

Block 1: Image basics

Session 1

- Acquisition module : La forma en la que podemos físicamente capturar una imagen.
- Focal length: En el pin-hole model camera, es la distancia que hay desde donde se proyecta la imagen con el hueco de la caja. Esta distancia determina cómo la imagen se formará. Cuanto más reducida sea, mayor será el FOV, cuanto más aumentada, menos FOV. Por esta razón, a más FOV, más cosas tienes que “comprimir”, por lo que las cosas más lejanas las verás más pequeñas.
- Lenses: Tiene imperfecciones, algunas “defractan” la luz. Otra imperfección es la optical aberration, que forma un efecto de “ojo de pez” o barrel distortion. Esto se soluciona con la cámara calibration.
- Sensor: Tenemos una capa de elementos fotosensitivos que generan pulsos de electricidad cada vez que reciben la luz, según lo fuerte de la intensidad de la luz podemos diferenciar entre los colores.
- Sampling and quantization: Sampling es cuántos píxeles tenemos, a más píxeles, más calidad tenemos. Quantization se refiere a la idea de cuántos bits por píxeles tenemos al hacer la conversión de continuo a digital. En este curso, usamos 8 bits que definen un rango de $[0,255]$, pero hay cámaras que usan 16 bits, etc. Por lo que, se entiende que usamos 8 bpp.
- Video grabbers: Transfiere la información de las imágenes/electricidad a la memoria de nuestros computadores, esto está formado por diferentes protocolos.

Session 2

- Una imagen se puede representar con una matriz de píxeles, en donde cada pixel tenemos un valor de 0 a 255 si estamos en blanco y negro o 3 componentes con 3 valores de 0 a 255 para poder representar imágenes con colores.
- Usaremos unsigned char para representar color. En color, tenemos 3 componentes en cada pixel que representan 8 bits, ya que las 3 van del mismo rango de 0 a 255.
- El valor alpha controla la transparencia.
- Los megapíxeles son el número total de píxeles de la imagen, esto es multiplicar las filas por las columnas. La ganancia de megapíxeles en términos de calidad es logarítmica, esto es, que si tienes pocos píxeles y aumentas un poco el número de filas y columnas, consigues mucha ganancia de calidad, pero si tienes muchos píxeles, un incremento de las filas y columnas resulta en aumentar mucho el número de píxeles pero sin ganar mucha calidad.
- A la hora de almacenar la imagen en términos de vectores, se guarda al final de cada fila un "offset", que no debemos de tocar.
- La píxel connectivity es la conectividad de píxeles, es intuitivo.
- Pixel wise operation, son operaciones que aplicamos en TODOS los píxeles de una imagen. Un ejemplo es añadir una constante, que es añadir a todos los píxeles de la imagen ese valor, pero debemos de tener cuidado en estas operaciones de no cometer desbordamiento, básicamente, que no nos pasemos del 255. También hay otras operaciones como suma de imágenes, multiplicación de imágenes, resta de imágenes, etc.
- Importante recordar que podemos tomar como funciones estas operaciones pixe-wise
- Otras pixel wise operation son por ejemplo la invertida de una imagen. Que toma el valor actual y se le resta a 255 para calcular su valor inverso.
- Threshold: Es una función de salto, en base a un valor, se cambian o no el resto de píxeles, dependiendo de si son superiores o inferiores a este.
- El contraste de una imagen es la relación entre los blancos y los negros, una imagen bien contrastada es una imagen que tiene una buena distribución de los blancos y los negros.

- La función logarítmica permite que los valores oscuros pasen a ser más brillantes pero aquellos que ya son blancos no cambiarán mucho.
- La función exponencial realiza lo contrario, donde podemos tomar una imagen con overexposure (con mucha luz) y obtener una más oscura.
- Media de imágenes: Es otra pixel-wise operación que consiste en tomar un conjunto de imágenes de una misma cosa y realizar la media de los valores de cada pixel de cada una de las imágenes, permitiendo con esto eliminar ruido, en concreto permite eliminar el ruido gaussiano y se usa también en background subtraction.
- Background removal: Consiste en tomar imágenes previas y compararlas con imágenes posteriores en las que al restarlas podemos obtener resaltar el fondo o nuevos objetos/personas que aparezcan.
- Histograma: Es una función que nos dice la frecuencia de un color en una imagen. Consiste en mostrar el número de veces que un valor aparece en una imagen. En las imágenes blanco y negro, dibuja una gráfica que muestra si una imagen es muy oscura o clara o si presenta un buen balance de estas. El histograma en 0 me dice cuántas veces aparece el número 0 en esa imagen.

Para transformar una imagen con un histograma con muchos valores cercanos a 0 es aplicar una imagen logarítmica, para hacerla más clara (El 0 es negro, el 255 blanco). Y para un histograma con muchos valores altos, deberíamos de aplicar una función exponencial.

La equalización de un histograma consiste en balancear el histograma para obtener un buen balance de distribución de blancos y negros, para ello usamos el histograma acumulativo.

El histograma acumulativo consiste en ir sumando para cada posición, el número de veces que ha ocurrido ese valor más los valores anteriores en el histograma.

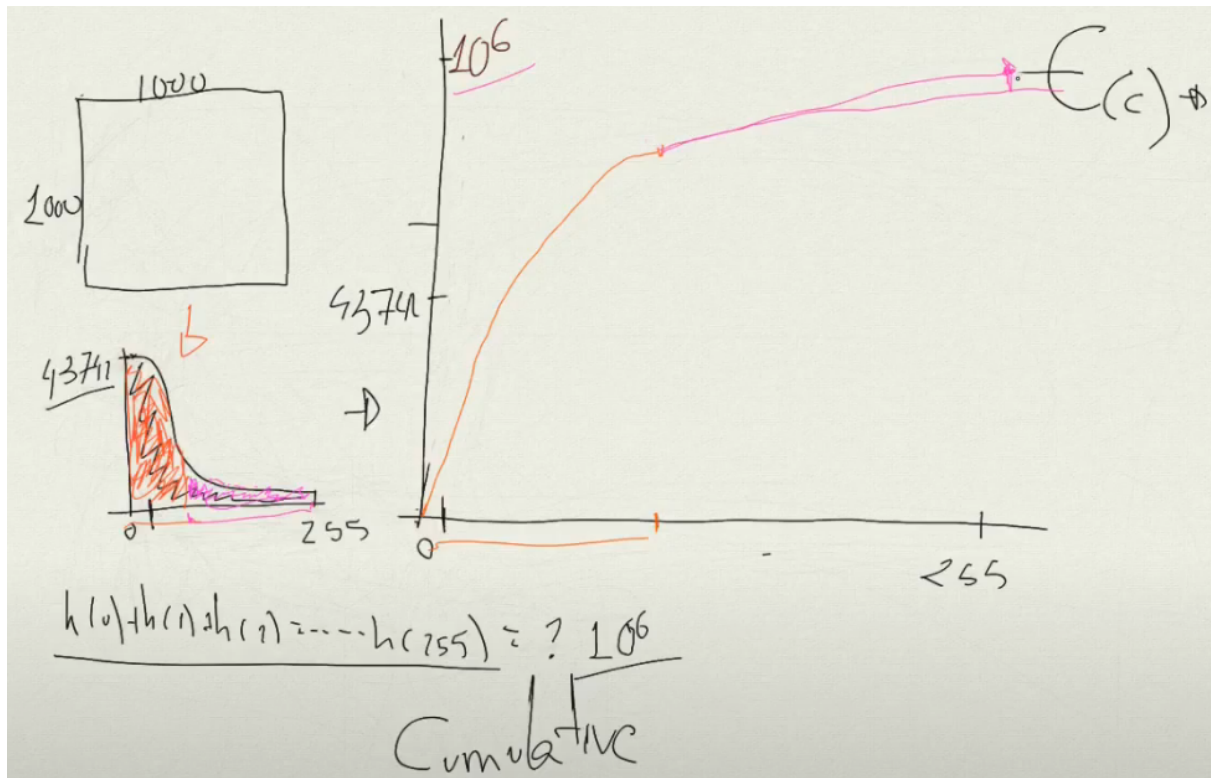
$$\text{Accu}(0) = h(0)$$

$$\text{Accu}(1) = h(1) + h(0)$$

$$\text{Accu}(2) = h(2) + h(1) + h(0)$$

$$\dots \text{Accu}(n) = h(n) + \dots + h(0)$$

De esta forma formamos una gráfica con un gran crecimiento en aquellas partes en las que el histograma tiene valores altos, teniendo otra parte donde el histograma acumulativo crece muy poco en sitios donde el histograma tiene valores bajos.



De esta forma obtenemos la función exponencial y logarítmica usando el histograma acumulativo.

Pero hay un problema, no podemos usar el histograma acumulativo directamente como función para ecualizar la imagen, primero necesitamos escalar el resultado a las métricas de nuestro programa, esto se consigue multiplicando por un factor.

Session 3

- RGB to Grescale conversion: Es otra pixel-wise operation que permite transformar de color a blanco y negro, hay varias formas.

Lightness: Es tomar el máximo y minimo y dividirlo entre 2

Average: Es sumar las tres componentes y dividirlos entre 3

Luminosity: Consiste en como vemos los colores los humanos, es aplicar:

$$0.21 \cdot R + 0.72 \cdot G + 0.07 \cdot B$$

- Hasta ahora hemos visto un conjunto de pixel-wise operations que se basaban en el propio valor del pixel, ahora veremos otro conjunto de operaciones donde el valor de un pixel depende de su valor y de los valores cercanos a él en la imagen.

- Esto se conoce como Neighbour-based processing

- Interpolación: Es una operación donde reescalar una imagen de una a mayor o menor tamaño. Sí queremos hacer una imagen más grande nos falta información, se puede intentar solucionar esto debemos de hacer interpolación, donde podemos aplicar algunos como:

- Nearest neighbour: Consiste en tomar píxeles que estén muy cerca.

- Bilinear interpolation: La idea es tomar valores de píxeles haciendo la media de estos pero en un solo sentido, por ejemplo los que estén en horizontal con el pixel en cuestión.

- Convolution: El concepto MÁS IMPORTANTE. Es la operación más importante en computer vision. Consiste en usar una ventana o kernel la cual nos indica que píxeles iremos tomando y a la vez usar esa ventana con determinados valores para ser usada como un filtro. La ventana se centra en un pixel y se usarán los píxeles que estén alrededor del central y que estén dentro del rango del kernel.

En sí, la convolución es aplicar esta multiplicación del kernel por los píxeles de alrededor del central. El kernel, por lo general, debe de ser impar.

- Blur: Aplicar un kernel lleno de 1 y dividir por el total del tamaño del kernel consiste en realizar la media de los píxeles de alrededor de uno central. Esto da lugar a una imagen más borrosa. Este es el box filter que se verá más adelante.

- Se puede ver la imagen como una señal bidimensional, donde se representa (en blanco y negro) brillo.
- High frequencies: Representan dónde están los pequeños detalles, como por ejemplos los bordes. Cuando nos referimos a las high frequencies, nos referimos a los cambios de la imagen. Son sensibles al ruido.
- Low frequencies: Representa información global, sin mucho cambio. Son robustos al ruido.
- Low pass filter: Elimina las altas frecuencias de una imagen, por tanto elimina el ruido.
- High pass filter: Resalta las altas frecuencias mediante la eliminación de las frecuencias bajas.

Principales low pass filters

- Box filter: Es el filtro con 1 en el kernel dividido por la suma total de los unos. NO es solamente hacer la media. Permite eliminar altas frecuencias, ya que es un low pass filter.
- Binomial filter: Es donde le damos diferentes valores al pixel. El efecto de esto es darle más peso a algunos píxeles.

La idea de dividir, es obtener al final unos valores que están escalados y que así no sea necesario normalizar.

$$\frac{1}{9} \times$$

1	1	1
1	1	1
1	1	1

Box Filter

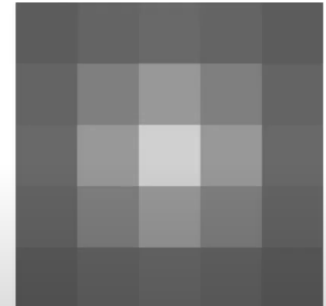
$$\frac{1}{16} \times$$

1	2	1
2	4	2
1	2	1

Binomial Filter

- Gaussian filter: Se basa en la función gaussiana. La idea es obtener un filtro que siga esta distribución a la hora de dar los pesos, se busca que esos pesos sigan esta distribución. Este filtro también corta las altas frecuencias, pero gracias a los parámetros sigma y tamaño del filtro, podemos jugar con cómo las recorta.

```
0.00291502 0.0130642 0.0215393 0.0130642 0.00291502  
0.0130642 0.0585498 0.0965324 0.0585498 0.0130642  
0.0215393 0.0965324 0.159155 0.0965324 0.0215393  
0.0130642 0.0585498 0.0965324 0.0585498 0.0130642  
0.00291502 0.0130642 0.0215393 0.0130642 0.00291502
```



Session 4

- Ahora jugamos con la idea de derivada, en la cual en la gráfica de una función, tras aplicar las derivadas, podemos obtener los picos donde hay grandes cambios.
- Podemos ver las imágenes como que en cada fila y columnas son funciones, por lo que nuestra imagen está formada por varias funciones en todas las direcciones.
- Para un píxel, tendremos la derivada en el eje X y la derivada en el eje Y.
- Según la forma gráfica explicada de calcular la derivada, podemos tomar los valores próximos a un pixel para obtener su valor derivado.
- Por lo tanto, la derivada de: $f'(x,y)$:
 - Derivada de X: $I(x+1,y) - I(x-1,y)$
 - Derivada de Y: $I(x,y+1) - I(x,y-1)$
- Esto es tomar el píxel los píxeles próximos a él.

- Para poder calcular la derivada de una imagen podemos usar kernels. El kernel esta formado por:

0 0 0	0 -1 0
-1 0 1	0 0 0
0 0 0	0 1 0

- De esta forma podemos calcular la derivada en X y en Y respectivamente con los filtros anteriores.
- La magnitud es un valor que indica cuánto de grande es la derivada tras aplicar un pixel, esto se calcula haciendo la raíz cuadrada de la suma al cuadrado de cada derivada (el cálculo de una distancia).

$$\text{Magnitud} = \text{Sqrt}(dx^2 + dy^2)$$

- NO se puede calcular la derivada con un único filtro.
- También podemos usar otro filtro para calcular la derivada PERO que tome en cuenta los pixeles vecinos para ser más robusto al ruido, el kernel sería:

-1 0 1	-1 -1 -1
-1 0 1	0 0 0
-1 0 1	1 1 1

- Luego existe el SOBEL filter, el cual permite dar más fuerza al pixel central, el cual es el kernel estándar para calcular la derivada de una imagen.

-1 0 1	-1 -2 -1
-2 0 2	0 0 0
-1 0 1	1 2 1

- Tras calcular la magnitud con este filtro, esa magnitud es la que se establece en la imagen.

- El resultado es 0 en aquellas zonas donde no hay gran cambio y destaca las zonas que sí hay, como por ejemplo los bordes o zonas donde hay un gran cambio, este filtro nos indica dónde están las altas frecuencias.

Session 5

- Laplacian filter: Es el filtro de segunda derivada, es la derivada de la derivada. Es otro tipo de high pass filter. Hay que transformar sus valores.

- High boost filter: Es otro high pass filter, donde resaltamos las altas frecuencias de una imagen, resaltamos los bordes, los hacemos más afilados. La idea consiste en que una imagen está compuesta por sus altas y bajas frecuencias, por tanto, podemos obtener la imagen correspondiente a las altas frecuencias, después de esto, podemos aplicar un kernel a esta imagen de altas frecuencias y tras esto sumarlo a la imagen original.

Original image = High frequency image + Low frequency image

Highboost image = Original image + kernel * high frequency image

- Donde el kernel es:

-c -c -c

-c 8c+1 -c

-c -c -c

- Hasta ahora hemos visto los filtros lineales usando un kernel.

- Ahora veremos filtros NO lineales. Donde tendremos una “ventana” similar al kernel, pero que NO usaremos para modificar valores, sino que solo la usaremos para tomar valores concretos cercanos a un pixel. Con estos píxeles, realizaremos una serie de operaciones, las cuales serán las que veremos.

- Median filter: Es un filtro que elimina el ruido del estilo “salt and pepper”, también llamado ruido gaussiano. Se puede usar uno de los filtros anteriores, no obtenemos resultados muy buenos, pero con en median filter sí tendremos buenos resultados.

- Este filtro consiste en tomar los valores alrededor del píxel central (según la ventana) y ordenarlos y tras esto tomar el píxel central de los valores ordenados y este valor será el que usaremos para ese pixel.

Session 6

- Bilateral filter: Permite suavizar, eliminar el ruido y permiten mantener los bordes, mantiene altas frecuencias.
- La idea consiste en usar una ventana para tomar los valores alrededor de un píxel y el resultado de ese píxel central será la suma de unas multiplicaciones, las cuales, multiplican unos pesos por las intensidades de los píxeles sobre los que está la ventana. El peso 1 significa que NO modificas ese valor. OJO, el kernel NO tiene pesos, estamos en nonlinear filters. Los pesos van del $[0, 1]$. Un pixel cercano a 0 indica que importa menos.
- Estos pesos se definen en base a la distancia del pixel con el centro y tambien la diferencia de color entre ese pixel y el central, osea, depende de la diferencia de valor y de distancia con respecto a un pixel y el central. Esto permite dar importancia a pixeles lejanos y que se parezcan al central, pero no darle importancia a píxeles que están lejos y que también no son similares al central.
- La distancia se puede ver como una mitad de la función gaussiana, que disminuye conforme nos alejamos del pixel central
- Similar para la función de valor de intensidad.
- En resumen, en áreas donde los valores son similares, es como aplicar un box o gaussian filter. En regiones donde hay bordes, zonas donde son diferentes, el filtro no tomará esas partes.
- Morphological filter/operations: Es un filtro que permite eliminar el ruido, simplificar las formas de los objetos, limpiar el resultado de una imagen threshold, etc. Tenemos dilatación y erosión.
 - Dilation: Permite aumentar de tamaño los objetos, da mayor valor a los vecinos.
 - Erosion: Permite reducir, hacer más fino, los objetos, de menor valor a los vecinos.
- Estas operaciones consisten en tomar el valor máximo o mínimo de la región que forma nuestra ventana, y aplicarlo como valor para el píxel central.
- Opening: Erosion + Dilation. Esto permite dejar la imagen tal cual estaba pero eliminando ruido.
- Closing: Dilation + Erosion. Permite lo mismo pero al revés.
- Estos filtros se pueden usar también para fotos a color.

Block 2: 3D Image

Camera Model

- Buscaremos extraer información en 3D con nuestra cámara. Buscamos saber cuan lejos están los objetos con respecto a nuestra cámara.
- Buscaremos calcular la ecuación que nos permite proyectar puntos 3D en los píxeles de nuestra imagen. Esta ecuación está formada en base a unos parámetros.
- El fotón que llega a nuestra imagen pasa siempre por el hueco de nuestro pin-hole camera, esa distancia es llamada distancia focal. Las imágenes capturadas están volteadas, por lo que usamos un sensor alternativo que está **por delante del hueco (centro \leftrightarrow focus)**, con distancia focal también, este sensor lo obtenemos volteando de arriba a abajo y volteando horizontalmente. Esto es solo para explicar que realmente capturamos las imágenes al revés.
- Sí proyectamos un punto $X Y Z$ en nuestro sensor con coordenadas de ese punto $x y$, podemos saber la ecuación que guarda ese punto en nuestro plano. Para ello miramos desde un lado, observando solo el eje Y y el eje Z . De esta forma observamos dos triángulos los cuales son proporcionales. Los triángulos se forman uniendo los puntos con el centro C . Con esta proporción de triángulos, podemos despejar la coordenada ' y ' de nuestro punto. Para la coordenada ' x ' es equivalente. Por tanto y resumiendo:
 - Sí sabemos un punto $P(X,Y,Z)$ y la distancia focal f , podemos obtener las coordenadas de la proyección del punto sobre nuestro plano.

- Hasta ahora hemos considerado que el eje de referencia está en el centro de la imagen, pero ahora tomaremos la esquina superior izquierda como eje, para ello, a la 'x' e 'y' debemos de sumarle un valor (**Optical center**) para lograrlo.

$$x = \frac{f_x X}{Z} + c_x; y = \frac{f_y Y}{Z} + c_y$$

- Podemos representar estas ecuaciones de forma matricial.
 - Para ello debemos de entender la idea de **coordenadas homogéneas**.
 - Esto es, multiplicar cada coordenada por un valor, indicando el resultado como un vector con el número por el que multiplicamos al final del mismo. Estas coordenadas nos ayudarán a obtener la proyección del punto usando operaciones con matrices.
- $(x,y) \rightarrow (wx, wy, w)$
- **Camera matrix**, matriz que nos permite obtener el punto en nuestro plano multiplicando esta matriz por el punto en 3D. Esta incluye la distancia focal y los centros ópticos.

$$q = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Camera Matrix

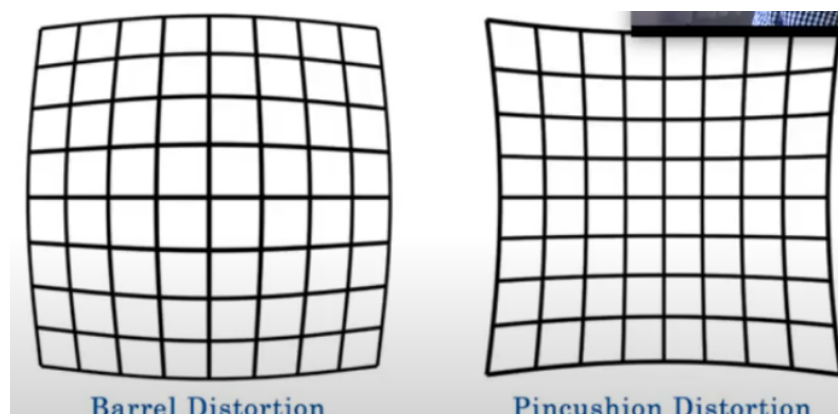
- Al realizar esta multiplicación (M·Q) obtendremos las **coordenadas homogéneas** de la proyección del píxel. Para obtener el punto con coordenadas normales, debemos dividir el resultado entre la 'w'.

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} p_x X + 0 Y + c_x Z \\ 0 + p_y Y + c_y Z \\ 0 + 0 + Z \end{bmatrix} \rightarrow \begin{bmatrix} p_x X + c_x Z \\ p_y Y + c_y Z \\ Z \end{bmatrix}$$

$$\frac{p_x X + c_x Z}{Z} \quad \frac{p_y Y + c_y Z}{Z}$$

Distortion Model

- Lo explicado anteriormente es en el caso ideal del pin-hole camera, en la vida real, las cámaras utilizan lentes y muchas de ellas presentan problemas, por lo que se producen distorsiones en las imágenes.
- Las lentes nos permiten aumentar o reducir el **campo de visión (field of view)**, en el caso del pin-hole camera, necesitamos hacer una caja más o menos grande.
- Este modelo anterior, **no** es válido para el caso de las cámaras reales.
- En nuestro modelo real, al proyectar un punto 3D, este pasa por la lente, por lo que sufre una pequeña distorsión en su proyección.



- Pese a estos problemas, aún podemos seguir encontrando las ecuaciones para obtener nuestro punto, este es nuestro principal objetivo desde el principio.

- El punto proyectado por la distorsión es llamado **punto real**, mientras que el punto que buscamos obtener es llamado **punto ideal**. Hay una función que transforma el punto ideal en un punto real.

- Ahora buscamos obtener esta distorsión que sufre el punto, viene dada matemáticamente por la “suma” de ambas distorsiones.

- Hay dos tipos de distorsiones que veremos:

· Radial distortion: Dobla las líneas rectas y es producida por la mala alineación entre el sensor y la lente (dado que la lente es “redonda”).

Deberían de estar así para que fuera perfecto: | | pero muchos de ellos no están así de alineados.

Donde x' e y' es el punto **real** (con la distorsión), x e y son el punto **ideal**, r es la distancia al **centro óptico** y k son constantes que definen una “serie” de Taylor. Esto significa, que a más términos, más preciso soy. Cuanto **más me alejo** del centro, **más distorsión** se sufre.

$$\begin{aligned}x' &= x(1 + k_1 r^2 + k_2 r^4) \\y' &= y(1 + k_1 r^2 + k_2 r^4)\end{aligned}$$

where (x', y') is the distorted point and $r = \sqrt{(x - c_x)^2 + (y - c_y)^2}$

· Tangential distortion: Sigue afectada por el radio, a más me alejo, más se afecta. Es producido por un problema de fabrica. Donde las p indican la inclinación del sensor. No se entra en mucho detalle.

- Si ahora mezclamos ambas distorsiones, obtenemos el **Brown's Model**. No hace falta memorizarlo.

$$x' = x + (x - x_c)(k_1 r^2 + k_2 r^4) + (p_1(r^2 + 2(x - x_c)^2) + 2p_2(x - x_c)(y - y_c))$$

$$y' = y + (y - y_c)(k_1 r^2 + k_2 r^4) + (p_2(r^2 + 2(y - y_c)^2) + 2p_1(x - x_c)(y - y_c))$$

- Si sabemos estas ecuaciones, podemos corregir el desplazamiento que sufre el punto. Es buscar la operación inversa a la de la ecuación (Despejar?).
- Pero ojo, también hay que aplicar las ecuaciones anteriores, por lo que tendremos una cadena de operaciones que hay que hacer.
- Si tenemos los parámetros de la Camera Matrix ($f_x \dots$) y tenemos los parámetros de la distorsión ($k_1, k_2, p_1, p_2 \dots$) podemos obtener el punto real (con la distorsión).
- **Todos** estos parámetros se conocen como **parámetros intrínsecos**. Y son producidos por cómo ha sido creada la cámara.
- Los **parámetros extrínsecos** se producen cuando mueves la cámara de posición.

Transformaciones 3D

- Hasta ahora, la cámara y el punto se encontraban en el mismo sistema de referencia. Ahora entendemos que nuestro sistema de referencia **no** está centrado en la cámara. Ahora la cámara puede moverse alrededor del sistema de referencia.
- Tendremos un punto que sabemos dónde está respecto al sistema de referencia **global**, pero **no** sobre los "ejes" de nuestra cámara. El problema es ver dónde se encontraría este punto en nuestra cámara.
- Para ello primero tenemos que saber **dónde** se encuentra nuestra cámara **respecto al** sistema de referencia global. Necesito saber la **transformación** que he hecho del sistema de referencia hasta la posición donde se encuentra ahora la cámara.

- Ahora veremos transformaciones 3D usando matrices. Suponemos que $w=1$, por lo que un punto se representara como $(X, Y, Z, 1)$

- Hay 3 transformaciones, traslación, rotación y escala, pero escala no la veremos.

- Traslación es mover en el eje de referencia. Esto es mover los puntos en el eje de referencia.

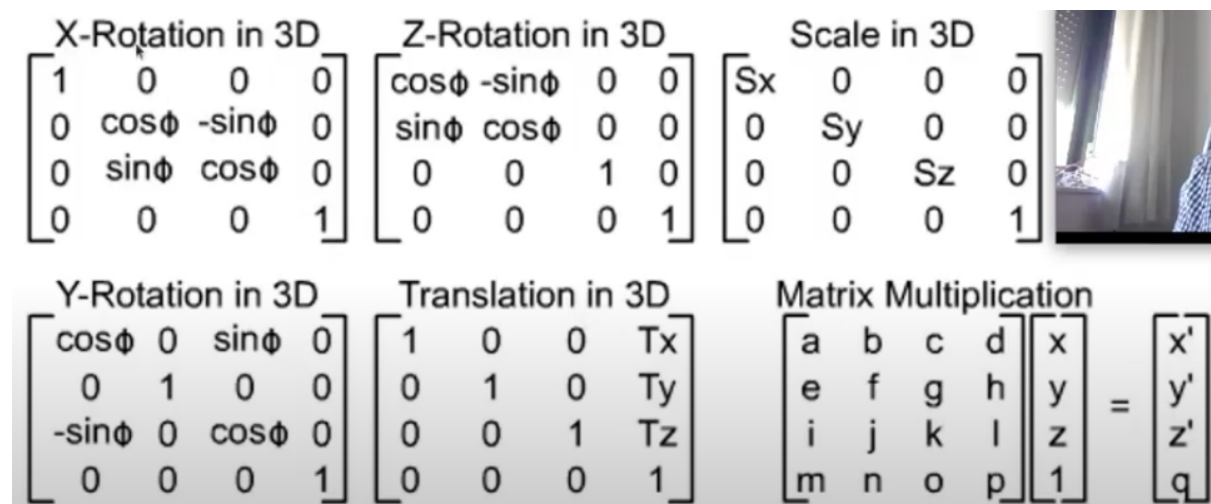
- Rotación es rotar sobre los ejes.

- La traslación es **añadir** un valor al punto para poder moverlo.

- La rotación es multiplicar por $\cos(\alpha)$.

- Lo **importante** es saber que estas operaciones se pueden lograr a través de una única matriz de 4×4 , que dependiendo de lo que pongamos en sus elementos, obtendremos una operación u otra.

- $R_x(30)$ significa rotar en el eje X 30 grados.



The image displays several 3D transformation matrices and a small photograph of a person's legs. The matrices are arranged in two rows:

- X-Rotation in 3D:**

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Z-Rotation in 3D:**

$$\begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Scale in 3D:**

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Y-Rotation in 3D:**

$$\begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Translation in 3D:**

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Matrix Multiplication:**

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ q \end{bmatrix}$$

A small photograph of a person's legs is visible on the right side of the image.

- Lo bueno de esto es que podemos concatenar operaciones y que en temas de computación resulten eficientes. Las multiplicaciones se hacen de derecha a izquierda.

- Podemos multiplicar todas las operaciones y obtener **una única** matriz la cual es la que multiplicamos por nuestros puntos y de esta manera obtener la transformación deseada para todos los puntos que queramos de forma más eficiente.

- De esta manera, podemos obtener una matriz la cual nos indica la transformación que sufre nuestro eje de la cámara respecto al eje global. Si aplicamos esta transformación sobre el punto 3D obtendremos el punto respecto a nuestro eje, pero aun no sabemos el punto en nuestro sensor, por lo que debemos de aplicar las multiplicaciones que hemos visto en las partes anteriores del tema, aplicando tras ello la distorsión.

- Estos son los parámetros, los parámetros de la izquierda son los intrínsecos y los de la derecha los extrínsecos.

$$\begin{array}{c} x' \\ y' \end{array} = d \left(\begin{bmatrix} x \\ y \\ w \end{bmatrix} \right) = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{ext} \\ Y_{ext} \\ Z_{ext} \\ 1 \end{bmatrix}$$

$(K_1, K_2, P_1, P_2, K_3) \{f_x, f_y, c_x, c_y\}$
 $\{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, t_x, t_y, t_z\}$

Camera Calibration

- Nota: Patron \Leftrightarrow patrón de ajedrez, chess pattern
- El problema ahora es saber los parámetros de nuestra cámara. Los parámetros externos cambian constantemente, pero los parámetros intrínsecos son siempre constantes, por lo que podemos calcularlos. La obtención de los parámetros intrínsecos se llama **calibración de la cámara**.
- Para ello, tomamos video/fotos de un patrón de tamaño que sabemos desde diferentes ángulos, podemos calcular los parámetros.
- Asumamos que en el interior del patrón es donde se encuentra nuestro sistema de referencia. Mientras movemos la cámara, el eje se mantiene igual. Si sabemos el tamaño del patrón, podemos estimar el punto en 3D, osea, los parámetros extrínsecos (*creo*).
- En cada imagen, los extrínsecos cambian, pero los intrínsecos siempre se mantienen constantes.

- En la calibración, establecemos una relación entre cada punto 3D con respecto a ese punto pero en nuestro patrón.
 - La **calibración** recibe como entrada los puntos 3D y la asociación de esos puntos respecto a nuestro ejes del patrón y genera como salida los parámetros intrínsecos de la cámara y los extrínsecos para cada imagen.
 - Aun así, si pregunta para que es la calibración, debemos de decir que es para obtener los parámetros **intrínsecos**.
-
- La **realidad aumentada** consiste en encontrar la proyección de los puntos (vértices) de los objetos que queremos generar con respecto a nuestra cámara y tras ello unir puntos. El problema viene con los extrínsecos, que cambian constantemente, para ello usamos el patrón de calibración anterior y resolviendo el problema PnP podemos obtener los parámetros extrínsecos para poder proyectar nuestro cubo, básicamente, nos da los parámetros extrínsecos para obtener la posición de nuestra cámara respecto al patrón.