CS7.505.S22: Computer Vision | Mid-Quiz

Author: Aditya Kumar Singh
ID: 2021701010

March 7, 2022

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Contents

Question 1

Why does the literature in SURF suggest taking det($\mathcal{H}$) (i.e., determinant of Hessian) for feature detection and why not *trace*($\mathcal{H}$) = Laplacian filter as used in SIFT and how robust this choice is against Harris-based Gradient matrix and *trace*($\mathcal{H}$), and why?

Novelty:

- Addressing this question tries to exploit on what basis SURF is superior to Harris corner detection and SIFT.
- And how the determinant of Hessian is rating the "interestingness of a point".

Challenges:

- Here we'll discuss the core process and mathematical formulation used to estimate the interest points.
- Plus, we'll demonstrate the geometrical intuition behind det($\mathcal{H}$).

Solution:
Interest points are those points where we can witness considerable variation in pixel intensity in more than one direction which basically suggest it to be a "corner-based" or a contrast point feature. To detect such interesting points, we have read about Harris corner detector that use gradient matrix of squared single derivatives terms, also **SIFT** that applies *trace*(Matrix) = Laplacian operator to find edges and some further preprocessing to find interest points. Now in case of **SURF** we're using the whole Hessian matrix and compute it's det to know the degree of interestingness of that point. Why so? Let's go through a basic concept in image processing i.e., the single derivative gradient ($\frac{\partial I}{\partial x}$ or $\frac{\partial I}{\partial y}$) considers not only abruptly changing intensity but also the smoothly changing intensity (like *Ramp*) as interesting feature whereas double derivative don't consider smoothly changing intensities which again shows us why most of the traditional feature detectors (VLAD) used *Laplacian or LoG* as their initial step in feature detection. Also less no. of well-deserved points always is a plus to efficiency of algorithm.

Further, how is Hessian efficient? After *David Lowe's* success with LoG approximation (i.e., DoG) in SIFT, the same approach but with a little tweak i.e., instead of *trace*(Hessian) = LoG, they (Bay et al.) computed det(Hessian) on downsampled images (i.e., gaussian smoothed images at different scales) or equivalently applying upsampled filter on same image. Now why they did that, we'll detail upon that in next paragraph. Further, they cropped differently scaled second order gaussian filter obtained for each term in Hessian matrix to a "box-filter" [1] instead of discretizing and upon evaluation the performance they got was comparable .

$$\det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{1}$$

Now this step is directly linked with the "integral image" concept so as to make it computationally more fast.

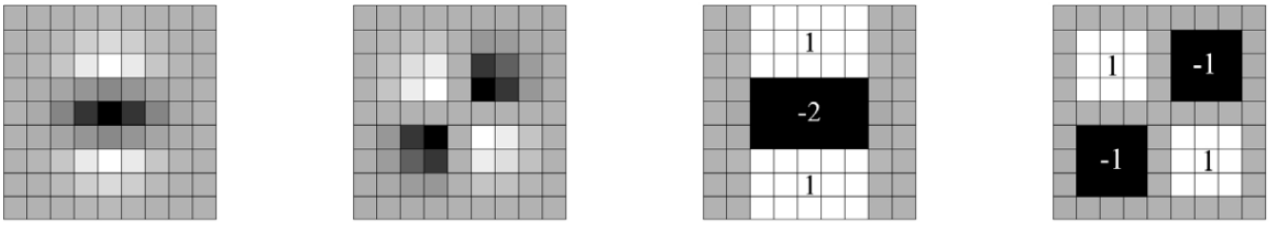

Figure 1: Left: Discretized Images; Right: Cropped Images

Now let's know the **intuition behind taking determinant** of Hessian matrix, $\mathcal{H}$. Generally, it is a second order partial derivative matrix defined upon a function, $f \in C^2$ (space of functions that are twice derivable w.r.t all it's inputs and it's derivative so obtained, are continuous). Denoted as:

$$\mathcal{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{2}$$

This matrix is more rigorously used in multi-dimensional *Taylor Series* expansion as stated below:

$$f(x + \Delta x) = f(x) + \nabla f(x)\Delta x + \frac{1}{2}(\Delta x)^T \mathcal{H}_f \Delta x + \mathcal{O}(|\Delta x|^3) \tag{3}$$

where $\Delta f(x)$ is the gradient of f at x, $\mathcal{H}_f$ is the Hessian of f at x and symmetric, of course. (*Note: Since symmetric, eigen values will be real and eigen vectors should be othogonal according to* ***spectral theorem***). So the behaviour of f near x depends largely on the quadratic form $Q(\Delta x) = (\Delta x)^T \mathcal{H}_f \Delta x$. Following is what all conclusions can we have on $f(x)$ and it's surrounding:

1. If $Q(\Delta x)$ is > 0 (i.e., $\mathcal{H}_f(x)$ is positive definite = p.d.) $\iff$ All eigen values of $\mathcal{H}_f$ are $> 0 \implies$ f increases in every direction $\longrightarrow$ a local minimum.

2. If $Q(\Delta x)$ is < 0 (i.e., $\mathcal{H}_f(x)$ is negative definite = n.d.) $\iff$ All eigen values of $\mathcal{H}_f$ are $< 0 \implies$ f decreases in every direction $\longrightarrow$ a local maximum.

3. For mixed positive and negative (but all nonzero) eigenvalues, f increases in some direction and decreases in other.

4. Lastly, suppose that there exists some $\Delta x \neq 0$ such that $\mathcal{H}_f \Delta x \neq 0$. This is true $\iff$ $\mathcal{H}_f$ has a 0 eigenvalue.

Geometrically, if we interpret intensity function, $f = I(x, y)$, as a surface, then the eigen-values of $\mathcal{H}_I$, say λ_1 and λ_2, correspond to the *principal curvatures* of the surface at that point, (x, y),

meaning the values of maximum and minimum curvature at that point. The eigenvectors are orthogonal and correspond to the directions of maximum and minimum curvature. And the product of *principal curvatures* is ***Gaussian curvature*** $= \lambda_1\lambda_2 = \det(\mathcal{H}_I) = \frac{\partial^2 I}{\partial x^2}\frac{\partial^2 I}{\partial y^2} - \left(\frac{\partial^2 I}{\partial y \partial x}\right)^2$, whereas their average is the ***Mean curvature*** $= k.(\lambda_1 + \lambda_2) = k.trace(\mathcal{H}_I)$. To us Gaussian curvature is of *prime* importance as its *sign* tells us a lot about nature of surface (i.e., how the intensity varies in both eigen direction):

1. $\det(\mathcal{H}_\mathcal{I}) > 0 \implies$ Surface have an *elliptic* point.
2. $\det(\mathcal{H}_\mathcal{I}) < 0 \implies$ Surface have a *hyperbolic or saddle* point.
3. $\det(\mathcal{H}_\mathcal{I}) = 0 \implies$ Surface have a *parabolic* point.

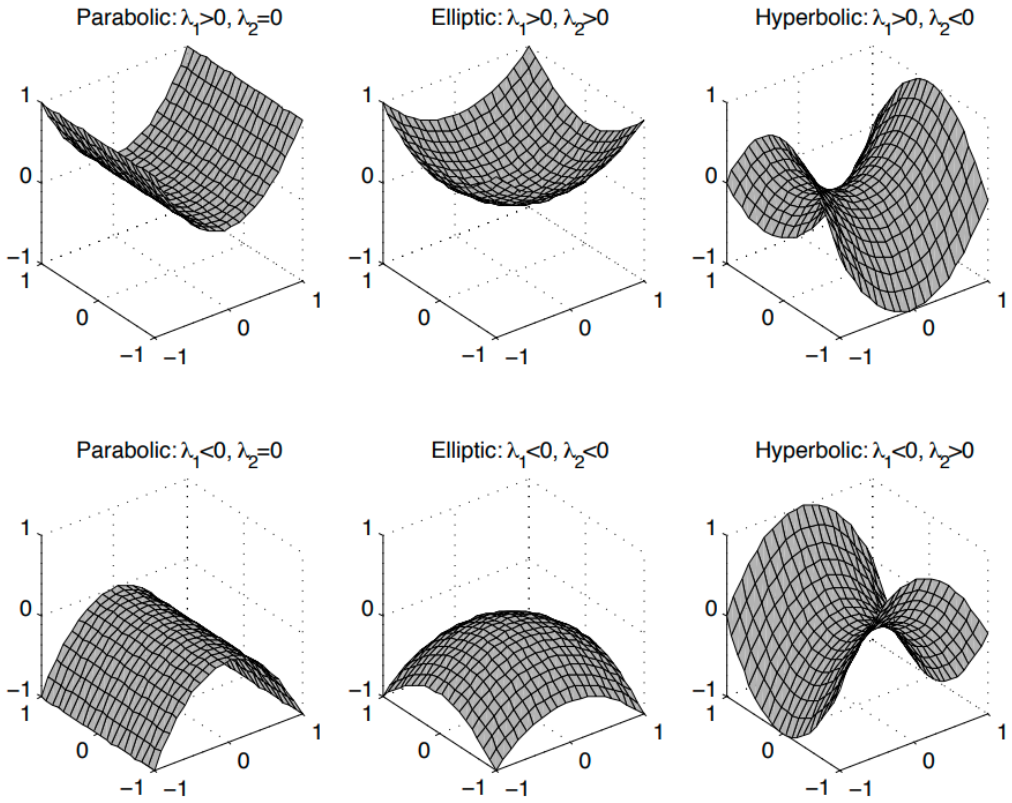

Figure 2: Eigenvalues (λ_1, λ_2) of the Hessian correspond to the principal curvatures of a surface and can be used to classify different surface types.

According to the above diagram, the pixels where their surrounding behaves like a paraboloid surface (as in 2^{nd} column) are the ones ("ideal ones") that behave like a corner. While the *Mean curvature* $= k.(\lambda_1 + \lambda_2) = k.trace(\mathcal{H}_I) =$ LoG considers all type of features including edges, corners, etc. (i.e., all type of columns shown above).

Question 2

Consider three images I_1, I_2 and I_3 captured by a system of three cameras, and suppose the fundamental matrices F_{13} and F_{23} are known. In general, given a point x_1 in I_1 and a corresponding point x_2 in I_2, the corresponding point x_3 in I_3 is uniquely determined by the fundamental matrices F_{13} and F_{23}. Then write an expression for x_3 in terms of x_1, x_2, F_{13} and

F_{23}. Also describe a degenerate configuration of three cameras for which the point x_3 cannot be uniquely determined by this expression.

Novelty:
Earlier we have seen how to correspond two image points, say $\mathbf{x_1}$ and $\mathbf{x_2}$, in two different plane using Fundamental matrix. Now, it will be using information from two image planes how to correspondence the 3rd image point, say $\mathbf{x_3}$, with $\mathbf{x_1}$, and $\mathbf{x_2}$

Challenges:

- 3-View geometry with more perspectives, meaning more epipolar lines and what all degeneracies to handle.
- Plus, we'll demonstrate how to use epipolar lines to estimate the 3rd image point, $\mathbf{x_3}$.

Answer:
First let's view the above problem in 3-view geometry:

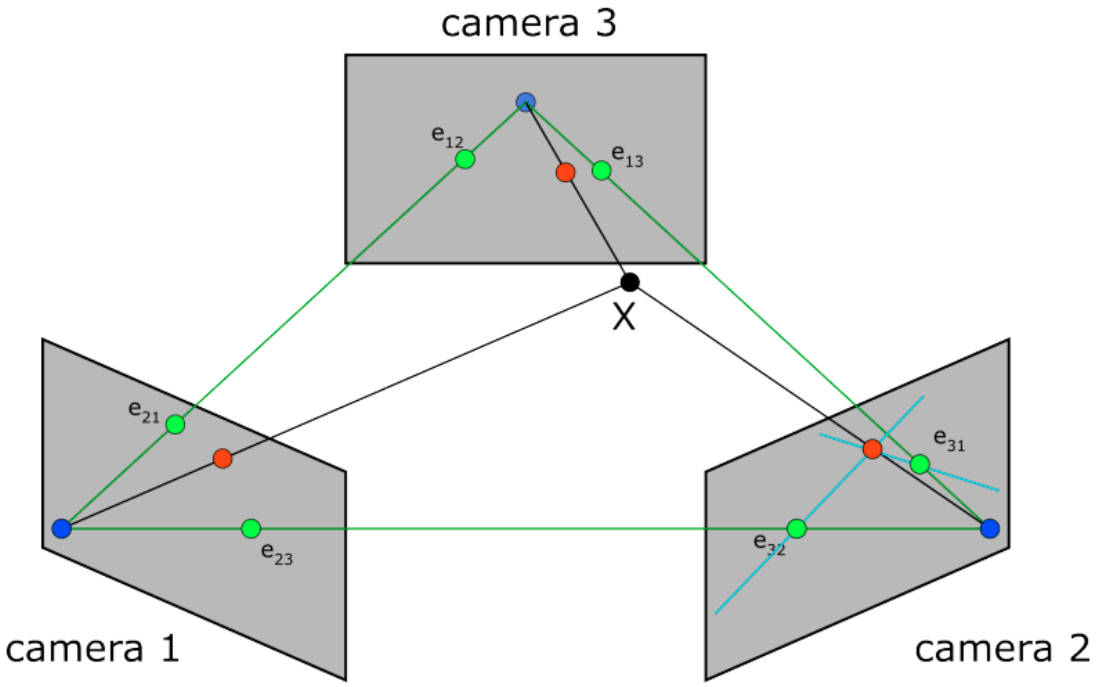

Figure 3: Projection of the point on camera 1 and 3 creates 2 epipolar lines that cross on camera 2

In perspective of 2-view geometry, the *Fundamental matrix* maps a point to its epipolar line. More formally, for each point $\mathbf{x}$ in one image, $\exists$ a corresponding epipolar line l' in the other image. Any point $\mathbf{x}'$ in the second image matching the point $\mathbf{x}$ must lie on the epipolar line l'. The epipolar line is the projection in the second image of the ray from the point $\mathbf{x}$ through the camera centre $\mathbf{C}$ of the first camera. Hence:

$$x \overset{F}{\mapsto} l' \tag{4}$$

According to the **correspondence condition** the fundamental matrix satisfies the condition that for any pair of corresponding points $\mathbf{x} \longleftrightarrow \mathbf{x}'$ in the two images, $\mathbf{x}'^T F\mathbf{x} = 0$. Where, $l' = F\mathbf{x}$ and $\mathbf{x}'$ lies on the epipolar line l'.

Now if we setup the above configuration in form of equation, where the fundamental matrix F_{ij} satisfies equation $\mathbf{x_j}^T F_{ij} \mathbf{x_i} = 0$ for any correspondence $\mathbf{x_i} \longleftrightarrow \mathbf{x_j}$ between images I_i and I_j, then:

$$\mathbf{x_3}^T F_{13} \mathbf{x_1} = 0 \tag{5}$$

$$\mathbf{x_3}^T F_{23} \mathbf{x_2} = 0 \tag{6}$$

From above, $F_{13}\mathbf{x_1}$ = the epipolar line, $l_{12,3}$ (line passing through epipole, e_{12} and x_3), and $F_{23}\mathbf{x_2}$ = the epipolar line, $l_{13,3}$ (line passing through epipole, e_{13} and x_3). As shown for image screen 2 in Figure 3, in image screen 3 too with same approach $l_{12,3}$ and $l_{13,3}$ intersect at point x_3. Hence, x_3 can be written as cross product of these two lines.

$$\begin{aligned} x_3 &= l_{12,3} \times l_{13,3} \\ &= F_{13}\mathbf{x_1} \times F_{23}\mathbf{x_2} \end{aligned}$$

Now coming to the degenerate configuration, epipolar transfer has a serious deficiency due to the degeneracy that can be seen from Figure 4.

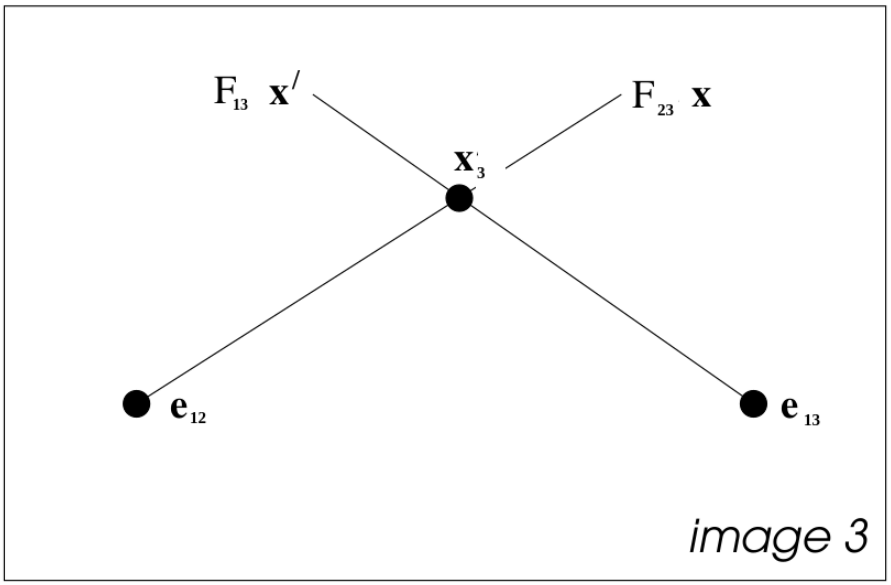

Figure 4: The configuration of the epipoles and transferred point x_3 as seen in the third image. Point x_3 is computed as the intersection of epipolar lines passing through the two epipoles e_{12} and e_{13}. However, if x_3 lies on the line through the two epipoles, then its position cannot be determined. Points close to the line through the epipoles will be estimated with poor precision.

Here epipolar transfer fails when the two epipolar lines in the third image are coincident (and becomes increasingly ill-conditioned as the lines become less "transverse"). The degeneracy condition that x_3 , e_{12} and e_{13} are collinear in the third image means that the camera centres C and C' and the 3D point **X** lie in a plane through the centre C" of the third camera; thus **X** lies on the trifocal plane defined by the three camera centres. Epipolar transfer will fail for points **X** lying on the trifocal plane and will be inaccurate for points lying near that plane.

Question 3

How PCA and **why** PCA in Face recognition?

Novelty:

- We'll get to know "intuitively" how PCA works.
- Plus, why we use Eigen-Decomposition i.e., how eigen-vector plays an important role here?
- How this is being turned into a **feature extraction** problem.

Challenges:
Brief overview on how PCA derives intuition from eigen-decomposition and project features into lower dimension.

Answer:
Taking flattened vector representation of a human face and matching them with vector representation of existing faces is formally a Template matching problem. Now if we embed these vectors in the corresponding euclidean space, then for sure we can obtain some pattern (or dependencies among features). For sake of simplicity, we assume a linear dependency which further lead us to the idea of "correlation" (or "covariance", the non-normalized version of correlation) among pair of features. Say X consists of flattened vectors of faces (mean-subtarcted and normalized) stacked up horizontally (# of samples = # of columns and # of rows = # of features). Then $Cov(X) = X_{n\times m}X^T_{m\times n}$, where, n = # of features, and m = # of samples.

Since we're expecting that some features would be interacting most among each other (i.e., having high correlation), one could always ask, "Is there any *hyper-plane* or any lower dimensional plane that captures these interactions?". In other words, can we project these vectors onto a lower dimensional space without much loss of information. And the answer is *yes* we can. There are certain features of human faces which are the basic constructional unit and can't be ignored, like eye, nose, placement of eyes, below nose mouth should be there, etc. And these commonalities as well as dependencies among features provides us the room to express them with less features. And the subspace (the lower dimensional space) which capture these patterns are spanned by some basis vectors which are nothing but the eigen-vectors of "**covariance-matrix**". How? Let's first understand why eigen-vectors point in the direction of maximal change or variation for a given matrix action and then we'll answer *How?*.

By definition the eigen-vectors are the direction in which vectors change upto a scale after matrix action. But if the matrix itself is *symmetric* then **norm of matrix is equal to its highest eigen value** or in other words the spectral radius is equal to the highest eigen value $\implies$ the maximum scaling in terms of magnitude is going to happen in the direction of the eigen-vector corresponding to max eigen-value. The second maximum scaling will be in direction of the eigen vector having second highest eigen-value and so on. For example, as in *Harris Gradient* matrix whose action is to capture max gradient direction, it's eigen vector so obtained points in maximum gradient. Likewise covariance matrix's eigen vector (whose eigen value is maximum) points in the direction of max correlation among the features as shown in Figure 5.

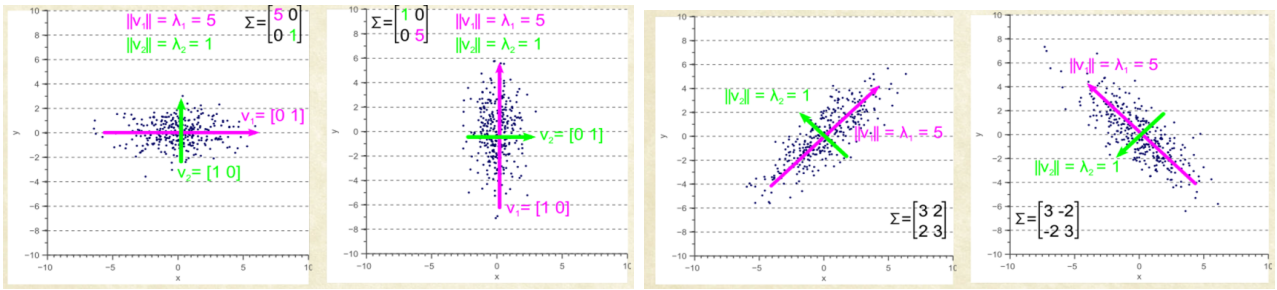

Figure 5: 2D spread can be effectively projected back to 1D representation; v_1 and v_2 are principal components

Representing the correlation among features through the eigen-vectors of covariance matrix captures overall semantics of our data without much reconstruction loss. This points us to some sort of "dimensionality reduction" and this is how **PCA** comes into picture. And then these eigen vectors are renamed as **principal components** or ghost-faces or eigen-faces or basic unit of a general face w.r.t to the domain of problem. What all we have discussed above, are the basic components of PCA algorithm which involves **data-preprocessing** (i.e., mean-subtraction and standardization) $\longrightarrow$ **SVD** (= computing covariance matrix + Eigen-value Decompostion) to arrive at a bunch of principal components. Now based on the eigen-values we rank the principal components and choose the first few that captures almost a major portion of variability (say, $> 95\%$). Due to this lower dimension represntation, an overall meaning of face is captured by discarding the minute details (or noise). And this add up for the *efficiency* of algorithm as well as reduce high-dimensional complicacies.

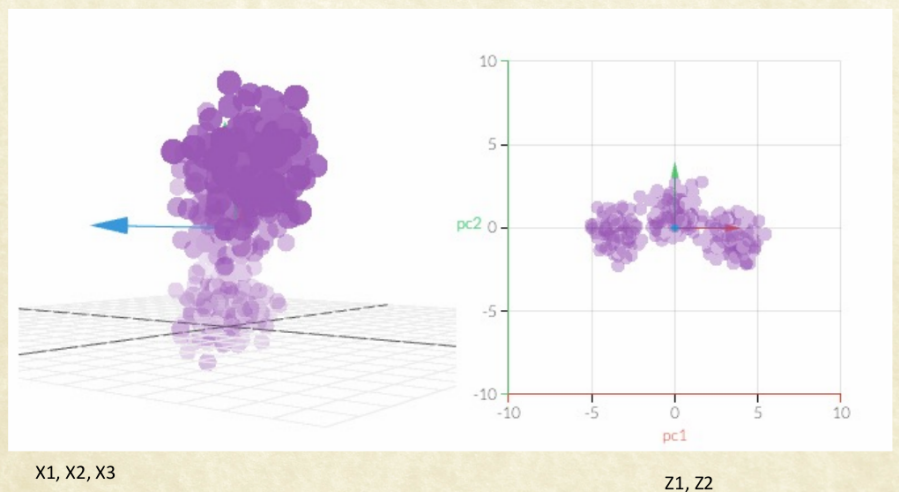

Figure 6: 3D to 2D represntation

Further, the assumption of "linear dependency" demands the face datasets to be aligned perfectly such that the local position of all facial features would locate in almost similar location for all the flattened vectors, which in itself is a limitation for PCA-based face recognition algorithm.