Εφαρμοσμένα Πληροφοριακά Συστήματα

12

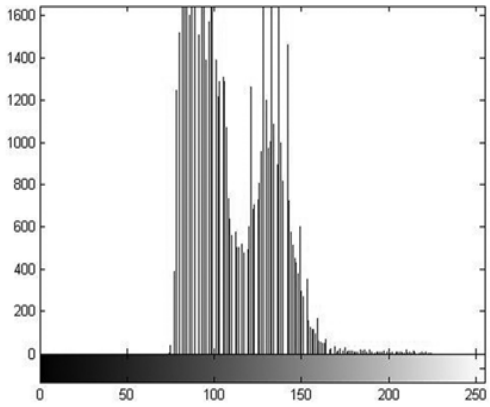
Επεξεργασία Εικόνας

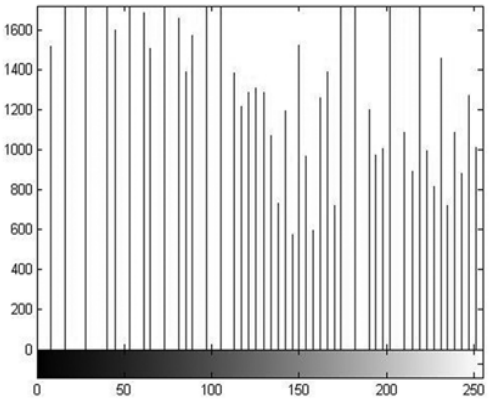
Ιωάννης Έλληνας

Επεξεργασία Εικόνας

- **Δυαδικές εικόνες**

- **Εξισορρόπηση ιστογράμματος**

- **Φιλτράρισμα**

Απομάκρυνση του θορύβου, ανίχνευση ακμών (φίλτρα πρώτης παραγώγου), ανάδειξη ακμών (φίλτρα δεύτερης παραγώγου), εξομάλυνση, κλπ. Ουσιαστικά, ένα φίλτρο υλοποιείται με την εφαρμογή μιας μαθηματικής σχέσης στη γειτονιά ενός pixel (χωρικά φίλτρα). Μια εικόνα με θόρυβο “salt & pepper” και την εικόνα που παράγεται μετά την εφαρμογή ενός φίλτρου μεσαίας τιμής (median filter).


- **Μετατροπή τύπων**

Η έγχρωμη εικόνα εισόδου μετατρέπεται πολλές φορές σε εικόνα μονοχρωματική και στη συνέχεια σε δυαδική εικόνα για την εφαρμογή αλγορίθμων τεχνητής όρασης.

- **Μεταβολή μεγέθους**

Η μεταβολή του μεγέθους μιας εικόνας (συνήθως σμίκρυνση) με υπο-δειγματοληψία ή υπερ-δειγματοληψία ή εφαρμογή μαθηματικής σχέσης (μέση τιμή, πολυωνυμική μεταβολή).

- **Μετασχηματισμοί**

Ο μετασχηματισμός μιας εικόνας από το χωρικό πεδίο σε ένα άλλο πεδίο (π.χ. πεδίο συχνοτήτων) εξυπηρετεί διάφορες λειτουργίες επεξεργασίας. Για παράδειγμα, ο μετασχηματισμός Fourier μπορεί να χρησιμοποιηθεί για τον προσδιορισμό της θέσης μιας υπο-εικόνας (π.χ. ενός χαρακτήρα) σε μια άλλη εικόνα μέσω συνέλιξης των δύο εικόνων. Ο μετασχηματισμός Hough χρησιμοποιείται συνήθως για τον προσδιορισμό αντικειμένων ορισμένου σχήματος.

Εργαλεία Επεξεργασίας

- ***Jon's Java Imaging Library (JJIL)*** – Ανοικτό λογισμικό το οποίο περιλαμβάνει αρκετές μεθόδους επεξεργασίας εικόνας και διατίθεται από τον ιστότοπο:

https://code.google.com/p/jjil/

- ***OpenCV*** – Ανοικτό λογισμικό με πλήθος αλγορίθμων επεξεργασίας εικόνας το οποίο διατίθεται από τον ιστότοπο:

http://opencv.org/platforms/android.html

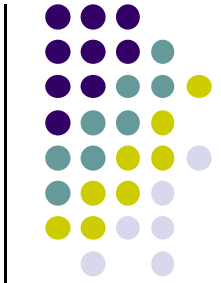
Εγκατάσταση της OpenCV3.0

- Εγκαθιστούμε στο κινητό μας το OpenVC Manager 3.0 από το Google Play.
- Κάνουμε download την τελευταία έκδοση της OpenCV για Android από το:
  http://opencv.org/downloads.html
- Το συμπιεσμένο αρχείο που κατεβαίνει έχει όνομα:
  OpenCV-3.0.0-android-sdk-1
- Το αποσυμπιέζουμε για παράδειγμα στα έγγραφά μου (OpenCV-android-sdk).

Πρέπει να δημιουργήσουμε ένα module για να μπορούμε να το ενσωματώνουμε στις εφαρμογές μας. Η διαδικασία στο Android Studio είναι:

- Import Eclipse Project
- Επιλέγουμε τον φάκελο sdk/java
- Δίνουμε το όνομα OpenCV300 (όχι υποχρεωτικά).
- Κλείνουμε το project (Close Project)

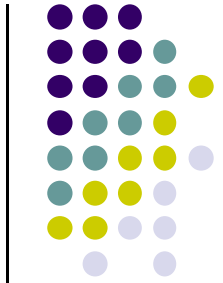
Δημιουργία εφαρμογής με OpenCV3.0



Δημιουργούμε ένα project για εφαρμογή επεξεργασίας των εικόνων που δίνει η κάμερα του κινητού.

- Start a New Android Studio Project με όνομα MyOpenCV3.
- Αφού ανοίξει, κάνουμε File → New → Import Module. Από το πλήκτρο Browse ψάχνουμε να βρούμε το module που έχουμε δημιουργήσει, δηλ. το OpenCV300, και πατάμε OK. Τσεκάρουμε το Import και στη θέση Module Name: όπου υπάρχει το :app τοποθετούμε το όνομα openCV300 (όχι υποχρεωτικά).
- Πατάμε στο project → δεξιό κλικ → Open Module settings. Επιλέγουμε την ετικέτα Dependencies, πατάμε το + και επιλέγουμε Module Dependency. Επιλέγουμε το :openCV300.
- Γίνεται αυτόματα Rebuild και η OpenCV3.0 είναι πλέον ενσωματωμένη στο project μας.

Διαμόρφωση εφαρμογής



- Κάνουμε Import από τα samples της νέας OpenCV το color-blob-detection. Κάνουμε copy-paste τα περιεχόμενα του αρχείου μανιφέστου, του activity_main και του αρχείου java (ColorBlobDetectionActivity) στη δική μας εφαρμογή.

- Σβήνουμε από το κύριο αρχείο αυτά που δεν χρειάζονται και φτιάχνουμε την εφαρμογή μας έτσι ώστε να λαμβάνει τα frames της κάμερας και στη συνέχεια να τα επεξεργάζεται με τη βοήθεια της OpenCV.

- Σαν πρώτο παράδειγμα να μετατρέπει την εικόνα σε δυαδική.

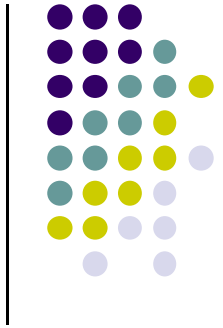
- **Αρχείο μανιφέστου:**

```
……
<activity
    android:name=".MainActivity"
    android:screenOrientation="landscape">
……

<supports-screens android:resizeable="true"
    android:smallScreens="true"
    android:normalScreens="true"
    android:largeScreens="true"
    android:anyDensity="true" />

<uses-permission android:name="android.permission.CAMERA"/>
<uses-feature android:name="android.hardware.camera"
android:required="false"/>
<uses-feature android:name="android.hardware.camera.autofocus"
android:required="false"/>
<uses-feature android:name="android.hardware.camera.front"
android:required="false"/>
<uses-feature
android:name="android.hardware.camera.front.autofocus"
android:required="false"/>
```

- **Αρχείο xml:**



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <org.opencv.android.JavaCameraView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/camerapreview" />

</LinearLayout>
```

- **Αρχείο java:**

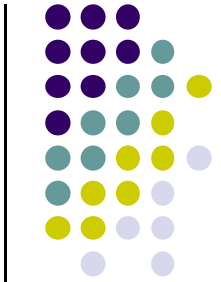
```
public class MainActivity extends AppCompatActivity implements
CameraBridgeViewBase.CvCameraViewListener2 {

……

public void onCreate(Bundle savedInstanceState) {
    Log.i(TAG, "called onCreate");
    super.onCreate(savedInstanceState);
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
    setContentView(R.layout.activity_main);

    mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.camerapreview);
    mOpenCvCameraView.setCvCameraViewListener(this);
}
```

- **Αρχικοποίηση και επεξεργασία των frames:**

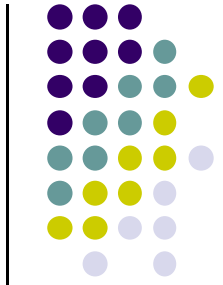


```
public void onCameraViewStarted(int width, int height) {
    mRgba = new Mat(height, width, CvType.CV_8UC4);
    hsv = new Mat();
    tgt = new Mat();
}

public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
    mRgba = inputFrame.rgba();

    // convert to HSV
    Imgproc.cvtColor(mRgba, hsv, Imgproc.COLOR_RGB2HSV_FULL);
    // make it binary with two thresholds
    Core.inRange(hsv, new Scalar(5, 50, 50), new Scalar(15, 255, 255), tgt);
    // track the edges
    //Imgproc.Canny(hsv, tgt, 40, 80);
    return tgt;
}
```

Δυαδική εικόνα



```
Imgproc.cvtColor(mRgba, hsv,
Imgproc.COLOR_RGB2HSV_FULL);
```

- Η λαμβανόμενη εικόνα αποθηκεύεται σε έναν πίνακα, ο οποίος στη συγκεκριμένη εφαρμογή καλείται `mRgba`. Το αντικείμενο αυτό δημιουργείται ως εξής:

```
mRgba = new Mat(height, width, CvType.CV_8UC4);
```

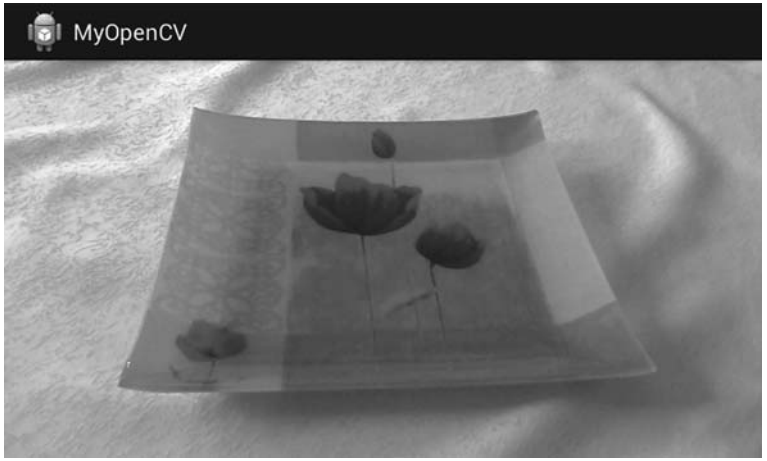
- Η κλάση *Mat()* της OpenCV χρησιμοποιείται για την αποθήκευση δεδομένων, όπως εικόνων, διανυσμάτων, πινάκων και είναι πολύ χρήσιμη στην επεξεργασία εικόνων.
- Στη συνέχεια μετατρέπεται σε HSV, `hsv`:

```
Imgproc.cvtColor(mRgba, hsv, Imgproc.COLOR_RGB2HSV_FULL);
```

- Τέλος εφαρμόζεται ένα κατώφλι μετατροπής σε δυαδική εικόνα, `tgt`:

```
Core.inRange(hsv, new Scalar(5, 50, 50), new Scalar(15, 255,
                                255), tgt);
```

Η Εικόνα αποτελεί την αρχική εικόνα που έχει ληφθεί μέσω της εφαρμογής και η οποία θα υποστεί την επεξεργασία. Να σημειωθεί ότι πρέπει στην κινητή μας συσκευή να έχει εγκατασταθεί ο *OpenCV Manager* από το Google Play Store.



Το χρωματικό μοντέλο της ληφθείσας εικόνας είναι RGB και αποτελείται από τρεις πίνακες, έναν για κάθε χρώμα. Η επεξεργασία μπορεί να γίνει ευκολότερα μετατρέποντας το RGB σε HSV (HueSaturationValue), με το οποίο η εικόνα περιγράφεται σε έναν πίνακα τιμών. Η μετατροπή αυτή πραγματοποιείται με τη μέθοδο *cvtColor()* της κλάσης *Imgproc*. Οι μεταβολές που πρέπει να γίνουν στην υποκλάση *myProcess* είναι:

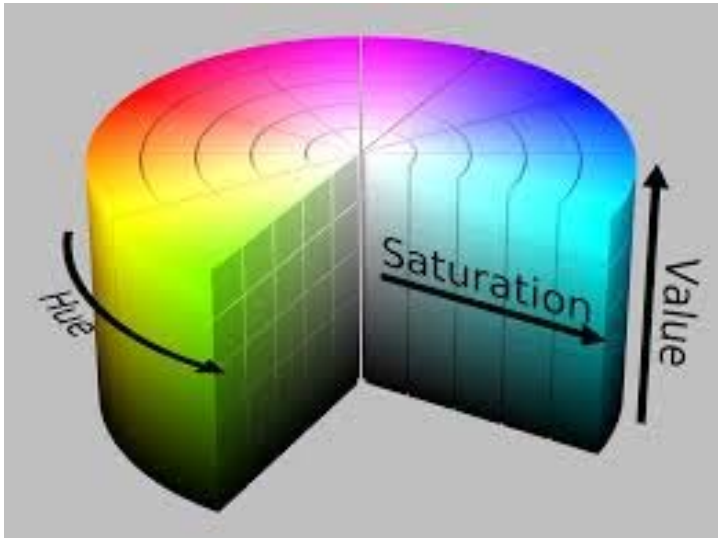
- Η μετατροπή του πίνακα *hsv* σε δυαδική μορφή πραγματοποιείται με τη μέθοδο *inRange()* χρησιμοποιώντας τιμές κατωφλίου (μία για κάθε συνιστώσα του χρωματικού μοντέλου) με την κλάση *Scalar*.
- Να σημειωθεί ότι η λειτουργία που πραγματοποιεί η μέθοδος *inRange()* είναι να μετατρέπει τις τιμές φωτεινότητας μεταξύ των δύο τιμών κατωφλίου σε λευκά pixels, ενώ όλα τα υπόλοιπα να τα κάνει μαύρα. Έτσι η μορφή της αρχικής εικόνας γίνεται ως εξής:

Χρωματικά Μοντέλα

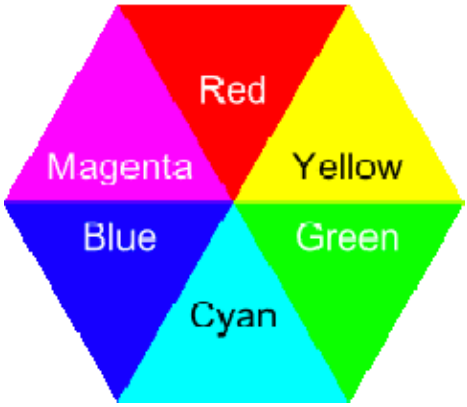
Στους αλγορίθμους επεξεργασίας εικόνας και ιδιαίτερα στην τεχνητή όραση χρησιμοποιείται το χρωματικό μοντέλο HSV (Hue Saturation Value) αντί του RGB (Red Green Blue) επειδή το HSV διαχωρίζει τη φωτεινότητα από το χρώμα. Σε πολλές εφαρμογές χρειάζεται το ιστόγραμμα φωτεινότητας μιας εικόνας, ενώ σε άλλες χρειάζεται να εντοπίσουμε ένα ορισμένο χρώμα. Ένα άλλο χρωματικό μοντέλο που διαχωρίζει τη φωτεινότητα από το χρώμα είναι το YCbCr.

Στο χρωματικό μοντέλο RGB εκφράζεται πόσο κόκκινο, πράσινο ή μπλε υπάρχει σε ένα pixel στην κλίμακα [0 255] για κάθε χρώμα. Επειδή οι τιμές των χρωμάτων προστίθενται και όσο αυξάνεται μια τιμή τόσο περισσότερο από αυτό το χρώμα προστίθεται, αν τα τρία χρώματα έχουν τη μέγιστη τιμή 255 το χρώμα του pixel είναι λευκό. Δεν είναι πολύ βολικό σύστημα για να μας δώσει την απόκλιση μιας απόχρωσης όταν παρατηρούμε τις μεταβολές χρωματισμού ενός pixel.

Ακριβέστερη περιγραφή ενός χρώματος μπορεί να γίνει με το χρωματικό μοντέλο HSV.



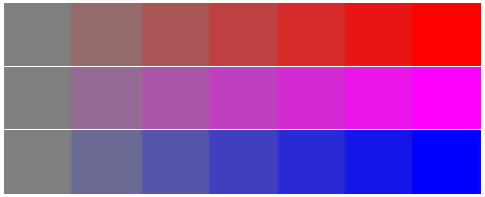
Παρατηρούμε ότι είναι πολύ εύκολο να προσδιορίσουμε την απόχρωση ενός pixel μεταβάλλοντας την τιμή Hue στην κλίμακα [0 360], την φωτεινότητά του μεταβάλλοντας την τιμή Value στην κλίμακα [0 100] χωρίς να αλλάξει η απόχρωσή του ή τον κορεσμό του στην κλίμακα [0 100]. Η τιμή 100 αποτελεί ποσοστό.
Στη χρήση του HSV στην OpenCV χρειάζεται προσοχή γιατί οι τιμές των τριών συνιστωσών διαφέρουν από αυτές που έχουμε πιο πάνω αναφέρει. Έτσι, η κλίμακα για τη συνιστώσα Hue είναι [0 180], για τη Saturation είναι [0 255], για τη Value είναι [0 255].



Hue

Απόχρωση

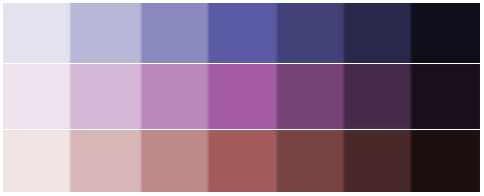
Μεταξύ κόκκινου και κίτρινου υπάρχουν οι πορτοκαλί αποχρώσεις



Saturation (Chroma)

Καθαρότητα χρώματος

Υψηλές τιμές δείχνουν το χρώμα πλήρες και πλούσιο



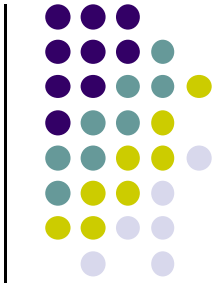
Value (Brightness)

Light or Dark colors

Ανάμιξη με λευκό ή μαύρο

Low saturation red, magenta and blue

Ανίχνευση Ακμών



Για να ανιχνεύσουμε τις ακμές των αντικειμένων σε μια εικόνα πρέπει πρώτα να την μετατρέψουμε σε εικόνα φωτεινότητας ή μονοχρωματική εικόνα. Αυτό γίνεται με τη μέθοδο *cvtColor()* ως εξής:

Imgproc.cvtColor(tab, hsv, Imgproc.COLOR_BGR2GRAY);

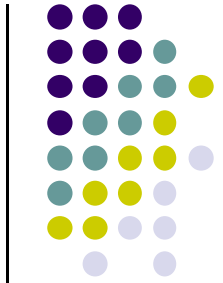
Στη συνέχεια εφαρμόζουμε τον αλγόριθμο ανίχνευσης ακμών Canny, ο οποίος χρησιμοποιεί δύο τιμές κατωφλίου, την *thr_min* και *thr_max*:

Imgproc.Canny(hsv, tgt, thr_min, thr_max);

Όσα pixels έχουν τιμή φωτεινότητας πάνω από τη μέγιστη θεωρούνται ότι ανήκουν σε ακμή. Όσα pixels βρίσκονται μεταξύ των δύο τιμών θεωρούνται ακμές εφόσον συνδέονται με pixels που ανήκουν σε ακμές. Όσα pixels είναι κάτω από την ελάχιστη τιμή δεν ανήκουν σε ακμές. Εάν θέσουμε thr_low=40 και thr_high=80 παίρνουμε τις ακμές του αντικειμένου της προηγούμενης Εικόνας.


MyOpenCV

Ασκήσεις



- Να δημιουργήσετε ένα δικό σας project παραγωγής δυαδικών εικόνων και ανίχνευσης ακμών. Να πειραματιστείτε με το χρωματικό σύστημα και τις τιμές κατωφλίωσης.