Augmented reality: Lesson 4 Omnidirectional cameras and unwarping algorithm

Francesco Di Giacomo

Augmented reality course
Francesco Di Giacomo
giacf@hr.nl

- camera with 360˚field of view.
- the surrounding space is projected on a mirror.
- a conventional camera looks at the mirror.
- the conventional camera can see the whole mirror, thus it can see all around it.

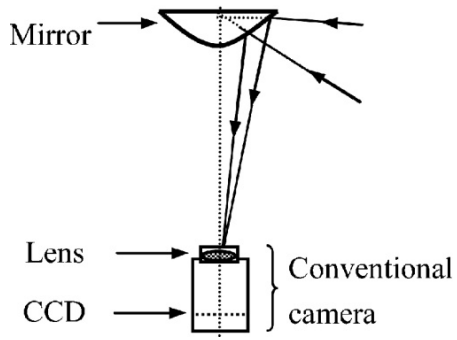


Figure: Omnidirectional camera schema

Omnidirectional cameras
Advantages

- No limited field of view.
- No camera rotation needed to control the whole area.

Figure: Omnidirectional view

- The surroundings are projected onto a circle.
- Humans have troubles visualizing objects in this representation.
- Computer vision algorithms usually works with perspective view (like the human eye can see).
- Conversion requires some additional computation that affects performance.

Figure: Converted perspective view

Omnidirectional cameras
Applications

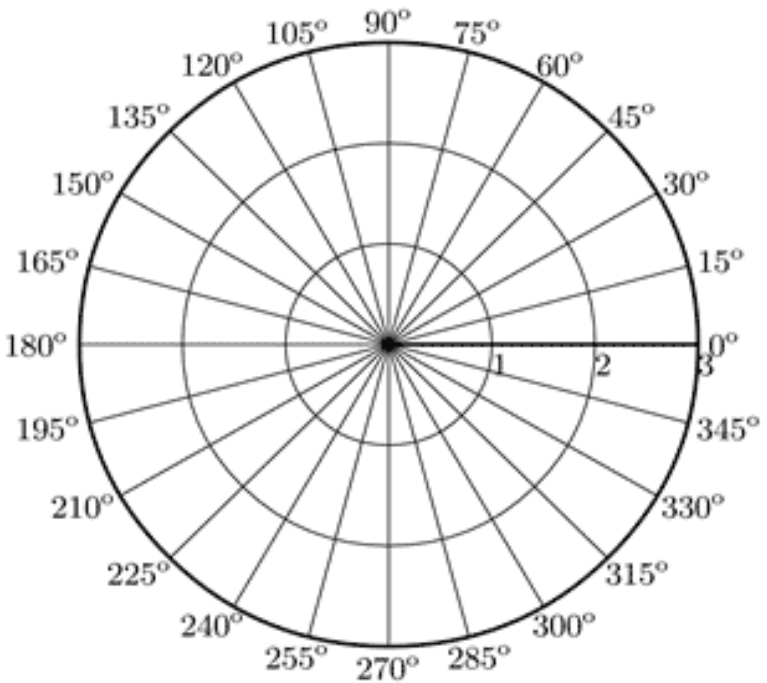
- Robotic.
- Surveillance.
- Panoramic art.

Unwarping algorithm

- Convert the omnidirectional view into a perspective view.
- Eliminate the disadvantages of omnidirectional cameras.
- Requirement: conversion between two different coordinate systems (polar to scalar).

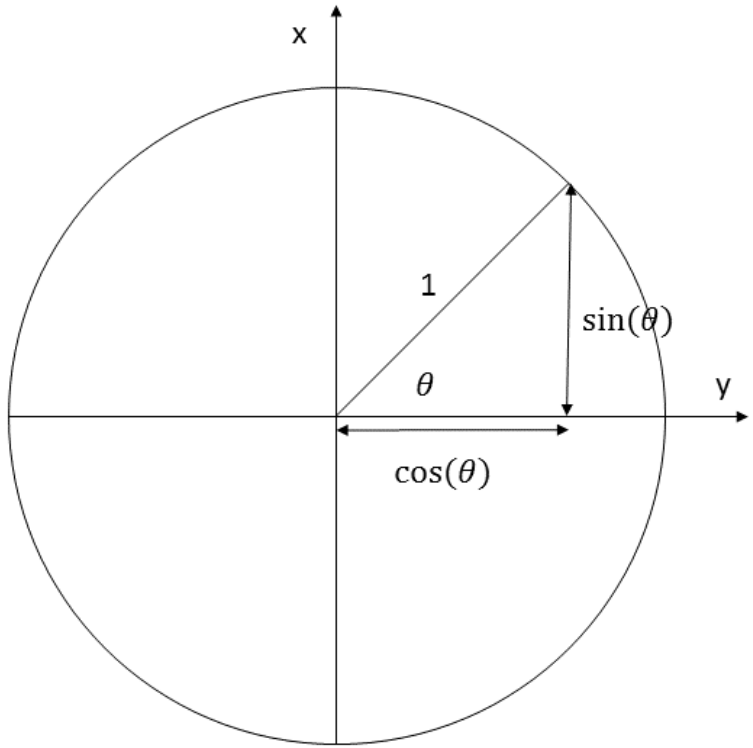
Polar coordinates
Definition

- A point in a cartesian system is defined by 2 or more components in space.
- The components are its position on 2 or more reference axes (their intersection is the origin).
- In polar coordinates a point is defined by an angle and the distance from the origin.
- They are useful to find the position of a point in a circular area.



Polar coordinates
Cosine and sine

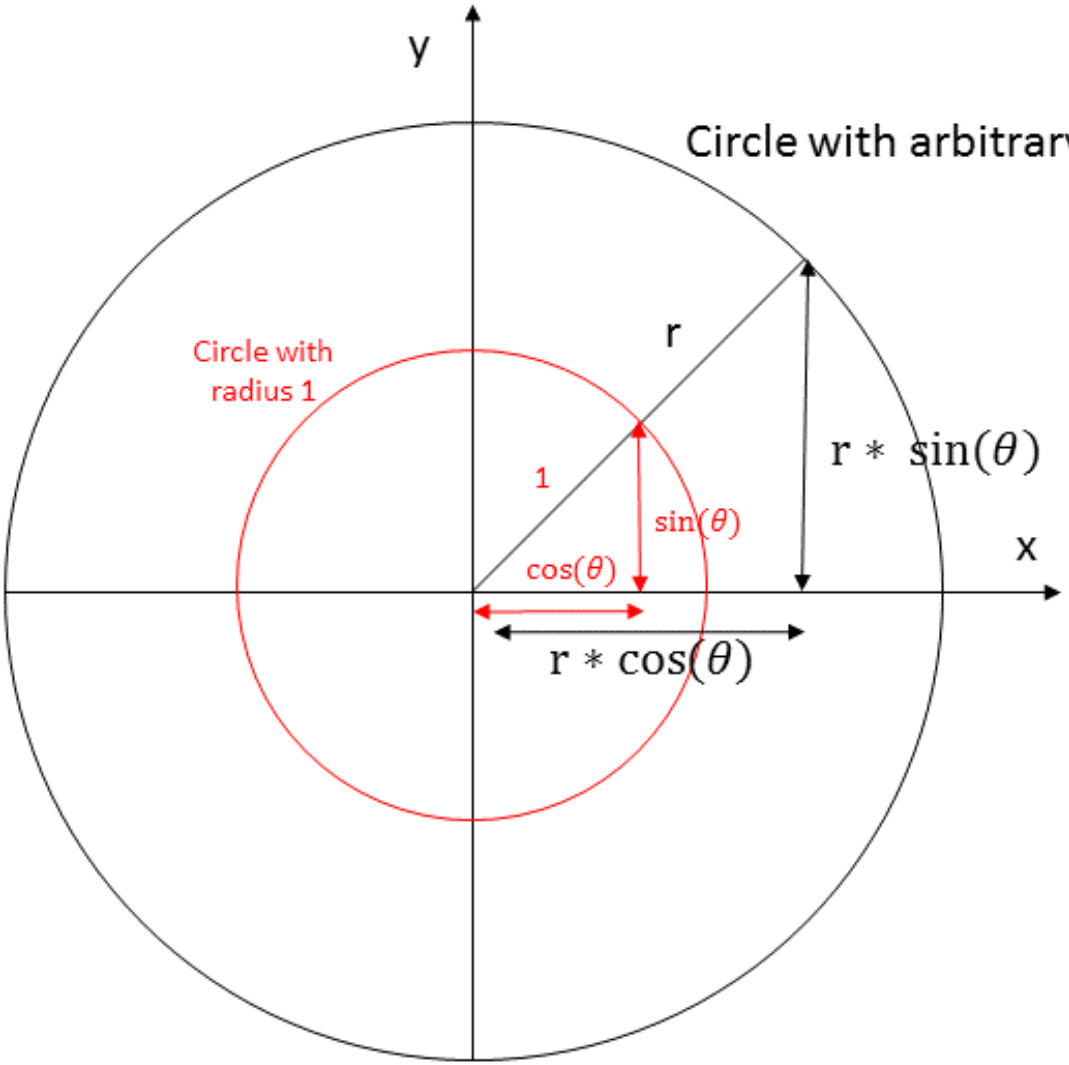
- Functions used to find the length of sides in a right triangle knowing an angle.
- Necessary to convert a point from polar to cartesian coordinates.
- Consider a circle with radius 1 and an angle θ. Then sine and cosine are defined as in the picture below (denoted as $\sin(\theta)$ and $\cos(\theta)$).
- $\cos(\theta)$ is the length of the base of the right triangle.
- $\sin(\theta)$ is the length of the height of the right triangle.



What happens if the radius is different from 1?

What happens if the radius is different from 1?

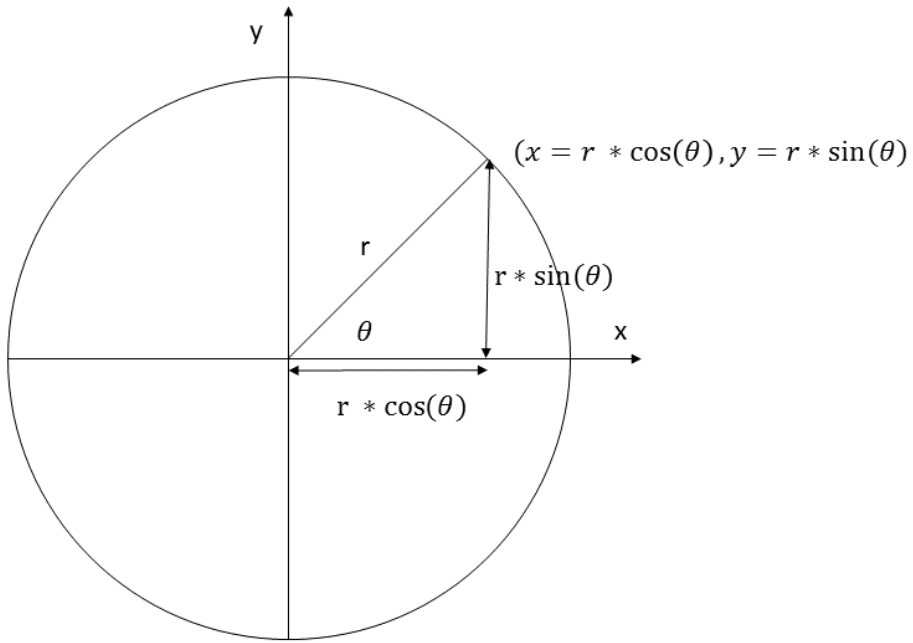
The length of the base and height is re-scaled according to the radius.



Polar coordinates
Conversion from polar to cartesian coordinates

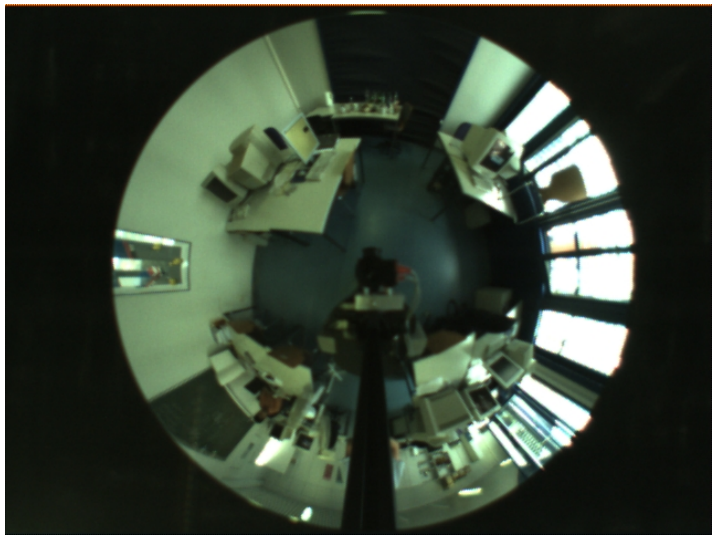
- Apply the definition of sine and cosine.
- We know the distance from the centre (radius) and the angle θ for the point from the polar coordinates.
- Then the cartesian coordinates are just the length of the base and height of the right triangle (see figure below).

$$\begin{cases} x = r * \cos(\theta) \\ y = r * \sin(\theta) \end{cases}$$



Unwarping algorithm
General idea

Change this (omnidirectional view):



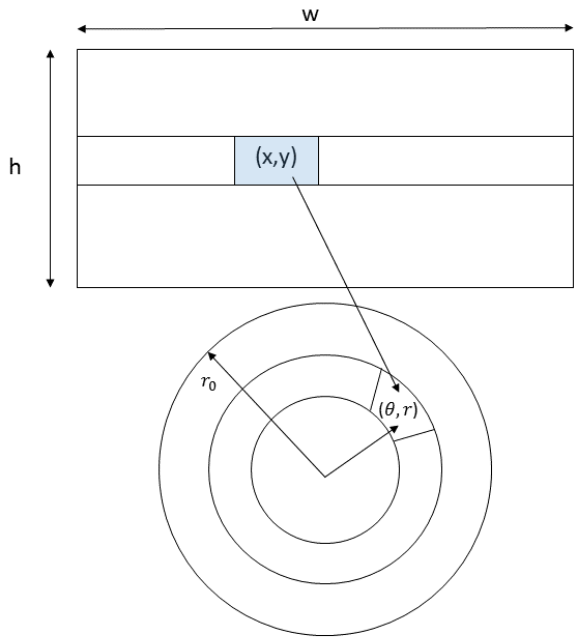
into this (perspective view):

Unwarping algorithm
General idea

- We create the perspective image (rectangular shape).
- We find where each pixel in the perspective image is positioned in the omnidirectional image, i.e. we find its polar coordinates.
- Given the polar coordinates of each point we convert it back into cartesian coordinates (because the pixels of an image are in a cartesian space).
- We apply bilinear interpolation to the fractional pixel coordinates given by the conversion.
- We insert the result of the interpolation in the current pixel of the perspective image.

Unwarping algorithm
Perspective pixel into omnidirectional pixel

- The omnidirectional image is a warped (curved) version of the perspective image.
- Note that a line in the perspective image becomes a circle in the omnidirectional image.
- We have to find the parameters θ and r in polar coordinates for a generic pixel at coordinates (x, y) in the image.



Unwarping algorithm
Step 1: Perspective image size

- Now we have to find the size of the perspective image given the size of the omnidirectional image.
- Note that the omnidirectional camera has a small blind spot, which we do not want in the final image, of a certain radius r_{in}.
- The width of the perspective image is the length of the circle in the omnidirectional image (rounded).
- The height of the perspective image is the radius (called r_{out}) of the circle in the omnidirectional image (rounded) without the radius of the blind spot.
- r_{out} and r_{in} are parameters of a specific omnidirectional camera. You have to measure them in the image with a image editor software (i.e. paint.net, which is free).

$$\begin{cases} width = round(2 \cdot \pi \cdot r_{out}) \\ height = round(r_{out} - r_{in}) \end{cases}$$

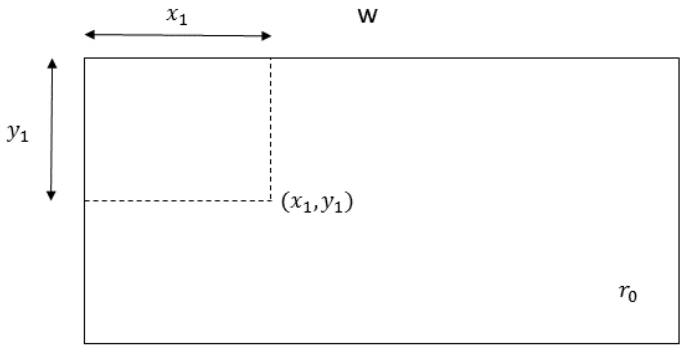
Assume we have the perspective image, where w and h are respectively the width and height of the image. Divide the coordinates of the pixel by w and h
to obtain values independent of the image size between 0 and 1.

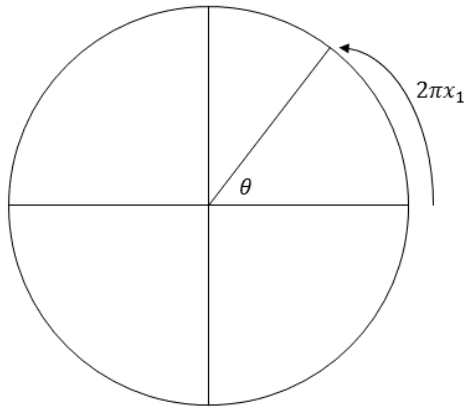
$$x_1 = \frac{x}{w}$$
$$y_1 = \frac{y}{h}$$

Unwarping algorithm
Step 2 : Pixel polar coordinates

- The x coordinate of a cartesian point is the distance from the origin of the cartesian axes.
- The cartesian axis x is converted into a circle in polar coordinates.
- The coordinate x_1 of the point is converted into a point on this circle.
- An angle, by definition, is the length of a portion of the circle with radius 1 (like ours: remember we normalized the width and the height of the pixel to be in between 0 and 1).
- The angle of the point is given by $\theta = -2\pi x_1$, which is the length of the portion of the circle until x_1.
- The negative sign is because the image y axis points downward, while in a cartesian system by convention it points upward. In this way the angles grow clockwise, as in the polar system. If you remove the sign you will get the symmetric image.

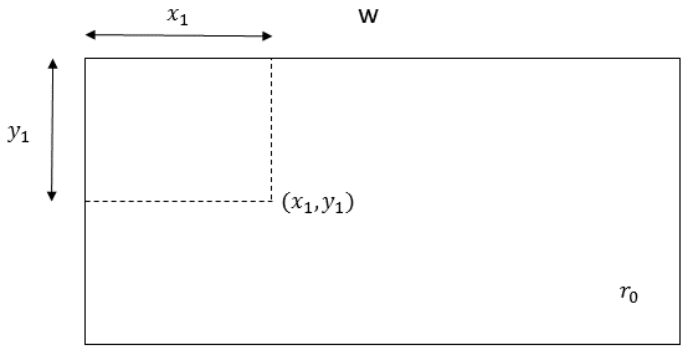
Unwarping algorithm
Step 2 : Pixel polar coordinates

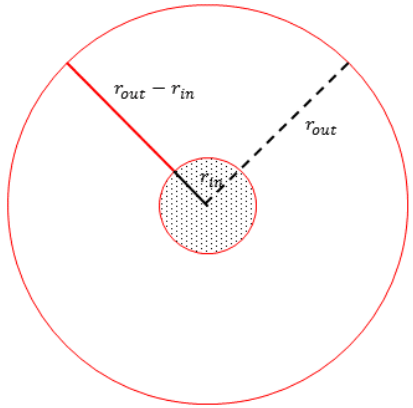




- The y component of the cartesian pixel is, in polar coordinates, the distance from the centre of the omnidirectional image.
- The radius component of the pixel in polar coordinates is given by $r = (1 - y_1)r_{out}$.
- **PROBLEM:** we are including the blind spot.
- We have to exclude the blind spot. The points in the perspective image will be those going from the edge of the blind spot to the outer circle.
- This is given by $r = r_{in} + (1 - y_1)(r_{out} - r_{in})$.

Unwarping algorithm
Step 2: Pixel polar coordinates





The formulas to get the polar coordinates of a pixel in the omnidirectional image (with normalized coordinates (x_1, y_1)) are:

$$\begin{cases} \theta = -2\pi x_1 \\ r = r_{in} + (1 - y_1)(r_{out} - r_{in}) \end{cases}$$

- Now we have the coordinates of the point in the omnidirectional image in polar coordinates.
- We need to convert it into cartesian coordinates because pixels are in cartesian coordinates into the image (row index, column index).
- The centre of the polar coordinates are the coordinates of the centre of the camera in the omnidirectional image. These are given cameras parameters as well, called c_x and c_y.

$$\begin{cases} x_p = r \cdot \cos(\theta) + c_x \\ y_p = r \cdot \sin(\theta) + c_y \end{cases}$$

Unwarping algorithm
Step 4: Bilinear interpolation

- Take the coordinates of the pixel in the omnidirectional image (x_p, y_p).
- Apply bilinear interpolation with x_p and y_p as input.
- Copy the interpolated colour in the perspective image.

Unwarping algorithm
Summary

1. Starting from a pixel in the perspective image, find the corresponding polar coordinates in the omnidirectional image.
2. Convert the polar coordinates into cartesian coordinates to find the row and column index to use in the omnidirectional image.
3. Interpolate the fractional coordinates obtained at the previous step to get the colour to put into the pixel in the perspective image.
4. Copy the interpolated colour in the perspective image.

Assignment 4

Implement the unwarping algorithm for an omnidirectional camera. Your function will take an omnidirectional image as input, and output the equivalent perspective image. Assume that you know the radius of the blind spot r_{in}, the outer radius r_{out} of the omnidirectional camera, and the coordinates of the centre of the camera (c_x, c_y) in the omnidirectional image. You can find these values for the standard images you should use in N@tschool in the `parameters.txt` file. If you want to add an additional image then you have to measure these values manually with an image editing software.