

# Visión por Computadora I

Ing. Andrés F. Brumovsky

([abrumov@fi.uba.ar](mailto:abrumov@fi.uba.ar))

Ing. Maxim Dorogov

([mdorogov@fi.uba.ar](mailto:mdorogov@fi.uba.ar))

Laboratorio de Sistemas Embebidos -FIUBA



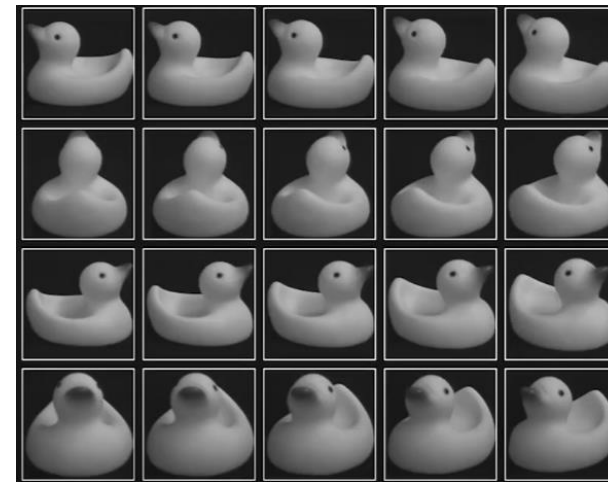
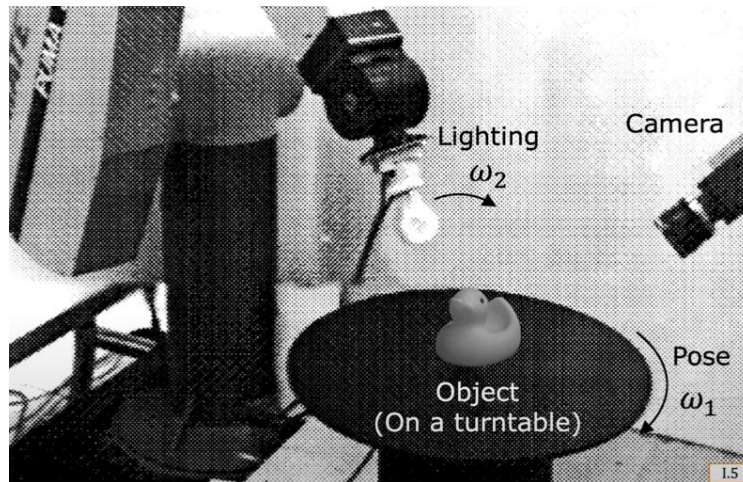
# DETECCIÓN DE OBJETOS

- Detección naive:

1. Template matching: Técnica simple invariante a desplazamientos pero no invariante a cambios de escala (se puede solventar en cierta medida utilizando pirámides) rotación y otras transformaciones. A partir de un template y una métrica de similitud se buscan máximos o mínimos de intensidad en la correlación del template con la imagen.

Una mejora propuesta por Hiroshi Murase:

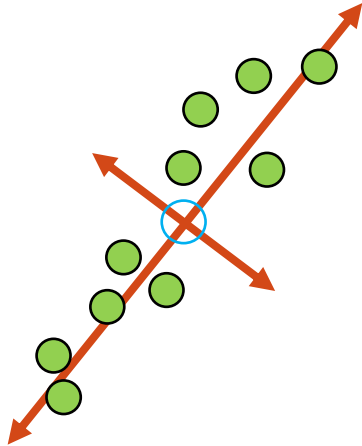
1. Appearance matching: Capturamos un dataset de imágenes con múltiples vistas de un objeto bajo diferentes condiciones de iluminación ([Visual Learning and Recognition of 3D Objects from Appearance, Murase 1994.](#)) Si solo nos quedamos con esto vamos a tener que probar cientos de templates por cada imagen en la cual queremos detectar el objeto de interés.



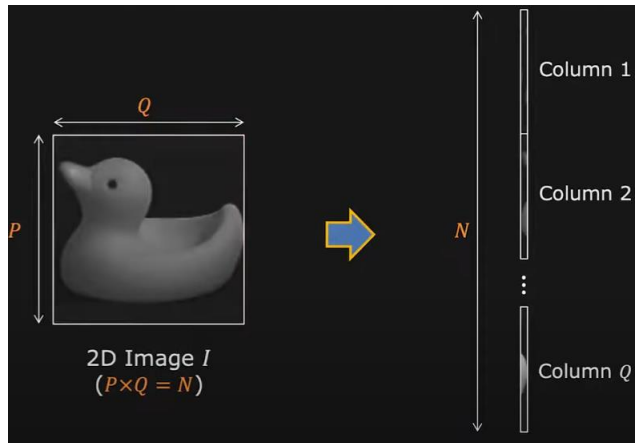
El dataset obtenido presenta redundancias que podemos aprovechar aplicando técnicas de reducción de dimensionalidad (PCA) y obtener una representación mas compacta que caracteriza el objeto de interés.



# PCA (PRINCIPAL COMPONENTS ANALYSIS)



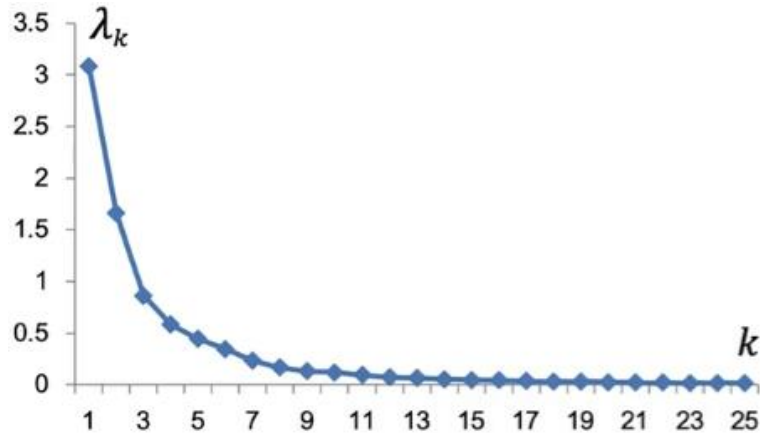
- Los modelos generativos funcionan mejor en espacios con pocas dimensiones (estimar las funciones de densidad en espacios de muchas dimensiones requieren una gran cantidad de datos)
- PCA es una técnica de reducción de dimensiones
  1. El primer componente principal es la dirección de máxima varianza (alrededor de la media).
  2. Los siguientes componentes principales son ortogonales a los previos y describen la máxima varianza residual
- Definimos:
- Matriz de características:  $F = [f_1, f_2, \dots, f_M]$  ( $N \times M$ )
- Matriz de covarianza:  $R = F * F^T$  ( $N \times N$ )
- Problema a resolver:  $Re = \lambda e$  (encontrar autovalores y autovectores de R)
- Autovalores:  $\{\lambda_1, \lambda_2, \dots, \lambda_K\}$
- Base ortonormal de autovectores:  $\{e_1, e_2, \dots, e_K\}$



Pregunta: Cuantos k-valores tomamos para definir el nuevo espacio de características ?



# PCA (PRINCIPAL COMPONENTS ANALYSIS)



## Elección de dimensión del espacio PCA

- A medida que aumenta K el aporte del autovalor asociado es cada vez menor debido a las redundancias presentes en el dataset.
- Podemos representar cada autovector como una imagen

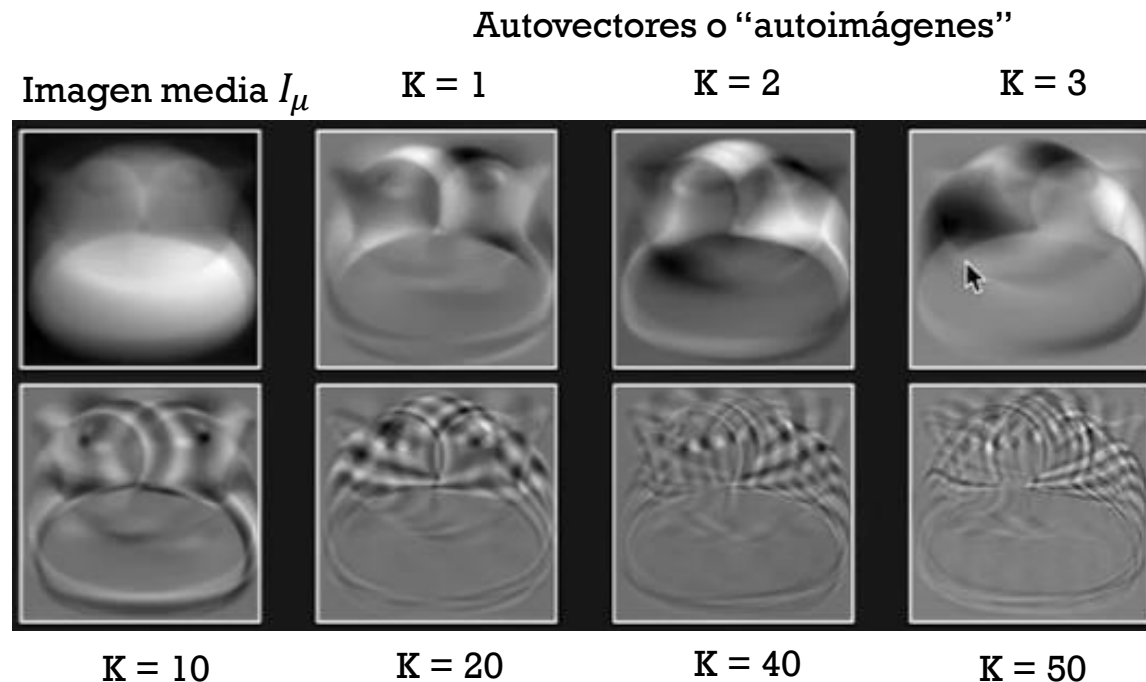
- Si queremos capturar el 95% de las variaciones presentes en el dataset → Buscamos el K mas chico que cumpla con:

$$\frac{\text{Suma de } K \text{ autovalores mas grandes}}{\text{Suma de todos los autovalores}} = \frac{\sum_{i=1}^K \lambda_i}{\sum_{i=1}^N \lambda_i} \geq 95\%$$

- Proyectamos cada imagen  $f(w)$  del dataset sobre el espacio de autovectores:

$$p(w) = [e_1, e_2, \dots, e_K]^T * (f(w) - I_\mu)$$

Con esta operación pasamos de una dimensión  $R^N$  a  $R^K$  con  $K \ll N$ .

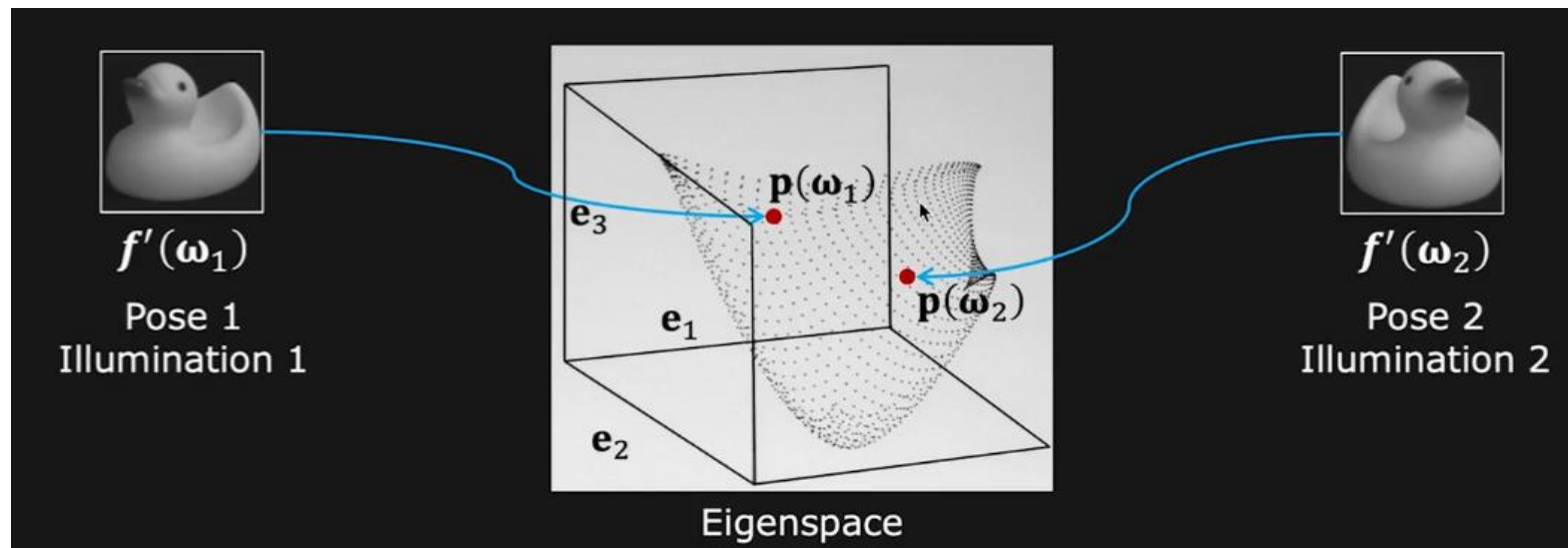




# PCA (PRINCIPAL COMPONENTS ANALYSIS)

Dada un espacio de autovectores y una imagen de entrada I:

1. Normalizamos la imagen para independizarnos del contraste (pasaje a coordenadas cromáticas)  $I' = I / |I|$
  2. Se convierte a un vector, se le resta la imagen media del espacio y se proyecta sobre el espacio obteniendo un  $p'(w)$
  3. Se busca el  $p(w)$  que minimice la distancia con  $p'(w)$  y se asigna esa clase a la imagen de entrada
- Extra: Se pueden mapear todas las imágenes del dataset al espacio y obtener un modelo paramétrico  $P(W)$  (K dimensional) como función de  $w$ . (El conjunto de datos queda representado por  $I_\mu$ , los autovectores y  $P(W)$ .)

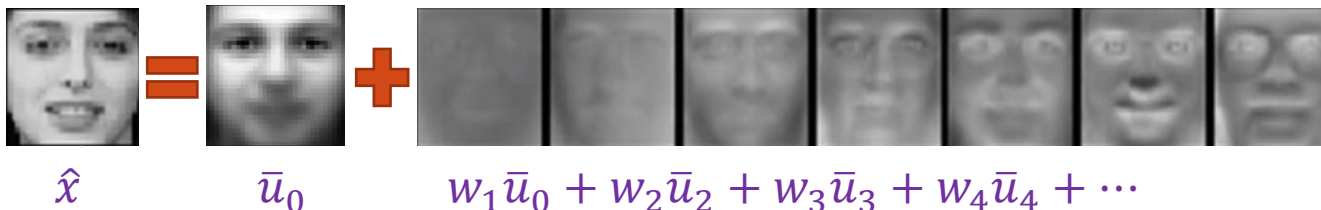


# EIGENFACES

- Mathew A. Turk y Alex P. Pentland – 1991
- 1. Asumen que la mayoría de las caras se encuentran en un subespacio de dimensión baja definido por las primeras  $k$  direcciones de máxima varianza ( $k \ll d$ )
- 2. Usan PCA para determinar los autovectores (o “eigenfaces”)  $u_1, u_2, \dots, u_k$  que representan el subespacio.
- 3. Representan todas las imágenes de rostros en el conjunto de datos (dataset) como combinaciones lineales de eigenfaces
- Imágenes de entrenamiento  $\bar{x}_1, \dots, \bar{x}_M$
- Las eigenfaces son en realidad vectores (arreglados con matriz y mostrados como imagen) de 10.000 posiciones (en este ejemplo)
- Si quisiéramos hacer el producto interno (proyección) de una cara nueva  $\bar{x}$  en este espacio de eigenfaces solo tenemos que superponer la imagen de la cara sobre cualquiera de estas eigenfaces y multiplicar punto a punto.

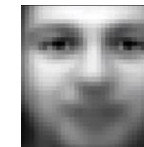
$$\begin{aligned}\bar{x} &\rightarrow [\bar{u}_1^t(\bar{x} - \bar{u}_0), \dots, \bar{u}_k^t(\bar{x} - \bar{u}_0)] \\ &= [w_1, \dots, w_k]\end{aligned}$$

- Donde estos pesos  $w_i$  son la representación de la cara, de 10.000 valores a  $k$

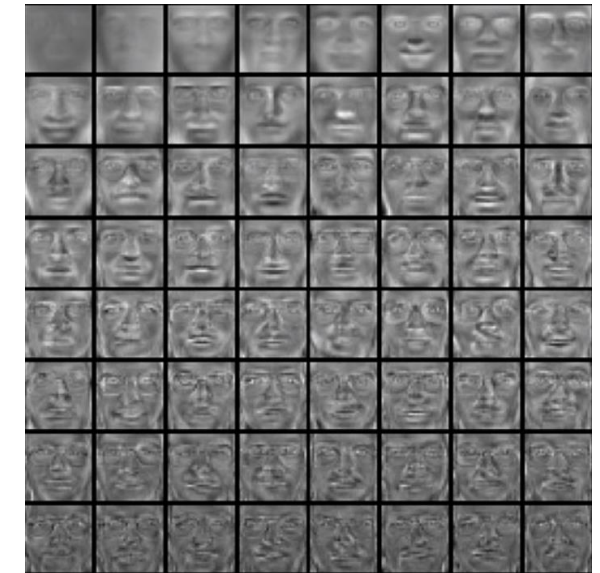


The diagram illustrates the Eigenface model. On the left, a face image  $\hat{x}$  is shown. This is followed by an equals sign, then the average face image  $\bar{u}_0$ , followed by a plus sign, and then a row of eigenface images. Below the eigenfaces is the expression  $w_1\bar{u}_0 + w_2\bar{u}_2 + w_3\bar{u}_3 + w_4\bar{u}_4 + \dots$ . Note that the first term in the sum should be  $w_1\bar{u}_1$  based on the context.

Dataset



Cara  
promedio  
( $\bar{u}_0$ )



Eigenfaces:  $u_1, \dots, u_k$



# EIGENFACES



$$\bar{u}_1, \dots, \bar{u}_k$$



$$\bar{u}_0 + 3\sigma_k \bar{u}_k$$



$$\bar{u}_0 - 3\sigma_k \bar{u}_k$$

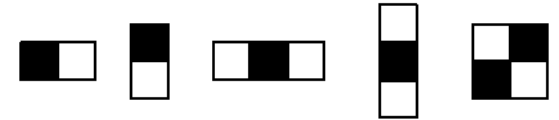
- ¿Cómo lo usamos para reconocimiento?. Dada una nueva imagen  $\bar{x}$ 
  1. Proyectamos en el subespacio:  $[w_1, \dots, w_k] = [\bar{u}_1^t(\bar{x} - \bar{u}_0), \dots, \bar{u}_k^t(\bar{x} - \bar{u}_0)]$
  2. Opcionalmente podemos verificar el error de reconstrucción  $\bar{x} - \hat{x}$  para determinar si es o no una imagen real
  3. La clasificamos como la cara de entrenamiento más cercana en el espacio k-dimensional
  4. Podemos pensar en este procedimiento como en un modelo generativo (se modelan las poblaciones)
- Contras
  1. No es robusto frente a desalineaciones
  2. No es robusto frente a fuertes variaciones del fondo
  3. A veces la dirección de máxima varianza no es la óptima para la clasificación



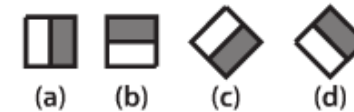
# FILTROS DE HAAR

- Buscan detectar patrones de brillo como características de la imagen (diferencias de intensidad entre zonas adyacentes)
- Se pueden definir a múltiples escalas y posiciones
- El resultado del filtro es la diferencia en la suma de los valores de los píxeles entre zonas claras y oscuras
- Se pueden computar muy rápidamente utilizando imágenes integrales → solo tres operaciones de suma para encontrar el área de cualquier tamaño de rectángulo

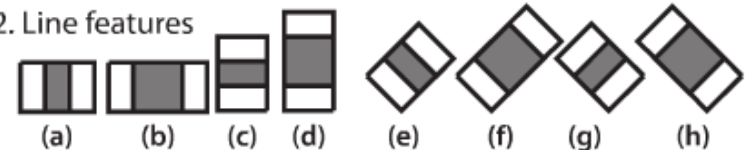
## Filtros de Haar Básicos



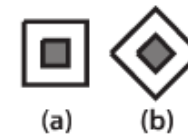
### 1. Edge features



### 2. Line features



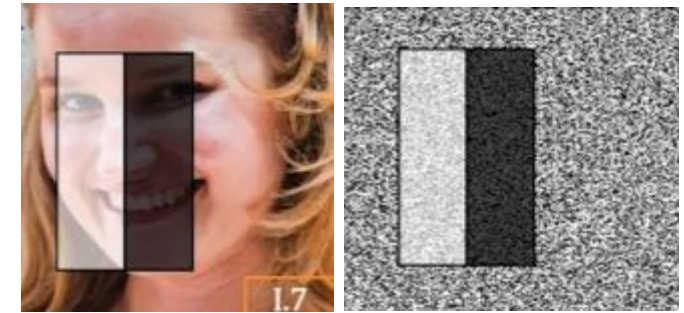
### 3. Center-surround features





# FILTROS DE HAAR

- [Viola Jones, 2001](#)
- Consideran una ventana de 24x24 pixeles como tamaño de partida
- Se optimiza el cálculo de características mediante **imágenes integrales**
- Se obtienen + 160.000 características (weak classifiers)
- Utilizan Adaboost para construir el mejor clasificador con la combinación lineal de todas las características
- Las características obtenidas son sensibles a la direccionalidad de los patrones que detectan

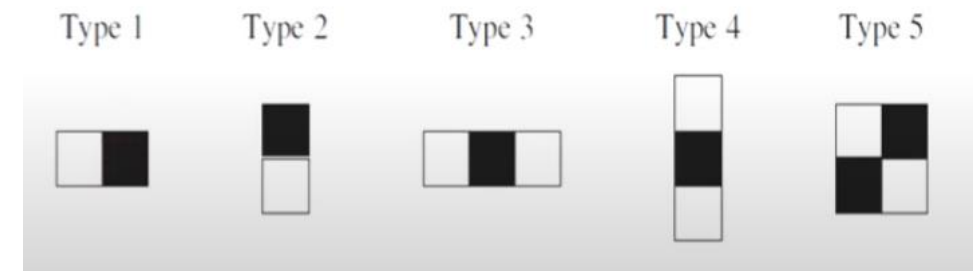


$V = 64$

$V \approx 0$



Características: Muy relevante a la izquierda y poco relevante a la derecha.



Filtros de Haar Básicos utilizado en el paper de Viola Jones.



# MEANSHIFT (DETECCIÓN Y TRACKING)

## Motivación

- Necesidad de detectar objetos sobre una secuencia de imágenes en tiempo real.
- Es difícil de lograr a partir de un único template en objetos de mucha variabilidad (non-rigid objects), necesitamos actualizarlo a medida que transcurre la escena.

## Algoritmo

- Se toma una ROI que contenga el objeto a detectar
- Extracción de características: Se convierte la ROI a HSV se calcula el histograma para el Hue.
- Se calcula la retroproyección del histograma sobre el frame actual
- Meanshift busca la zona donde se maximiza la retroproyección de manera iterativa
- Se actualiza la ROI y se repite el proceso sobre un nuevo frame



# MEANSHIFT (DETECCIÓN Y TRACKING)

## Pros

- Tracking en tiempo real
- Bajo uso de recursos, fácil de implementar
- Se puede combinar con otras técnicas ([meanshift-Kalman](#)) para robustecer el algoritmo
- Invariante a rotación (siempre que no cambie la distribución de características)

## Contras

- Dependiente de la cantidad de bins del histograma
- Se asume que los desplazamientos son en un entorno de la ventana
- Falla cuando hay oclusión total o parcial del objeto a detectar
- No es invariante a la escala
- La performance se ve afectada en escenarios con mucho ruido



# TP6

- Una empresa de transporte quiere monitorear la actividad de sus conductores durante el manejo. Para ello se propone implementar un detector de somnolencia en tiempo real tomando como datos de entrada frames provenientes de una cámara que apunta al rostro del conductor.
- 1. Implementar un detector Haar de caras (vista frontal)
- 2. En esa ROI detectar los ojos (Se puede usar otro detector Haar).
- Mediante alguna de las técnicas vistas en clase detectar si ambos ojos están abiertos o cerrados y mostrar el estado de la detección sobre el frame.

Algunos modelos de detectores pre entrenados se pueden encontrar [aca](#)

