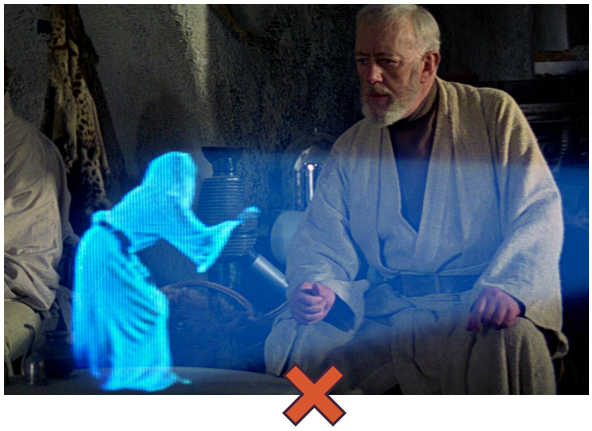
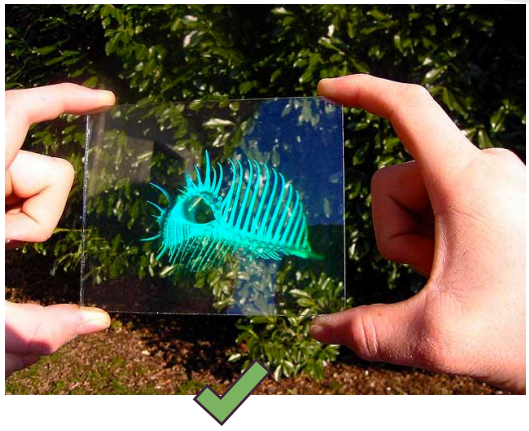
Computer Generated Holography

Luís Paulo Santos, Waldir Pimenta

Universidade do Minho · Nov 2014

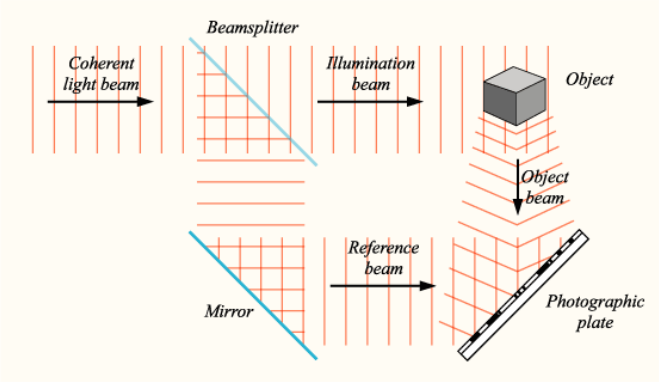
What is holography?



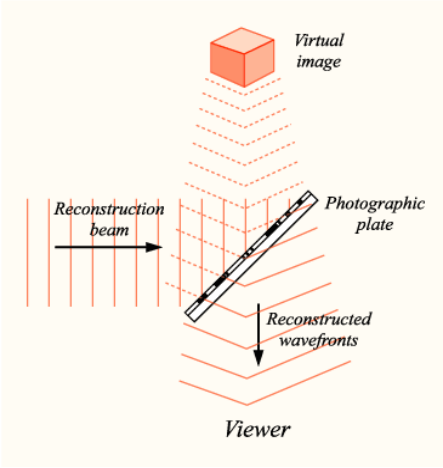


Classic holographic setup

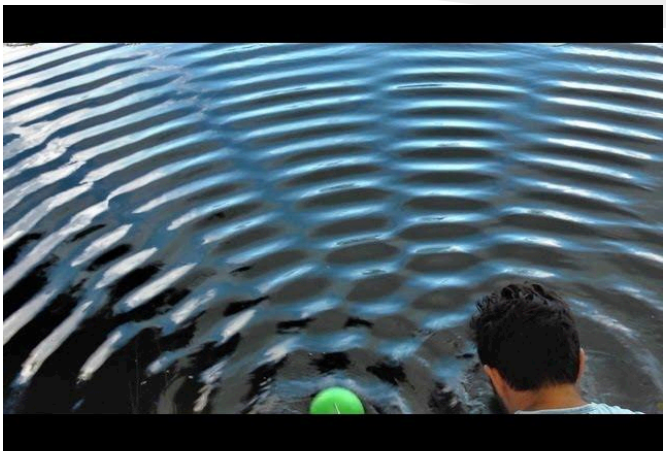
Recording:



Reconstruction:



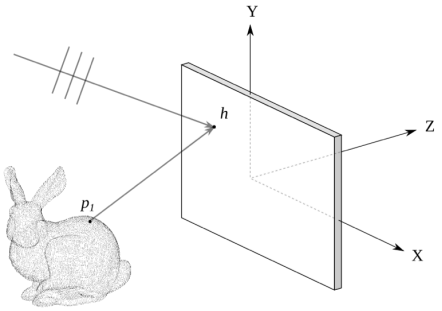
Interference patterns



Demo:
http://waldyrious.github.io/learning-holography/2d-hologram-viz.xhtml

Computer-Generated Holography

- Physically-based CGH:
  Simulate wave propagation and interference



- Combinatorially large:
  number of object points × number of hologram pixels

  → huge data and processing requirements

Computer-Generated Holography

Back-of-the-envelope calculations:

- 10×10 cm hologram @ 10μm (2 540 dpi)
  max. diffraction angle: only 2°!
- object point cloud: 1M vertices
- 10^{10} hologram pixels × 10^{6} object points
  = **10^{16} iterations**

Equation

amplitude of a wave

intensity

$$I(h) = \int_0^T \left[\overbrace{A_1 \cos(\omega t - \varphi_1)} + A_2 \cos(\omega t - \varphi_2) \right]^2 dt$$

initial phase (0..2π) at measuring point

= distance × cycles per meter (spatial angular frequency, k)

= time elapsed × cycles per second (*temporal* angular frequency)

Equation

$$I(h) = \int_0^T \left[A_1 \cos(\omega t - \varphi_1) + A_2 \cos(\omega t - \varphi_2) \right]^2 dt$$

$$\lim_{T \to \infty} I(h)/T = \frac{A_1^2}{2} + A_1 A_2 \cos(\varphi_1 - \varphi_2) + \frac{A_2^2}{2}$$ (for 2 points)

$$\lim_{T \to \infty} I(h)/T = \sum_{n=1,\ m=1}^{N} A_n A_m \cos(\varphi_n - \varphi_m)/2$$ (for n points)

Bipolar intensity

$$\lim_{T\to\infty} I(h)/T = \sum_{n=1,\ m=1}^{N} A_n A_m \cos(\varphi_n - \varphi_m)/2$$

$A_R A_R \cos(\varphi_R - \varphi_R)/2$	+	$A_R A_1 \cos(\varphi_R - \varphi_1)/2$	+	$A_R A_2 \cos(\varphi_R - \varphi_2)/2$	+	...	+	$A_R A_N \cos(\varphi_R - \varphi_N)/2$
$A_1 A_R \cos(\varphi_1 - \varphi_R)/2$	+	$A_1 A_1 \cos(\varphi_1 - \varphi_1)/2$	+	$A_1 A_2 \cos(\varphi_1 - \varphi_2)/2$	+	...	+	$A_1 A_N \cos(\varphi_1 - \varphi_N)/2$
$A_2 A_R \cos(\varphi_2 - \varphi_R)/2$	+	$A_2 A_1 \cos(\varphi_2 - \varphi_1)/2$	+	$A_2 A_2 \cos(\varphi_2 - \varphi_2)/2$	+	...	+	$A_2 A_N \cos(\varphi_2 - \varphi_N)/2$
...	+	...	+	...	+	...	+	...
$A_N A_R \cos(\varphi_N - \varphi_R)/2$	+	$A_N A_1 \cos(\varphi_N - \varphi_1)/2$	+	$A_N A_2 \cos(\varphi_N - \varphi_2)/2$	+	...	+	$A_N A_N \cos(\varphi_N - \varphi_N)/2$

Bipolar intensity

$$\lim_{T\to\infty} I(h)/T = \sum_{n=1,\ m=1}^{N} A_n A_m \cos(\varphi_n - \varphi_m)/2$$

$A_R A_R \cos(\varphi_R - \varphi_R)/2$	+	$A_R A_1 \cos(\varphi_R - \varphi_1)/2$	+	$A_R A_2 \cos(\varphi_R - \varphi_2)/2$	+	...	+	$A_R A_N \cos(\varphi_R - \varphi_N)/2$
$A_1 A_R \cos(\varphi_1 - \varphi_R)/2$	+	$A_1 A_1 \cos(\varphi_1 - \varphi_1)/2$	+	$A_1 A_2 \cos(\varphi_1 - \varphi_2)/2$	+	...	+	$A_1 A_N \cos(\varphi_1 - \varphi_N)/2$
$A_2 A_R \cos(\varphi_2 - \varphi_R)/2$	+	$A_2 A_1 \cos(\varphi_2 - \varphi_1)/2$	+	$A_2 A_2 \cos(\varphi_2 - \varphi_2)/2$	+	...	+	$A_2 A_N \cos(\varphi_2 - \varphi_N)/2$
...	+	...	+	...	+	...	+	...
$A_N A_R \cos(\varphi_N - \varphi_R)/2$	+	$A_N A_1 \cos(\varphi_N - \varphi_1)/2$	+	$A_N A_2 \cos(\varphi_N - \varphi_2)/2$	+	...	+	$A_N A_N \cos(\varphi_N - \varphi_N)/2$

$$2\sum_{n=1}^{N} A_R A_n \cos(\varphi_R - \varphi_n)/2 = A_R \sum_{n=1}^{N} A_n \cos(\varphi_R - \varphi_n)$$

Basic algorithm

```
for ( p = 1 --> numPts ) {
        for ( h = 1 --> holoWidth * holoHeight ) {
                hologram[h] += cos ( k * dot(ref, h) - k * dist(p, h) );
        }
}
```

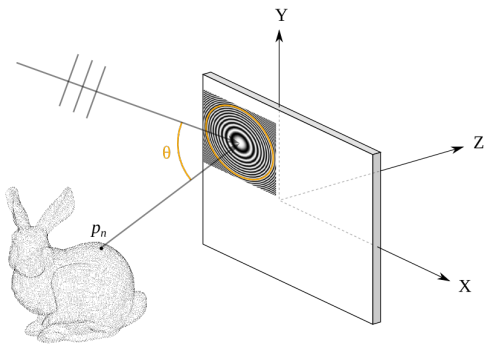
$$I(h) = A_R \sum_{n=1}^{N} A_n \cos(\underline{\varphi_R - \varphi_n})$$

difference of phases determines resulting phase for this pair
(wave interference)

Irregularity

Not all object points contribute to all hologram pixels:

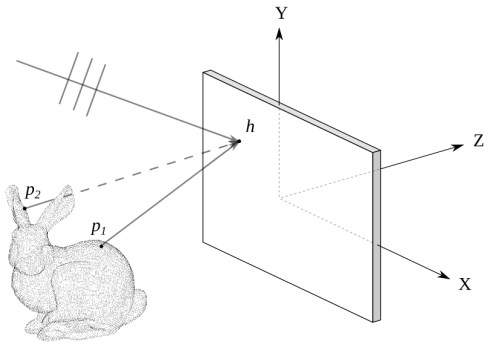
- Aliasing



Irregularity

Not all object points contribute to all hologram pixels:

- Aliasing
- Occlusion



Irregularity

algorithm with alias verification

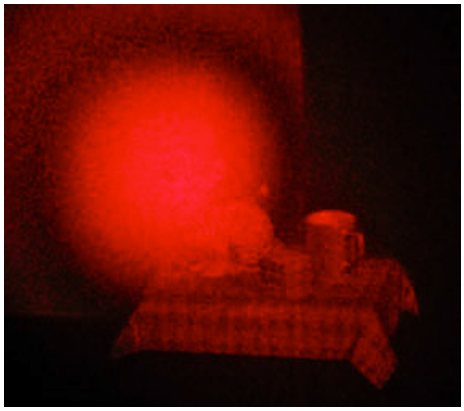
```
for ( p = 1 --> numPts ) {
      for ( h = 1 --> holoWidth * holoHeight ) {
            if ( angle( vec(p, h), ref ) <= maxFreqAngle ) {
                  hologram[h] += cos ( <phase> );
            }
      }
}
```

Irregularity

algorithm with visibility evaluation

```
for ( p = 1 --> numPts ) {
      for ( h = 1 --> holoWidth * holoHeight ) {
            if ( angle( vec(p, h), ref ) <= maxFreqAngle ) {
                  if ( visible(p, h) ) {
                        hologram[h] += cos ( <phase> );
                  }
            }
      }
}
```

Sample results



optical reconstruction

digital reconstruction