3D Tongue Tip Tracking

Navaneeth Balamuralidhar, *n.balamuralidhar@student.utwente.nl, s2070707, M.Sc. Systems and Control*
Saumya Kumaar Saksena, *s.k.saksena@student.utwente.nl, s2084627, M.Sc. Systems and Control*
and Zoe Chambre, *z.c.m.chambre@student.utwente.nl, s2260298, M.Sc. Computer Science*

***Abstract*—In this project, we attempt to track the tip of tongue in 3-dimensions using Farneback's optical flow tracking technique and estimating its 3D visualization using multi-view geometry. We also present certain benchmarks for this research. The remaining sections deal with tracking of other facial features in image frames in both 2D and 3D. We also compare two different approaches to achieve the objective of 3D tracking.**

***Keywords*—Computer Vision, Tongue Tip Tracking, Facial Feature Detection, Multi-view Geometry**

I. INTRODUCTION

PATIENTS undergoing surgery and/or radiotherapy in the oral region, especially the tongue, have the risk of limited tongue mobility with serious deterioration of oral functions, such as speech, food transport, swallowing, and mastication. The mobility of tongue is expressed in the so-called 'Range of Motion'. To study the statistical correlation between the range of motion and a given treatment, this particular range of motion should be measured in a population of patients. This is to be done before and after the treatment. We now can write the official statement of project as to measure the range of motion, patients are asked to move their tongue to standardized, extreme positions, left, right, forward, downward and upward.

Although tongue tip tracking using computer vision is not a very new concept, it is still a significant challenge to tackle, given the current developments in image processing and pattern recognition. Research in this field dates back to 1986 when Stone *at. al.* [13] registered an ultrasound examination of tongue movement during swallowing, and was later followed by tongue extraction from ultrasound images [12] and several other research articles [14, 10, 11]. Very recently, Kim *et. al.* [1] used a wireless magnetic tongue tip tracking system for silent speech detection. They however, claim that it is difficult to capture the movements of the tongue tip due to obstruction by the jaw and hyoid bone. Lu *et. al.* [2] attempted joint magnetic calibration and localization for tongue tracking with some promising results, without the implication of computer vision. Multiple other literature articles have been presented in the field of speech tasks [3, 4, 5, 8], multi-modal PC access for people with disabilities [6] and identification of tongue behavior for impairment assessment [7, 9].

A survey of this field reveals that most of the tongue tracking research is based on ultrasound images [11, 10, 14, 13, 12]. Even the 3D models are based on the same concept [15,16]. In this research, we model the same dynamics using computer vision instead of functional MRI or ultrasound.

II. METHODOLOGY

This section is divided into two parts here. We attempt to track the facial features (tongue tip and other features) using deep learning and then with optical flow. The flow will become increasingly clear as we proceed further with the explanations.

A. Facial Feature Based Estimation

As suggested in the assignment instructions, the first recommended step was to manually pin-point some facial features in the face in order to get a 3D visualization of the tongue tip later. The approach is now further illustrated here.

The inputs provided were videos from 3 different orientations of people rolling out their tongues at different instances of time and the desired outputs would be a 3D estimation of the tongue movement. Color/contrast, pixel-level based tracking of facial features is not only tedious and needs to be hard-coded, it also presents challenges such as robustness to environment switching and accuracy in tracking. The conventional approach in modern literature is to train a modified convolutional neural network (CNN) that automatically predicts the facial features present in the current image. We adopt a similar approach in this part of the research.

Instead of manually pin-pointing facial features, we decided to use a bench-marked feature system for generic facial recognition (Fig. 1). Automatic facial landmark detection is a long-standing problem in computer vision, and 300-W Challenge is the first event of its kind organized exclusively to benchmark the efforts in the field. The data-set presented in [17] has been used to train a recently proposed Conditional Local Neural Fields (CLNF) [18] for facial landmark detection and tracking (Fig. 2). CLNF is an instance of a Constrained Local Model (CLM) [19], that uses more advanced patch experts and optimization function. The two main components of CLNF are: Point Distribution Model (PDM) which captures landmark shape variations; patch experts which capture local appearance variations of each landmark. The mathematical foundation of CLNFs is beyond the scope of this research and hence, has not been included in this report.

1) Estimating the Head Pose and Bounding Box Visualization: The model is able to extract head pose (translation and orientation) information in addition to facial landmark detection. We are able to do this, as CLNF internally uses a 3D representation of facial landmarks (visualization given in Fig. 3) and projects them to the image using orthographic camera projection. This allows us to accurately estimate the head pose once the landmarks are detected by solving the PnP problem. For accurate head pose estimation the model needs to be provided with the camera calibration parameters (focal

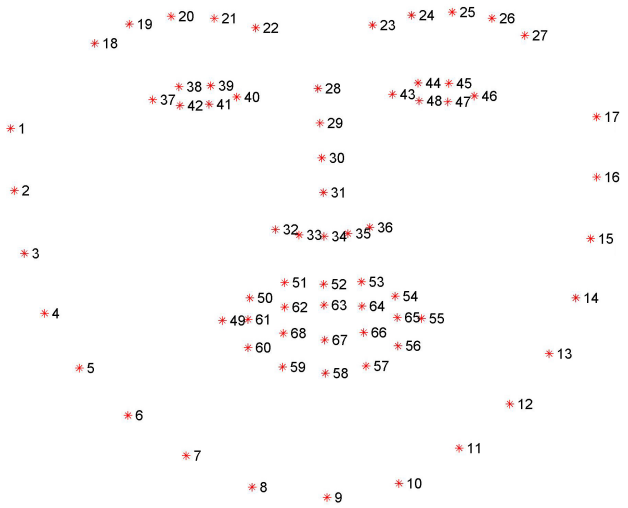

Fig. 1: Locations of the 68 features present in the 300-W dataset

length and principal point). The projected 3D box is a cube estimated from the square box surrounding the face. In the absence of calibration parameters, we programmed the model to use a rough estimate based on image size. This trained model was requested from the research group at Cambridge and was further fine-tuned on the current provided images. Training was intensive and is tedious to carry out on the available medium-powered GPUs.

2) Performance Evaluation: Performance evaluation of this research comprises of evaluating the L2-norm between the estimated points by the tracker and the actual ground truth of the specific frame. Images were first annotated manually with the true locations of the tongue tips as shown in Fig. 7. This annotation is then compared with the predicted location of tip of the tongue.

Note that this evaluation is in 2D and not in 3D. Evaluating in 2D makes more sense as the 3D projection of the tongue tip is estimated directly from the 2D source, so if the 2D prediction is inaccurate, the 3D result would be inaccurate as well. We will be using the error metric to evaluate both our methods mentioned in this report.

It was however observed that interestingly, error metrics were very low for except for the tongue tip point. All points were being consistently tracked in every frame with an error less than 3 pixels on an average. However, the tongue tip was off the charts by 50-100 pixels and upto 200 pixels in worse cases. Since the tongue tip estimation was not accurate from the other estimated facial features, the 3D tracking was also inaccurate while testing. An elaborate explanation of the failure is mentioned later in the discussion section where we compare the better performance of our algorithm than the one suggested in the assignment.

B. Optical Flow Based Estimation

In this approach we estimate the motion of the tongue using optical flow. We first use Lucas-Kanade method, but

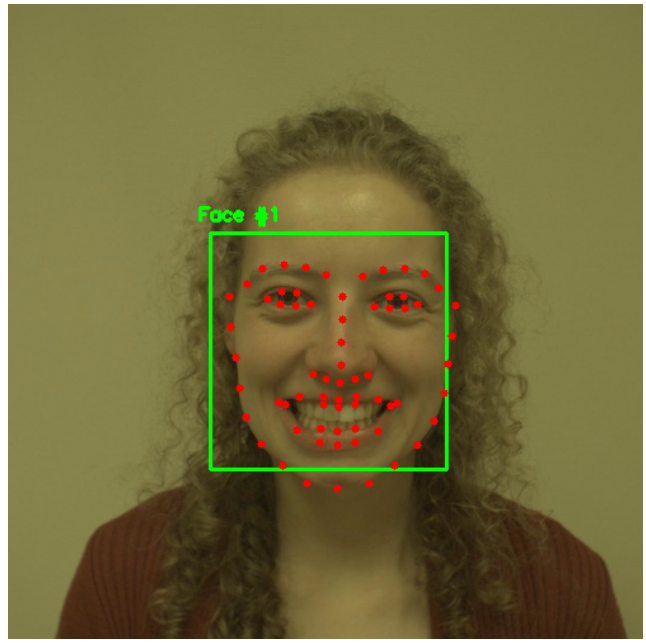

Fig. 2: 2D detection and tracking of facial features in almost real-time using CLNF models. This technique can also be used for designing facial recognition systems.

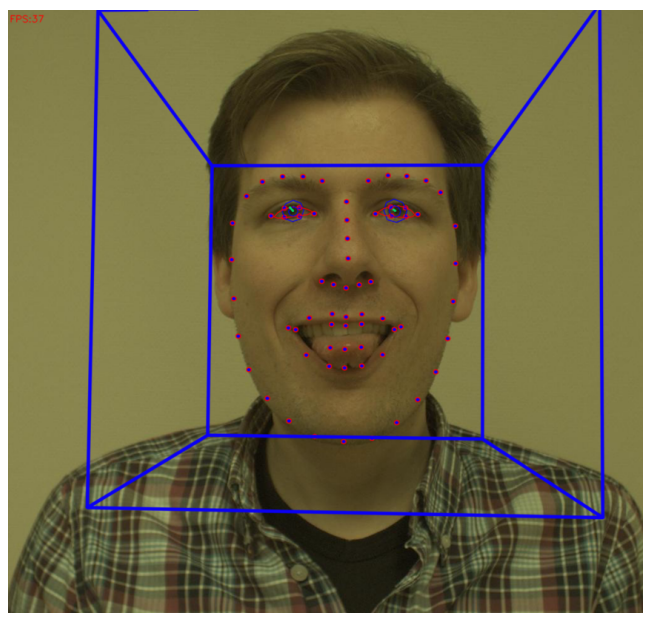

Fig. 3: 3D head pose estimation and feature detection and placement using CLNF models. The origin of the head box is taken at the lower-left corner near the face. The box orientation is simply calculated from the pose of the face estimated by the model. The visualization code has not been added to the appendices considering the article length.

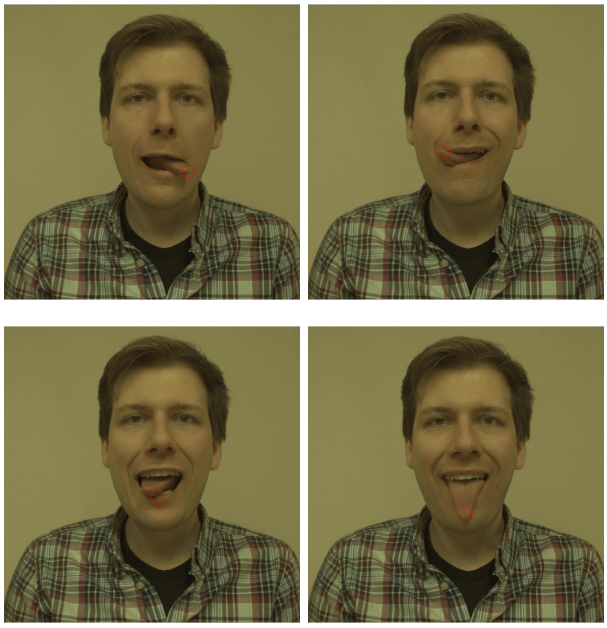

Fig. 4: Tongue Tips Detected and marked with red circles

it computes optical flow for a sparse feature set. However, MATLAB provides another algorithm to find the dense optical flow. It computes the optical flow for all the points in the frame and is based on Gunner Farneback's algorithm mentioned in [20]. Sparse techniques only need to process some pixels from the whole image, dense techniques process all the pixels. Dense techniques are slower but can be more accurate, which was the primary motivation behind using this method.

In order to increase the computational speed the images are first cropped slightly to decrease the evaluation area. Since there is no motion in most of the frame parts, it does not make sense to evaluate dense optical flow for these sections. Once the optical flow in the cropped image is detected, we need to apply a threshold to estimate the location of the maximal flow. This threshold was empirically found to be of magnitude 6. When we filter out the other magnitudes, we get almost the exact location of the tongue and place a marker for further visualization. An indication is shown in the images in Fig. 4.

The next task is to estimate the 3D positioning of the detected tongue tip. The following sub-tasks were identified to complete the the tracking aspect:

1) Points matching between the images from all 3 videos: We use SURF features for this part. SURF stands for seeded-up robust features. As the name suggests, the feature detection technique is more accurate than Harris detection algorithm and is also rotation-invariant, thereby making it one of the most robust algorithms in modern literature. The algorithm is similar to SIFT and has three main components: interest point detection, local neighborhood description and matching. SURF uses square-shaped filters for approximating Gaussian smoothing. The integral image representation is given by

$$S(x,y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(x,y) \tag{1}$$

Blob detection is implemented in SURF with the help of the Hessian matrix to find points of interest. The determinant of this Hessian determines the local change around the current pixel points and features are selected on the maximal value of this determinant. The Hessian matrix is defined in the algorithm as:

$$H(x,y) = \begin{bmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{xy}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix} \tag{2}$$

where L_{xx} is the second order derivative of Gaussian of the image $I(x,y)$ at the point p.

2) Estimate the camera poses of the images relative to each other: Camera pose estimation is a necessary step to be carried in order to reconstruct the 3D points from 2D estimates. We compute relative rotation and translation between camera poses and the camera pose of the current view in the global coordinate system relative to the first view.

```
Result: Pose Estimation
Reconstruct the projection rays from the image points;
marker foreach Projection Ray R do
    foreach 3D Contour c do
        Estimate the nearest point P1 of ray R to a
         point on the contour;
        if n == 1 then
            choose P1 as actual P for the
             correspondance
        else
            compare P1 with P;
            if dist(P1, R) is smaller than dist(P,R) then
                then we choose P1 as the new P
            else
                continue
            end
        end
    end
end
Next, we use (P,R) as the new correspondance set;
Pose estimation with this contour set;
Transform contours and then goto marker
```

Algorithm 1: Camera Pose Estimation

3) Triangulate the initial 3-D world points: This is main part of the 3D tracking of the tongue tip and all previous steps enable us to define a 3D world coordinate system for estimating the movement of the points in the frame with maximal dense optical flow. We compute the 3-D locations of undistorted points matched across multiple images taken into consideration. The triangulation process is explained graphically in the Fig. 5

4) Bundle Adjustment: Further and the last step in the process is to refine camera poses and 3-D points which is done using the classic bundle adjustment technique. Bundle

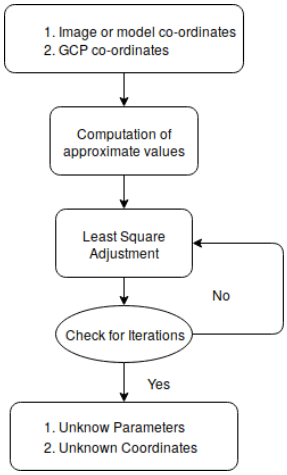

Fig. 5: TRIANGULATION ADJUSTMENT PROCESS

adjustment minimizes the total re-projection error with respect to all 3D point and camera parameters, specifically

$$A = \min_{a_j, b_i} \sum_{i=0}^{n} \sum_{j=0}^{m} v_{i,j} d(Q(a_j, b_i), x_{ij})^2 \tag{3}$$

where a_j is the vector parameter for each camera j, b_i is the vector parameter for each 3D point i, $Q(a_j, b_i)$ indicates the projection of the point i on image j and d is the L2-distance between the points.

The final result after the bundle adjustment is shown in the image below (Fig. 6).

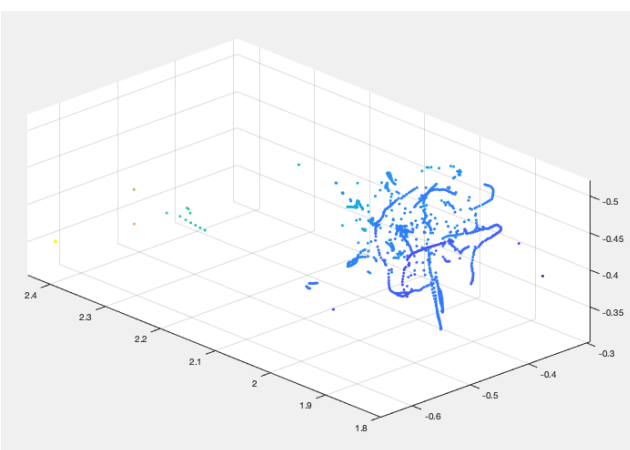

Fig. 6: This plot shows the 3D movement of the tongue in the global coordinate system. The points tracked are in synchronization with the tongue tips detected in 2D. Range of motion of tongue tip for a normal person is indicated for statistical correlation.

III. Results

The overall performance of both the algorithms in terms of L2-errors has been evaluated and presented in Table 1. As it is observable, optical flow based method provides more accurate results as compared to the estimation from facial features. The 3D movement of the tongue is clearly indicated in Fig. 5.

Fig. 7: Data Annotation Samples

TABLE I: Error Comparison of Algorithms for Tongue Tip Tracking

Algorithm	Avg. L2-Norm Error	FPS
Facial Features	123.44	37.3
Optical Flow	15.65	13.5

The optical flow was found to operate at a comparatively lower FPS than the facial feature tracking method. Optical flow based system tends to under-perform in cases of extreme head movements, as there is relatively larger motion vectors in throughout the image frame. However, the facial feature tracking system is more robust to environment noise as it is a learning framework and does have samples with extreme environment noise, multiple faces with different orientations, background variations etc.

IV. Discussion

Although constrained local neural field (CLNF) models are robust enough to track multiple facial features in an image frame, but the detection accuracy depends directly on the training samples. While testing the model for tongue tip tracking, we witnessed multiple *feature slips* in between the successive frames of the videos. These feature slips indicate the loss in tracking of the desired point to be tracked. This could be attributed to the small number of training samples we had for tongue tip detection. The original model was trained on 12,000 images whereas for tuning we had effectively 300 frames. Moreover, in the research article [18], the authors have also mentioned that a possible low contrast or a low

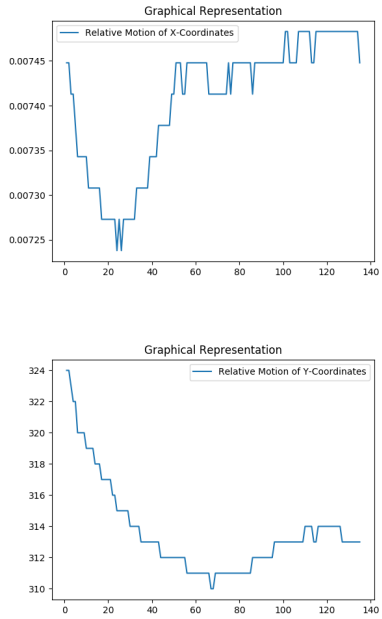

Fig. 8: Relative movement of x and y coordinates

gradient of change in surrounding pixels might also lead to this result. Most of the features in the face are present along the boundaries (eyelashes, face-cut line etc.) along edges basically. This makes it easier for the model to learn the weights. The tongue tip however, does not have that many interest points as others which probably is the reason for the model's under-performance. The accuracy of the tracking can be increased if we add more variations in the fine-tuning dataset, and more data as well.

The approach that uses optical flow to track the tip of the tongue is slower in processing than the CLNF model, but is more accurate. As the method evaluates the motion vectors in the select region, the part of the image with the tongue has the maximal flow as it has movements throughout the videos. The CLNF model does not estimate the flow and is purely feature based, hence cannot detect the moving tip of the tongue as precisely as the optical flow method. However, since this is a more computationally intense process it takes more time.

One interesting observation is that the literature does not really reveal any metrics for tongue tip tracking using computer vision. If however, we treat it the same way as generic object recognition and tracking, we can compare the estimated locations with the actual ground-truths of the tips, which is exactly what we have presented in Table 1.

During the research, as we were observing the problem statement, we realized that it would be relatively significant to have a variation plot of the (x, y) coordinates of the features detected in the mouth region. Considering partial facial movement loss, which might be difficult to observe with the naked eye for a normal person, the relative movement of these detected features would look something like the plots shown in Fig.8 for a specified movement strategy. If the paralyzed person follows the same movement strategy (roll out tongue, go left etc.), the plots would show a difference in the movement of the detected features, and hence an indication of some distorted movement. Based on the offset from the actual variations, relevant exercises could be suggested for repair.

V. Conclusion

The research presented in this report is by no means complete, as the problem statement is open-ended. We have presented two methods to track tongue tips in 3D, with one showing a limited implementation while the other demonstrating a significant improvement over the former one. The problem statement was to track the tip of the tongue in an image frame in 3D which has been presented in the **Methodology Section 2**. A triple camera system is used to measure the 3D positions of the tongue tip. This research could have direct impacts in the medical fields of radiotherapy where tongue mobility is questionable with some patients. The *Range of Motion* of the tongue has been measured for one normal person and this could be used as the ground-truth or the **'should be'** graph for statistical correlation with abnormal movements in patients for better understanding of the medical problems.

There is scope of improvement in the presented research in the sense that tongue tip is sometimes off the tracker. One approach to detect tongue could be to treat it as an object and perform a object detection/segmentation algorithm to get the tongue contour. The tip could then be extracted using the predicted triangular shape and then could tracked in 3D using the approach presented in this research.

Appendix A
Codes for the Research

The fine-tuned model we prepared for this part of the project can be downloaded from here.

```
# USAGE
# python facial_landmarks_video.py --shape-predictor
    shape_predictor_68_face_landmarks_finetuned.dat
from imutils import face_utils
import numpy as np
import argparse
import imutils
import dlib
import cv2

# construct the argument parser and parse the
    arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=
    True,
  help="path to facial landmark predictor")
args = vars(ap.parse_args())

# initialize dlib's face detector (HOG-based) and
    then create
# the facial landmark predictor
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["
    shape_predictor"])
```

```python
# define the arrays for further appending the coordinates
mouth_array_x = []
mouth_array_y = []

cap = cv2.VideoCapture('proefpersoon 2_M.avi')
frame_count = 0
frame_count_arr = []
while(True):
  # Capture image-by-image
  ret, image = cap.read()
  image = imutils.resize(image, width=500)
  # Our operations on the image come here
  #image = imutils.resize(image, width=500)
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# detect faces in the grayscale image
  rects = detector(gray, 1)

  for (i, rect) in enumerate(rects):
  # determine the facial landmarks for the face region, then
  # convert the facial landmark (x, y)-coordinates to a NumPy
  # array
    shape = predictor(gray, rect)
    shape = face_utils.shape_to_np(shape)
    for (x, y) in shape[48:49]:
      mouth_array_x.append(x)
      frame_count = frame_count + 1
      frame_count_arr.append(frame_count)
      mouth_array_y.append(y)

    # convert dlib's rectangle to a OpenCV-style bounding box
    # [i.e., (x, y, w, h)], then draw the face bounding box
    (x, y, w, h) = face_utils.rect_to_bb(rect)
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # show the face number
    cv2.putText(image, "Face #{}".format(i + 1), (x - 10, y - 10),
      cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    # loop over the (x, y)-coordinates for the facial landmarks
    # and draw them on the image
    for (x, y) in shape:
      cv2.circle(image, (x, y), 3, (0, 0, 255), -1)
  cv2.imwrite('image.png',image)
  cv2.imshow('image',image)
  if cv2.waitKey(1) & 0xFF == ord('q'):
    break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()

print(len(frame_count_arr))
print(len(mouth_array_x))

# Plotting the results for estimation

import matplotlib
import matplotlib.pyplot as plt
from scipy.signal import medfilt, find_peaks

y = mouth_array_y
x = mouth_array_x/np.sum(mouth_array_x)
peak_estimates = find_peaks(x)

print (peak_estimates[0])
array_len = len(peak_estimates[0])

fig = plt.figure()
ax = plt.subplot(111)
ax.plot(frame_count_arr,medfilt(x), label='Relative Motion of X-Coordinates')
plt.title('Graphical Representation')
ax.legend()
fig.savefig('plot_x.png')

fig = plt.figure()
ax = plt.subplot(111)
ax.plot(frame_count_arr,medfilt(y), label='Relative Motion of Y-Coordinates')
plt.title('Graphical Representation')
ax.legend()
fig.savefig('plot_y.png')
```

facial_landmarks_video.py

```python
close all
clear variables


opticFlow = opticalFlowFarneback;


r = [367.5 350.5 361 365];  %midtest
%r = [135.5 479.5 367 261]; %ltest
%r= [595.5 431.5 412 302]; %rtest
frameRGB = readFrame(vidReader);
frameRGB = imcrop(frameRGB,r);
frameGray = rgb2gray(frameRGB);
pointTracker = vision.PointTracker('MaxBidirectionalError',1);

cp=1;
rp=1;
initialize(pointTracker,[cp rp],frameGray);
count=1;
while hasFrame(vidReader)
    frameRGB1 = readFrame(vidReader);
    frameRGB = imcrop(frameRGB1,r);
    frameGray = rgb2gray(frameRGB);
    frameGray=imsharpen(frameGray);
    %frameGray = adapthisteq(frameGray,'clipLimit',0.01,'Distribution','rayleigh');
    flow = estimateFlow(opticFlow,frameGray);
    [row,col]=find(flow.Magnitude == max(flow.Magnitude(:)));
    imshow(frameGray)
    if max(flow.Magnitude(:))>=6
        cp=col;
        rp=row;
        release(pointTracker);
        initialize(pointTracker,[cp rp],frameGray)

    else

        [pt,point_validity] = pointTracker(frameGray);
        cp=pt(1);
        rp=pt(2);
    end
    hold on
    plot(cp, rp, 'ro', 'MarkerSize', 30);
```

```
    hold off
    drawnow

    prevc=cp;
    prevr=rp;
    mpoints(count,1)=cp;
    mpoints(count,2)=rp;
    count=count+1;
end
```

tracking_tongue.m

```
clearvars -except cameraParams camPoses mpoints
    lpoints rpoints

figure;
plotCamera(camPoses, 'Size', 0.2);

for i=1:length(mpoints)
points = [lpoints(i,1),lpoints(i,2);mpoints(i,1),
    mpoints(i,2);rpoints(i,1),rpoints(i,2)];
viewIDs = [1 2 3];
tracks(i) = pointTrack(viewIDs,points);
xyzPoints(i,:) = triangulateMultiview(tracks(i),
    camPoses, cameraParams);
end
grid on
% Refine the 3-D world points and camera poses.
 [xyzPoints, camPoses, reprojectionErrors] =
    bundleAdjustment(xyzPoints, ...
          tracks, camPoses, cameraParams);

% Display the dense 3-D world points.
pcshow(xyzPoints, 'VerticalAxis', 'y', '
    VerticalAxisDir', 'down', ...
    'MarkerSize', 45);
hold on
% Specify the viewing volume.
loc1 = camPoses.Location{1};
xlim([loc1(1)-5, loc1(1)+4]);
ylim([loc1(2)-5, loc1(2)+4]);
zlim([loc1(3)-1, loc1(3)+20]);
camorbit(0, -30);

title('Dense Reconstruction');
hold off
```

tracking_in_3d.m

REFERENCES

[1] Kim, Myungjong, Nordine Sebkhi, Beiming Cao, Maysam Ghovanloo, and Jun Wang. "Preliminary Test of a Wireless Magnetic Tongue Tracking System for Silent Speech Interface." In 2018 IEEE Biomedical Circuits and Systems Conference (BioCAS), pp. 1-4. IEEE, 2018.

[2] Lu, Jun, Zhongtao Yang, Klaus Z. Okkelberg, and Maysam Ghovanloo. "Joint magnetic calibration and localization based on expectation maximization for tongue tracking." IEEE Transactions on Biomedical Engineering 65, no. 1 (2018): 52-63.

[3] Sigona, Francesco, Massimo Stella, Antonio Stella, Paolo Bernardini, Barbara Gili Fivela, and Mirko Grimaldi. "Assessing the position tracking reliability of Carstens? AG500 and AG501 electromagnetic articulographs during constrained movements and speech tasks." Speech Communication 104 (2018): 73-88.

[4] Xia, Wenfeng, Simeon J. West, Malcolm C. Finlay, Rosalind Pratt, Sunish Mathews, Jean-Martial Mari, Sebastien Ourselin, Anna L. David, and Adrien E. Desjardins. "Three-dimensional ultrasonic needle tip tracking with a fiber-optic ultrasound receiver." JoVE (Journal of Visualized Experiments) 138 (2018): e57207.

[5] Laporte, Catherine, and Lucie Mnard. "Multi-hypothesis tracking of the tongue surface in ultrasound video recordings of normal and impaired speech." Medical image analysis 44 (2018): 98-114.

[6] Sahadat, M. Nazmus, Arish Alreja, and Maysam Ghovanloo. "Simultaneous multimodal PC access for people with disabilities by integrating head tracking, speech recognition, and tongue motion." IEEE transactions on biomedical circuits and systems 12, no. 1 (2018): 192-201.

[7] Woo, Jonghye, Jerry L. Prince, Maureen Stone, Fangxu Xing, Arnold D. Gomez, Jordan R. Green, Christopher J. Hartnick et al. "A Sparse Non-negative Matrix Factorization Framework for Identifying Functional Units of Tongue Behavior from MRI." IEEE transactions on medical imaging 38, no. 3 (2019): 730-740.

[8] Chen, Weiyi, Dani Byrd, Shrikanth Narayanan, and Krishna S. Nayak. "Intermittently tagged real?time MRI reveals internal tongue motion during speech production." Magnetic resonance in medicine (2019).

[9] Ghovanloo, Maysam, and Jacob Block. "Wireless real-time tongue tracking for speech impairment diagnosis, speech therapy with audiovisual biofeedback, and silent speech interfaces." U.S. Patent 9,911,358, issued March 6, 2018.

[10] Li, Min, Chandra Kambhamettu, and Maureen Stone. "Automatic contour tracking in ultrasound images." Clinical linguistics & phonetics 19, no. 6-7 (2005): 545-554.

[11] Stone, Maureen. "A guide to analysing tongue motion from ultrasound images." Clinical linguistics & phonetics 19, no. 6-7 (2005): 455-501.

[12] Akgul, Yusuf Sinan, Chandra Kambhamettu, and Maureen Stone. "Extraction and tracking of the tongue surface from ultrasound image sequences." In Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231), pp. 298-303. IEEE, 1998.

[13] Stone, Maureen, and Thomas H. Shawker. "An ultrasound examination of tongue movement during swallowing." Dysphagia 1, no. 2 (1986): 78.

[14] Akgul, Yusuf Sinan, Chandra Kambhamettu, and Maureen Stone. "Automatic extraction and tracking of the tongue contours." IEEE Transactions on Medical Imaging 18, no. 10 (1999): 1035-1045.

[15] Chen, Shicheng, Yifeng Zheng, Chengrui Wu, Guorui Sheng, Pierre Roussel, and Bruce Denby. "Direct, Near Real Time Animation of a 3D Tongue Model Using Non-Invasive Ultrasound Images." In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4994-4998. IEEE, 2018.

[16] Woo, Jonghye, Jerry L. Prince, Maureen Stone, Fangxu Xing, Arnold D. Gomez, Jordan R. Green, Christopher J. Hartnick et al. "A Sparse Non-negative Matrix Factorization Framework for Identifying Functional Units of Tongue Behavior from MRI." IEEE transactions on medical imaging 38, no. 3 (2019): 730-740.

[17] Sagonas, Christos, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. "300 faces in-the-wild challenge: The first facial landmark localization challenge." In Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 397-403. 2013.

[18] Baltrusaitis, Tadas, Peter Robinson, and Louis-Philippe Morency. "Constrained local neural fields for robust facial landmark detection in the wild." In Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 354-361. 2013.

[19] Cristinacce, David, and Timothy F. Cootes. "Feature detection and tracking with constrained local models." In Bmvc, vol. 1, no. 2, p. 3. 2006.

[20] Farnebck, Gunnar. "Two-frame motion estimation based on polynomial expansion." In Scandinavian conference on Image analysis, pp. 363-370. Springer, Berlin, Heidelberg, 2003.