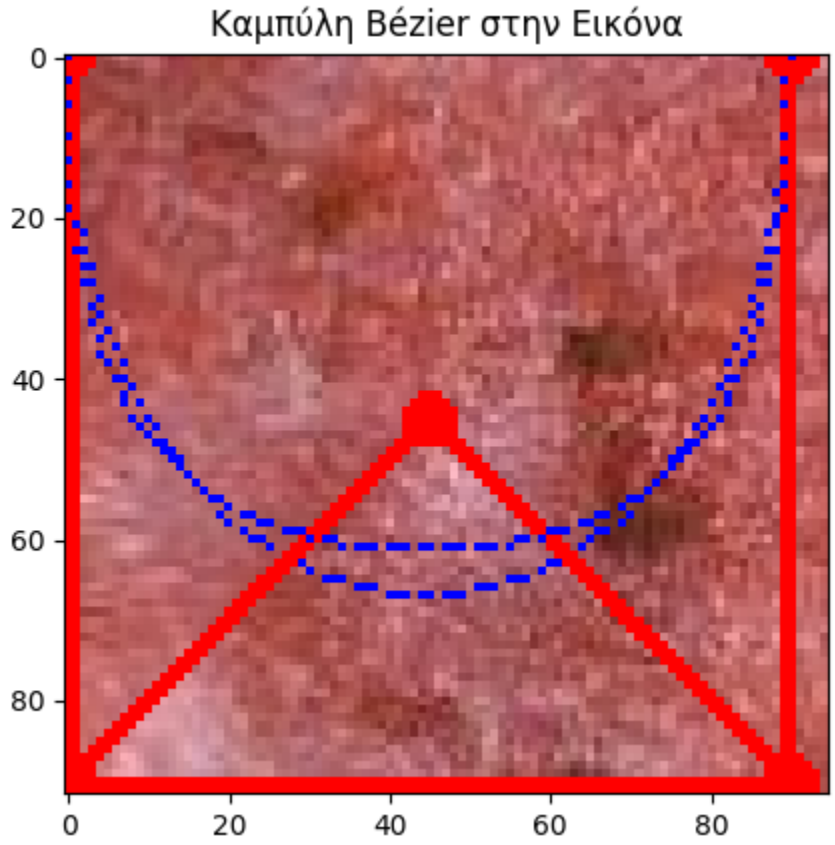


Εικόνα 3 . Η εικόνα που περιλαμβάνει την Bézier καμπύλη εμφανίζεται σε ένα παράθυρο χρησιμοποιώντας τη βιβλιοθήκη Matplotlib.

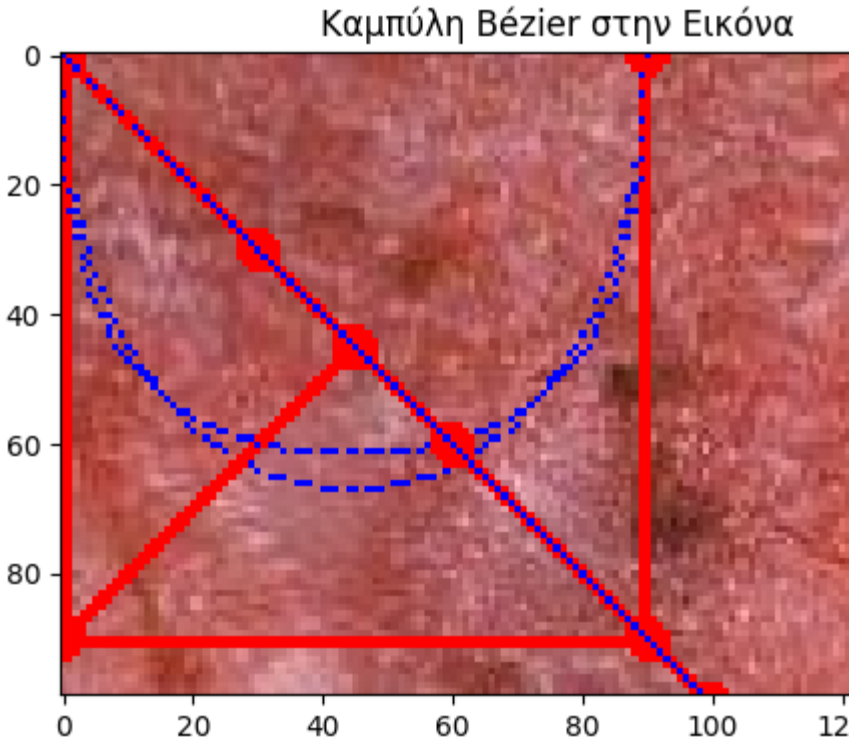
4.4 Παραδείγματα με περισσότερα σημεία

Στις παρακάτω εικόνες μπορούμε να παρατηρήσουμε την συμπεριφορά των καμπυλών σε περισσότερα σημεία.

5 Σημεία:



6 Σημεία:



Κεφάλαιο 5:Πηγές και Βιβλιογραφία:

Paper "BézierSeg: Parametric Shape Representation for Fast Object Segmentation in Medical Images":

Authors: Not specified

Title: BézierSeg: Parametric Shape Representation for Fast Object Segmentation in Medical Images

Journal: Life

Volume: 13

Issue: 3

Pages: Not specified

Date: March 2023

Publisher: MDPI

URL: https://www.mdpi.com/2075-1729/13/3
https://www.semanticscholar.org/paper