עיבוד תמונה

מה זה עיבוד תמונה

- עיבוד תמונה זו דרך לקבל מידע על הסביבה של הרובוט בעזרת מצלמה פשוטה ולהשתמש בו.

- אנו משתמשים במידע כדי למצוא את מיקום הרובוט ביחס לפסים מחזירי האור (ומכאן ביחס למגרש) או ביחס לאובייקטים אחרים שבחרנו לזהות (למשל,Power Cube).

סוגי מצלמות

pixy cam
יתרונות: בעלת הרבה פיצ'רים שימושיים לעיבוד תמונה כגון זיהוי אובייקטים ומעקב אחרי קו.
חסרונות: יקר, אין אפשרות לקנות בארץ (אבל יש אפשרות למשלוח).

(Ethernet (axis
יתרונות: אפשרות לכבל ארוך, מעבר מידע מהיר במקצת.
חסרונות: יקר, אין הרבה חיבורים כאלה על רכיבים.

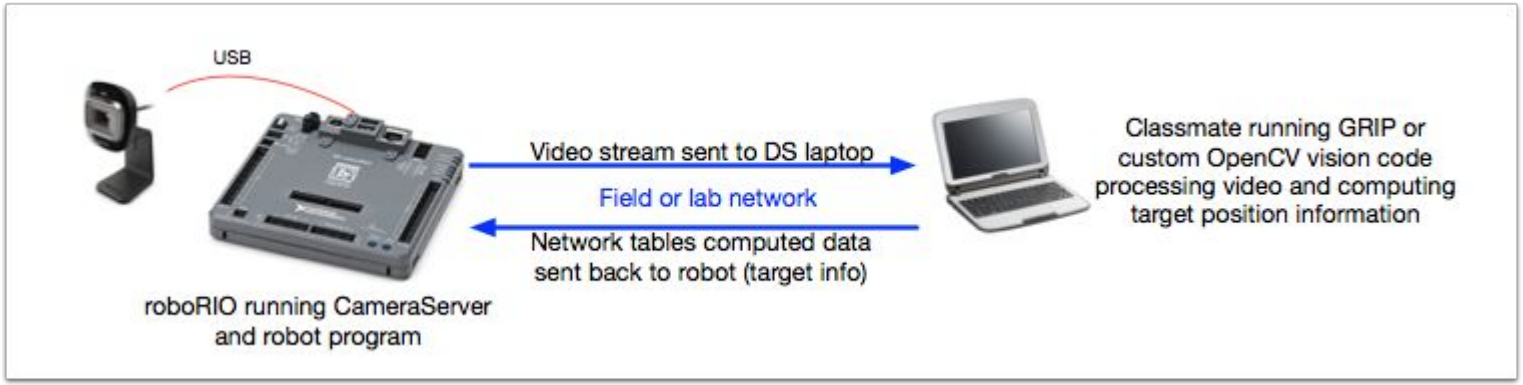
Usb
יתרונות: זול, לכל רכיב יש הרבה חיבורים כאלה.
חסרונות: אין פיצ'רים מיוחדים.

אסטרטגיות לעיבוד תמונה: עיבוד תמונה על הרובוריו

יתרונות: לא צריך להעביר את המידע מרכיבים אחרים.

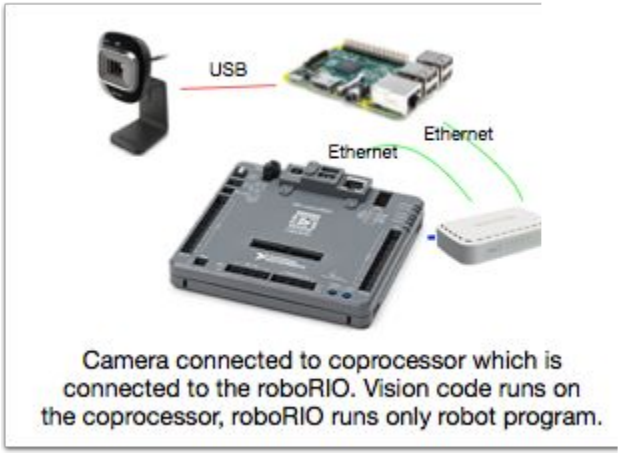
חסרונות: הרובוריו עסוק כבר בשליטה של הרובוט ולכן ייווצר עליו עומס והוא יעבוד לאט.

אסטרטגיות לעיבוד תמונה: עיבוד תמונה על הDS



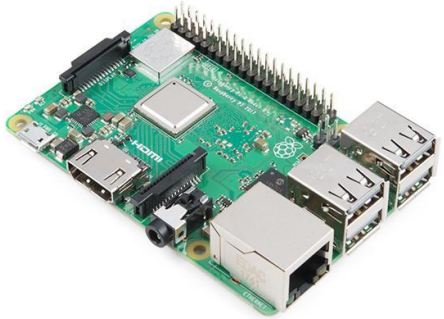
יתרונות: המחשב יכול לעשות את עיבוד התמונה מהר. לא צריך חומרה נוספת.
חסרונות: צריך לשלוח את כל התמונה דרך הרשת למחשב ואז לשלוח את הנתונים חזרה מהמחשב.

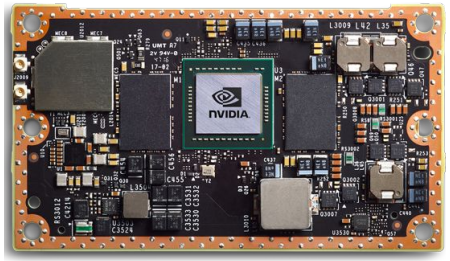
אסטרטגיות לעיבוד תמונה: עיבוד תמונה על CO-PROCESSOR



יתרונות: לא מעמיס על הroborio ולא דורש העברת מידע דרך רשת אלחוטית.
חסרונות: עולה כסף ודורש ידע בסיסי במעבד.

סוגים של מעבדים





שלבים בעיבוד תמונה

<u>עיבוד תמונה מורכב משלושה שלבים:</u>

- זיהוי התמונה - זיהוי ובידוד האובייקטים הרצויים מהתמונה וקבלת המידע הרצוי כגון גודל ומיקום.

- העברת המידע על התמונה מהמעבד לRoborio.

- ניתוח המידע שזוהה ושימוש בו למטרת בקרה על המערכת/מערכות.

שלב ראשון - זיהוי התמונה

הרעיון הבסיסי

הרעיון הבסיסי הוא לקבל את התמונה מהמצלמה, ולסנן ממנה את כל הדברים שהם לא הפסים מחזירי האור (או כל אובייקט אחר שתרצו), כך שהאובייקטים היחידים שנשארים על המסך (כלומר - עברו את תהליך הסינון) הם המטרות אותם אנו מעוניינים לראות.

אנחנו עושים את הסינון הזה באמצעות שימוש בפילטרים.

openCV - ספרייה לעיבוד תמונה

זו ספרייה לעיבוד תמונה שיש בה המון המון פונקציות שימושיות ל:

קריאת תמונות וסרטונים

המון פונקציות לעריכה ועיבוד של התמונות (או פריימים)

פונקציות שמחזירות ערכים מספריים מהתמונה לאחר העריכה והעיבוד.

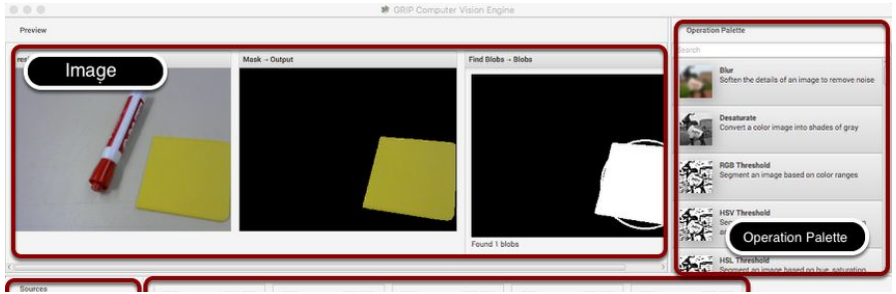
אנחנו משתמשים בכל הפונקציות האלה כדי ליצור pipeline

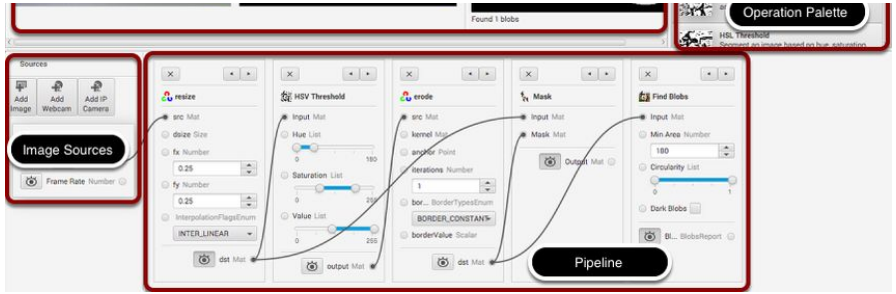
Grip

גריפ היא אפליקציה שפותחה על ידיWPILib והיא מאפשרת לנו ליצורPipeLine שמעבד את התמונה בצורה מאוד נוחה על ידי שימוש בפונקציות של openCV.

הממשק של Grip הוא ממשק GUI שמאפשר לנו לקבל input ממקור מסוים ולהעביר אותו מספר פילטרים, ולאחר מכן ליצור קוד מוכן בשפת תכנות מסוימת בהתאם לפילטרים שהגדרנו.

אנחנו נציג שלושה פילטרים עיקריים שבד"כ מספיקים לעיבוד תמונה מהסוג שאנחנו עובדים איתו : HSV Threshold, findContours, filterContures

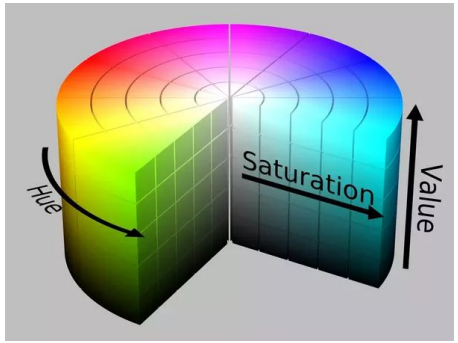


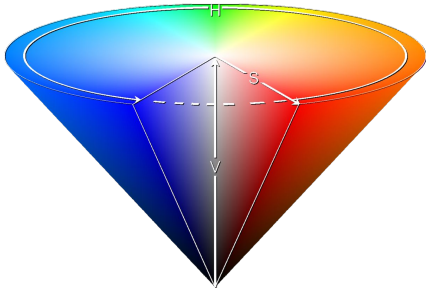


HSV Threshold

HSV/B stands for Hue, Saturation and Value/Brightness

- Hue : סוג הצבע (לדומה, אדום, כחול, ירוק).
  - הטווח ערכים נע בין 0 ל360 (מעלות). כל ערך מתאים לצבע: 0 זה אדום בזמן ש45 זה גוון של כתום ו55 זה גוון של צהוב.
- Saturation : העוצמה של הצבע.
  - נע בין 0 ל100%. 0 זה גוון של אפור בין לבן לשחור, 100 זה גוון ממש חזק של הצבע.
- Brightness (or Value) : הבהירות של הצבע.
  - בין 0 ל100%. 0 זה תמיד שחור, 100 זה גוון בהיר של הצבע.





find contours

את הoutput של פילטר הhsv אנחנו מעבירים כ input אל תוך פילטר בשם findcontours.
תפקידו של הפילטר הוא למצואcontours מתוך הinput שהוא קיבל.

הcontours הם הקווים המתאר של כל גוש צבע שעבר את הסינון בפילטר הקודם. אנו מסמנים אותם כדי להשיג מהם מידע בהמשך (מידע כמו אורך, רוחב, ומיקום X וY).

הוא לא עושה שום סינון בעצמו אבל מעביר את מה שהוא מצא לחלק האחרון.

filter contours

אחרי שביצענו את הסינון של הHSV, סיכוי סביר שעדיין נשארו המון רעשי רקע שעברו את הסינון של מסנן ומפריעים לנו לזהות אך ורק את האובייקטים אותם אנו רוצים לזהות.

הדרך להתמודד עם הפרעות אלה היא להשתמש בפילטר בשם filterContours. פילטר זה מסוגל לקבלcontours ולסנן אותם על פי פרמטרים שונים כגון אורך, גובה ושטח.

בדרך כלל ההפרעות שלנו יהיו קטנות יותר מהמטרות אותן אנו רוצים לזהות ולכן ניתן להסתמך על זה בתהליך הסינון של הcontours.

Exposure של המצלמה

אחת ההגדרות של המצלמה שלנו היא החשיפה שלה לאור (exposure). באפשרותנו לקבוע את חשיפת המצלמה. אם נקבע את ערך הexposure לערך נמוך יחסית, נראה רק את הדברים שמחזירים הכי הרבה אור למצלמה, פסים מחזירי האור. כלומר - נוכל להפחית משמעותית את כמות הרעשים שהמצלמה קולטת עוד לפני הרצת הקוד! (אבל שיטה זאת יעילה רק למציאת פסים מחזירי אור)

הערה- בהירות המצלמה חוזרת לברירת מחדל כל הפעלה מחדש, ועל כן אנחנו צריכים לקבוע אותה בכל פעם שהRPI נדלק (אנחנו עושים את זה באמצעות פקודה של video4Linux2 בקובץ rc.local, את הפקודה של בסוף המצגת).

העברת מידע ושימוש במעבד המשני

What do we use?

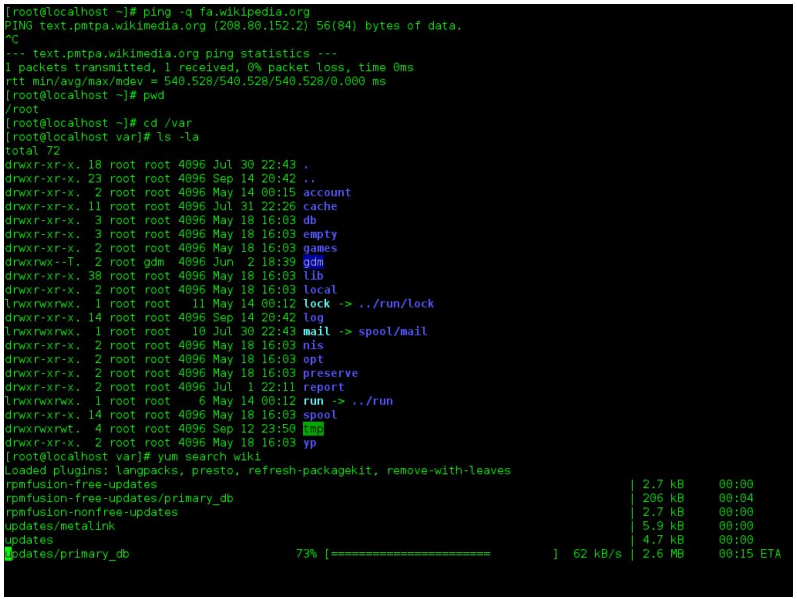
- Programming Language - Python3

- Communication Protocol - Network Tables

- OS - raspbian linux

Interface With The Pi

Command Line Interface



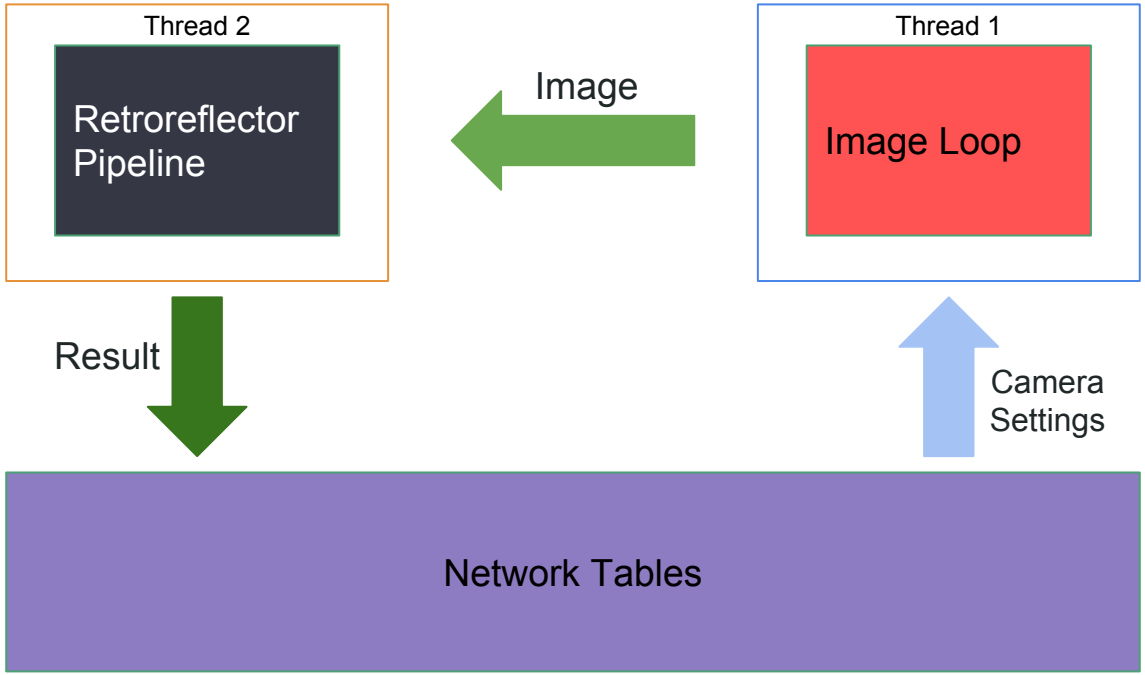
Network Tables

- Key-Value based communication protocol provided by WPILib

```
from networktables import NetworkTables

NetworkTables.initialize("10.22.12.2")
table = NetworkTables.getTable("ImageProcessing")

table.putNumber("number", 2212)
table.putString("name", "spikes")

x = table.getNumber("x")
```

Our Code

Old Version


Thread 2
Retroreflector Pipeline
Image
Thread 1
Image Loop
Result
Camera Settings
Network Tables

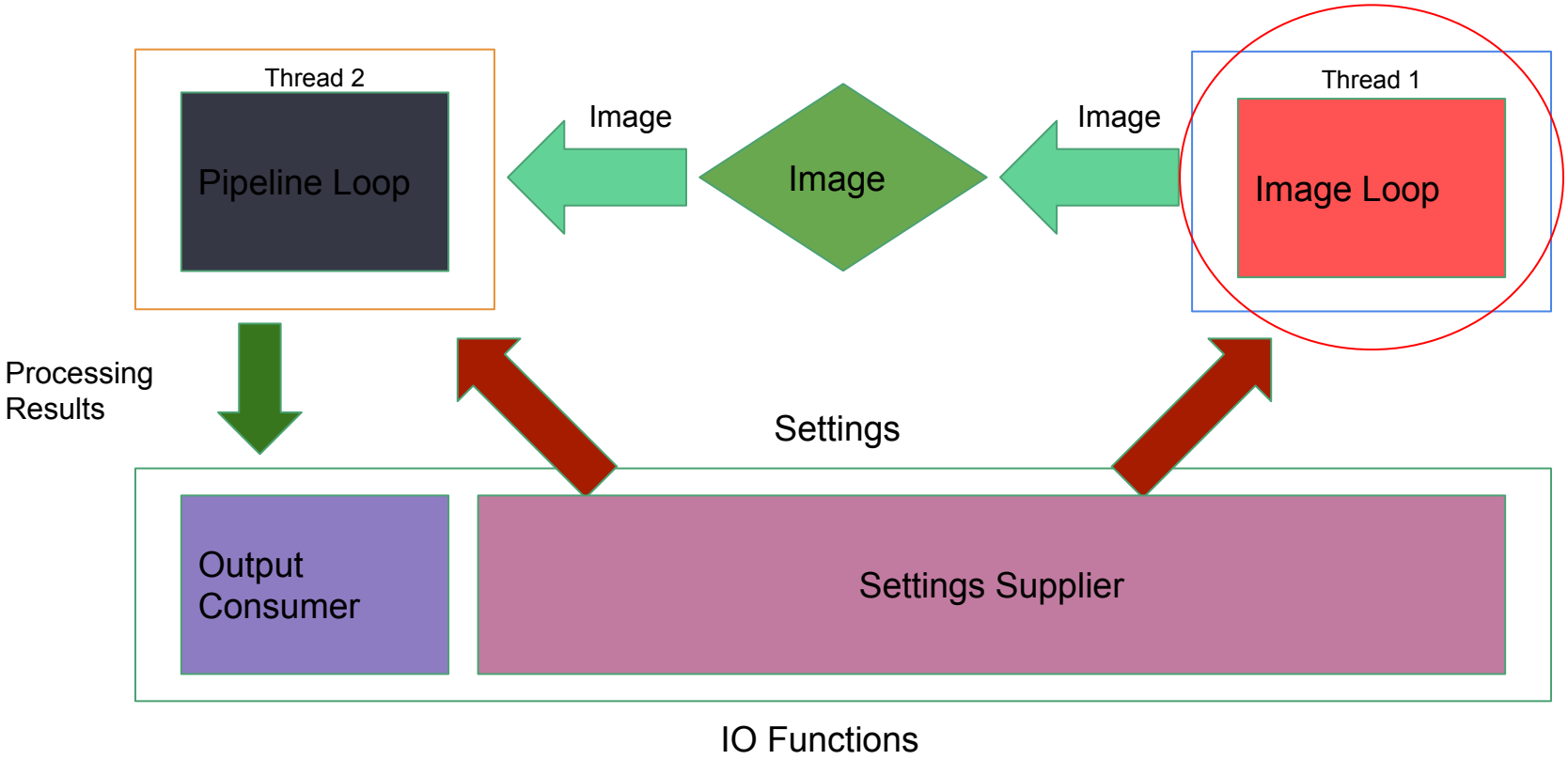
FIRST

New Requirements

- Change Pipelines On Demand
- Change Camera Exposure On Demand
- Change Camera On Demand
- Generic IO

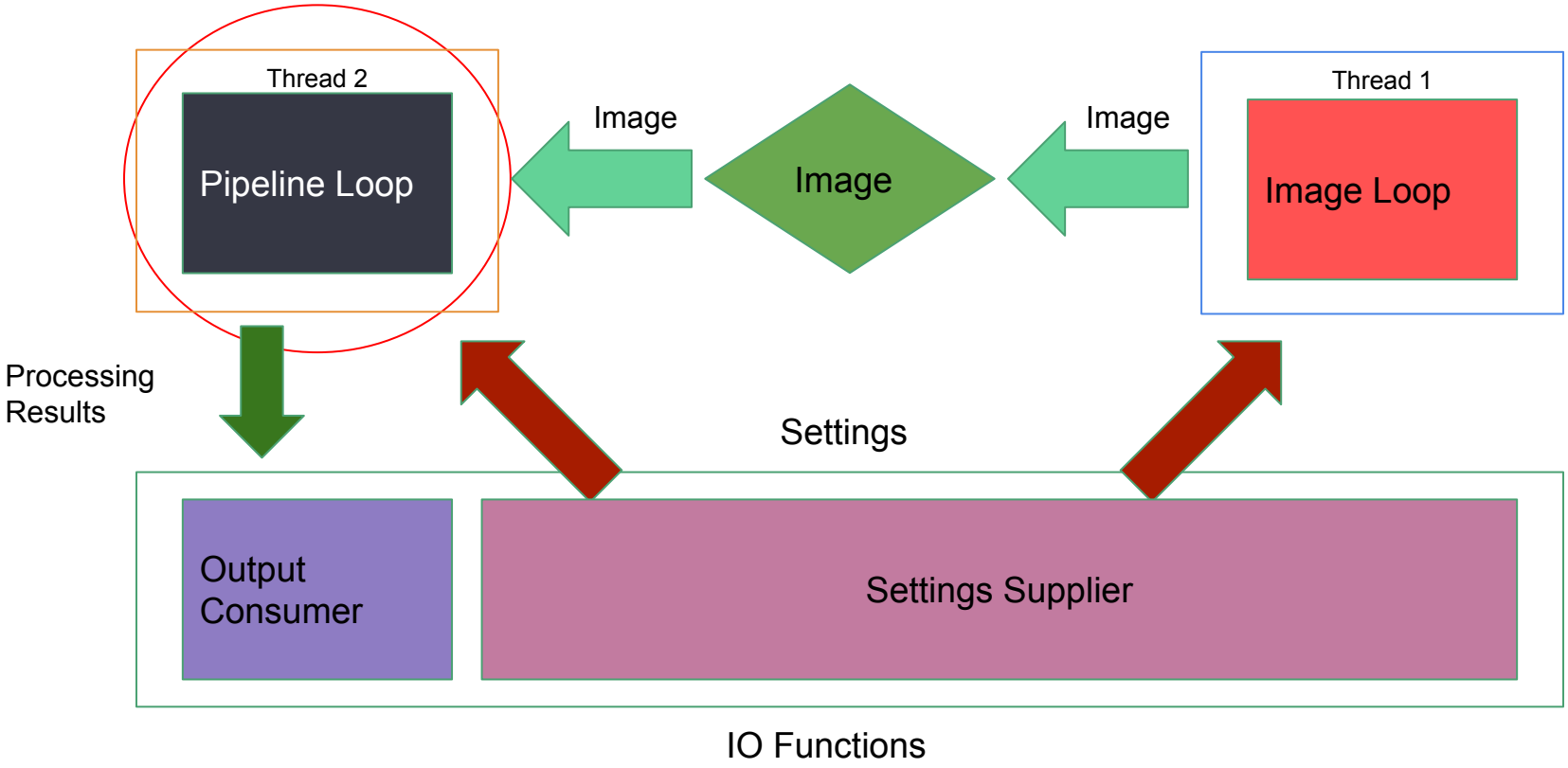
Spikes computer vision Framework

ScvF Architecture



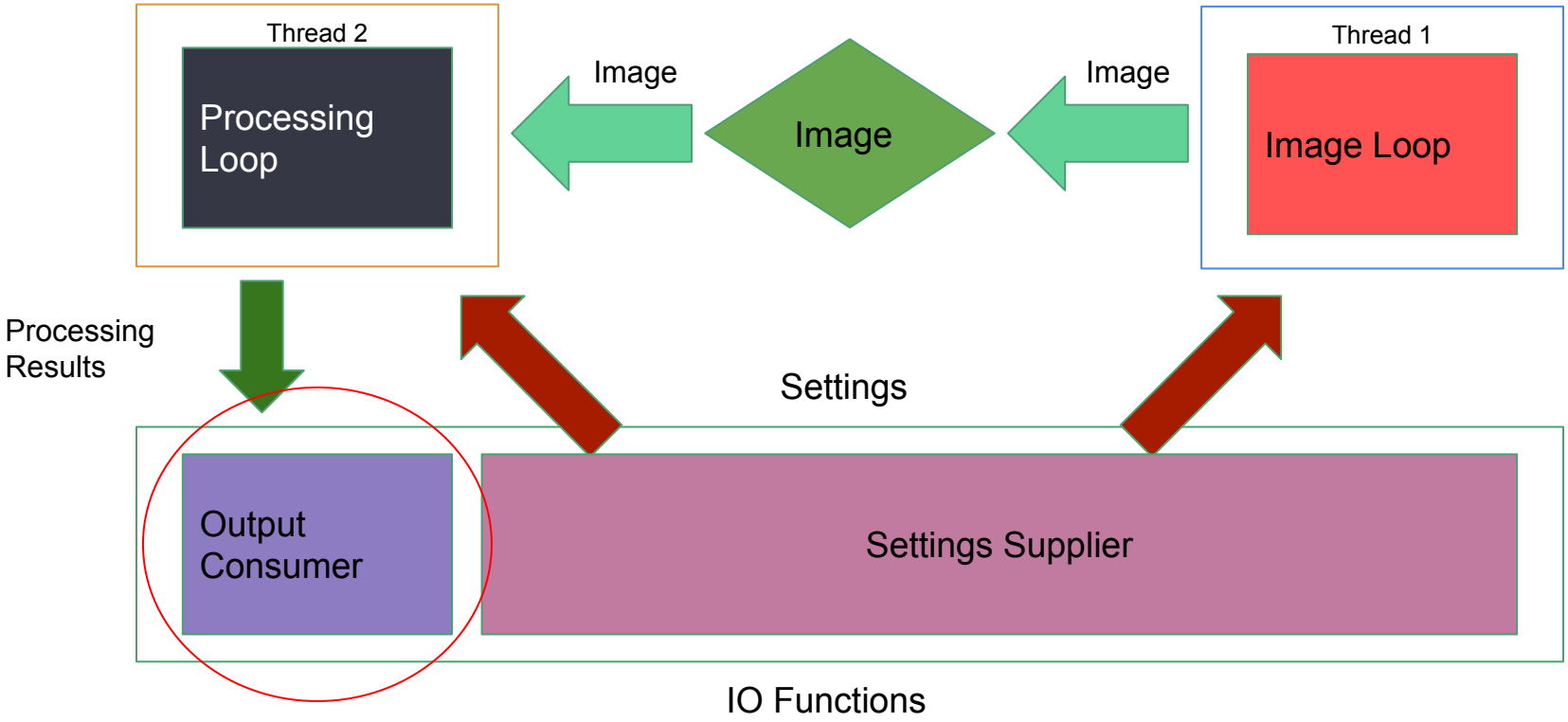
Camera Loop

```
def camera_loop(image_container, camera_manager, settings, running):
    while running():
        if settings.is_updated():
            camera_manager.set_exposure(settings.get(settings_keys["exposure"], ""))
            camera_manager.set_camera(settings.get(settings_keys["camera_id"], ""))

        img = camera_manager.get_image()
        if img is not None:
            image_container.set(img)
    camera_manager.release()
```


Thread 2
Pipeline Loop
Image
Image
Image
Thread 1
Image Loop
Processing
Results
Settings
Output
Consumer
Settings Supplier
IO Functions

Pipeline Loop

```
def pipeline_loop(image_container, pipeline_manager, settings, output_consumer, running):
    while running():
        if settings.is_updated():
            pipeline_manager.set_pipeline(settings.get(settings_keys["pipeline_name"], ''))
        img = image_container.get()
        if img is not None:
            pipeline_manager.process(img)
            output_consumer(pipeline_manager.get_output())
```


Thread 2
Processing Loop
Image
Image
Image
Thread 1
Image Loop
Processing Results
Settings
Output Consumer
Settings Supplier
IO Functions

Output Consumer

```
def output_consumer(self, output):
    contours = sorted(output, key=cv2.contourArea, reverse=True)
    self.contour_count = max(self.contour_count, len(contours))

    # sends info about all the filtered contours received by the function
    for i, c in enumerate(contours):
        self.nt.putNumber("contourArea{}".format(i), cv2.contourArea(c))

        x, y, w, h = cv2.boundingRect(c)

        self.nt.putNumber("width{}".format(i), w)
        self.nt.putNumber("height{}".format(i), h)
        self.nt.putNumber("x{}".format(i), x)
        self.nt.putNumber("y{}".format(i), y)

        self.nt.putBoolean("isUpdated{}".format(i), True)
        self.nt.putNumber("numberOfContours", len(contours))

    # turning off isUpdated flag for contours that were not updated
    for i in range(len(contours), self.contour_count):
        self.nt.putBoolean("isUpdated{}".format(i), False)
```

Our Main File

```
from scvf import cv_loop
from scvf.io import NetworkTableIO

from pipelines.retroreflective_pipeline import GripPipeline as RetroreflectivePipeline
from pipelines.powercube_pipeline import GripPipeline as PowercubePipeline

nt_io = NetworkTableIO("127.0.0.1", "ImageProcessing")

pipelines = {"reflective": RetroreflectivePi                    ercubePipeline()}
                           do_image_processing()
cv_loop.start(pipelines, output_consumer=nt_io.output_consumer, settings_supplier=nt_io.settings_supplier)

try:
    while True:
        pass
except KeyboardInterrupt:
    pass
finally:
    cv_loop.end()
```

Deployment

- On raspbian, make sure to add your ***app.py*** file to *<u>/etc/rc.local</u>* in order to start it when the pi boots

- We suggest you turn off the default gui that comes preinstalled with raspbian in order to improve performance

- in order to use grip pipelines with scvf add *get_output* function to the pipeline

שלב שלישי - שימוש במידע על הרובוט

שימושים למידע המסופק על ידי עיבוד תמונה

לאחר שהמידע הגיע לnetworkTables , אנחנו יכולים לקרוא אותו משם ולהשתמש בו בשביל לבצע מגוון משימות:

- התכוונות ואוריינטציה למרכז בין שני פסים מחזירי אור
- חישובי מרחק מהמטרה
- התכוונות לקובייה כדי להרים אותה
- ועוד ועוד

בשקופיות הבאות נראה את דוגמה אחת לשימוש כזה.

התכוונות ואוריינטציה למרכז בין שני פסים מחזירי אור

- חישוב מרכזי שני הפסים (האובייקטים הכי גדולים) בפיקסלים בעזרת הגודל של האובייקט בתמונה.
- המרת הערך לערך בין 1 ל0 ואז חיסור 0.5 כדי לקבל טווח בין -0.5 ל0.5 כדי שאמצע המצלמה יהיה 0.
- חישוב ממוצע של שני מרכזי הפסים כדי לקבל את המרכז שלהם.

```
public static final Supplier<Double> BIG_OBJECT_CENTER = () -> ((NETWORK_TABLE.getEntry("x0").getDouble(0)
                + 0.5 * NETWORK_TABLE.getEntry("width0").getDouble(0))                    ;
// calculates the center of the small reflective
public static final Supplier<Double> SMALL_OBJECT_CENTER = () -> ((NETWORK_TABLE.getEntry("x1").getDouble(0)
                + 0.5 * NETWORK_TABLE.getEntry("width1").getDouble(0))                    ;
```

ועכשיו, הדגמה

לעוד מידע

אם יש לכם שאלות על המצגת, אתם רוצים להשתמש בקוד שהוצג פה או רוצים עזרה בכתיבת קוד משלכם מוזמנים לפנות אלינו:

עמרי כהן - ראש צוות תוכנה - 0546264324

סימון חרמצקי - 0544855808

קורס תכנות ב-Java

מפגש ראשון למתחילים: 2.11

- בקרי מהירות וחיישנים
- שימוש במחלקות חדשות ב- Java 8
- בעיות תוכנה נפוצות ופתרונן

מפגש שני למתקדמים: 9.11

- בקרת PID
- SpikesLib

לפרטים נוספים והרשמה, מוזמנים
לבקר בפייסבוק שלנו