Learning and Optimization Final Project

Gilad Atias – 026511956 and David Keisar - 305088759

Tuft Visualization – TuftViz

Abstract

An indispensable part of experimental aerodynamics is "flow visualization", which can provide the overall understanding of the aerodynamical behavior of the vehicle or structure that is being investigated [1]. It is particularly useful for problems where a basic understanding is lacking, or computational solutions are impractical, impossible or inaccurate. Visualization is typically employed on or near to a surface, or in the flow field adjacent to the surface. The former usually involves surface affixed materials (tufts), oil films or so-called "china clay" (titanium dioxide and kerosene that evaporates). The latter employs seed particles, such as smoke, that follow the flow, or exploit density gradient for schlieren methods in heated or high-speed flows. Flow visualization is typically several orders of magnitude cheaper, and radically simpler than well-established quantitative method such as hot wire anemometry, LDA and PIV **שגיאה! מקור ההפניה לא נמצא.**,2].

One of the simplest, cheapest and oldest flow visualization techniques is 'Tuft Visualization' where high aspect ratio (length/diameter) threads are attached to the surface under investigation [3-7]. The method has been applied in wind tunnel experiments for flight tests for at least 90 years [3] and remains a primary tool to this day (see [8] Boeing 757 ecoDemonstrator flight tests). The primary advantages are: simple installation, cheap materials, and easy analysis. The main disadvantages are that the physical presence of tufts can affect the state of the boundary layer and act and act as a "trip" [6].

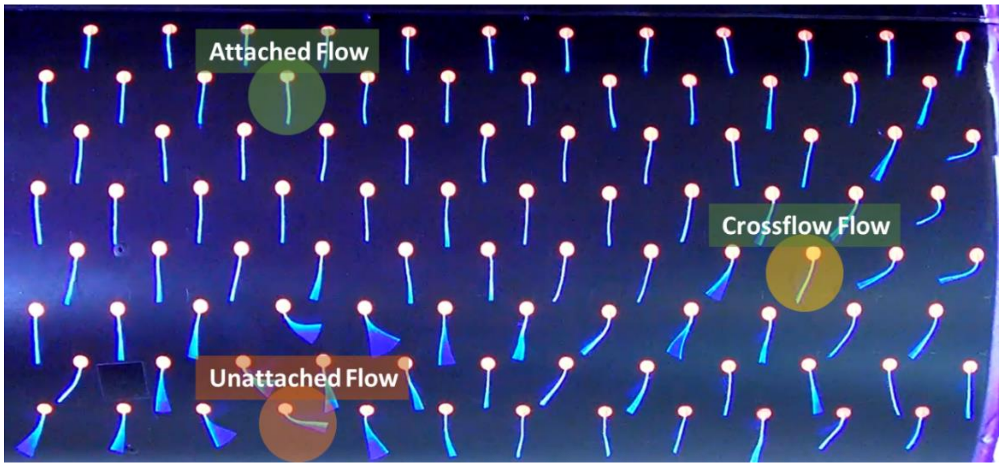


Figure 1 - Snapshot from the Analyzed video whit examples of: Attached, Unattached and Crossflow (explained in the 'Data' Section)

Typically, images or videos are acquired during the experiment or flight and then analyzed manually. <u>The objective of the present research is to develop and Open Source tool that facilitates quantitative analysis unsteady aerodynamic flows meaning analyzing “How good” the flow is over the subjective body</u>. The idea is to use this well-established technique in combination with Machine Learning to develop techniques that less dependency on subjective analysis. The program is called TuftViz and it written in the popular Matlab software environment.

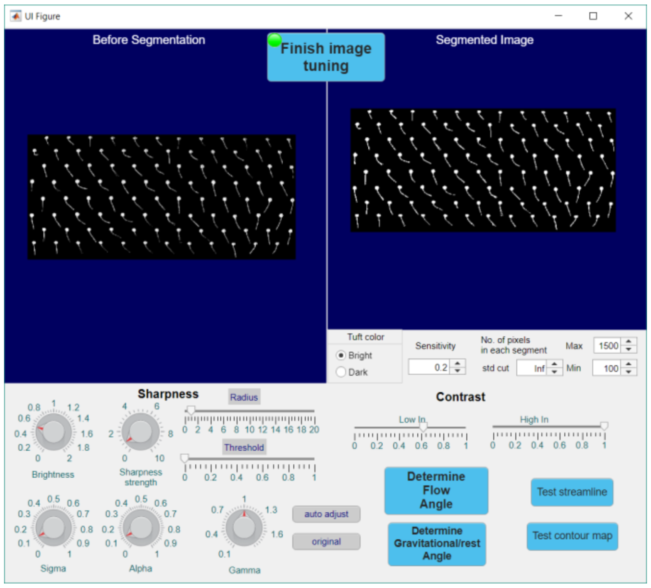
In the sections below, the ML algorithms of the program will be described and initial (on 1 video set) results are shown.

Data

The data of that is presented in this work is based on a video of a moving cylinder in a wind tunnel (see in figure 1) that is also available in the GitHub folder under “christ.mov”.

Preliminary process on of the video includes:

1. Diving the video into individual, but sequentially, frames.
2. Cropping of the ‘Area of Interest’.
3. Black and white color editing (Brightness, contrast, Alpha, Gamma…) (See figure 2)
4. Image segmentation to extract all the “Tufts”.
5. Deciding the “Main wind” orientation.
6. Gridding the Tufts to 2D X-Y plane.
7. Extraction of the features about each ‘Tuft’.



Tuft color: [✓ Bright] Dark; Sensitivity 0.2; std cut Inf; No. of pixels in each segment: Max 1500, Min 100

Figure 2 - Preliminary Process Tool

For each tuft we are extracting the following **features**:

1. Wind related angle
2. Length – How long is it compare to other 'Tufts'.
3. Straightness – How straight is the 'Tuft'
4. Edge related angle – The average, relative orientation of the edge of the 'Tuft' compare to the direction of the wind.
5. Physical location – The location of the 'Tuft' in the X-Y plane.
6. 4 neighbors – The 4 closest neighbors that comes "before" the 'Tuft', considering the direction of the wind.

Those features will be the key to consider our tagging of every tuft separately and understand whether it is attached (considering its surrounding) or not to the general stream of the wind.

The Tagged Data: We will use about 50 tagged frames, each consist of about 96 tufts with 3 kind of tags (see Figure 3): Attached tuft, cross wind (when part of a group that directed to the same direction but it's different from the wind angle) or Unattached tuft (see Figure 1) – Total of about 5,000 Tagged 'Tufts'.

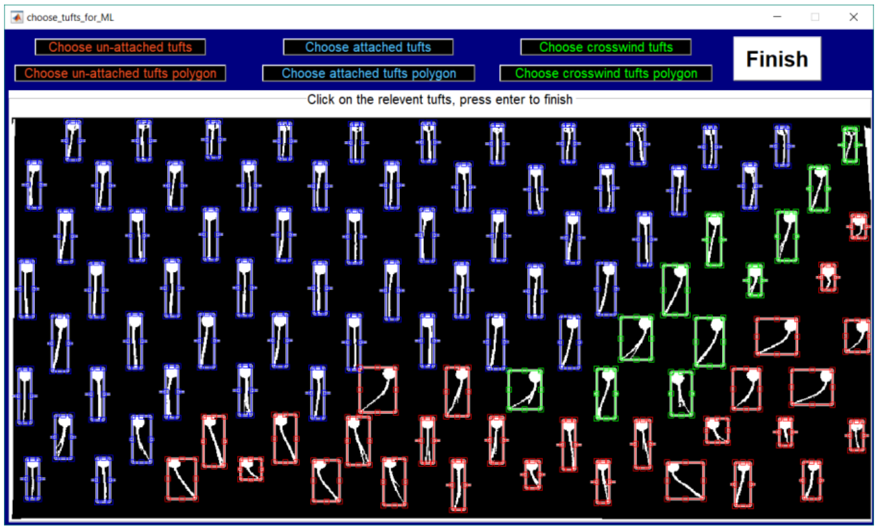

choose_tufts_for_ML
Choose un-attached tufts
Choose attached tufts
Choose crosswind tufts
Finish
Choose un-attached tufts polygon
Choose attached tufts polygon
Choose crosswind tufts polygon
Click on the relevent tufts, press enter to finish

Figure 3 - Tagging tool created for this program

Data Processing

Algorithm

The base algorithm implemented is a ‘Structured Perceptron’. The structure of the model is a frame consist of about 96 tufts, in our examples the wind direction is from the upper side of the frame (wind angle 270). Those tufts stand in rows (8 rows of 12 tufts) and we describe the dependency of certain tuft by choosing the closest tuft from the opposite direction of the wind (hence tufts from lower rows depends on tufts from higher rows).

Structured perceptron

We chose to implement the structured perceptron in tow main steps:

1. Creating frame topology – by using topology search we determined the order of execution of the calculations over the sequence.
2. ‘Viterbi’ algorithm - Given a frame X (consist of n tufts), a tag result Y and a frame topology, we are using ‘Viterbi’ algorithm to run evaluation of the best prediction tags for the frame – we then use the perceptron update step to improve the weight vector.

Model theory

The structured perceptron we use try to divide the data into 3 classes {0 – unattached, 0.5 – cross wind, 1 – attached}.

The goal is to discover a linear divider (represented as weight vector - W), that for every frame, calculate his corresponding feature vector, and create a linear divider to 3 groups.

We define $i\vdash$ as the tuft neighbor of the tuft i (the tuft that i depends on). We then create a local feature vector $\Phi(y_{i\vdash}, y_i, x)$ for every x_i, so the global feature vector is $\sum_i \emptyset(y_{i\vdash}, y_i, x)$.

The goal of the algorithm is for every frame to find a tag y’ out of all possible tags, that maximize $\sum_i W * \emptyset(y_{i\vdash}, y_i, x)$, and then by using perceptron learning step to update W to be $W + \sum_i \emptyset(y_{i\vdash}, y_i, x) - \sum_i \emptyset(y'_{i\vdash}, y'_i, x)$.

To find the maximum we use ‘Viterbi’ algorithm.

Features and thresholds

We run the model for several different thresholds over the following features:

1. cosine similarity of a tuft and his dependent tuft –cosine of the difference between the wind related angle of both Tufts. The greater the number, the bigger probability the 'Tufts' will have the same tag (by logic).
2. Wind related angle – The average, relative orientation of the 'Tuft' compare to the direction of the wind.
3. Length – How long is it compare to other 'Tufts', normalized by the biggest 'Tuft'.
4. Straightness – How straight is the 'Tuft' compare the "Straightest" 'Tuft'.
5. Edge related angle – The average, relative orientation of the edge of the 'Tuft' compare to the direction of the wind.
6. For all features, as probably noticed, the data runs between 0 to 1. When the feature is closer to 1, the probability of it to be 'Attached' is greater.

After various tries with different parameters we set the best thresholds to be

1 = 0.92 (above means they are similar)

2 = 0.77 (above means it is wind related)

3 = 0.79 (above means it appears straight)

4 = 0.93 (above means the edge is straight)

5 = 0.95 (above means the tuft is long)

In addition, we tested how many iterations we had to make until the algorithm will converge.

Experiment

The algorithm runs over a data set of 50 frames with cross validation technique (separation to 4 sets of the all data and using 3/4 of the portions as training set and 1/4 as test – 4 times).

Extensions

Advance Part

We then decided to give each neighbor, out of the four, his own independent weight vector and then make the prediction by averaging through the whole 4 neighbors vectors we learned.

By using the same algorithm 4 times every time on another neighbors (who are less important – hence a neighbor that is more far from the tuft then the others). We then calculate 4 different weight vectors and predict a new observation with a majority decision). The influence of 4 different neighbor means that for every ‘Tuft’ we can make sure that the area surrounds it and the behavior of few more tufts around gave better results on the tuft.

The calculations for every neighbor kind were made independently. The results were combined the to create a single prediction. The results were improved as we described later.

Creative Part

In the last two parts we tried to solve the problem by using structured perceptron algorithms, that are part of Supervised ML. In this part of the course, we approach our problem in a bit different way then we learned in the course, by Using structured Unsupervised ML algorithm called ‘Normalized Cuts’.

Given a set of features, we construct a weighted graph, $G=f(V,E,w)$. If A and B are two subsets of vertices, we can define:

$$w(M,N)=\sum_{i\in M,j\in N} w_{ij}$$

$$\text{ncut}(M,N)=\frac{w(M,N)}{w(M,V)}+\frac{w(M,N)}{w(N,V)},\text{nassoc}(M,N)=\frac{w(M,M)}{w(M,V)}+\frac{w(N,N)}{w(N,V)}$$

For any cut (M,N) in G ‘ncut’ measures the similarity between different parts, and ‘nassoc’ measures the total similarity of vertices in the same part. To find the optimum (M^*,N^*) we will look for the subset that minimize ‘ncut’. Using spectral techniques, we can find in polynomial time the cut (M^*,N^*).

The partitional algorithm (for solving in polynomial time) is:

(1) computing weight on each edge and then placing the data into W and D.
(2) Solving $(D-W)y=\lambda Dy$ for eigen vectors (λ) with the smallest eigenvalues.
(3) Using the eigen vector corresponding to the second smallest eigenvalue to bipartition the graph into two groups.
(4) Recursively repartition the segmented parts if necessary.

To learn about our data using ‘Normalized Cuts’ we:

1. Made a main grid of the averaged location of the Tufts
2. Made a 3D Table of each frame containing: on X-Y plane– the grid and on the Z dim – the normalized information about its behavior.
3. Using PCA/weight vector we reduced the vectors in the Z dim- to a 1-member vector (scalar).
4. Using a scalar named: ‘labelDistanceFactor’ we decided “the Distance” of the Z dim relative to the X-Y dim.
5. Using ‘Normalized Cuts’ we labeled the 3D matrix to ‘noMaxCluster’ subsets in the X-Y plane and used the algorithm again to retrieve a 2D ‘Normalized Cuts’ subsets (‘noOfSuperPixels’ subsets) (The first time only reduced it from 3D to 2D).
6. We then averaged the Z-dim score of each subset and gave teach tuft in the subset that score.
7. By using the data, we labeled before, we optimized the Z-dim scores that separate between guessing the subset is Attached, Unattached or Crossflow by choosing ‘Low’ and ‘High’ thresholds that divide the sets into 3 groups (seen in Figure 4).

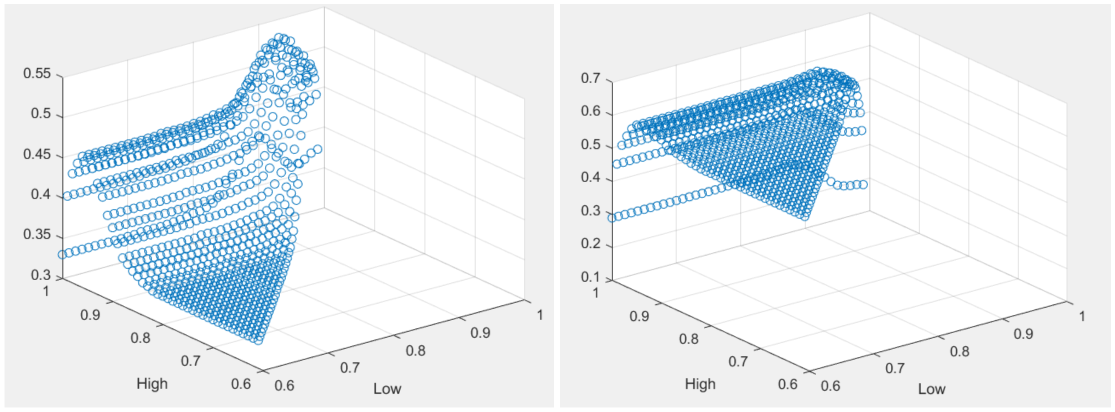


Figure 4 Graphs showing the effect of parameter selection on the avraged (left) and Total (right) accurcy of the algorithm for a randon set of features

Results and a short discussion

Basic algorithm

The algorithm results show accuracy of up to 69% using the structured perceptron, the selected parameters and considering the most significant neighbor.

The outcome for relatively ordered frames (around 70% of the frame true tag is attached) was higher (between 75% - 80%), while for more disordered frames, lower results were achieved (around 50%).

Advance Part

As we predicted, a bit contrary to logic, the averaged algorithm result improved by 5% to 73% accuracy on the test set.

A jump of 5% in the results is a big leap, and it seems a bit unusual that using more structures, but that have less influence on each 'Tuft', give much better result. But, our prior course knowledge taught us that usually averaging structure perceptron weight vectors will lead to better result, subsequently its possible that averaging different structures ('Tufts' Topologies) will lead to a better result.

Creative Part

By using the Unsupervised algorithm, we achieved accuracy of up to ~64.5%.

In this part we also checked the influence of the number of subsets, the "Distance" of Z-dim and influence of PCA/weight vector which, at the beginning, were picked randomly, only by our guts. In addition, we calculated the "Total accuracy" of the algorithm and the "averaged accuracy" – meaning the accuracy of guessing each kind of Tuft – Attached, Unattached and Crossflow.
The results of the feature selection of the algorithm are showed in the table (it only shows the creative part results, and not others, to not make this section "heavy"):

	PCA/weight vector	Features			Best Accuracy	
		Z-dim distance	'noMaxCluster' subsets	'noOfSuperPixels' subsets	"Total accuracy"	"averaged accuracy"
1	avrage feature	12	15	10	58.1%	52.3%
	PCA				57.8%	48.0%
	Edge related angle				62.0%	54.4%
2	Edge related angle	12	10	8	63.1%	53.8%
			15	10	62.0%	54.4%
			25	20	64.6%	56.5%
			90	90	62.0%	49.4%
3	Edge related angle	1	15	10	62.2%	54.2%
		3			62.4%	54.6%
		12			62.0%	54.4%

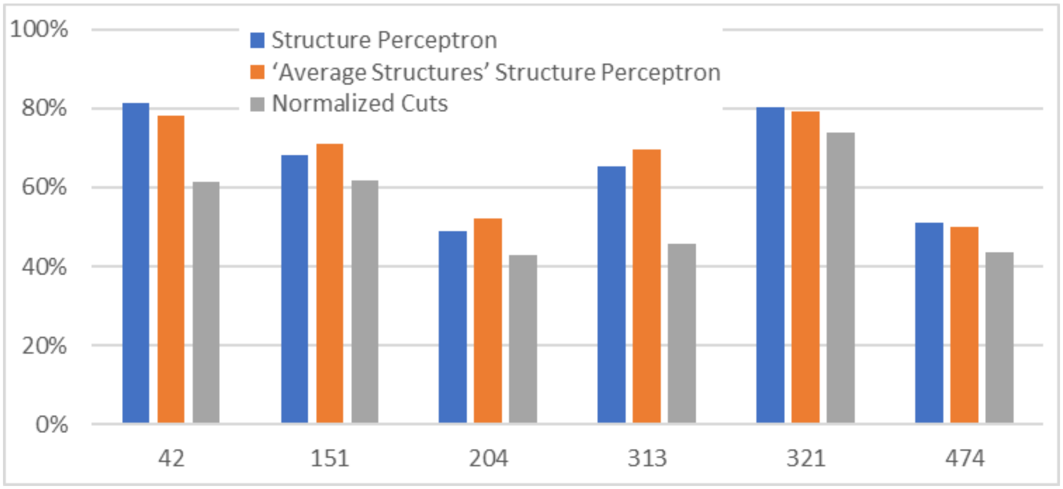
Table 1 - feature selection of the Unsupervised Normalized Cuts' algorithm

As seen in the table, using only the 'Edge related angle' feature, we achieved better results than averaging the feature vector or using Principal Components Analysis. In addition, it seems that having 1/4-1/3 subsets from the total nodes, will give a bit better result. Contrary to our guess, the distance of the Z-dim doesn't change the results, as make sense from the algorithm purpose.

In addition, in this part, we can notice the difference between the total accuracy of the algorithm and the averaged accuracy, meaning the average of the accuracy of each state. This is happening due to a higher number of 'Tufts' attached (~59%). Meaning, we will expect that for disordered frame, the algorithm will have an accuracy ~55% (as face up to with tests).

Comparison of the results

As seen in the graph, we compared the result for several frames with each algorithm:



Graph 1 - accuracy of each algorithm for different frames

We notice several things:

1. The advanced algorithm is not always “superior” to the ‘Basic’ algorithm.
2. Normalized cuts always achieve a worst result than the perceptron.
3. For more order frames, all the frames achieved better results and worse for ‘less ordered’ ones – this may be due to problems in extracting right data from the image.

Conclusion

In this project we wrote and applied three “Structure Machine Learning” algorithms in order to analysis the behavior of a wind over a cylinder in a wind tunnel. We used Structure Perceptron, ‘Average Structures’ Structure Perceptron and Normalized Cuts algorithms that achieved 69%, 74% and 64.5% accuracy (in that order).

The results and the process brought us the following conclusions:

1. The ‘Structured Perceptron’ algorithm provided superior results comparing to the ‘Normalized Cuts’ algorithm. Although a noticeable disadvantage in using Supervised Learning over Unsupervised is the need to label a lot of data.
2. Using less but more ‘Important’ features can achieve better results especially in the Unsupervised algorithms, due to a misleading information it brings. This issue is usually suppressed in the Supervised algorithms by using the weight vector.
3. Lack of diversity in the images lead to more “un-useful” learning – meaning, learning frames similar to others, as happens in a short video, led to a less accurate algorithm compare to a similar size of diverse set.
4. As mentioned in the abstract, it is hard to tag each tuft accurately only by looking at each frame separately. Due to this problem, the tagging itself suffered from low accuracy (we will guess 85-95% accuracy). Meaning that the max accuracy of an optimum algorithm must be much lower than 100%, and probably many mistakes in the learning process happened due to this issue.
5. Another problem can be due to wrong data extracted about the features – as explained I the section above.

References

[1] M. Van Dyke, "An album of fluid motion," Vol. 176. Parabolic Press, Stanford, 1982.

[2] W. Merzkirch, “Techniques of flow visualization,” tech. rep., No. AGARD-AG-302. (France), 1987.

[3] F. Haslam, “Woolen Tufts: A direct method of visually discriminating between steady and turbulent flow over the wing surfaces of aircraft in flight’,” Rep.Mem. Aeronaut. Res. Council, vol. 1209, 1928.

[4] M. N. Gough and E. Johnson, “Methods of visually determining the air flow around airplanes,” National Advisory Committee for Aeronautics, 1932.

[5] J. P. Crowder, “Flow visualization in flight testing,” in 5th Biannual Flight Test Conference,
1990.

[6] J. P. Crowder, “Fluorescent mini-tufts for non-intrusive flow visualization,” McDonnell
Douglas Report, vol. 1600, no. S57374, 1977.

[7] J. P. Crowder, “Add fluorescent minitufts to the aerodynamicist’s bag of tricks,” Astronautics Aeronautics, vol. 18, pp. 54–56, 1980.

[8] E.A. Whalen, A. Shmilovich, M. Spoor, J. Tran, P. Vijgen, J.C. Lin, M. Andino, "Flight Test of an Active Flow Control Enhanced Vertical Tail," AIAA Journal, 2018, Vol.56, pp. 3393-3398.