CONFIDENTIAL | AUTHORIZED HPE PARTNER USE ONLY

**HPE EZMERAL  
DATA FABRIC AI   
EVENT DEMO**

INFORMATION AND INSTALLATION DOCUMENTATION

Alex Ollman

Last Updated: 4th November, 2022

**PHYSICAL REQUIREMENTS**

*Edge Machine:*

* 8 cores
* 64GB RAM
* 2x 500GB Disks
* Ethernet NIC

*Camera:*

* Network (IP) camera, capable of RTSP streaming over WiFi. Must be connected to the same network and subnet as the Edge Machine.
* **OR** a USB webcam, connected and visible by Edge Machine and demo OS.

*Robot:*

* DJI Robomaster EP or EP Core
* **OR** DJI Tello Drone
* **OR** custom mobile robot, capable of communicating over WiFi and from packets generated by a Python script.

*Printer:*

* Canon SELPHY CP1300
* **OR** any printer with drivers compatible with your OS.

*Networking - Standardised*

* A non-enterprise modem router (internet connection optional, only if cloud/external Data Fabric cluster component is desired)
* A PoE injector (for IP camera, optional)

Connect all devices, including controlling laptops, Edge Machines, [robot](https://www.youtube.com/watch?v=5uyMpggPU3U&ab_channel=DJISupport) and [printer](https://www.youtube.com/watch?v=gUDDQuccDBE&ab_channel=CanonPrintAssist) to the same local network.

*Networking – Aruba*

PoE Injector -> Aruba 505H -> Aruba PoE Switch -> IP Camera

Option B:

Generic Router -> PoE Injector -> IP Camera

*ZeroTier*:

Connect to a [ZeroTier](https://www.zerotier.com/) network to enable multiple clusters to communicate with one another over the same virtual subnet. Great for setting up multi-cloud demo environments. Installation details can be found on their website. An account (free) is required and with one, you will be provided a Network ID.

To setup on Linux machines, run the following:

curl -s [https://install.zerotier.com](https://install.zerotier.com/) | sudo bash

sudo zerotier-cli join {Network ID}

sudo shutdown -r now

**(VIRTUAL) EDGE MACHINE SET UP**

Operating System:

1. Install a Linux OS. Ubuntu 20.04 is **highly** suggested and used for this demo. Create a virtual machine or load it bare metal. Rest of steps still apply to other Linux distributions but are not guaranteed to work.
2. During set up, ensure that both (virtual) disks are connected and present (/dev/sdb as the second, non-boot disk)
3. Create user called “hpe”. Set password to “mapr”.
4. Open Terminal
5. **sudo passwd root** – enter in user password, then set root password to “mapr”.
6. sudo apt-get update && upgrade
7. Run the following:

apt-get install curl wget nano net-tools openssh-server git pssh expect

sudo ufw disable

iptables -F

iptables -Z

1. Add the IP addresses of each of the nodes/clusters in the hostname. Edit the hostname (Suggested: edge1.com)
   1. **nano /etc/hostname**
   2. **nano /etc/hosts**
2. Edit **nano /etc/ssh/sshd\_config**:
   1. PermitRootLogin yes
   2. PasswordAuthentication yes
3. Run the following:

service sshd restart

shutdown -r now

1. Repeat for every node in every cluster.

**INSTALLING EZMERAL DATA FABRIC**

Initial Setup:

1. Run the following:

systemctl stop numad

systemctl disable numad

echo "vm.overcommit\_memory=0" >> /etc/sysctl.conf

echo "vm.swappiness=1" >> /etc/sysctl.conf

echo "net.ipv4.tcp\_retries2=5" >> /etc/sysctl.conf

sysctl -p

echo "umask 0022" >> /etc/profile

echo never > /sys/kernel/mm/transparent\_hugepage/enabled

echo "mapr - nofile 65536" >> /etc/security/limits.conf

echo "mapr - nproc 64000" >> /etc/security/limits.d/90-nproc.conf

echo "session required pam\_limits.so" >> /etc/pam.d/su

1. Repeat for every node in the cluster.
2. Run the following to get the Ezmeral Data Fabric installer script:

wget  <https://package.mapr.com/releases/installer/mapr-setup.sh> -P /tmp

chmod +x /tmp/mapr-setup.sh  
cd /tmp

./mapr-setup.sh

1. Follow the prompts. Where prompted and applicable, keep the default value [shown by square brackets] by clicking “Enter” (or Return) on them, **unless entering the password**:   
     
   Enter cluster admin password: **mapr**   
   Re-enter password: **mapr**
2. Once finished, open a web browser and navigate to   
   https://{IP address of the machine}:8443.

Cluster Setup:

Sign in with the credentials that were entered during the Initial Setup (default: mapr/mapr)  
For a video walkthrough of this installation, [click here](https://www.youtube.com/watch?v=lkjjiYONN2A&ab_channel=AntonFedorov).

*Services:*

Select the latest version of the Data Fabric (**7.0** at time of writing, known good).

Ensure “Enterprise Edition” is checked.

License Option: Add License After Installation Completes

Ensure “Enable Secure Cluster” is **unchecked.**

Ensure “Enable NFS” is **checked,** with version 4.0.

Select the latest MEP version (**7.0.1** at time of writing, known good)

Select “Batch, interactive and real-time analytics” for a **normal install.**   
For **Demo,** click “Show advanced service options”. Select Object Store Gateway with S3-Compatible API, Airflow, Binary DB OJAI and all Kafka-related services.)

Ensure “Enable Verbose Logging” is **checked.**

Click Next.

*Databases:*

For all selected databases, select “Install shared MySQL Server”. Enter credentials in as “mapr/mapr.”

Click Next

*Monitoring:*

Ensure “Install and set up log collection infrastructure” is **checked**.

Under “Enter password for Elasticsearch Administrator ID”, enter in the same password as cluster admin user (default: mapr)

Click Next

*Cluster:*

Enter the cluster admin password (default: mapr) where prompted.

Set the cluster name   
(demo defaults: **Edge -** edge1.enterprise.org **Cloud/DC -** dc1.enterprise.org)

*Nodes:*

Enter the **hostname** of each node in the cluster. For the Ezmeral AI demo, only single node clusters are required. In this case, enter the hostname of the primary machine only. (as declared in Step 8 of Operating System in the Machine Setup chapter. Demo default: edge1.com).

Under “Disks”, enter “/dev/sdb” (the secondary drive declared in Step 2 of Operating System in the Machine Setup chapter).

Set the “Login Method” to **SSH – Password**

SSH Username: **root**

SSH Password: **mapr**

SSH Port: **22**

**Ensure these are correct, most verification/installation errors are related to an inability to connect to nodes.**

Click Next:

*Verification:*

The cluster nodes should verify without errors. Ignore warnings. If present, please inform and email [**jimmy.bates@hpe.com**](mailto:jimmy.bates@hpe.com)**.**

When complete, click Next.

*Layout:*

Here is where certain services can be assigned/installed only to certain nodes. Useful if particular nodes have existing applications and more limited resources than others.

For a demo environment, keep the suggested/default layout.

Click Install.

*Installation:*

The cluster nodes should install without errors. Ignore warnings. If present, please inform and email [**jimmy.bates@hpe.com**](mailto:jimmy.bates@hpe.com)**.**

When complete, click Next.

*Licensing:*

Skip. Click Next.

*Completion:*

You have now completed a demo installation of the HPE Ezmeral Data Fabric. To navigate to the Mission Control System (MCS), open a browser and go to https://{IP address of machine}:9443.

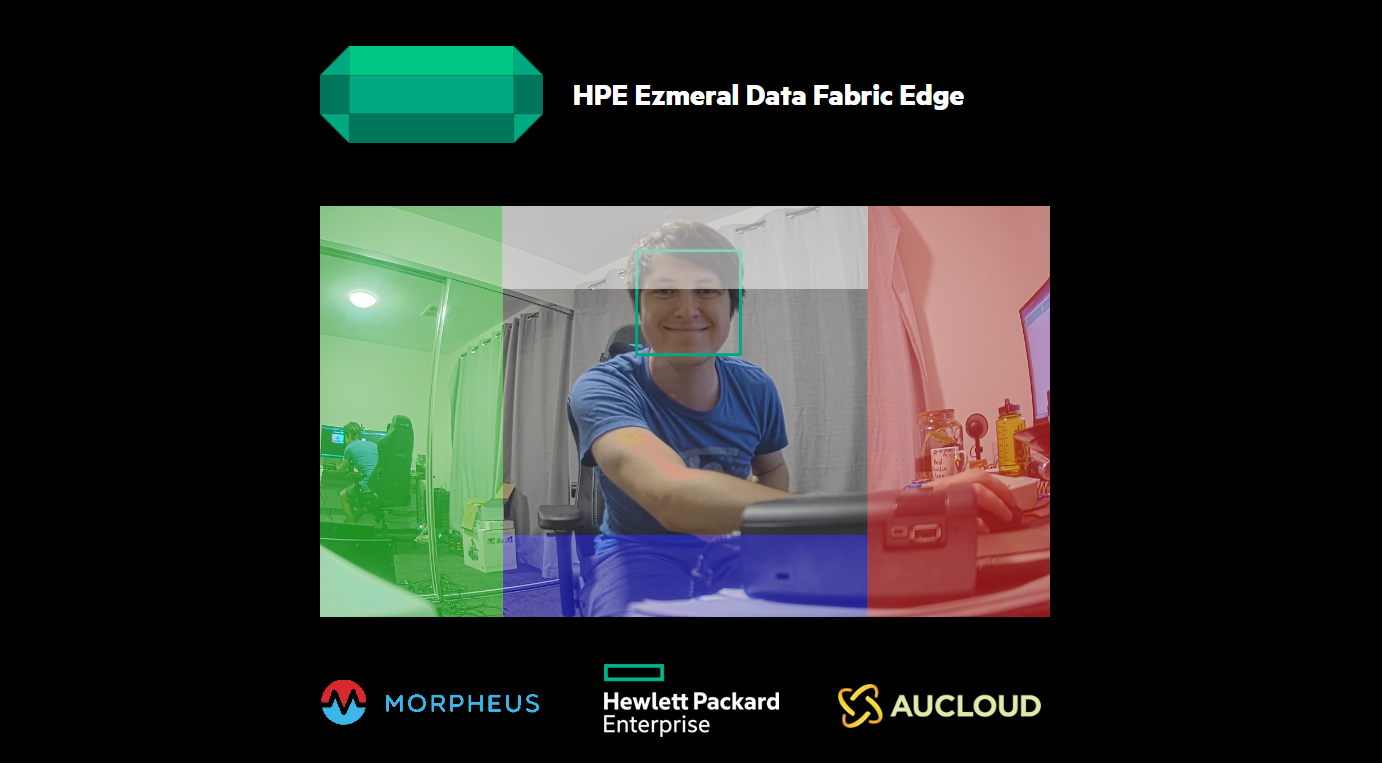
**GET LICENSE**

You are now ready to install the Ezmeral AI Demo.

**DEMO OVERVIEW**

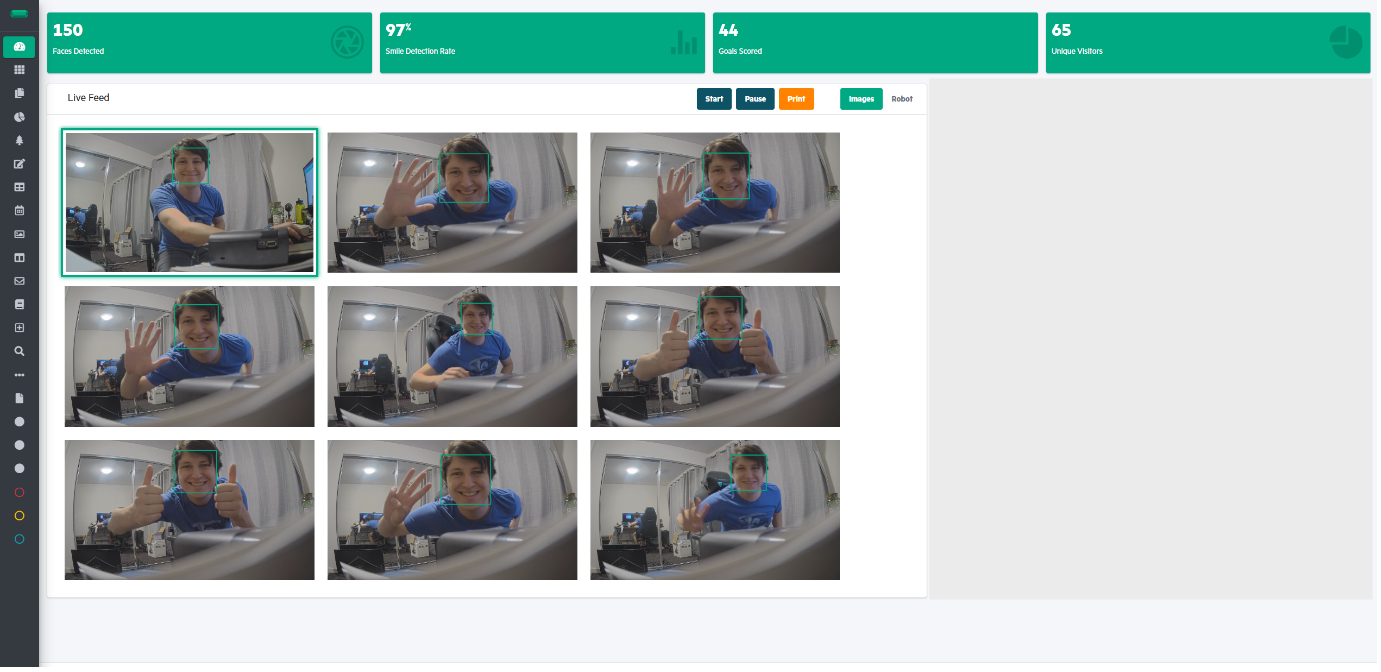
*Summary:*

The Ezmeral AI demo aims to showcase several features of the HPE Ezmeral Data Fabric, including File and Object Store, Event Streams and Databases. The gamification of the core Data Fabric features is to have a user control a mobile robot using the location of their face in a frame. A user will look at a camera, placing their face in the centre of the frame. The user will then move into the coloured rectangles to send commands to move the robot forward (white) or backward (blue), or rotate the robot left (green) or right (red).



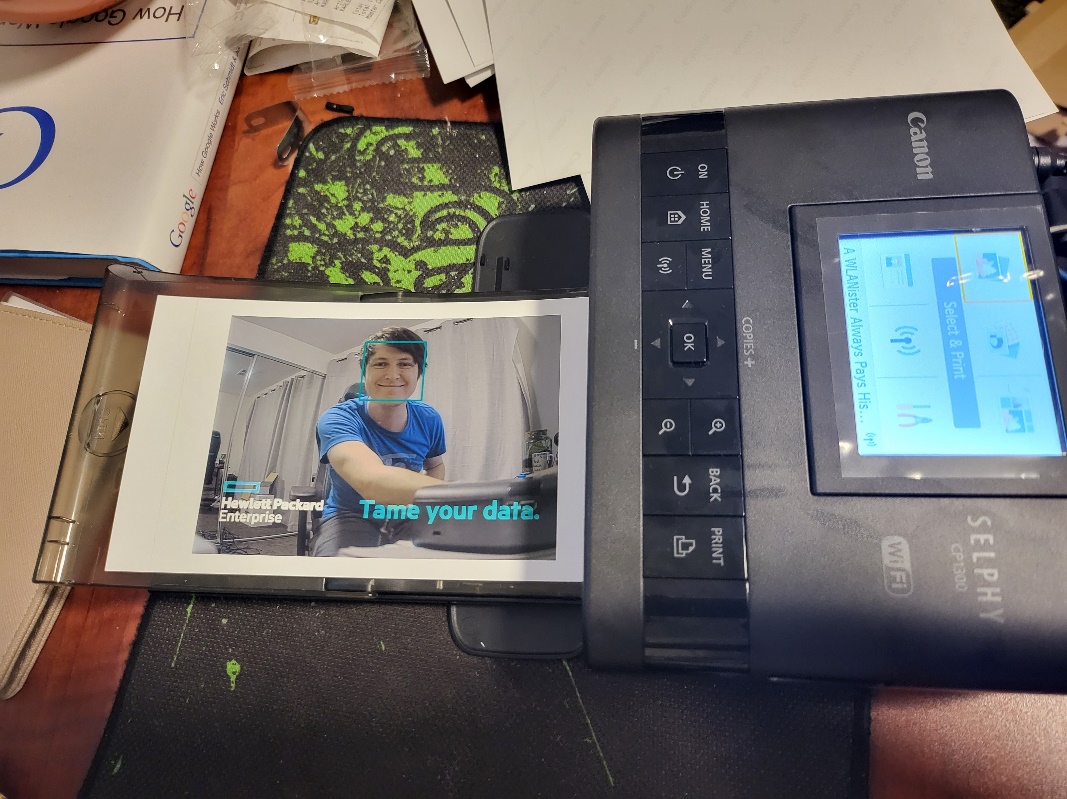
*In the case of this frame, with the detected face overlapping the top white rectangle, the Face Recognition component would send a command to the robot to move forward.*

When a user smiles, the frame of them smiling is captured as a JPG and saved into the Data Fabric. A Live Dashboard, appearing on another screen, will show the saved smiling images in real time as they are saved into the Fabric. Similarly, when a DJI robot is connected, the robot’s camera will capture a frame every 10 seconds and save it to the Data Fabric.



*When the Live Dashboard is running, smiling images and images produced by the robot appear in real-time as they are saved into the Data Fabric.*

A user can then take their experience with them with a snapped photo of their smiling selves, selected from the Live Dashboard and superimposed with the HPE photo and tagline (for a recent event, this was “Tame your data”.



*Using a Canon SELPHY CP1300, a user or demo operator can select one of the images on the Live Dashboard and print it over WiFi.*

**DEMO COMPONENTS:**

The demo is made up of three core components: **Facial Recognition**, **Robot Control** and **Live Dashboard.**

*Facial Recognition (main.py):*

The Facial Recognition component is a Python Flask application which takes in video from the declared camera, performs face and smile recognition, determines the location of the detected face to send controlling commands to a DJI robot, and outputs frames to a web application for showing the user where their detected face is in the frame.

It does so by running several functions as individual processes in parallel, which pass information from one another using “queues in memory (*multiprocessing.Manager.Queue())*:

* *cam\_loop()* opens up a connection to the declared camera. It then continuously captures a frame and pushes it to the *queue\_from\_cam.*
* *detectFaces()* pops the last frame on *queue\_from\_cam,* performs facial recognition using Haar Cascade detection, draws a HPE Green-coloured bounding box around the persons face on the frame, then pushes both the edited frame and the bounding box coordinates to *queue\_from\_faces.*
* *detectSmiles()* pops the last frame on *queue\_from­\_faces,* performs smile recognition using Haar Cascade detection, saves frames with detected smiles on them to the “static/images” folder, making them available to the other Flask applications, and publishes the saved file locations to an Event Streams topic.
* *robotRectangles()* pops the last frame on *queue\_from­\_faces,* draws the coloured rectangles on the frame, and pushes the edited frame to *queue\_from\_rectangles.*
* *detectOverlap()* pops the bounding box coordinates from *queue\_from\_faces*, and compares with the declared coordinates of each of the four control rectangles. If there is any overlap, a message is published to an Event Stream with the appropriate command to move the robot forward, backward, left, or right. If no overlap is detected, a message with the “stop” command is published.
* *robotControl()* is the process which runs the **Robot Control** component. The “Start” and “Stop” buttons on the Facial Recognition web application either call or kill this process (starting or stopping connection to the robot). It does not start when the script does, only starting when the “Start” button is pressed.

*Robot Control (robot\_control.py):*

The Robot Control component starts the connection to the robot by creating a Robot object (from the robot.py script). This object begins the connection to the robot and enables the image streaming service. The Robot Control component subscribes to the appropriate Event Stream, receiving commands that have been published by the *detectOverlap()* function of the Facial Recognition component. These commands are then mapped to the movement commands, which are then sent to the robot.

*Live Dashboard (cloud\_dash.py):*

The Live Dashboard component is a Python Flask application which finds the ten most recently saved images in the Data Fabric (“static” folder) and displays them on a webpage. The Live Dashboard is built on the [AdminLTE 3](https://adminlte.io/themes/v3/) Bootstrap template and is highly modular. Components from the AdminLTE board can be copied into the HTML code of the dashboard (“templates/index3.html”) as desired. Javascript functions written within the index3.html file are parsed arrays of the numbered suffixes of the last ten images and their file paths from the Flask application. This is what enables the real-time “refresh” of the images. The “Start” and “Pause” buttons on the Live Dashboard start and stop the stream of images. A user can select an image on the Live Dashboard and press the “Print” button, which will send the path of the selected image to the Flask application via JSON and call the function to print that image.

**DEMO INSTALL**

*Installation*

1. Log into Edge Machine as “mapr” user. (p/w: “mapr”)
2. maprcli volume create -name projects -path /projects -type rw
3. su hpe (p/w: “mapr”)
4. git clone <https://github.com/AlexanderOllman/hpe_ezmeral_demo.git>
5. cd hpe\_ezmeral\_demo
6. nano settings.py

In *settings.py*, check or edit the following:

* **USERNAME:** Primary user, **not** root or mapr user. Default is “hpe”.
* **PASSWORD:** Password for primary user. Default is “mapr”.
* **PEM\_FILE:** Used for database secure connection. Leave default unless required.
* **CAMERA\_IP:** The IP address of the camera (in inverted commas). If using a USB webcam, set **CAMERA\_FEED** equal to 0 (no inverted commas).
* **PRINTER:** Boolean if printer is being used in demo. Default “False”.
* **PRINTER\_NAME:** Name of the printer according to the OS (in inverted commas). To find, run *lpstat -l* in the terminal of the Edge Machine. Set the desired printer as default using *lpadmin -d {Printer Name}*
* **ROBOT\_IP:** IP Address of Tello Drone or Robomaster robot. **Only need to declare if using an Aruba/enterprise-grade access point.** Otherwise, leave blank.

1. sudo ./sudo-setup.sh
2. ./setup.sh
3. python3 configure.py

The demo has now been installed and is ready to run. Navigate to “/mapr/projects/ezai/” and run the following:

1. source init.sh (**very important,** and **must** be logged in as hpe user. Scripts will not work otherwise).
2. python3 main.py **to run the Facial Recognition component.**
3. python3 cloud\_dash.py **to run the Live Dashboard component.**
4. python3 robot\_control.py **to run the Robot Control component.** However, this is not advised as robot\_control.py should be called from within the Facial Recognition web application using the “Start” button.
5. Navigate to the following in web browser of any device connected to the same network as the Edge Machine:
   1. To load the Facial Recognition component:  
      http://{IP Address of Edge Machine}:9990
   2. To load the Live Dashboard component:  
      http://{IP Address of Edge Machine}:9991

**PRINTER**

Model: Canon SELPHY CP1300

Printing scripts only work with **Linux** operating systems.   
  
Whenever the printer is connected or restarted, the following **must** be performed or the printer will only print blank images:

1. Power on the printer.
2. **Ensure** printer is connected to the WiFi network (**must** be thesame subnet as Edge Machine)
3. Log into the Edge Machine UI (via VNC, vSphere or equivelant)
4. Check Settings > Printers and ensure printer has been found and is connected (has green tick) and its name is “Canon\_SELPHY\_CP1300”.
5. Manually print off any PDF (there is one on the desktop if needed [face\_1.pdf]). Do so by double-clicking on the image and printing with Ctrl-P.

Images printed off from the Cloud Dashboard should now print successfully.

**ADDITIONAL INFORMATION**

*ORB-Deface Library*

ORB-Deface is a Python application which performs facial detection on media files. It is capable of processing frames of crowds at a distance and achieve incredible accuracy with only a small pixel area. It is built on a the cv2 Deep Neural Network library and uses an ONNX model trained on public datasets. It uses CUDA drivers to leverage GPUs (specifically NVIDIA ones) to perform the inferencing at speed.

The facial recognition is near best-in-class for both speed and accuracy. The application for use in the Ezmeral AI demo is the fact that using the current Haar Cascade facial classifier only detects front-on face and, if desired (and would require another function/process), side profiles. ORB-Deface is able to seamlessly detect all faces, regardless of orientation or partial obstruction.

In the Ezmeral AI *main.py* script, the function *detectFaceCNN()* applies the ORB-Deface detection. This is not being called (instead *detectFace()* is called for the *face\_process*). During testing on a system with no dedicated GPU (HPE e920 blade), the inferencing on frames in real-time took nearly 30 seconds. This function has not yet been tested on a GPU.

The ORB-Deface application can be installed to run as a standalone or installed as a package. The *sudo-setup.sh* script in the Ezmeral AI demo repository installs ORB-Deface (as “deface”) as a package by default. To use it in scripts:

*from deface* *import deface  
from deface.centerface import Centerface  
  
dets, \_ = centerface(frame, threshold=threshold)   
  
#frame is image, threshold is accepting face based on accuracy. Default is 0.2, so even if only 20% confident. dets are a [n x 4] array of x1,y1,x2,y2 coordinates for the bounding box, where n is the number of faces detected in the frame*