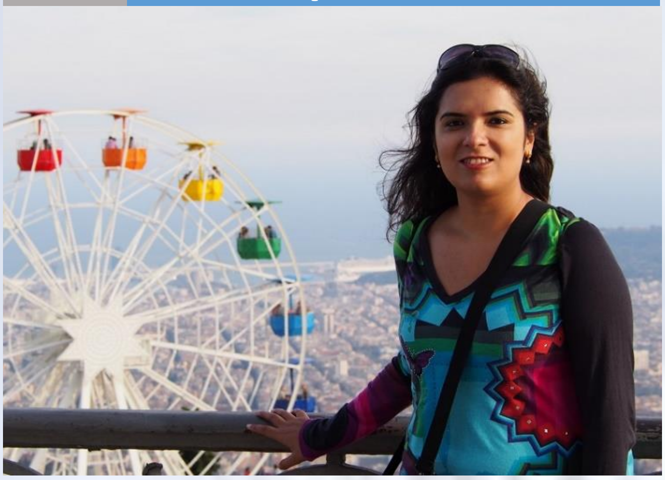
# Computer Vision News

The magazine of the algorithm community

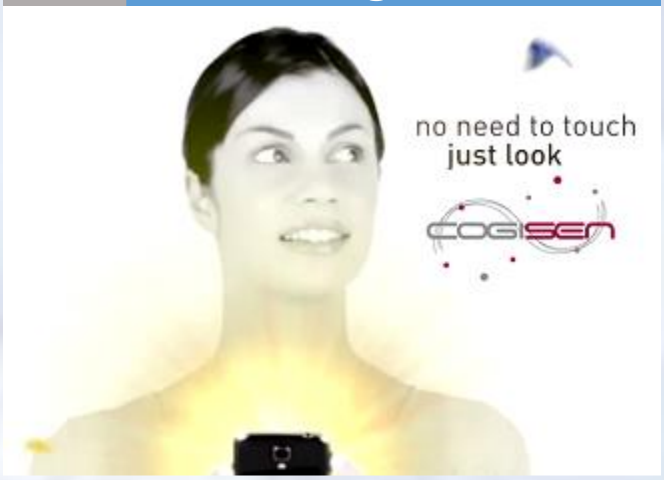



## September 2016


A publication by
RSIP VISION



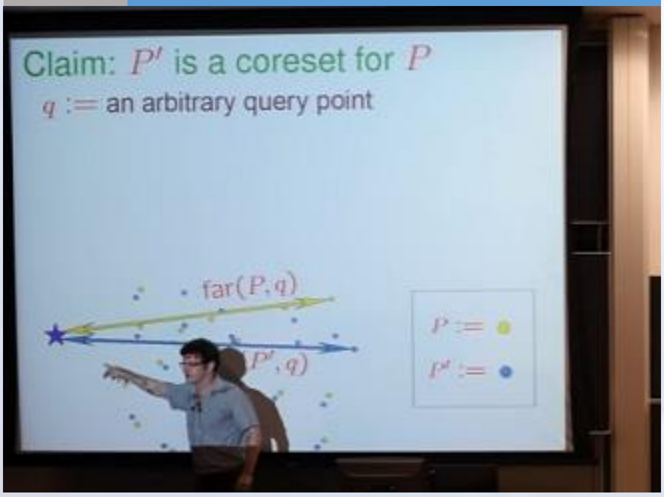



no need to touch
just look








Claim: P' is a coreset for P
q := an arbitrary query point
far(P,q)




Mark your calendars!
ICIP 2016
IEEE
September 25-28, 2016 • Phoenix, Arizona, USA
2016.ieeeicip.org
IEEE Signal Processing Society

**Computer Vision News**
Editor: **Ralph Anzarouth**
Publisher: **RSIP Vision**

Click to contact us

Click to give us feedback

Click for free subscription



Dear reader,

We invited several scientists to share their views with us in this September issue of **Computer Vision News**. First and foremost, we keep interviewing **Women in Computer Vision** with **Dr. Sophia Bano of the University of Dundee**. It is so inspirational to see women scientists showing so much passion in what they do! This is true also about **Luisa M. Zintgraf**, who spoke to us with much verve about the challenge of **Decoding Brain Signals** organized by **Microsoft** earlier this year. She offered her precious insights about the competition.

The same strength of motivation animates **Dr. Lina J. Karam**, the General Chair of **ICIP 2016 (International Conference on Image Processing)**: when you read what she told us, you will understand why we chose ICIP 2016 as our **Event of the Month** of September.

Of course, our admiration for women scientists should not make us forget about men's contribution to progress in our field: we interviewed **Dr. Dan Feldman**, who manages the **Robotics and Big Data Lab of the University of Haifa**. He told us about the **coresets**, which he chose as his main technique and which enable the use of smaller sets of data, without compromising the quality of the output.

The **application of the month** for September comes from **Cogisen**, a start-up founded by **Christiaan Erik Rijnders**: building on computational neuroscience models and on his own experience as an engineer for **Ferrari**, Rijnders developed a new image processing platform with powerful **gaze tracking** capabilities. Don't miss his memories with **Michael Schumacher**!

Read also the other sections: **Spotlight News, Project Management, Tools and Tricks and the Research of the month**. I am sure that you will love this magazine: please keep sharing it with friends and colleagues.

**Enjoy the reading!**

**Ralph Anzarouth**
Marketing Manager, RSIP Vision
Editor, Computer Vision News

# Sophia Bano

We continue our series of interviews with women in computer vision. This new section, which we started with the CVPR Daily at CVPR 2016, hopes to help mitigate the severe gender imbalance in the computer vision community by getting to know better some remarkable female scientists and their career paths. We think that some of what they did might serve as an example for other young females who wish to enter this field. This month, we interview **Dr. Sophia Bano**, who is currently working as a postdoctoral researcher with Professor Stephen McKenna and Professor Annalu Waller at the **University of Dundee in Scotland**.



**Computer Vision News:** *Where were you born, Sophia?*

**Sophia Bano:** I was born in **Islamabad, Pakistan.**

**CVN:** *How was it for a technically oriented young girl to grow up in Islamabad?*

**Sophia:** In Islamabad, education of women is encouraged. In my case, my family was very supportive. They were always encouraging me and my siblings to pursue higher education.

**CVN:** *When did you discover that you had a passion for technology?*

*"The idea of discovering something new really excited me"*

**Sophia:** It was probably because my father was working as a scientist. When I was in high school, I was more interested in engineering subjects. For that reason, I wanted to apply to an engineering university rather than follow medical studies.

At the time I entered engineering college, I didn't really know the differences between electrical engineering and computer engineering, for example. **I just knew that I wanted to be an engineer**. Eventually, I found my tract, and I was mostly interested in things relating to programming. The idea of discovering something new really excited me.

**CVN:** *Why didn't you continue that path in your home country?*

**Sophia:** I did really well in my home country, where I completed my Bachelors and Masters education. Then I got lucky and secured a position with **Erasmus Mundus** for a **Masters scholarship in Vision and Robotics**. I wasn't sure how an international Masters would differ from a Masters in Pakistan, but I wanted to take this opportunity abroad.

Actually, I discovered in my experience that Pakistan is still a bit behind in research and technology compared to the rest of Europe. I can gain more knowledge working here than working in Pakistan.

*"My future plan is to return to my country after gaining some experience. Then I can make a contribution for my country"*

**CVN:** *So you decided to stay in Europe?*

**Sophia:** My future plan is to return to my country after gaining some experience. Then I can make a contribution for my country.

**CVN:** *I know that you went to many other places before Scotland. What did you find interesting there?*

**Sophia:** Yes - I would say that I was lucky in the sense that I got the positions for my Masters and for my PhD. My positions were from Erasmus Mundus. What that entails is that you have to travel while you are studying. Or study while you are traveling... *[she laughs]*

Starting with my initial experience with the Erasmus Mundus Masters program, I learned a lot about different cultures in Europe. My program included international students from 15 different countries so I got to know a lot about people living in different places in the world. I would say that people in general are really friendly. They are always helping, no matter from which background or from which country they come.

What I found really helpful was that we formed communities. For example, **I really liked living in Spain** because they have very strong family ties. Their **family culture** is very similar to what I knew in Pakistan. Also, they are really friendly and encouraging people.

**CVN:** *It seems that during the first years of your career you had very positive experiences with people around you. Did you meet with any difficulties being a woman in this field?*

**Sophia:** I wouldn't say difficulties, probably because I never felt that I am any different than a man. For example, in my current position, most of the time I am probably the only woman in my group or during group meetings. Then it depends on me how I look at others. I know I can work just as well as a man. I can compete with others on the same level. If I think that I can do my job very well, then others will see that I am capable. I never felt that I am different probably because **I am very motivated** or maybe because **I always came across good colleagues and friends**.

Also it's a common trend that usually in engineering careers, there are less females than males. If you check in the academia or in the industry, you will see more males than females. Most of the time, it's a lower proportion of women to men.

**CVN:** *Why did you choose Scotland?*

**Sophia:** Because it's beautiful! *[she laughs again]* Personally, I like to travel for my career. I like to explore and live in different places. I had lived in **Edinburgh**, and I already knew that the Scottish people are quite nice and also friendly. I thought that I should try this position because of the good gesture of the **Scottish people**. It's also because I was really interested in the project which I am part of right now.

> ***"Building voice output communication devices for people with speaking difficulties"***

**CVN:** *Let's talk about it. What is the subject of your work?*

**Sophia:** I am working on a project called **ACE-LP**, which stands for **Augmenting Communication using Environmental Data to drive Language Prediction**. This project is about building voice output communication devices for **people with speaking difficulties**. Those who cannot speak very well use some kind of device for communication. Since this product doesn't use any environmental feedback such as sensors, cameras, or GPS location sensors, my role is to incorporate these sensors and to extract useful information from video, for example. Then I view this information for predicting language. This language prediction can help make communication easier for people with these kinds of difficulties.

**CVN:** *Is it only in English or can it work also in other languages?*

**Sophia:** At the moment, we are looking at English.

**CVN:** *Do you think that you've come a long way since you were a little girl in Islamabad?*

**Sophia:** I would say yes.

> *"If I think that I can do my job very well, then others will see that I am capable. I never felt that I am different"*

**CVN:** *Do you feel privileged?*

**Sophia:** In some sense yes, because not many people get this opportunity. Not many women reach this stage. For a Pakistani woman, you probably have to end your career at a certain stage. In my case, my family was really encouraging, and I kept pushing forward. **I am what I am because of my family.**

**CVN:** *What is your ambition? What would you like to achieve in the future?*

**Sophia:** I really want to make a difference and to work on some computer vision technologies which can impact the future society. For instance, with this project, if we can make it work, it will have a huge contribution for people with language difficulties. **In my case, I really want to get something done which makes significant changes in the society**.

**CVN:** *What chances do you give yourself to achieve that objective?*

**Sophia:** I am really ambitious... *[she laughs again and then smiles]* When I plan something, most of the times I achieve it!



# Cogisen



Are our eyes going to be the most powerful of our organs? That might happen with **gaze tracking**, one of the hottest issues in computer vision. A very much talked about start-up in the field is Italian-based and European-funded **Cogisen**, headed by **Christiaan Erik Rijnders**. Trained as an aerospace engineer, Rijnders later worked for the **Ferrari Formula 1 team**, when they were dominating with **Michael Schumacher**. With Ferrari, Rijnders developed simulations and simulators. He was also a technical reference at the circuits.

In those years, simulations were very different from today, when a simulation engineer opens up a software package and puts in parameters. Back then, they had to develop the simulations and the simulators by themselves. It was nonlinear and transient cross physics and Rijnders explains how they were very difficult to model.

He came to the conclusion that engineering needed to make a change when it came to what could be modeled. At a certain point, he started modeling algorithms which he considered necessary, taking the ideas from computational neuroscience, while the actual model itself is inspired by biology. He developed it trying to model what our **visual cortex** impressively does, when it preprocesses before our consciousness, solving signal to noise problems and recognizing single photons.

*Cogisen CEO Christiaan Erik Rijnders controlling a smartphone with gaze tracking from a 2m distance*

Rijnders found that on the level of neurons, our brain develops models like the chair and table in the first years of our lives: our consciousness doesn't process all of the single photons. Instead, our consciousness stores these models for faster recognition and processes models that have been recognized by our visual cortex. Then we make a conscious model out of this. That's the inspiration behind his idea.

He soon realized that computer vision would need **much more refined algorithms**. As an engineer, his philosophy is that the underlying data has to be as good as possible. Not only that, but you should give the engineers a model that they can understand. That is how Cogisen is becoming very relevant in areas like **machine learning and deep learning**, which are becoming very effective but are still limited by the underlying data.

***"The movements of the iris don't even register in pixels if one is standing one meter from the camera"***

Here comes the idea of gaze tracking, one of Cogisen's applications in computer vision: you can't just provide a deep learning model of many faces with their eyes moving. First of all, you must have a relevant model of the eyes moving. Secondly, you have to be able to do things sparsely and in a non-linear way. You have to consider the infinite number of light conditions, the faces, and the points of view to the camera.

Then you consider the fact that the pupil and iris movements are subpixel. The movements of the iris don't even register in pixels if one is standing one meter from the camera. Think about the area on the inside of the eye, the outside of the eye and the irises. You can't measure these movements by counting the pixels unless under perfect conditions. **Instead of counting pixels, Cogisen makes the underlying data much better and creates a model from it**.

With this work, difficult nonlinear problems in industry can find a solution: for example, the detection of distant objects for autonomous vehicles with very few pixels. This could also include gaze tracking interfaces for people standing meters away from the video as well as adaptive video compression where it is difficult to get a good model of visual saliency.

Currently, the most advanced solutions in image processing need heavy GPU use. If you look at what they are doing, such as the cloud image processing or image processing for autonomous vehicles, it is extremely advanced and computationally intensive. This cannot be the solution for the real needs of the industry which is **Internet of Things**. We will have billions of devices for which such a huge amount of GPU usage is out of question.

That really goes back to **the core of image processing**. An image is defined by positional data which are the indexes in the image or, in other words, counting pixels. The image itself is defined by many sinusoidal contrasts. Put lots of sinusoidal contrasts together, and you get your complex image of clouds and mountains.

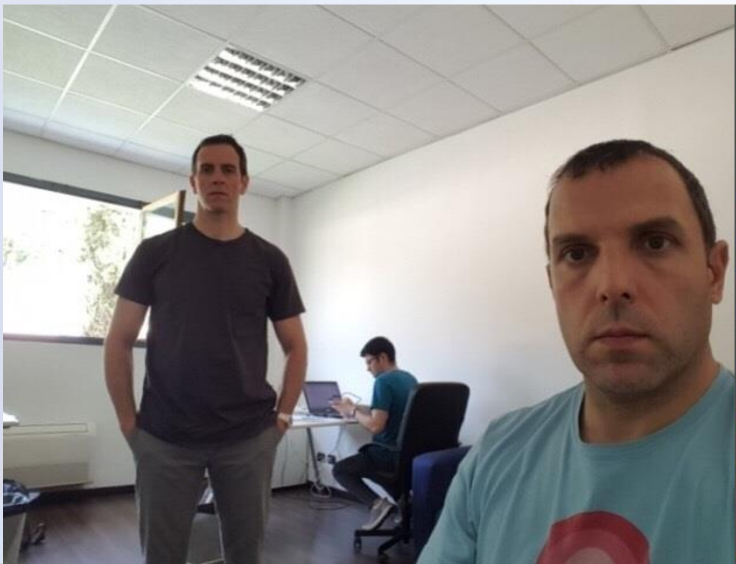
That comes from the theory books saying that the sinusoidal contrasts in the frequency domain become an index position. What people always forget is that these theory books are showing you magnitude. With magnitude, you've lost the information to go back to the spatial domain.

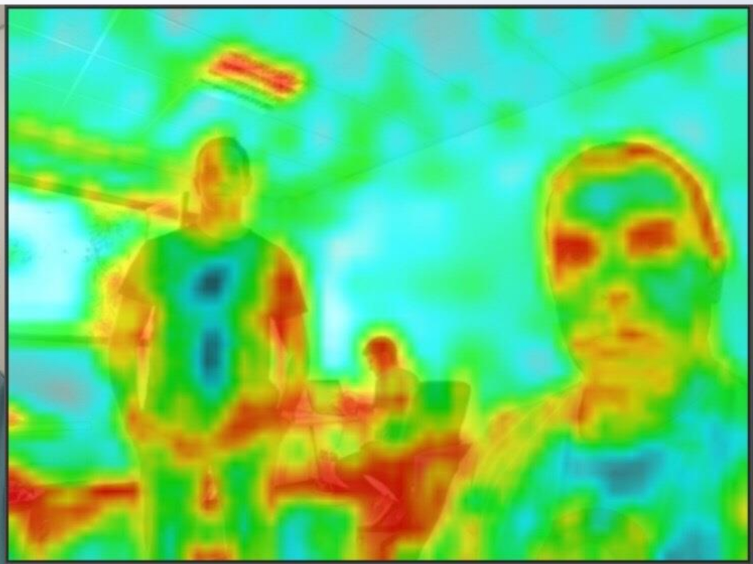
What actually is happening is the opposite of the spatial domain. You have the positional data and the movement data in the frequency domain defined as sinusoidal contrasts. You need those two things. You need the positional data with the sinusoidal contrasts. You also need your indexes or the image itself to go back again to the spatial domain.

If you have the ability to recognize positional data like shapes and movements in the frequency domain, you have something which is much faster and more robust. It manages to model much more complex things because **you're not counting pixels anymore**. You're really looking at changes in information.

Think of it from the point of view of **gaze tracking**. Imagine having a gaze tracking solution in which you are not counting pixels around the irises and around the edges of the eyes. You're simply looking at the change of information as the eyes move. Using the change of information in the movement data in the frequency domain, you have something much more solid and robust which needs less data. Plus you have something which is a model.

That is what is able to capture much more difficult processes in a model. Therefore, **Cogisen**'s vision is to improve the underlying data for industry, allowing all of the deep learning and machine learning methods to reach a better performance. Not only that, if you work like this in the frequency domain, there is much less data required to do learning and training. You won't have this incredible need for GPU acceleration. In this way, you also have the solution for the **Internet of Things**.





We asked Rijnders about the algorithm techniques which are most suited to solve these problems. In his view, you cannot use the traditional methods of algorithms such as DCT or FFT. There are lots of problems with aliasing, periodic data repeating, how to normalize this data, how to make the high frequency data relevant, or how to avoid an input that is a power of 2. To solve all of these things, they converted differently to the frequency domain.

Rijnders and his team worked for two years to create their tool chain and platform. Last summer they started to go often to **the Silicon Valley** and ask the industry what their problems were. Obviously, they have a lot of things that need to be solved and reportedly reception was very good. They clearly want to specialize in quickly creating new solutions for the industry (during the initial phase of Cogisen, they also worked with a Formula 1 racing team): basically, **being asked to invent technology and quickly come back with solutions**.

Now they concentrate mostly on customers in Silicon Valley, who ask them to create technology in an accelerated way using their platform to solve problems they have. Up until now, Rijnders claims there hasn't really been a problem that they haven't been able to solve for Silicon Valley.

Now that they have the structure to do an accelerated type of prototyping for the invention of new technologies, their next step is to provide a product technology, not just an accelerated prototyping for new technologies. They want to open an office in the United States (in the Silicon Valley), close to the customers. Secondly, they would like to have the structure to push these new technologies forward to real products in an accelerated fashion. The time is probably coming for them to do series B funding with an American investor or, even better, with an industrial partner who will help raise their app **Sencogi** to the next level.

## "At Ferrari, everyone was 100% open!"

I asked Rijnders if his work at **Ferrari** had any influence on his management style at Cogisen. His first hand testimony of the great years of the **Italian Scuderia** deserve to be told in first person. So here is his reply:

*"I have to say that, when I was at Ferrari, it was really a dream team with **Ross Brawn, Jean Todt, Marco Fainello**. We had a top team. Everyone's opinion was respected. The information flow was open. Everyone knew what the others were up to, but nobody interfered with the work of the other. If you were an electronics expert, you needed to know what was happening with the tires. If you were a tire expert, you needed to know what was happening in aerodynamics. If you were in aerodynamics, you needed to know what was happening in the chassis. Everyone would know what the others were up to, but nobody would interfere with their work.*

*People were not afraid to share information. Often, if the work culture is not correct, then people will not share information to protect themselves and their work from future problems or politics. There was none of this. **Everyone was 100% open**.*

## "There was a culture in which taking risks with technology was encouraged"

*On top of this, there was a culture in which taking risks with technology was encouraged. If it didn't work out, it was fine. Nobody was punished for it. Another thing that was really fantastic was that there was really no politics. Jean Todt really encouraged that and all the managers were protected, feeling that taking risks with technology was accepted and encouraged. All the engineers felt that they had a voice, they were listened to and appreciated by the team. It was really an incredible work environment for engineering and stability. Ferrari was incredibly strong in those years.*

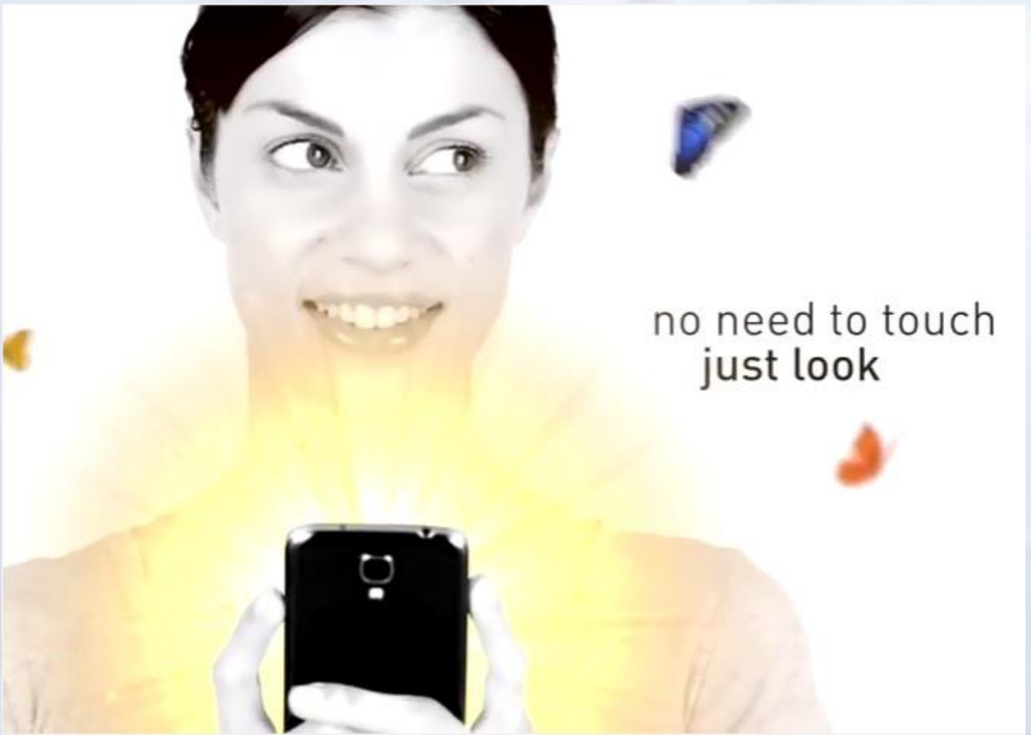
*During that time, I woke up and couldn't wait to get to work because it was a fantastic work environment. It was really a joy to work for people like **Rory Byrne**. Everyone had **so much confidence in what the other person was doing** that you could completely concentrate on your own work.*

***I've followed many of these guidelines** at Cogisen. I really tried to recreate the management style that makes Ferrari such a fun place to work. We have a lot of stability at Cogisen with many top experts. The fact that they all enjoy working with us shows that we have a very good work environment."*

## "Michael is a very sensitive, funny, and generous person"

Rijnders also has sweet souvenirs of **Michael Schumacher**, which are quite off-topic here, but too precious to leave unpublished:

*"Michael was also special in those years. People thought of him as a robot or super human. This was something that the press created, and Michael was very happy to play along. He always knew how to speak Italian, but he never spoke Italian in front of the press because he didn't want to show weaknesses. Of course, if your opponents think that you are super human, you're already halfway to winning the race. Psychologically, I think that his trick worked. In reality,* [Rijnders and I decide that we should speak of Schumacher in the present tense] ***Michael is a very sensitive, funny, and generous person**."*



Computer Vision News lists some of the great stories that we have just found somewhere else. We share them with you, adding a short comment. Enjoy!

**Comparison of Symbolic Deep Learning Frameworks**

Great job by **Microsoft Data Scientist Anusua Trivedi**: a series describing her **deep learning** experience and the reasons behind her choices. In the first post the commonly used open-source deep learning frameworks are compared, with pros and cons of each, including **Theano (with Lasagne)**, which she chose for her work. The second part describes **Deep Convolutional Neural Networks** and how transfer learning and fine-tuning helps the training process for domain-specific images. Read 1 and 2


**Canada Startups Bring Drones, Big Data, IoT to Farming**

Canada's long agricultural history makes it fertile ground for digital innovation in the farming industry. **Precision agriculture** startups are writing **algorithms building on a combination of drones, robotics, sensors and predictive analytics** to give farmers nearly real-time data so they can maximize crop yields while preserving resources. Read...

**Scientists Will Turn Your Smartphone Into a Lie Detector**

The next image processing software claiming to **decode hidden emotions and to double as a lie detector** comes from Toronto. **NuraLogix**'s **Transdermal Optical Imaging** is based on the control of our blood flow by the sympathetic and parasympathetic nervous system, which is responsive to emotions. Cameras capturing re-emission of light from our skin enable to track hemoglobin concentration and correlate its changes to our emotions. Read...

**Q&A with Google Brain's Scientists and Engineers**

They are a group of research scientists and engineers working in the Google Brain team. Their mission is to make intelligent machines to improve people's lives. That's how they have used their incredible talent and skills during the last 5 years. And now **they have answered our questions about machine learning** and the Brain team. Read...

**Baidu Brings Intelligent Augmented Reality to the Masses**

The Chinese Internet giant announced a new AR platform called **DuSee** that will allow many of the company's apps to understand the 3-D environment and make use of sophisticated computer vision and deep learning. Read...

**Read also:** The Facebook story you all know    Microsoft's Open source commitment
Automatic Alt Text with Microsoft's Computer Vision API

# TensorFlow - Part 2

**It's more than a deep learning package, it's an architecture!**

We showed you in our issue of August how to use TensorFlow as a framework for doing computer vision; specifically we demonstrated, with a simple example, how to find the grid lines in a picture of a Sudoku puzzle. In this month's issue we will expand this example and demonstrate more advanced techniques in TensorFlow: **(1) Modular programming** - splitting the code and having a graph with a pipeline of multiple functions, each doing one independent piece of code; **(2) Function parameters** - having function's parameters being part of the TensorFlow graph and changing/updating those parameters via the graph; **(3) Queue reader** - analyzing batches of files and maintaining them in parallel via the queue reader thread.

First, a brief recap of what we did in our last issue: we used the getHoughlines function for finding the straight lines in a given image. The function starts by converting the image into grayscale. Next, it extracts the edges in the grayscale image using the Canny edge detector. Then, it invokes the HoughLines function which finds the lines in the image with the Hough transform.

```
def getHoughlines(img):
    image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray  = cv2.convertScaleAbs(image)
    edges = cv2.Canny(gray, 10, 150, apertureSize=3)
    lines = cv2.HoughLines( edges,1.0, np.pi, 200)
    return lines
```

The code for executing this with TensorFlow is as follows:

```
import tensorflow as tf
# place holder - a tensor that will always be fed with the input image
sudoku = tf.placeholder(tf.float32, shape=shape, name='sudokuP')
# Hough Transform warper
board_update= tf.py_func(getHoughlines,[sudoku],[tf.float32], name='HoughF')
# Read the image and invoke the getHoughlines by TensorFlow
filename = 'sudoku.png'
raw_image_data = cv2.imread(filename)
with tf.Session() as session:
    session.run(tf.initialize_all_variables())
    lines = session.run(board_update, feed_dict={sudoku: raw_image_data})[0]
```



Let's now start adding some new stuff!

**(1) Modular programming**

We will demonstrate how to make the code more modular. TensorFlow has a built-in support for defining a pipeline of functions, each processing the output of the function before it.

We will update the above code by splitting it into two separate functions, each performing one independent process: (a) getEdges for computing the images edges; and then (b) getHoughlines for extracting grid lines with the Hough transform method. The two functions will now be:

```
# Hough transformation for straight lines
def getHoughLines(edges ):
    edges = np.squeeze(edges)
    lines = cv2.HoughLines(edges,1.0, np.pi / 180, 200)
    return lines

# The canny edge detection
def getEdges(img ):
    image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray = cv2.convertScaleAbs(image)
    edges = cv2.Canny(gray, 10, 150, apertureSize=3)
    return edges
```

Doing so would allow us, in the development process, to update/change one part of the code without changing all of it. Specifically, if we want to have a different edge detection method, we can replace it without touching the Hough method (getHoughlines) and vice versa, as these are now two independent processes in the graph.

The TensorFlow code for executing this two-operation graph is as follows:

```
1.    my_img = tf.placeholder(tf.float32, shape=shape, name='data')
2.    X1 = tf.py_func(getEdges, [my_img], [tf.uint8], name='Edges')
3.    X2 = tf.py_func(getHoughlines, [X1], [tf.float32], name='Hough')
4.    raw_image_data = cv2.imread('dave.png')
5.    with tf.Session() as session:
6.        session.run(tf.initialize_all_variables())
7.        lines = session.run(X2, feed_dict={my_img: raw_image_data})[0]
```

***"It's more than a deep learning package, it's an architecture"***

Lines 2 and 3, in the above code, are a pipeline of two functions: the getEdges (line 2) computes the image's edges and those are fed into the getHoughlines (line 3). The session.run() function executes the 'X2' which is a symbol for the getHoughlines. The TensorFlow "understands" this pipeline and it first executes the getEdges to get its output for the getHoughlines function.

It is worth noting that TensorFlow has a powerful feature for visualizing the graph: it can help you understand and debug it.

To visualize the graph, call the SummaryWriter function at the end of the "*with*" section in the code.

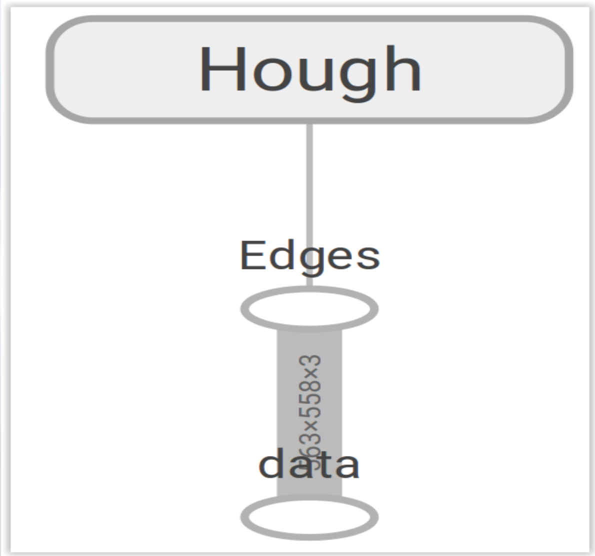
```
with tf.Session() as session:
    . . .
    tf.train.SummaryWriter('/tmp/log', session.graph)
```

Then, after the code was executed, in the command line you can type:

```
tensorboard --logdir=/tmp/log
```

Now, using your Web Browser, you will be able to observe the TensorFlow graph:

This graph is quite simple and it provides an excellent visualization of the pipeline of two functions in the code. It starts at the bottom, having a data placeholder tensor of a size of (563x558x3). This is an RGB image to be processed: it is fed into the Edges function which perform the Canny Edges Detection. Next, the Hough transform is executed for getting the Sudoku grid lines.




**(2) Function parameters**

In our code, the cv2.HoughLines() in the getHoughlines function has several parameters, which influence how the Hough transform method works. The best practice is to have those parameters supplied to the function and not have them as fixed numbers included in the function code.
Luckily, TensorFlow allows you to define variables being part of the graph; those could be changed and evaluated for different values as part of the Framework.

We will update the getHoughlines function to receive two variables, 'rho' and 'theta'. The function will now be:

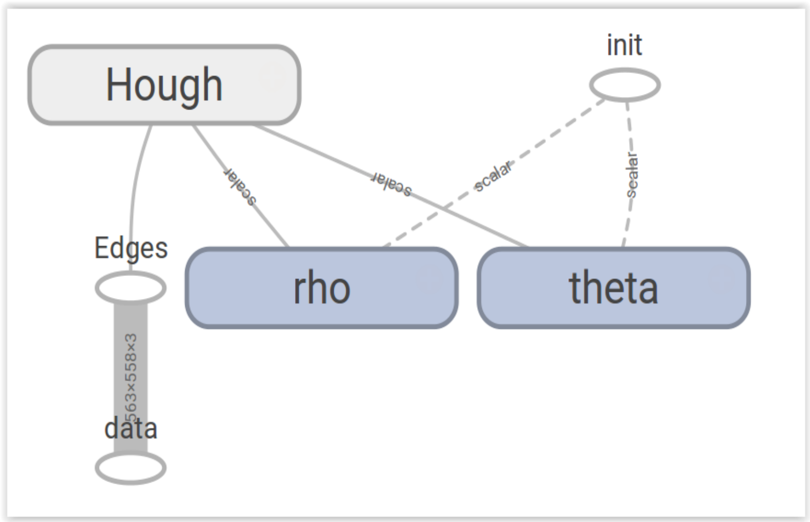
```
def getHoughlines(edges, _rho , _theta ):
    edges = np.squeeze(edges)
    lines = cv2.HoughLines(edges, _rho ,  _theta , 200)
    return lines
```

To have those two variables as part of the TensorFlow graph, we update our code as follows:

```
_rho   = tf.Variable(1.0, name='rho')
_theta = tf.Variable(np.pi / 180, name='theta')
X1=tf.py_func(getEdges, [my_img], [tf.uint8], name='Edges')
X2=tf.py_func(getHoughlines,[X1,_rho,_theta],[tf.float32], name='Hough')
```

We first have to define two variables using the tf.Variable() function (lines 1-2). Then, we can define them as parameters being sent to the function (line 4). Note that each variable has an initial value which can later be changed.

The updated graph now looks as follows:




You can see the 'rho' and the 'theta' parameters now being part of the TensorFlow graph and fed into the Hough function.

Parameters, of course, are meant to be changeable; later we will see how to do it.

***" Nice and modular "***

**(3) Queue reader**

Obviously, we didn't write all this code for analyzing a single image and most probably you have many files which you need to examine, analyze and evaluate. The process of reading many data files and feeding them into the module can be quite tedious. Luckily, **TensorFlow comes with several functionalities** for easing this process. One of them is the queue reader. Queue reader parallelizes the reading process by performing it on a different thread from the one executing the rest of the code.

Here is the code defining the Queue reader for reading a list of .png images:

```
1.  filename_queue =
    tf.train.string_input_producer(['im1.png','im2.png','im3.png','im4.png'],name="FileNames") #  list of files
    to read
2.  with tf.name_scope("RederType"):
3.     reader = tf.WholeFileReader(name="red")
4.  key, value = reader.read(filename_queue, name="Reader")
5.  my_img = tf.image.decode_png(value, name="pngDec") # use png decoder
```



In the code above, we start by defining a queue initialized with a list of filenames (four in this example). Those names are fed into our model string_input_producer (line 1). Next, the wholeFileReader (line 3) tells TensorFlow to read a complete file at a time. Then the reader.read (line 4) extracts and reads those files from the queue. Lastly, the files are decoded by their type - png (line 5), so that you can have a decoder per file type (nice and modular, isn't it?).

The code for executing this Queue reader graph is as follows:

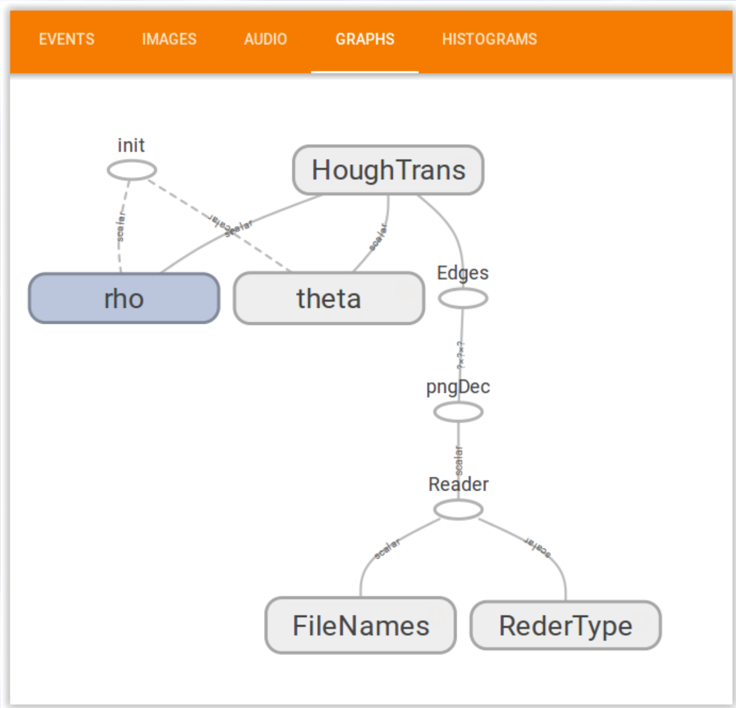
```
with tf.Session() as sess:
  sess.run(init_op)

  coord = tf.train.Coordinator()
  threads = tf.train.start_queue_runners(coord=coord)
  for i in range(4):  # length of your filename list
    lines = sess.run(X2)[0]
```

It starts the QueueRunners by calling tf.train.start_queue_runners(sess=sess). This call is functional, it creates the threads for enqueueing data to the queue, which will later be processed by the operations defined by the TensorFlow graph.

Note that we omitted the cv2.imread and the feed_dict={my_img: raw_image_data} as now the files will be read by the queue.

The updated graph with the Queue reader looks now as follows:




**Update parameters**

Last but not least, we will demonstrate how the parameter can be updated. For example, setting different values of theta for each image by raising it by pi/1800 at each iteration. Here is the code for doing that:

```
1.   with tf.Session() as sess:
2.     sess.run(init_op)
3.
4.     coord = tf.train.Coordinator()
5.     threads = tf.train.start_queue_runners(coord=coord)
6.     for i in range(4):  # length of your filename list
7.       assign_op = _theta.assign_add(np.pi / 1800)
8.       lines = sess.run([assign_op, X2])[0]
```

The only update we have to make is in lines 7-8, where we add pi/1800 at each iteration using the _theta.assign_add function; and we add this operation into the run() session.

# The Unbearable Lightness of Algorithm Development

Our CEO **Ron Soferman** continues his series of lectures providing a robust yet simple overview about how to ensure that computer vision projects respect goals, budget and deadlines. Mr. Soferman shares one of his project management tips in each issue of Computer Vision News. Today we learn about what he calls the **Unbearable Lightness of Algorithm Development**.

"Multiple algorithms, one on top of the other, might just be a cover for the inefficiency in applying the right algorithm"

We always like to recruit very creative and innovative thinkers to work in our algorithm group. These engineers always find **new directions and new solutions for the development**, making it possible to solve the problems.

In this article we want to shed a light on the other side of creativity in algorithm development. We notice it most often among young developers who just finished school and wish to solve **all kind of problems in computer vision** with the power of the countless algorithms that they have learned in school. Moreover, today you can easily apply algorithms from Matlab or OpenCV in only a few minutes.

The problem arises when one starts to solve a problem using an algorithm which gives some kind of good result for part of the problem at hand. Seeing that about 20% of the problem is yet to be solved, the developer will choose to add **another algorithm layer that would increase the yield and give better results**. And after that, another algorithm and then another one.

On the contrary, it is advisable to check more thoroughly each algorithm, to tweak it and find the right adaption of this algorithm to the specific problem and hence to be sure that the new developer is very rigorous in debugging the code. Otherwise, you might find that the **multiple algorithms, one on top of the other, might just be a cover for the inefficiency in applying the right algorithm**. This will turn as a boomerang against the development effort.

The objective is to find one algorithm that will do most of the work. It has to be very neat and clean, so that all the mathematical theorems that supplement the use of this algorithm will play the right role with the right parameters. Using algorithms over algorithms will only produce a very confused process, that will rarely converge; and in many cases, the assumptions which are needed for the algorithm will not hold, following the input of yet another algorithm.

# Dan Feldman, University of Haifa

It is the first time that we discuss the **coresets**, a powerful technique which enables the use of smaller sets of data instead of larger ones without compromising the quality of the output. To learn more about it, we decided to interview one of the most important researchers in the field: **Dr. Dan Feldman**, who after three years at the **Computer Science and Artificial Intelligence Lab of the MIT (CSAIL)**, is now Director of the **Robotics and Big Data Lab** and Senior Lecturer at the **Computer Science Department of the University of Haifa**. He was kind enough to tell us more.

**Computer Vision News:** *What are coresets and why are they so important? Why are they so powerful? Why should we use them?*

**Dan Feldman:** The main idea is that in computer science we usually have a problem that someone suggests, let's say clustering. Different solutions will be suggested to solve this problem. We usually want to develop a better algorithm over time with an improved running time, memory or space.

With coresets, the philosophy is different. Instead of trying to suggest another algorithm, we want to prove that we can reduce the data, so that running existing algorithms on the reduced "small data" will provably give approximated result, as running them on the original "Big data". We can usually reduce the data, not by half, but by order of magnitude: for example, from n to log(n). This is done, not by designing a new algorithm for solving the problem, but by just running the new algorithm and existing algorithm on small compressions.

Unlike other compression techniques like zip or mp4, coreset is data reduction and not just compression of the input in the sense that it's problem-dependent. A point may be important for one problem, but not important for another problem.

We keep seeing papers on optimization for new problems, but we still have general techniques such as linear and quadratic programming. We have Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). We have the derivatives. We have general techniques on how to optimize functions if they have specific properties. These days we also try to find more general techniques for coreset constructions.

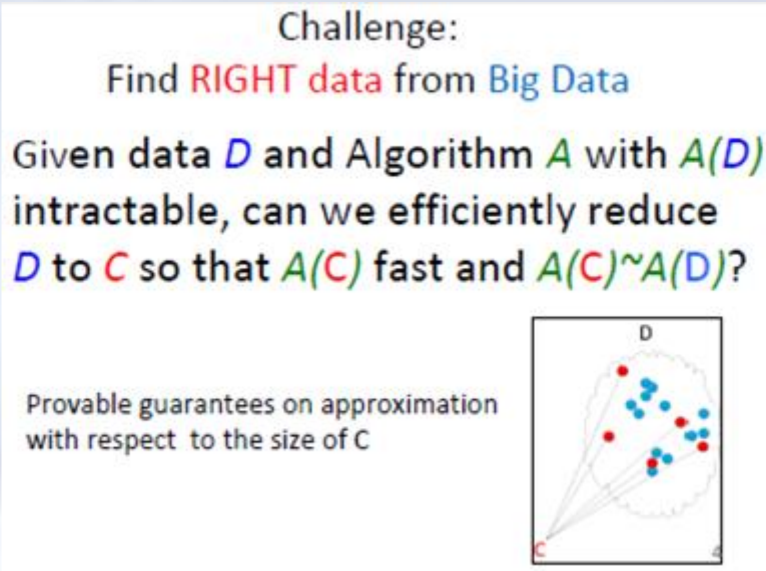
*"Unlike other compression techniques like zip or mp4, coreset is data reduction and not just compression of the input"*

**CVN:** *Is it because coresets are so powerful that you chose them as your main technique?*

**Feldman:** Yes - There are many social, academic and industrial reasons why we use coresets these days. If you know how to optimize a program and use the original data with the small data, of course you get some errors. Surprisingly, **the results on the coresets are usually better than the original data**. If you give me data to find the optimal solution, I can do better by moving some of the data. If we just have strange heuristics that give you some number without any proof of why it's good or bad, usually these heuristics only find local minima. The coresets remove a lot of noise, thus the local minima are much smaller and better. In some sense, data reduction removes most of the noise. That's how we get better results compared to running the algorithm on the original coreset. It's very good for business: we still use all of your expert knowledge. But also academically, I started using the coresets for theoretical problems. These days, I'm using the coresets for drones, image processing, computer vision robotics and EEG.

**CVN:** *Which kind of algorithm works better and on which kind of problems?*

**Feldman:** For many problems we can prove that a small coreset does not exist: removing one input point would yield a very bad approximation. In this discussion we assume that a small coreset exists, but we need to find it and prove its guarantees for every new problem. As I said, we try to find general solutions for problems that don't satisfy specific requirements. This is a kind of optimization. **We don't expect anyone to find one single technique to optimize all the problems in the world.**




***"I hope to bridge the gap between theory and practice using these coresets"***



**CVN:** In what way is the coreset that you use different from the coreset for k-means used by Sariel Har-Peled?

**Feldman:** It's not! Over the years, we have developed coresets for different problems and we keep improving and redefining them so that we can solve the problems.

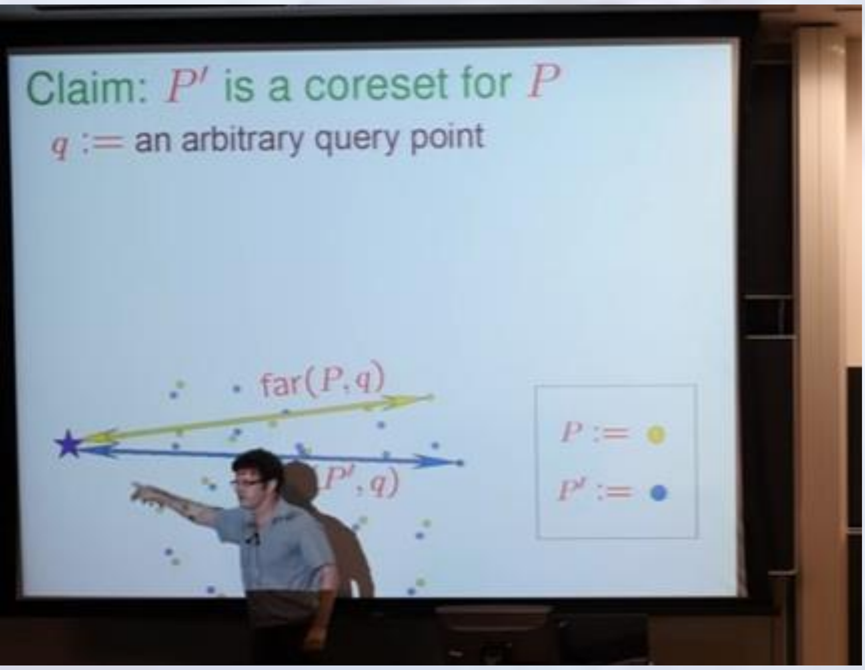
With every problem, we have a long line of research. In computer vision, there are numerous problems that I believe we can solve using coresets. Researchers from theoretical computer science are not interested in or (in the more common case) not familiar with this kind of problems.

**Coresets are a new paradigm. It is more a state of mind than an exact mathematical definition** because the exact definition changes from paper to paper. Actually, the number of definitions is very similar to the number of papers on coresets.

**CVN:** *I understand that coresets are less useful when you have too many layers or too many parameters.*

**Feldman:** Right - If you have n points and you're looking for n parameters, like in the Traveling Salesman Problem for example, you probably can't use coresets again unless you assume something about the input. Every polynomial time algorithm is a coreset in the sense that you have an exponential number of solutions, but you only search a small number. You still guarantee that the solution will be there. You compress the solution space. In some way, every efficient algorithm uses this kind of approach for compressing data.

The other related fields which we are trying to connect with coresets are compress sensing, sketching, all the "sufficient statistics" and property testing. Unlike with coresets, with property testing you're not allowed to look at all of the data; the challenge is the same: **to solve the problem with small data**. However, they have a much harder constraint because they cannot read all of the data, but only some of it. In coreset, we usually assume that we can scan all of the data. That's how we can solve more problems than property testing, for example.




## "We don't expect anyone to find one single technique to optimize all the problems in the world"

**CVN:** *If our readers want to try to experiment with coresets themselves, is there any place where they can find tutorials, courses or examples?*

**Feldman:** We upload new coresets on my lab's website. We hope to publish a library within the next few weeks, but you can also get it online or send me an email. We'll try to have one library for everything.

**CVN:** *What surprised you from working on the coresets?*

**Feldman:** Recently we had a big surprise. We now examine coresets for two fields using differential privacy. The idea is to do machine learning, while preserving the privacy and anonymity of the users, so that we extract statistics from the data without revealing information about the individuals.

Surprisingly, our STOC'11 paper that we now implement shows a formal connection that says that if you can compress the data, it also means that you can have a private version; we refer to it as private coresets of the data, or a sanitized database. It means that you can add small noise to data so that the statistics will be preserved and the k-means will still

Guest

be the same. Yet you cannot reveal anything about the individuals from the k-means. There shouldn't be any connection between compressing the data and adding this small noise to preserve their privacy, but there is such a connection.

> ***"Coresets are a new paradigm. It is more a state of mind than an exact mathematical definition"***

**CVN:** *It sounds like we have a distinctive coreset in our mind.*

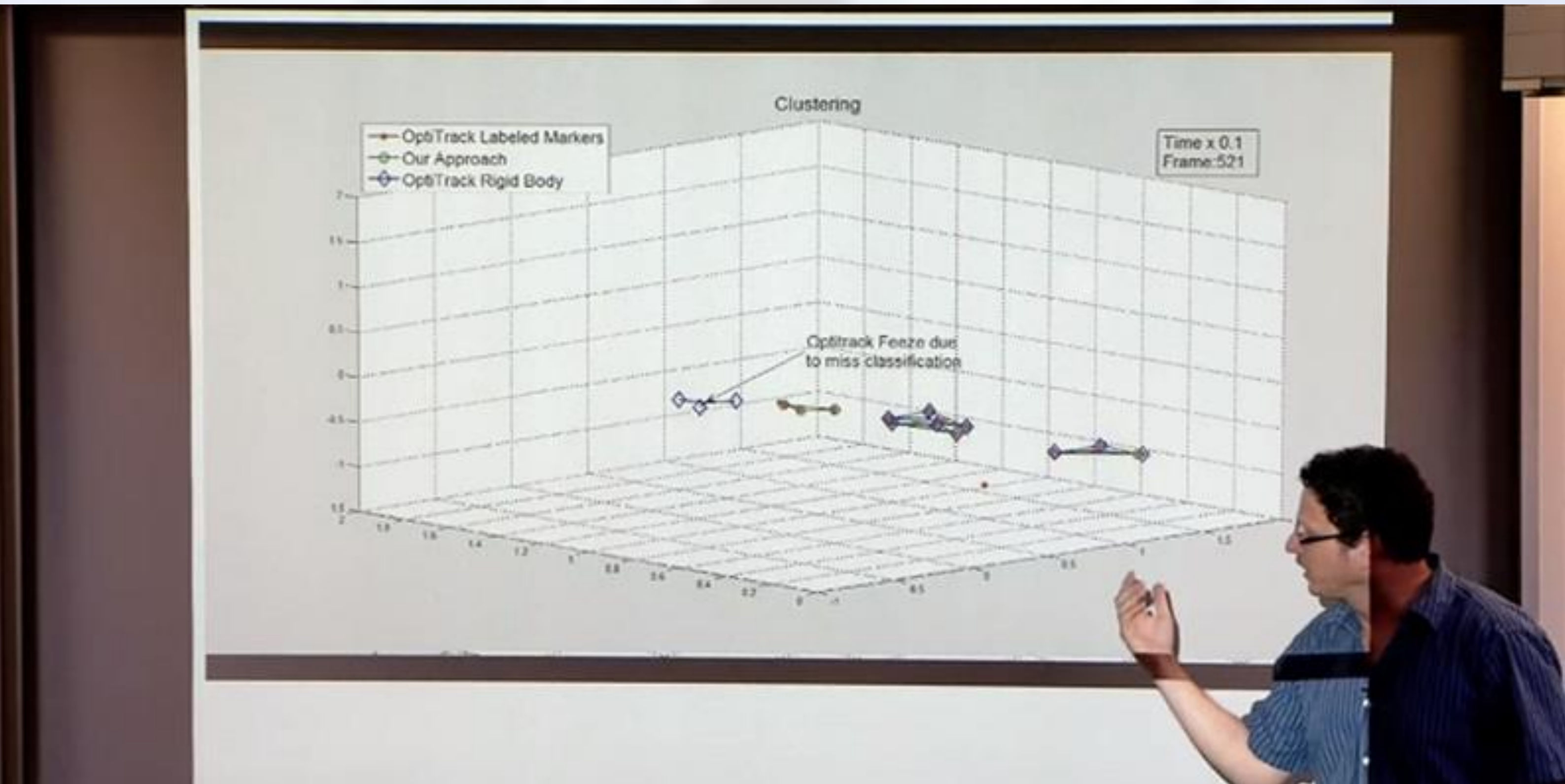
**Feldman:** Kind of… We always need to decide what is important and what is not. Think at how we manage our time, what we see, what we hear and so on. The idea of compression and compressing the right thing is problem-dependent. I am sure that we do it all the time for all the senses that we have. Some people do it better than others.

**CVN:** What is the biggest breakthrough that you have seen?

**Feldman:** The main breakthrough in this field is the fact that **we now have a general framework coreset for any problem**. Unfortunately, I think the coresets are still buried in theoretical computer science conferences. Engineers are not using them because there is very little code out there.

I hope that in coming years we will have more implementations so that people who cannot read my papers can still run the coresets and evaluate them for their business or for their research. We want to **bring the coreset to the people** and bring the research into the industry.






*This image and the previous one were taken on June 28 at the Simons Institute for the Theory of Computing: Core-sets for Real-Time Tracking using Caratheodory Theorem, with Applications to Drones. The video of the conference is [here](#).*

# Call Python code from Matlab

Every month, Computer Vision News shows you a nice trick that will make your work easier. This time, our engineers will tell you **how to call Python code from Matlab**.



Both Matlab and Python have efficient and powerful tool-boxes for computer vision and it is not rare to have a scenario where you have a computer vision project in Matlab but you need to incorporate already written Python code in it.

Since version 2014b, Matlab comes with built-in support for calling Python code. However, there are some bits and bytes that need to be tuned in order to get it to work correctly. In addition, some of the settings change from one Python version to another. This month's trick comes to aid you in this process. After all, this is what our trick of the month is for, isn't it?

## A. Initialization

The initialization consists of two steps: setting and loading the Python interpreter. Setting is done by the *pyversion* command and loading is done automatically when you type the first py.command.

The Matlab's *pyversion* command is a bit tricky as it has two uses: both for setting and for getting the version of the Python interpreter. In addition, once the Python interpreter is loaded you can't change it in the current session. Only restarting the Matlab will allow you to load a different version of Python, for example, in case you want to switch between version 2.x and 3.x. Let's see the *pyversion* command in action:

| Details | The code |
|---|---|
| pyversion argument telling Matlab which Python version to use.<br><br>First time we run the pyversion command: The 'isloaded' field is 0 indicating that Python is not loaded yet. | >> pyversion 'C:/Anaconda3/python.exe'<br>>> pyversion<br>version: '3.5'<br>executable: 'C:\Anaconda3\python.exe'<br>library: 'C:\Anaconda3\python35.dll'<br>home: 'C:\Anaconda3'<br>isloaded: 0 |
| Python will be loaded only after you execute the first Python command. For example after running this command which define a Python list. | >> myList = py.list([1.0,2.0,3.0,4.0]) |
| Calling the pyversion again, after running a Python command, Now the is 'isloaded' field is '1' | >> pyversion<br>version: '3.5'<br>executable: 'C:\Anaconda3\python.exe'<br>library: 'C:\Anaconda3\python35.dll'<br>home: 'C:\Anaconda3'<br>isloaded: 1 |
| As mentioned, once Python was loaded, it cannot be changed. Only restarting Matlab would allow you to do so. | Error using pyversion<br>Python is loaded. The version cannot be changed. |

This is the initialization scenario in most cases. Why most and not all? Because when Matlab finds the Python interpreter both in the path environment variable and in the Windows registry, it automatically loads it. In this case, for changing the Python version you will have to change your path environment variable.

After successfully loading the Python interpreter, you can call Python commands via the py. Model (e.g. py.list([1.0,2.0,3.0,4.0])). More examples follow below.



## B. Calling Python Module

Now, let's see how we can call our Python code and model. For this demonstration, we will assume our model is saved in ThePythonCode.py file and it has a testFunc function, see code snippets below:

The Python code:

The testFunc function get a list1 and define a new list (list2), print the type of those list and return the concatenation of those two list.

For concatenating a 'list' sent from Matlab and a list define in Python a type conversion is needed.

ThePythonCode.py

```
def testFunc(list1):
  list2 = [1.0,2.0,3.0,4.0]
  print(type(list1))
  print(type(list2))
  return list2+list(list2)
```

The Matlab code:

We start by adding the local directory (''), empty path, to the Python environment variable. As the location of the ThePythonCode.py script has to be in the Python path even if it is in the current directory.

After the path has been set, we call the testFunc Python function located in ThePythonCode.py file.

Note, the different types between a 'pure' Python list and a 'list' sent from Matlab which is array.array. Array objects are sequence types and behave very much like lists, except that the type of objects stored in them is constrained. For more on data type conversion see here.

```
>> if count(py.sys.path,'') == 0
   insert(py.sys.path,int32(0),'');
  end

>>  my_list = [5.0, 6.0, 7.0, 8.0]
>>  my_list2 =
py.ThePythonCode.testFunc(my_list)

<class 'array.array'>
<class 'list'>
ans =
  Python list with no properties.
   [1.0, 2.0, 3.0, 4.0, 5.0, 6.0,
     7.0, 8.0]
```



## C. Return Values

Returns values, much like send values, should get special care according to their type. For example the 'my_list2' above is a Python list.

```
>> whos
  Name          Size              Bytes      Class
  my_list        1x4                 32     double
  my_list2        1x8               112     py.list
```

For a full list of return values and their type see here.

Note that even though the Python list was returned to Matlab, the operators are still Matlab. For example, you cannot use the '-1' to get the last variable as you normally do in Python, but you need to use the 'end' operators as in Matlab. Example in the code snippets below:

| | |
|---|---|
| my_list2 defined in section B above.<br><br>Note that you have to use the '{}' and not the '()' for extracting a value from the py.list. | >> my_list2(1)<br>ans =<br>  Python list with no properties.<br>    [1.0] |
| For a full list of Python/Matlab operators of other types see here. | >> my_list2{1}<br>ans =<br>    1 |
| For example, to access the last variable in the list. | >> my_list2{-1}<br>Error using py.list/subsref<br>Subscript indices must be real, positive integers.<br><br>>> l1{end}<br>ans =<br>    8 |

## D. Passing Image (3D matrix) to Python Module

We are dealing here with computer vision applications. Thus, it is most likely that we would need to pass an image (3D matrix - RGB image of the size nRow x nCol x 3) to Python and not just a list/string etc. Unfortunately, matrices can't be passed directly into Python since **this option is not supported yet**, as of Matlab 2016a.

The way of doing so is by converting the image into a 1D matrix, i.e. a vector.

For this demonstration we will assume that we have a Python file named ThePythonCode in the current directory with our famous functions: **getEdges, getHoughlines and PlotHoughLines**. Not famous yet? **See the Tool of the month at page 14 for more info**.

We will add a new function to this file: dataFromMatlab. This function will get the image as vector sent from Matlab, convert it "back" into a 3D image and perform the Hough Transform on it.

Here is the Matlab and Python code for this:

The Matlab Code:

Start by reading the image (lines 1-2), reshape it into a vector (line 3). Load our Python model ThePythonCode.py (line 4-5) and calling the dataFromMatlab Python function.

Note that the py.reload
Has a different syntax for [Python 2.X]

```
1.  im = imread('dave.png');
2.  im = uint8(im);
3.  asVec = reshape(im ,1 , []);
4.  mod =
    py.importlib.import_module('ThePythonCode');
5.  py.reload(mod);
6.  py.ThePythonCode.dataFromMatlab(l1 ,
    size(im,1) , size(im,2) )  ;
```

The Python Code:

Lines 1-2: Reshape the vector back to an RGB image matrix of the size (nRow x nCol x 3)
Lines 3-6: compute and display the HoughLines of the image.

```
def dataFromMatlab(
        imageAsVector, nRow, nCol):
1.      im = np.array(imageAsVector
                 , dtype=np.uint8 )
2.      im = im.reshape(( 3, nCol,
          nRow)).transpose().copy()
3.      eg = getEdges( img )
4.      lines = getHoughlines(eg )
5.      img = PlotHoughLines( img,
                       lines )
6.      plt.imshow(img)
7.      plt.show()
```

Running the six lines of the Matlab code above would popup the Sudoku image with grid lines marked in blue, as we saw in the [Tool section of last month].

The new cv2 (OpenCV 3.X) interface for Python integrates [numpy] arrays into the OpenCV framework, which makes operations much simpler, as they are represented with simple multidimensional arrays. If you are using the OpenCV 2.X, you will have to use the cv.fromarray() function for converting the Numpy array into CV::Mat.

For the trick of this month we have used: Win10, Matlab-2015b, Python-2.7, OpenCV 3.1 and Numpy-1.10.

N.B. To call MATLAB functions from Python applications, see [MATLAB Engine API for Python].

# Decoding Brain Signals

Every month, Computer Vision News reviews a challenge related to our field. If you can't find time to read challenges, but are interested in the new methods proposed by the scientific community to solve them, this section is for you. This month we have chosen to review the **Decoding Brain Signals competition** held by **Microsoft**. The website of the challenge, with all its related resources, is [here].

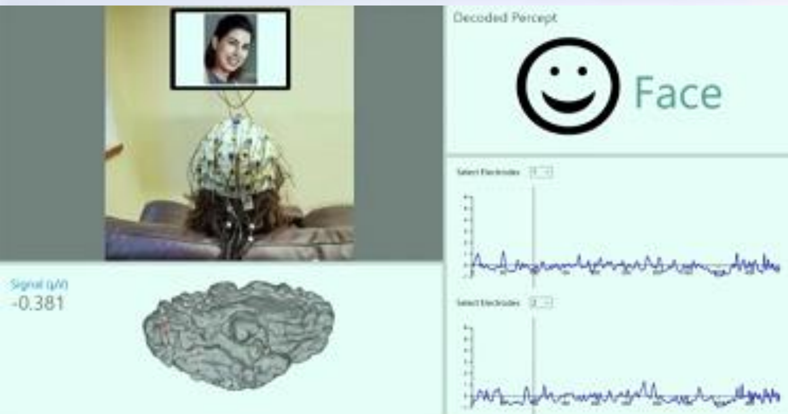


## Background

**Microsoft Corporation** saw an opportunity to further the cause of neuroscience research. To that purpose, it partnered with a **Stanford neurosurgeon, Dr. Kai J. Miller**, to create a competition based on his original research.

On one hand, millions of people are affected every year by brain injuries, related disorders and strokes. On the other hand, science does not possess yet a good understanding of **how our brain interprets electric signals**, e.g. those originated by looking at an image. Hence, the call for **experts in machine learning and data science** to help decode these signals and play a key role in advancing neuroscience research to bring the next generation of care to patients.

### How does our brain interpret electric signals?




*Brain signal data gathering: from Microsoft's Channel 9 video introducing the Decoding Brain Signals competition.*

## Challenge

The challenge asked participants to build a learning model capable of accurately predicting the image shown to a human subject based on electric signals in the brain. The model needs to predict whether the person is seeing the image of a house or of a face, following the Electrocorticographic (ECoG) signals collected from a sample of four epileptic patients. That makes it an image recognition neuroscience and data science project.

The Grand Prize winner of this competition is **Alexandre Barachant**, whose model reached an accuracy just below 94%. Actually, his final solution is a blend of 5 different models: 2 were dedicated to detection of evoked potential and 3 to induced activity. Among the general ideas which he followed to build them:

- train the models independently on each individual patient;
- avoid any preprocessing of the data and directly feed the raw signal to the models.

These and other principles are carefully explained by the author of the winning entry and presented with his code here. Those interested can also use the solutions proposed by the runner-ups here and here. We are more than happy to find the name of **Luisa Zintgraf** (one of the authors of the research which we reviewed in Computer Vision News of July 2016) as high as the **7th place among the overall 688 participants**. So we asked her about her experience participating in this challenge. It seems that she actually agrees on some points with the winner of the challenge; here is what she told us off the record (but kindly agreed to let us publish):

## "A really fun challenge, and the first one I ever did!"

*"It was a really fun challenge, I must say (and the first one I ever did, so I was quite happy about the 7th place).*

*I spent most of my time on feature engineering and feature selection, actually. I talked to some other people who also participated in the challenge who spent a lot of time and effort on training deep neural networks, but that didn't work out too well: accuracies were below 70%! There were just not enough data points to train a good neural net.*

*So I used a **logistic regression classifier** (because it's fast and I could test many different things) and focused on the features. One thing that really boosted performance was for example to use only a subset of the electrodes, and this subset was specific to the patient. Again, I think the biggest challenge was that there were so many more features than data points, which is why I spent most of my time reducing the features/noise (like taking a smaller time frame than 800ms, etc.).*

*I did everything per-patient, and had some discussion about this with the other participants I talked to (we actually teamed up in a **Kaggle competition** now!). They said that it's unrealistic to have specific features per patient, and that in reality it would be better to generalize, and have the same preprocessing+model for all patients.*

*Luisa M. Zintgraf is graduate of Artificial Intelligence at the University of Amsterdam, Netherlands.*

## "There were just not enough data points to train a good neural net"

*But I strongly disagree! The performance is so much worse, also because the electrodes are at different places in the brain and therefore **the signals don't have the same meaning across patients**. I think it should be totally feasible to do some feature engineering and individual model training per patient (also since the potential benefits of decoding brain signals for example of paralyzed patients are huge). And even if that's too expensive, it would be possible to automate the feature selection process to some extent."*

# Learned Invariant Feature Transform (LIFT)

Every month, Computer Vision News reviews a research from our field. This month we have chosen to review the **Learned Invariant Feature Transform (LIFT)**, a research paper introducing a novel Deep Network architecture that implements the full feature point handling pipeline: detection, orientation estimation and feature description. The paper, which will be presented at **ECCV 2016 in Amsterdam**, shows how to learn to do all three in a unified manner while preserving end-to-end differentiability. This Deep pipeline model outperforms state-of-the-art methods on a number of benchmark datasets, without the need of retraining. We are indebted to the authors (Kwang Moo Yi, Eduard Trulls, Vincent Lepetit and Pascal Fua) for allowing us to use their images to illustrate this review. The full paper is here and the source code will be available here.

> ***"Deep Network architecture that implements the full feature point handling pipeline, that is: detection, orientation estimation and feature description"***



## Background:

Finding and matching image features is a key step in many computer vision applications. The amount of literature relating to local features extraction is enormous, but it always revolves around three main steps: (a) finding feature points, (b) computing their orientation, and (c) describing them. Prior to the deep learning revolution, the best techniques relied on carefully hand-crafted image features. However, Deep Learning based techniques have started to outperform these traditional methods.

## Motivation:

So far, all deep learning methods address only a single step in the feature extraction process (a to c above) and there is no existing method tying all three components together.

## Challenge:

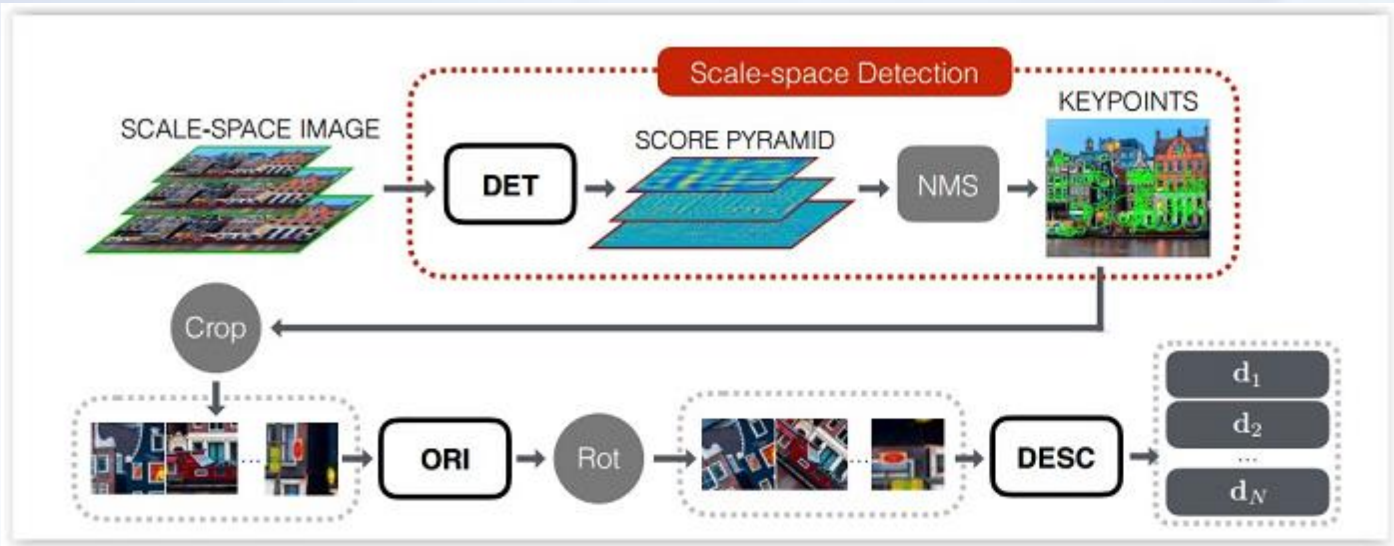
To train a Deep Network architecture that implements the full feature point handling pipeline: detection, orientation estimation and feature description.

## Novelty:

A new framework for image features extraction called **LIFT**, which stands for **Learned Invariant Feature Transform**, the architecture that implements the full feature point handling pipeline (detection, orientation and description) is based on **Convolutional Neural Networks**. The framework is trained in an effective manner and outperforms the state-of-the-art methods on a number of benchmark datasets, without the need for retraining.

## Method:

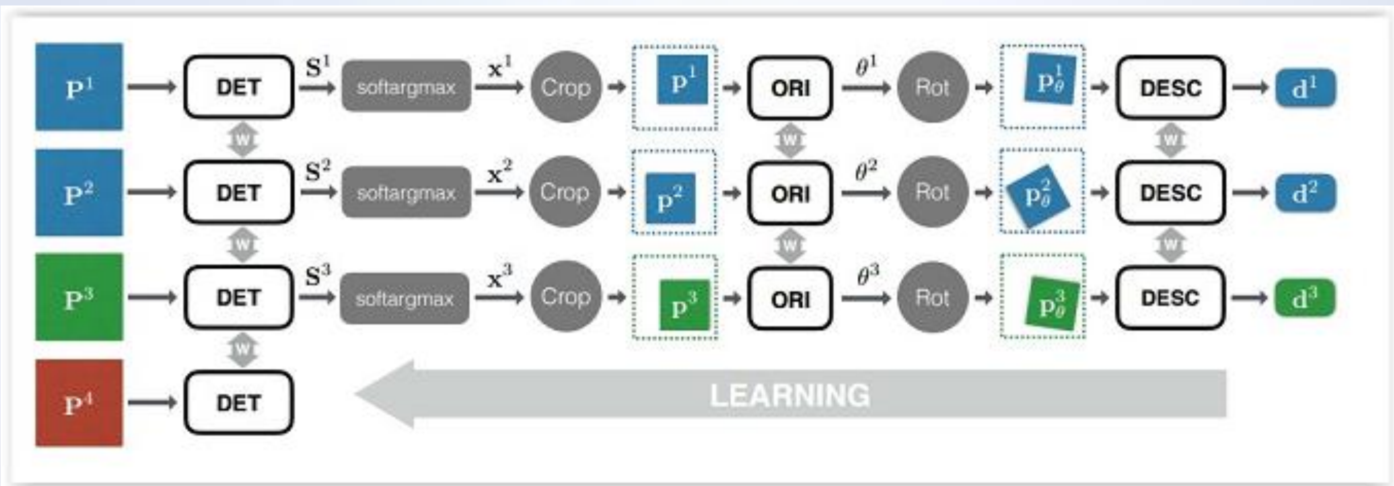
The LIFT pipeline




The Detector (DET), the Orientation Estimator (ORI) and the Descriptor (DESC) are all CNN-based, coupled with an end-to-end differentiability framework.

For training, the LIFT framework uses a four-branch **Siamese architecture** (see figure below), where each branch contain three CNNs marked by DET, ORI and DESC. More details on these three networks follow.

The Siamese architecture: given an input image patch P, the Detector (DET) provides a score map S, which feeds into the softargmax and returns the location x of one feature point. Then, a smaller patch p centered on x is extracted along with the Spatial Transformer Crop. Next, the patch is used as the input to the Orientation Estimator (ORI), to predict the patch orientation and rotate it according (Rot). Finally, this patch is fed into the Descriptor network, which computes a feature vector d.



The CNNs are trained in reverse order (DESC → ORG → DET); namely, the Descriptor, then the Orientation Estimator given the learned descriptor, and lastly the Detector, conditioned on the other two.

The inputs to train the Siamese architecture are quadruplets of image patches with the following properties:

- $P^1$, $P^2$ - Two different views of the same 3D physical point used by the Descriptor as a positive example during training.
- $P^3$ - 3D point different from that of $P^1$ and $P^2$ used by the Descriptor as negative example
- $P^4$ - 3D point with no distinctive information used as negative example to train the Detector.

The image patches which are used as input are assumed to be small enough to contain only one dominant local feature.

The table below details each of the three CNNs in the LIFT framework:

| **The Descriptor** | | |
|---|---|---|
| | Original method | Simo-Serra, E et al. Learning of Deep Convolutional Feature Point Descriptors. In: ICCV (2015) |
| | Training input / output | The last layer is the first to be trained. Therefore, for the training process, the image locations and orientations of the feature points used by the SFM to generate image patches p |
| | Adaption made in the LIFT framework | The increasing mining scheme starts with r = 1 and double every 5000 batches. use balanced batches with 128 positive pairs and 128 negative pairs, mining each separately |
| **The Orientation Estimator** | | |
| | Original method | Jaderberg, M., Spatial Transformer Networks. In: NIPS (2015) |
| | Training input / output | Only the positive patches, $P^1$, $P^2$ are used to train the Orientation Estimator. The already trained Descriptor is used to compute the description vectors and the input locations are used from SFM as in the Descriptor layer |
| | Adaption made in the LIFT framework | None |
| **The Detector** | | |
| | Original method | Verdie, Y., et al. TILDE: A Temporally Invariant Learned Detector. In: CVPR (2015) |
| | Training input / output | The Detector is trained on the full LIFT pipeline as the Orientation Estimator and the Descriptor are already learned by this point. |
| | Adaption made in the LIFT framework | To let S have maxima in places other than a fixed location retrieved by SFM, we treat this location implicitly, as a latent variable. |

## Datasets:

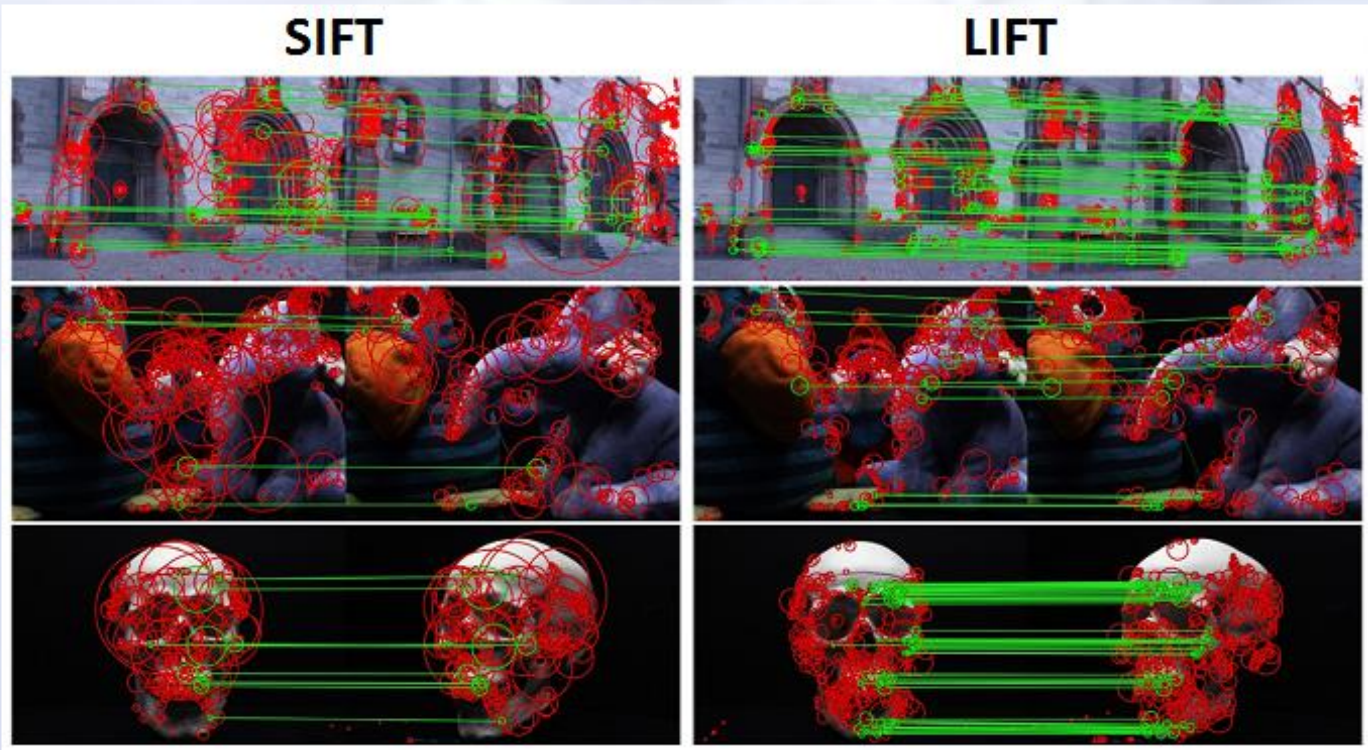
**The Datasets used for training** are the Piccadilly Circus in London and the Roman Forum in Rome. On these datasets, the VisualSFM was used to generate invariance images that capture the same views with different illumination conditions and different perspectives. The Piccadilly dataset contains 3384 images and the reconstruction has 59k unique points. The Roman Forum contains 1658 images and 51k unique points. Only the feature points that survive the SFM reconstruction process were used to train the LIFT framework.

**The Datasets used for testing** are the *Strecha* dataset, which contains 19 images of two scenes; the *DTU* dataset, which contains 60 sequences of objects with different viewpoints and illumination settings; and the *Webcam* dataset.

***“Outperforms state-of-the-art methods on a number of benchmark datasets, without the need of retraining”***

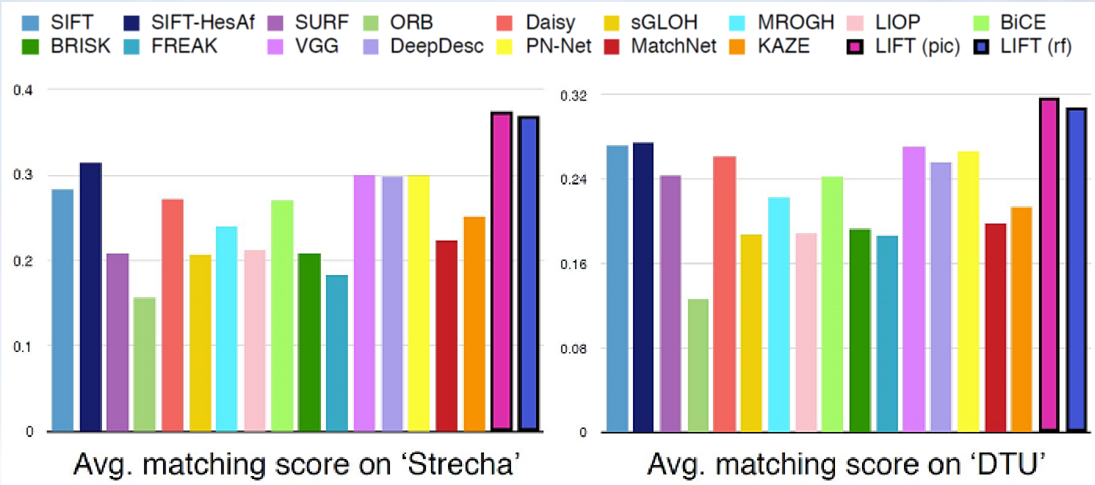
## Results:

To stimulate your appetite, we present three qualitative results. As expected, LIFT returns a higher number of correct correspondences across the two images. Correct matches are shown in green lines and the descriptor support regions are shown in red circles. First row: Strecha; second and third row: DTU.



## Qualitative Results:

The authors conducted an impressive and exhaustive comparison to 16 other feature extraction methods, outperforming each one of them on each of the 3 datasets used:




M. Score - Matching Score: the ratio of ground truth correspondences that can be recovered by the whole pipeline over the number of features proposed by the pipeline. This metric measures the overall performance of the pipeline.

In the deep evaluation procedure, the metric used were as follows: to demonstrate on Strecha that each of the 3 LIFT layers is indeed essential, 2 LIFT models were evaluated (one trained on Piccadilly dataset and the other on Roman Forum dataset). In each model, the LIFT layer was interchanged with its SIFT counterpart,  showing that each element of the pipeline is indeed crucial:

| | | | Trained on *Piccadilly* | | | Trained on *Roman-Forum* | | |
|---|---|---|---|---|---|---|---|---|
| Det. | Ori. | Desc. | Rep. | NN mAP | M.Score | Rep. | NN mAP | M.Score |
| SIFT | SIFT | SIFT | | .517 | .282 | | .517 | .282 |
| SIFT | LIFT | SIFT | .428 | .671 | .341 | .428 | .662 | .338 |
| SIFT | SIFT | LIFT | | .568 | .290 | | .581 | .295 |
| SIFT | LIFT | LIFT | | .685 | .344 | | **.688** | .342 |
| LIFT | SIFT | SIFT | | .540 | .325 | | .545 | .319 |
| LIFT | LIFT | SIFT | **.446** | .644 | .372 | **.447** | .630 | .360 |
| LIFT | SIFT | LIFT | | .629 | .339 | | .644 | .337 |
| LIFT | LIFT | LIFT | **.446** | **.686** | **.374** | **.447** | .683 | **.369** |

The metrics used are:

- Rep. (Repeatability of feature points): the ratio of key-points that are found consistently in the shared region.
- NN mAP (Nearest Neighbor mean Average Precision): Area Under Curve (AUC) of the Precision-Recall curve, using the Nearest Neighbor matching.

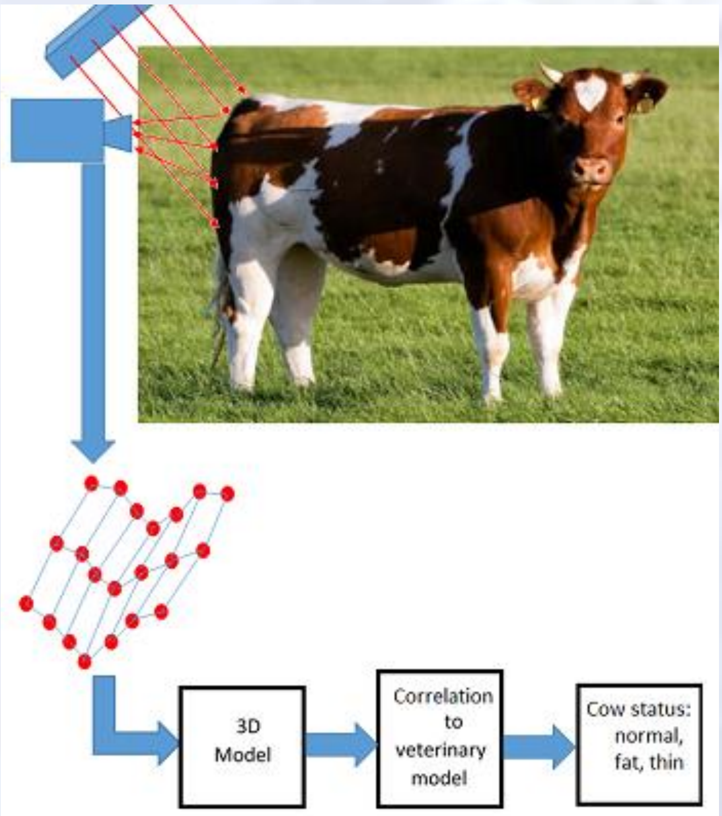
Results show that training the pipeline as a whole is important for optimal performance.

# Cow Feeding and Optimization of Milk Output

Every month, Computer Vision News reviews a successful project. Our main purpose is to show how diverse image processing applications can be and how the different techniques can help to solve technical challenges and physical difficulties. This month we review a software for **Cow Feeding and Optimization of Milk Output**, developed by **RSIP Vision** for one of its **Precise Agriculture** clients. Do you have a project in computer vision and image processing? Contact our consultants.

Our client wanted a system to provide the optimal quantity of food to cows, in order to **maximize the production of milk for each individual animal**. This involves giving each cow the quantity of food which best fits it: a fat cow does not increase production by overfeeding it. A thin cow does not generally produce as much milk as it would, were it properly fed. Bringing these two cows into the "normal" group prevents food waste on one side and increases milk output on the other.

A veterinary study found a correlation between the cow's fat and the size of her rear part. This enables to estimate how fat the cow is by measuring the puffiness of its rear, which is done by technologies of **image processing**.




At the beginning we wanted to photograph the part, then we understood that the simplest solution was a **laser scanner**: 3 times a day, the cow slowly walks its way through the milking process line. That is the ideal spot for a laser scanner, which scans red markers on the cow's back at video frequency (25 frames per second).

The process developed by **RSIP Vision** stitches together 10 or 20 images into one, in which the markers describe the extent of the cow's back. This is compared to the **veterinary's model** to assess if the cow is fat, thin or normal. With this data in hand, the veterinary decides the daily quantity of food intake which is proper for that cow.

The challenge is not as easy as it sounds: lighting conditions in a cowshed are far from optimal and dirt is everywhere, which makes the problem very different from regular conveyor belt imaging. **Using a laser scanner solved those problems**: the devices scans the red markers and disregards all kinds of noise.

The result proved **extremely effective** and allowed our client to sell his system in international markets: first, it is entirely automatic and does not require any intervention. Second, it is fast. Finally, being repeated 3 times per day, it gives a very quick feedback about the cow's condition and enables immediate corrective measures about food intake.

## ICIP: International Conference on Image Processing

Every month, Computer Vision News points the spotlights on a specific event which we think deserves the attention of our readers. The Event of the Month of September is ICIP, the International Conference on Image Processing which will be held in Phoenix, Arizona on September 25-28. You can visit the event page with speakers, schedule, topics, registration and venue information, but before you do we suggest you read what **Professor Lina J. Karam, Professor at the Electrical, Computer & Energy Engineering at the Arizona State University and General Chair of IEEE ICIP 2016**, told us about it.

**Computer Vision News:** *The technical program at ICIP is very rich for those who are interested in* ***computer vision****. What brings presenters to display their work at ICIP rather than at CVPR, held only 2 months before?*

**Lina Karam:** IEEE ICIP is a more comprehensive conference than CVPR and covers a wide array of topics including image/video processing, image/video communication, computer vision, computational imaging, biomedical imaging, visual-based forensics and security, perception-based visual processing, visual quality assessment, and other visual-based applications. IEEE ICIP contributes and seeks to advance



ICIP 2016
2016 IEEE International Conference on Image Processing

the theory and practice in above areas. This also enables networking and collaboration among attendees working on various vision-related fields. Most of the above topics (quality, perception, processing, ...) are very important for computer vision.

As with CVPR, there is a significant increase at IEEE ICIP in work related to **machine learning based computer vision** including **deep learning based visual processing and analysis**.

**CVN:** *What are the most captivating cross-disciplinary connections which computer vision people might expect to find at ICIP?*

**Karam:** Computer vision people would be able to interact and connect with people working on a variety of disciplines including perception and cognition, probability and statistics, signal processing, transforms, machine learning, visual quality, reconstruction, 3D, HDR, VR, embedded real-time processing, image/video processing, image/video analysis, image/video compression and transmission, to name a few.

**CVN:** *What kind of novelties can we expect to find at ICIP this year?*

**Karam:** Some novel initiatives include:

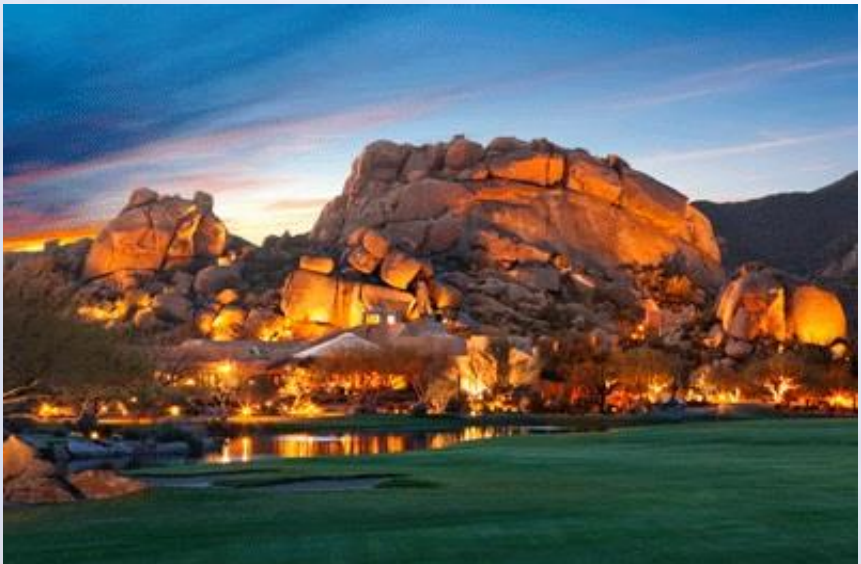
Open Preview: ICIP 2016 papers are published on IEEE Xplore late August and available freely for download to everyone about a month before the conference (with the exception of a few papers that opted out of the Open Preview).

Tutorials and Student Tutorials: IEEE ICIP 2016 is offering a Tutorial-Only registration for people wishing to attend a tutorial without attending the full conference. Students attending ICIP will also be eligible to receive one free tutorial as part of their registration (subject to space availability).

Visual Innovation Award: The World's First Visual Innovation Award Finalists are now listed on the ICIP2016 website.

Half day will be exclusively reserved to **Panel Discussions on controversial topics** including the following panels:

1. **Compressive Sensing** 10 Years Later: Has it Changed Image Acquisition and Processing?
2. **Deep Learning**: Is this the end or the beginning of Computer Vision?
3. Is **Compression** Dead, or Are We Wrong Again?
4. Quantity vs Quality in **Publications**: Should we be publishing less?



*The Boulders Resort: Venue of the Award Dinner & Show*

*Edward J. Delp*
*Purdue University*
*Moderator of Panel 3*

Event


*Anna Scaglione*
*Arizona State University*
*Moderator of Panel 4*

Event

***"4 Panel Discussions on controversial topics"***

**CVN:** *In your opinion, what makes ICIP special?*

**Karam:** ICIP is the flagship conference of the **IEEE Signal Processing Society**, the first society that was formed in the IEEE.

The ICIP community includes researchers, developers, product creators, educators, and students from diverse background, who want to share, learn about, and advance the state-of-the-art in the areas of image/video processing, image/video communications, computer vision, computational imaging, biomedical imaging, and visual technologies based applications. Papers and presentations at IEEE ICIP are expected to be of high quality, to spur discussions, and to significantly contribute to the advancement of these areas.

**CVN:** *I heard that this year the keynote speakers choice is particularly exciting.*

**Karam:** Yes, the IEEE ICIP 2016 Innovation Program Chairs put together an Innovation Program with **twelve Keynote Speakers** to share with attendees their insights on the future vision innovations.

The speakers are: Hanno Basse (CTO, 20th Century Fox), Bo Begole (VP, Huawei), Achin Bhowmik (VP, Intel), Bill Dally (SVP, nVidia), Michael Antonov (Co-Founder, Oculus), C.–C. Jay Kuo (Dean's Professor, University of Southern California), Matthew Mengerink (VP, Google), Tim Milliron (VP, Lytro), Anthony Park (VP, Netflix), Jamie Shotton (Co-Inventor of Kinect, Microsoft), Raj Talluri (SVP, Qualcomm) and Susie Wee (VP and CTO, Cisco).

***"ICIP is a must-have experience to anyone working in the computer vision field!"***

**CVN:** *What else don't we know about ICIP?*

**Karam:** It is hard to explain the ICIP experience in words. ICIP is a **must-have experience** to anyone working in the computer vision field!


## FREE SUBSCRIPTION

Dear reader,

Do you enjoy reading Computer Vision News? Would you like to receive it **for free in your mailbox** every month?

**Subscription Form**
(click here, it's free)

You will fill the Subscription Form in **less than 1 minute**. Join many others computer vision professionals and receive all issues of Computer Vision News as soon as we publish them. You can also read Computer Vision News on our website and find in our archive new and old issues as well.


**We hate SPAM and promise to keep your email address safe, always.**

**First European Machine Vision Forum**
Heidelberg, Germany Sept. 8-9 Website and Registration

**Vision Demo Night: Boston Imaging and Vision**
Cambridge MA, USA September 15 Website and Registration

**International Conference on Computer Vision and Graphics**
Warsaw, Poland Sept. 19-21 Website and Registration

**British Machine Vision Conference: BMVC 2016**
York, UK Sept. 19-22 Website and Registration

**RE•WORK Deep Learning Summit**
London, UK Sept. 22-23 Website and Registration

**International Conference on Image Processing: ICIP 2016**
Phoenix AZ, USA Sept. 25-28 Website and Registration

**ECCV - European Conference on Computer Vision**
Amsterdam, Netherlands Oct. 8-16 Website and Registration

**RE•WORK Women in Machine Intelligence & Healthcare**
London, UK Oct. 12 Website and Registration

**MICCAI - Medical Image Computing and Computer Assisted Intervention**
Athens, Greece Oct. 17-21 Website and Registration

**RE•WORK Deep Learning Summit**
Singapore Oct. 20-21 Website and Registration

**International Conference on Computer Vision and Image Processing**
Malaga, Spain Oct. 20-21 Website and Registration

**ACIVS - Advanced Concepts for Intelligent Vision Systems**
Lecce, Italy Oct. 24-27 Website and Registration

**HCOMP 2016 - Human Computation**
Austin TX, USA Oct. 30-Nov. 3 Website and Registration

**RE•WORK Machine Intelligence Summit**
New York NY, USA Nov. 2-3 Website and Registration

**International Conference on Computer Vision and Image Processing**
Paris, France Nov. 21-22 Website and Registration

**ACCV'16: The 13th Asian Conference on Computer Vision**
Taipei, Taiwan Nov. 20-24 Website and Registration

**International Conference on Pattern Recognition and Computer Vision**
London, UK Nov. 24-25 Website and Registration

Did we miss an event? Tell us: editor@ComputerVision.News


Improve your vision with
Computer Vision News
The Magazine Of The Algorithm Community
The only magazine covering all the fields of the computer vision and image processing industry
Subscribe
(click here, it's free)
RE•WORK
% invoke the matlab debugger
function STOP_HERE()
[ST,~] = dbstack;
file_name = ST(2).file; fline = ST(2).line;
stop_str = ['dbstop in ' file_name ' at ' num2str(fline+1)];
eval(stop_str)
Mapillary
Building
Traffic Sign
Car
License Plate
Gauss Surgical
A publication by
RSIP VISION