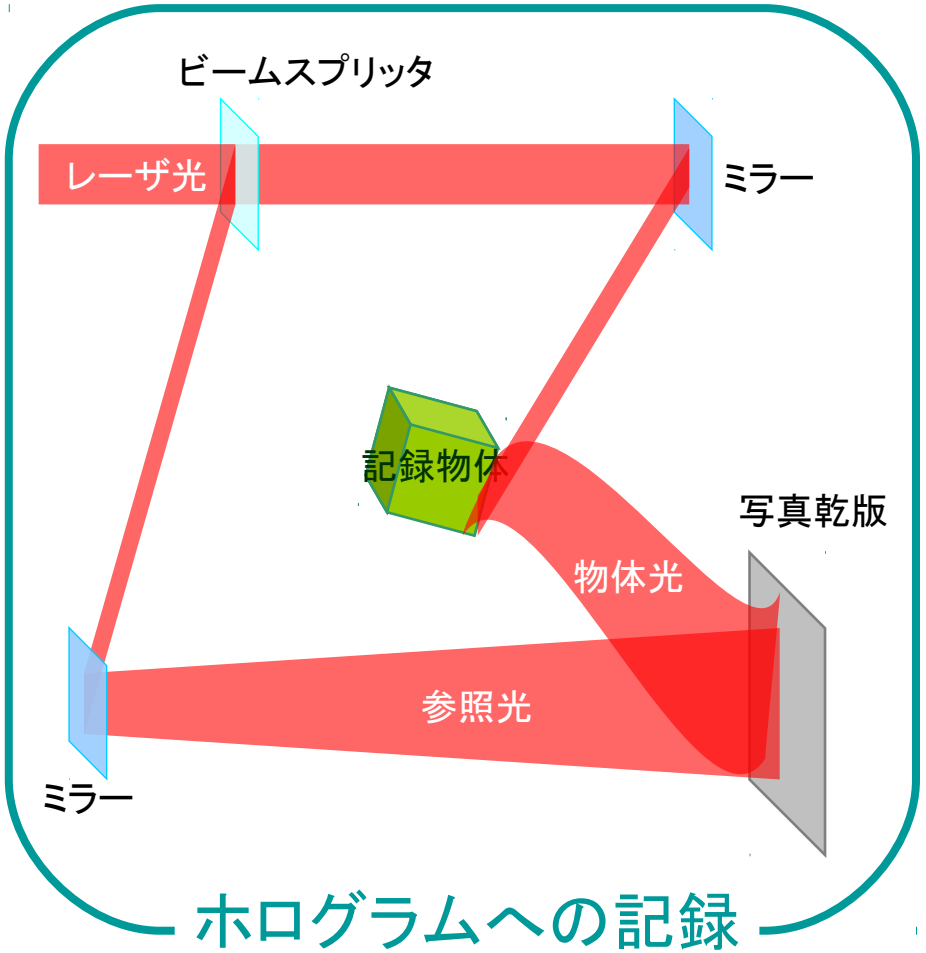
電子ホログラフィ技術を用いた立体映像システムに関する研究開発

下馬場 朋禄

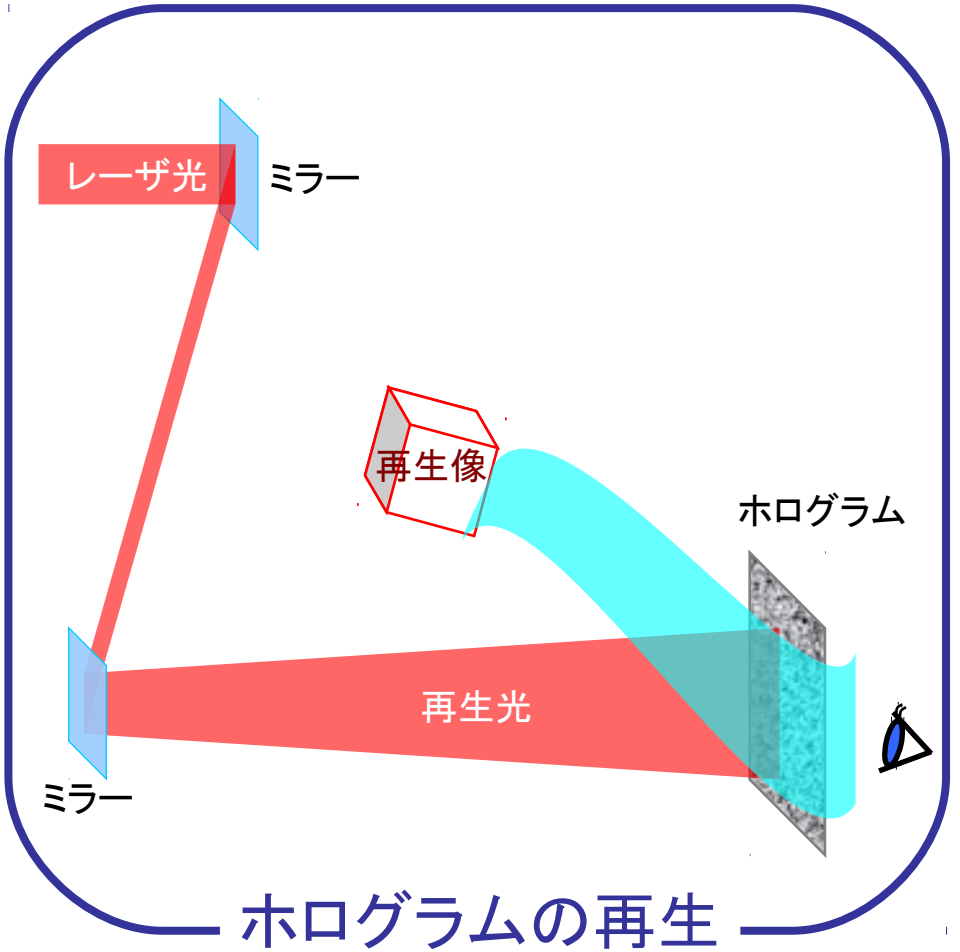
千葉大学大学院工学研究科

ホログラフィ



ホログラムへの記録

光の干渉を利用

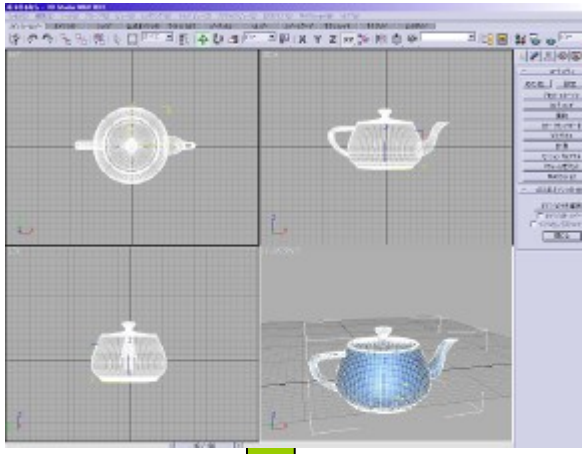


ホログラムの再生

光の回折を利用

電子ホログラフィ

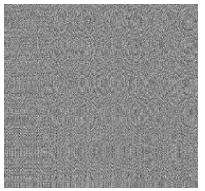
コンピュータ

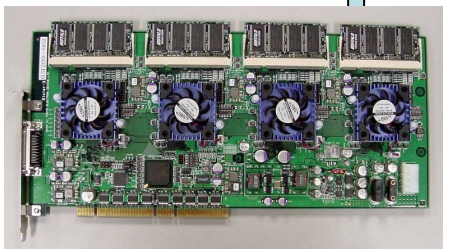


CGHを表示

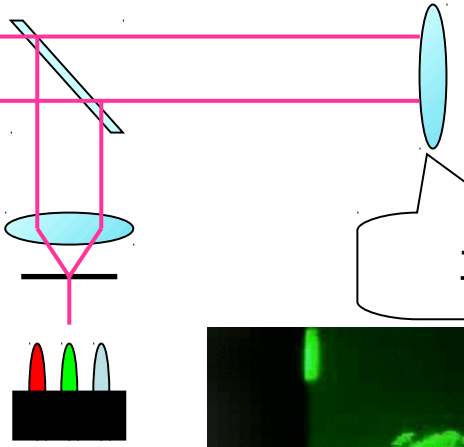
LCD Controller

LCD



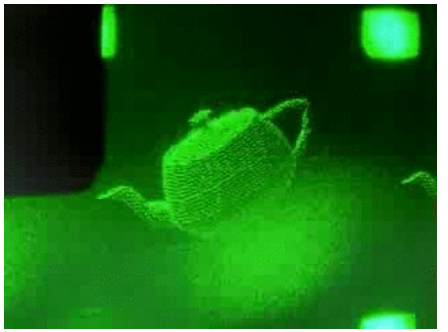


CGH計算



再生光

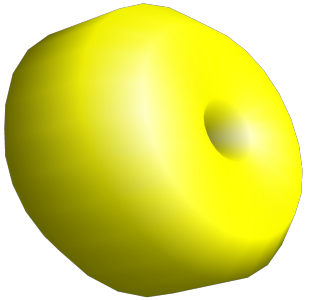
再生像

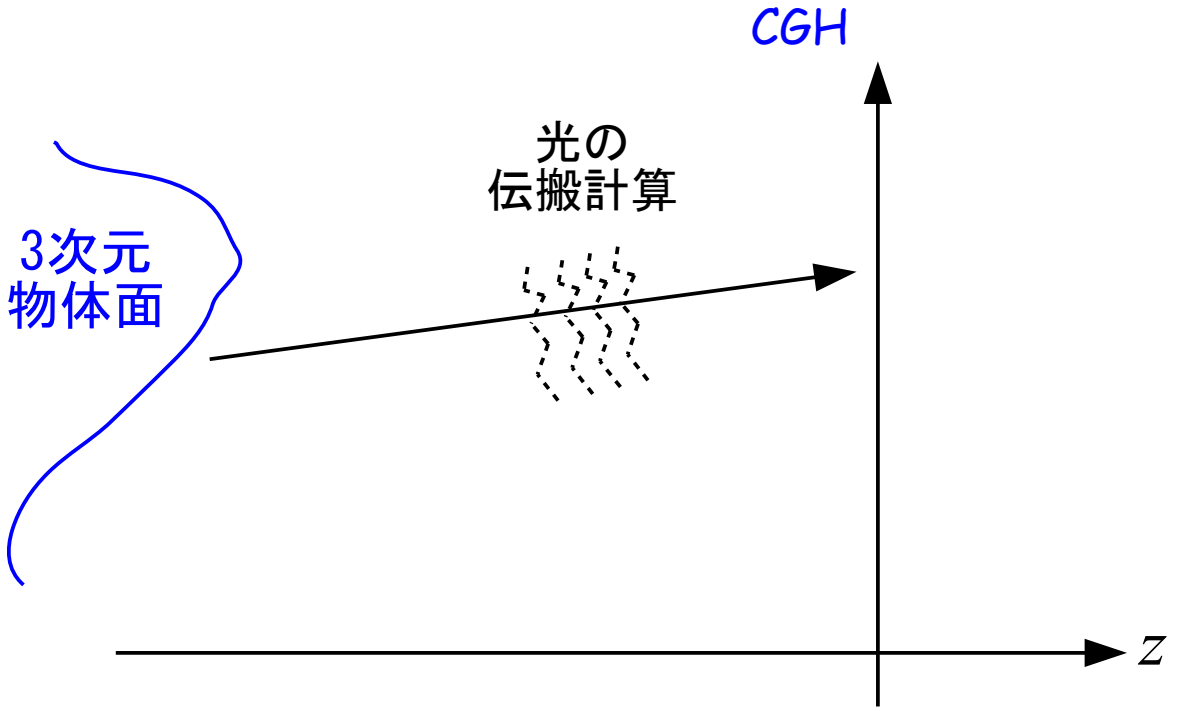


空間に3次元像を確認できる

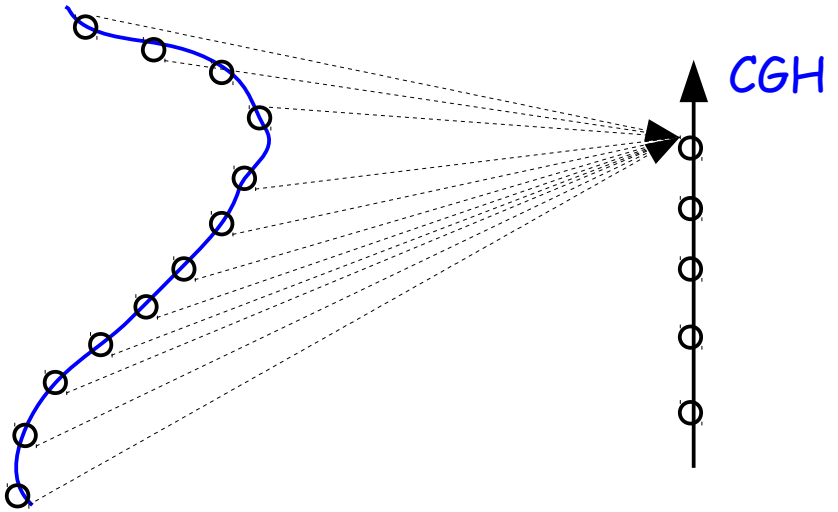
電子ホログラフィの問題点
（1） 再生像のサイズと視域
（2） ホログラムの計算量
（3） カラー化

CGHの計算方法

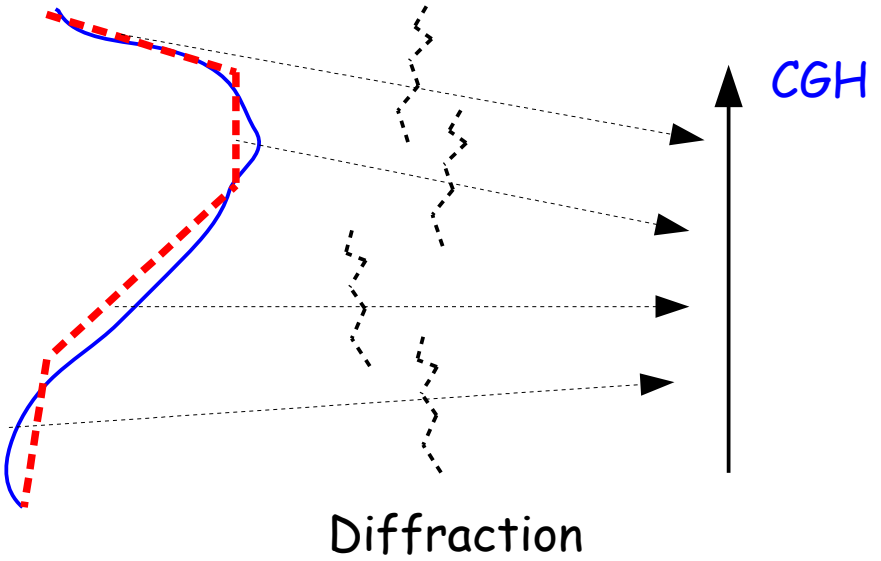




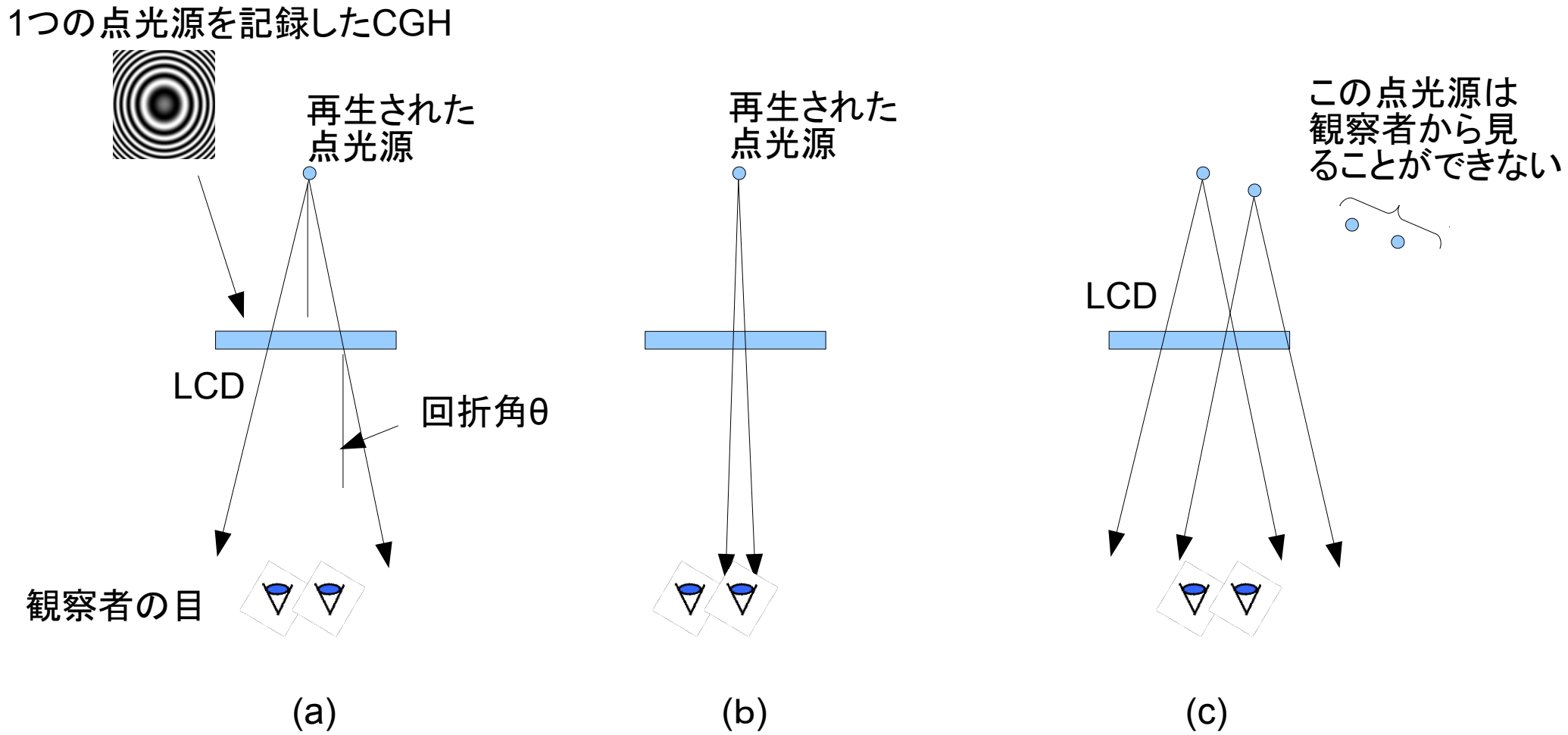
点光源モデル



ポリゴンモデル



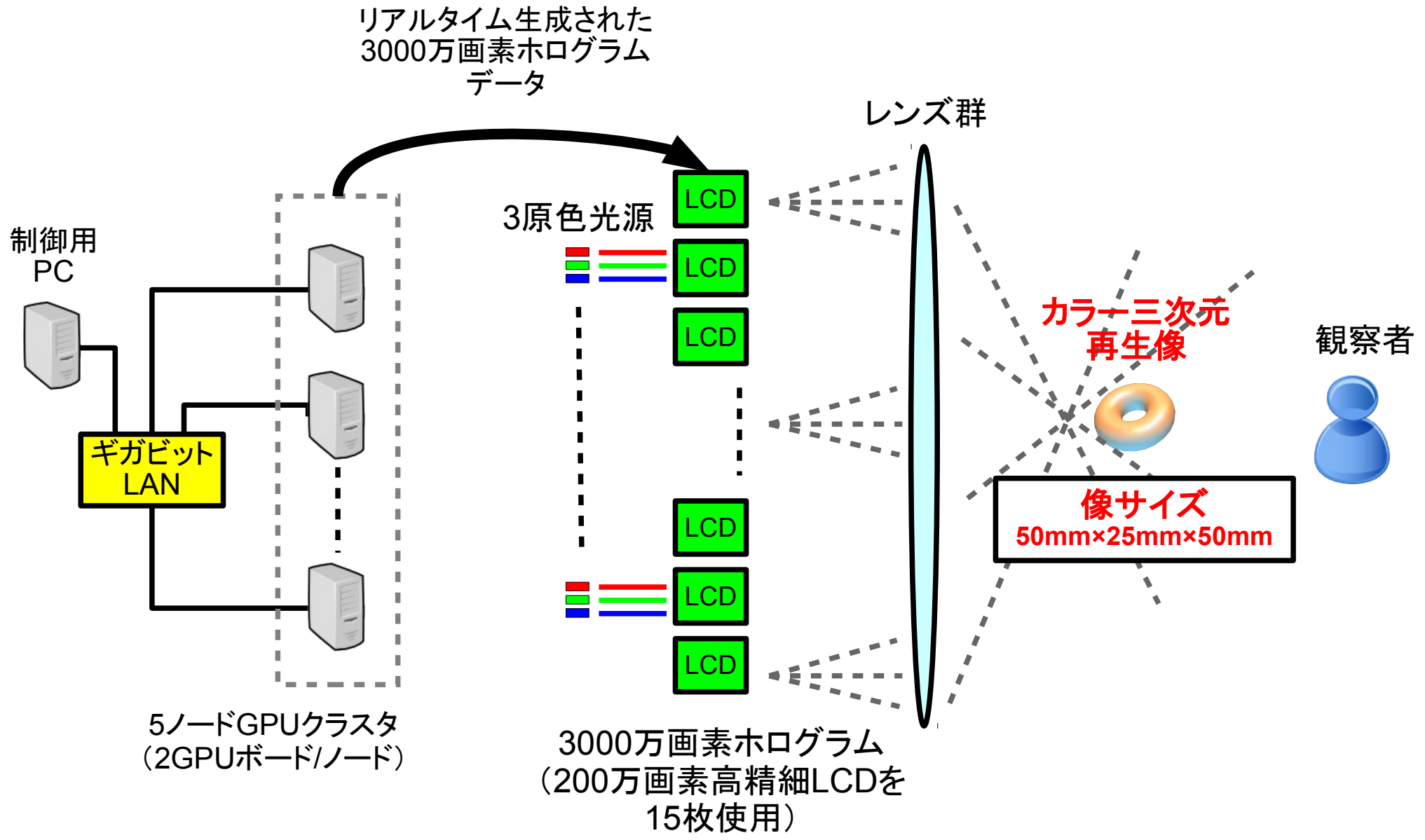
再生像のサイズと視域


1つの点光源を記録したCGH
再生された
点光源
LCD
回折角θ
観察者の目
(a)
再生された
点光源
(b)
この点光源は
観察者から見
ることができない
LCD
(c)

本研究の実施内容

- 複数枚のホログラム表示素子を用いた光学系の開発
- ホログラムを高速生成できるアルゴリズム（波面記録法）の開発
- 波面記録法のGPUへの実装
- カラー再生手法の開発
- 副次的な産物
  - 波動光学計算ライブラリ：CWOライブラリ
  - 任意曲面フレネル回折計算

システム全体の概略図

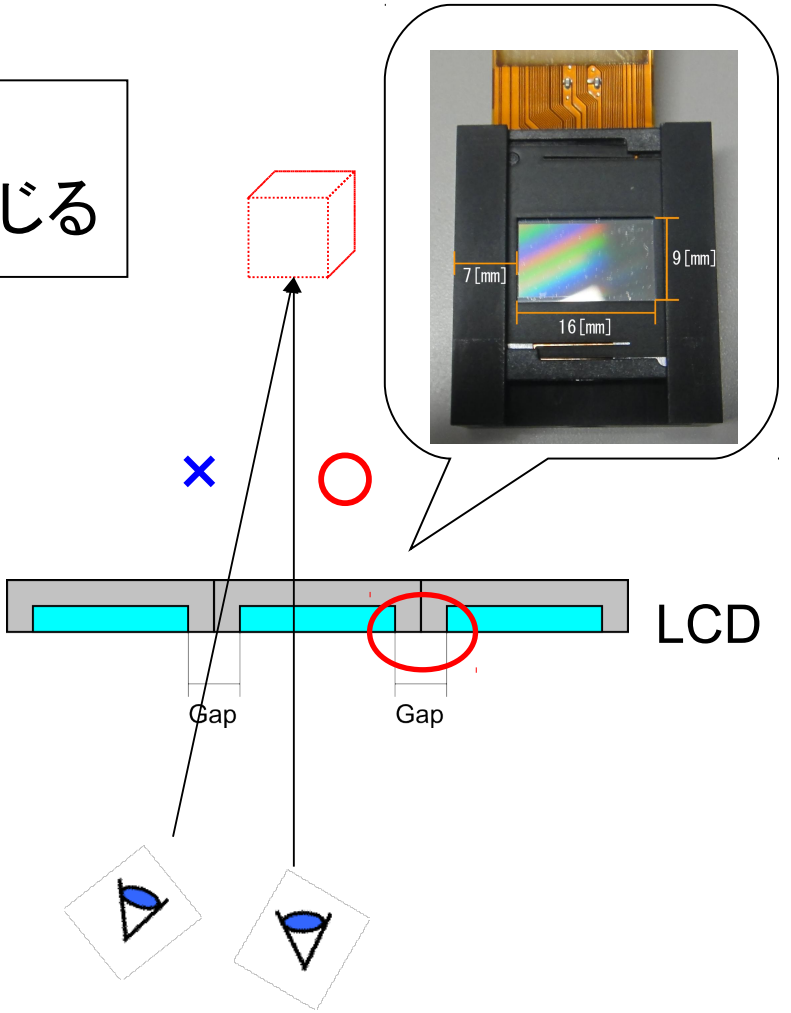

リアルタイム生成された
3000万画素ホログラム
データ
レンズ群
制御用
PC
3原色光源
LCD
LCD
LCD
LCD
LCD
LCD
ギガビット
LAN
カラー三次元
再生像
観察者
像サイズ
50mm×25mm×50mm
5ノードGPUクラスタ
（2GPUボード/ノード）
3000万画素ホログラム
（200万画素高精細LCDを
15枚使用）

複数枚のホログラム表示素子を用いた光学系

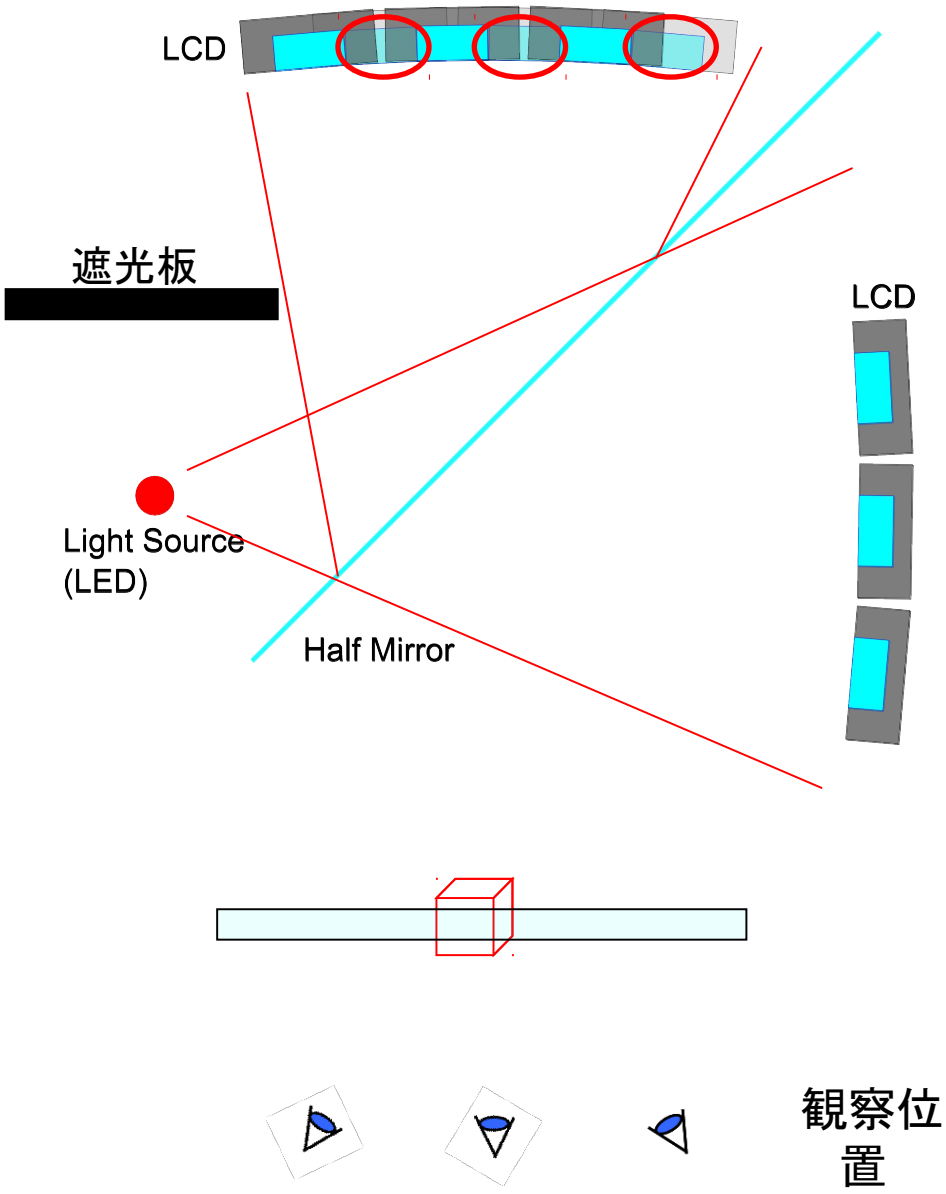
ディスプレイを並べると
ディスプレイ間に隙間（ギャップ）が生じる

ギャップによって再生像が
途切れてしまう

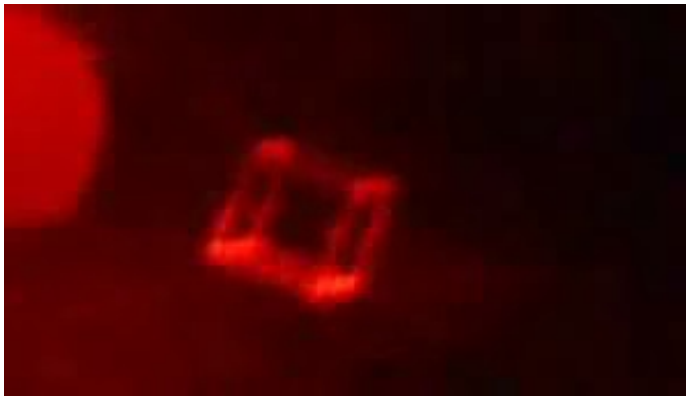
ギャップを解消する光学系の作成



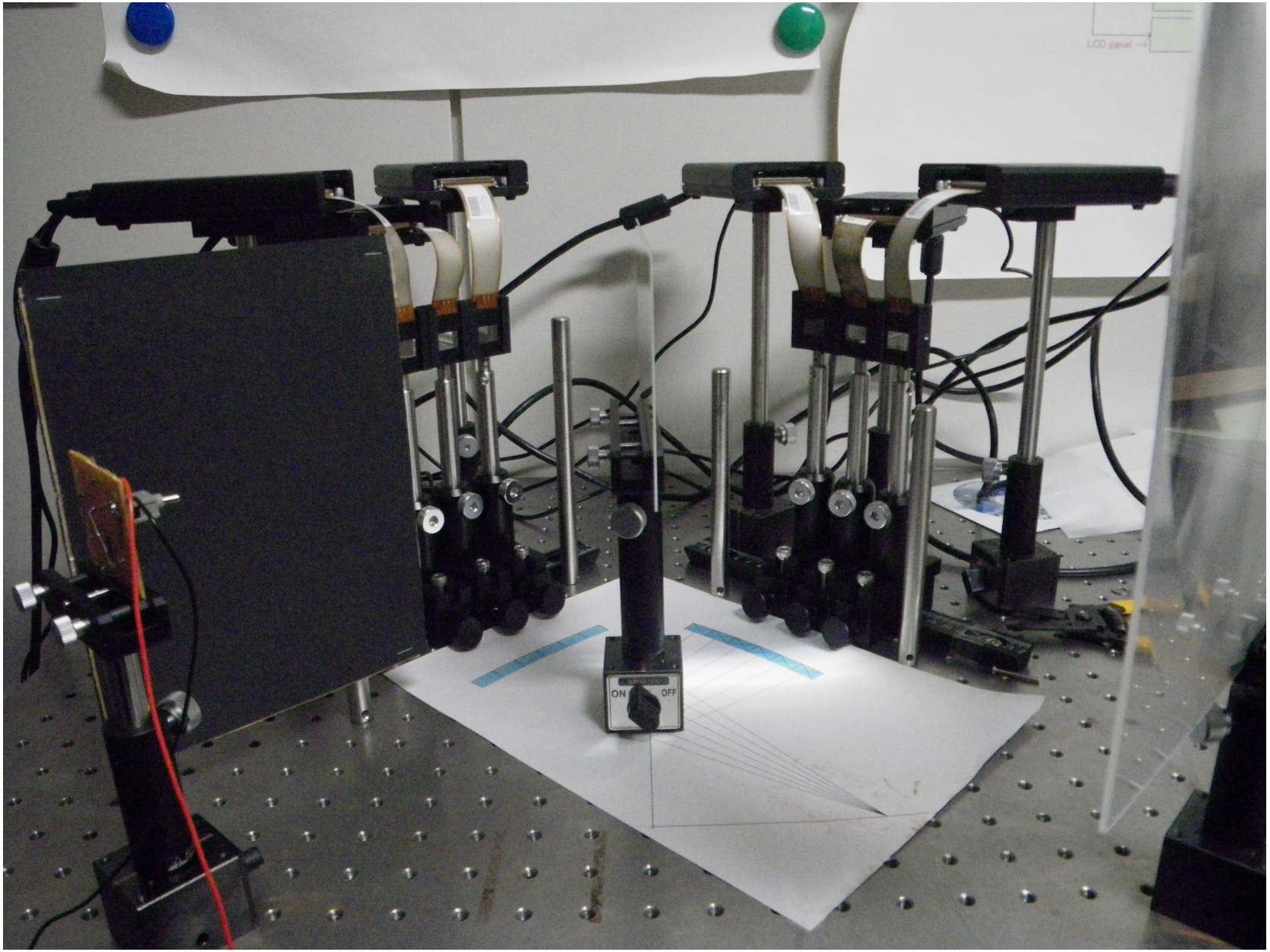
複数枚のホログラム表示素子を用いた光学系



	視域角 [deg]	視域幅 [cm]
LCD1枚（理論値）	4	3.5
本光学システム（理論値）	14	12.3
本光学システム（計測値）	13.1	11.5

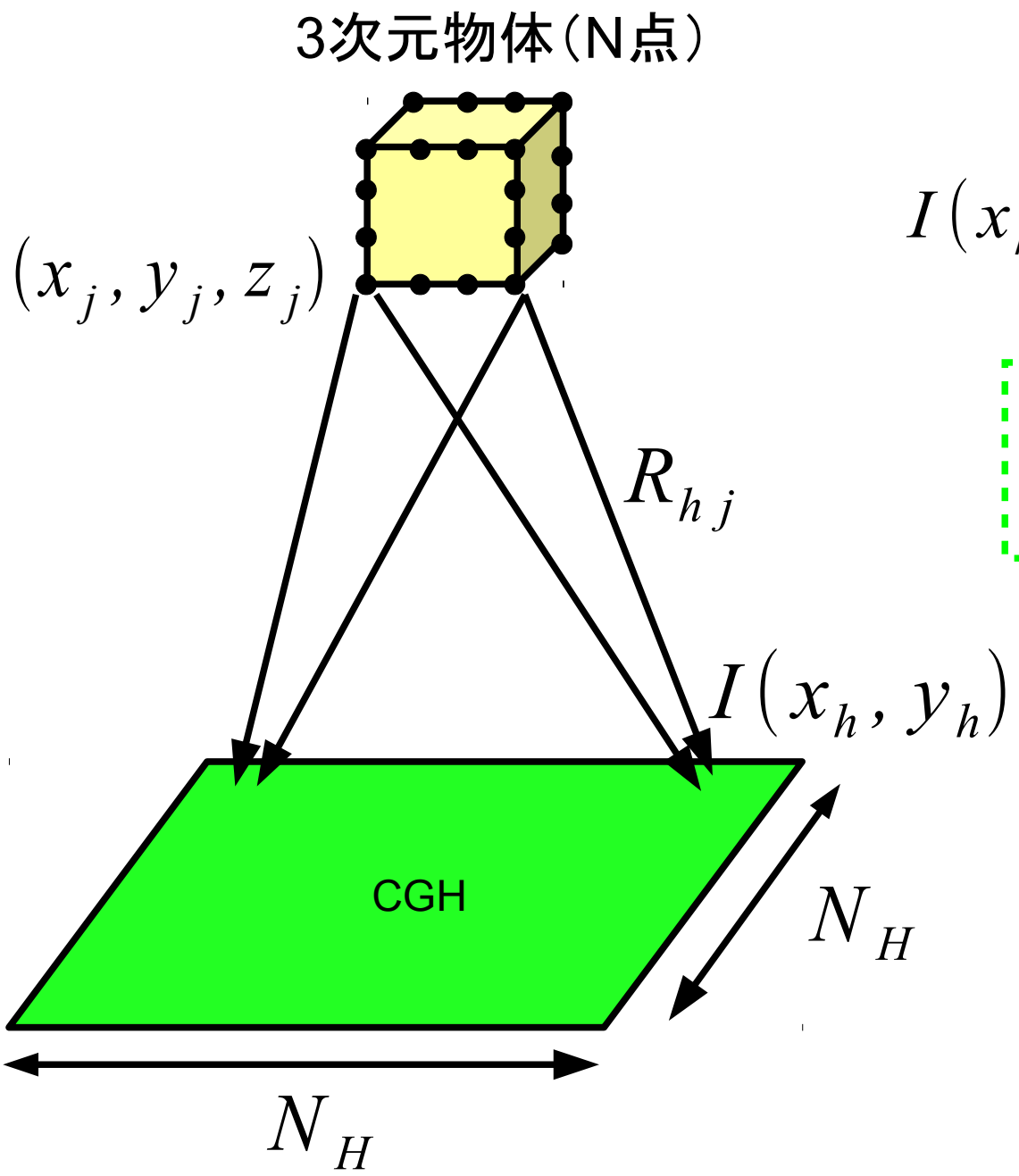


開発した光学系




NP901WJ
ViewLight
GALAXY
EVO

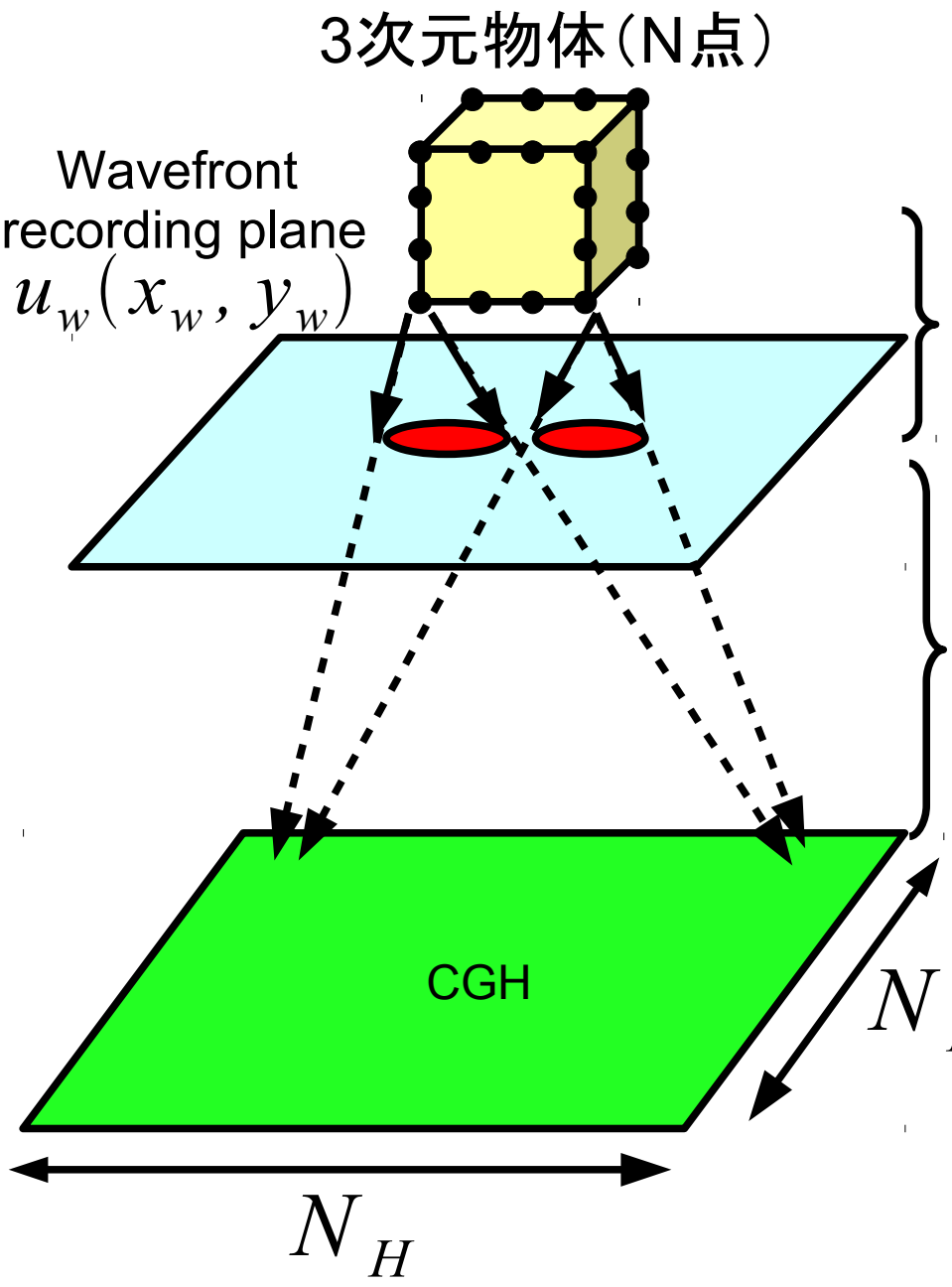
点光源モデル



$$I(x_h, y_h) = \sum_{j}^{N} A_j \cos\left(\frac{2\pi}{\lambda} R_{hj}\right)$$

計算量 $O(N N_H^2)$

ホログラムを高速生成できるアルゴリズム（波面記録法）の開発



第1ステップ：直接計算

$$u_w(x_w, y_w) = \sum_j^N \frac{A_j}{R_{wj}} \exp(i \frac{2\pi}{\lambda} R_{wj})$$

第2ステップ：FFTを用いた回折計算

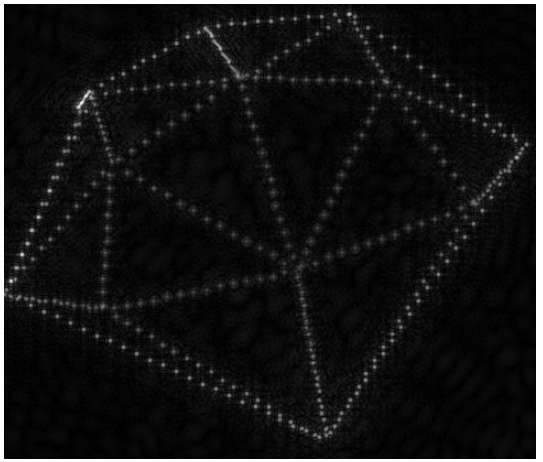
$$\begin{aligned} u(x,y) &= \frac{\exp(i\frac{2\pi}{\lambda}z_2)}{i\lambda z_2} \iint u_w(x_w, y_w) \times \\ &\quad \exp(i\frac{\pi}{\lambda z_2}((x-x_w)^2 + (y-y_w)^2))dx_w dy_w \\ &= \frac{\exp(i\frac{2\pi}{\lambda}z_2)}{i\lambda z_2} \mathcal{F}^{-1}\left[\mathcal{F}\left[u_w(x,y)\right] \cdot \mathcal{F}\left[h(x,y)\right]\right] \end{aligned}$$

CGH計算

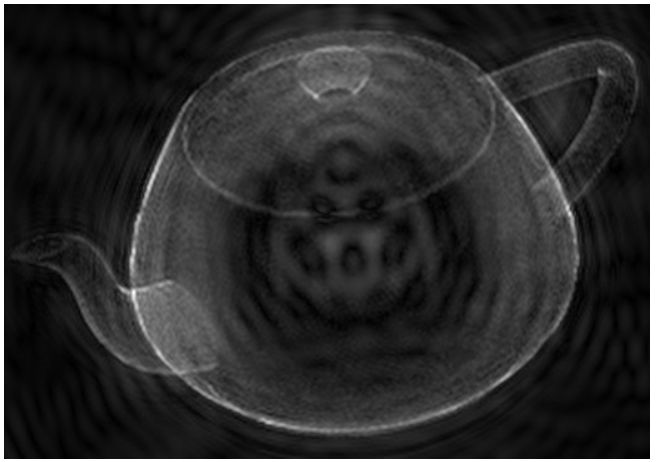
$$I(x_h, y_h) = \Re[u(x,y)r^*(x,y)]$$

波面記録法（CPU，2009年）

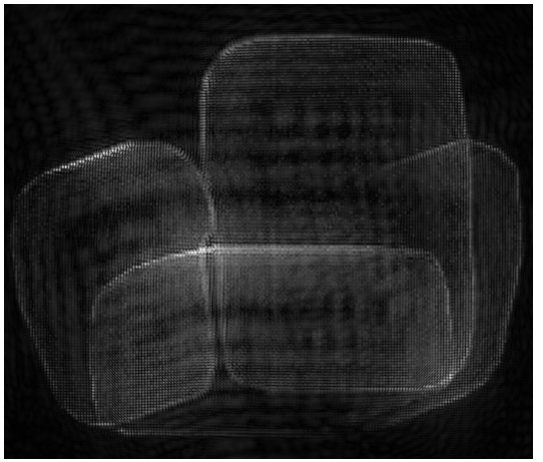
Emerald（407点）



Teapot（48277点）



Sofa（52380点）



CGHサイズ：1024×1024

参照光波長：633nm

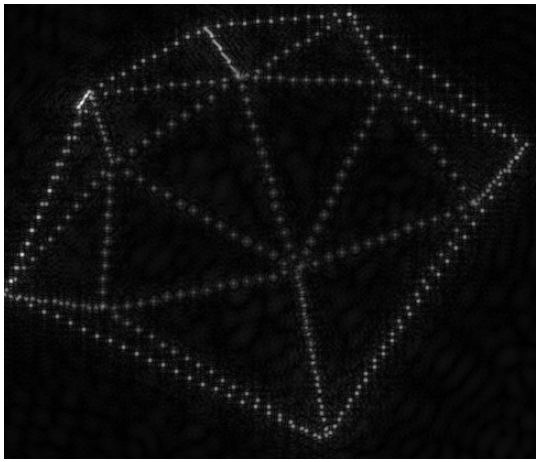
波面記録面とCGHのサンプリング間隔：8.5μm

3次元物体~波面記録面の距離 z_1： 5mm

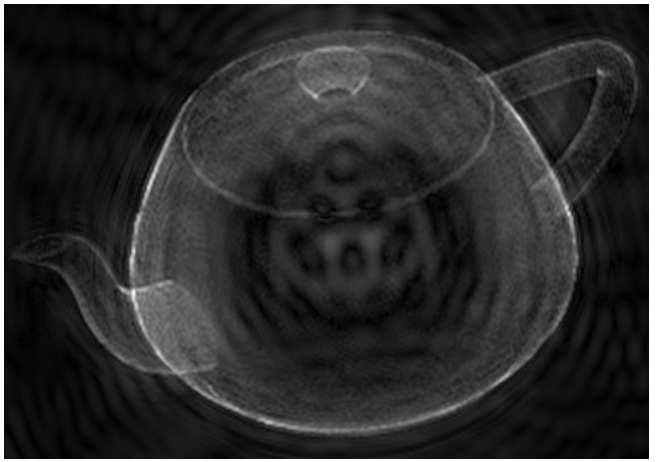
波面記録面~CGHの距離 z_2： 300mm

波面記録法（CPU，2009年）

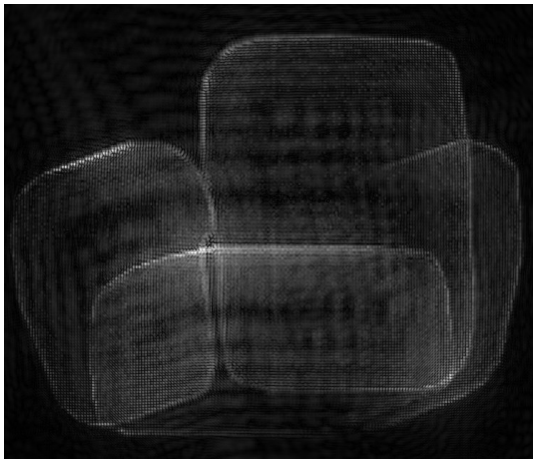
Emerald（407点）



Teapot（48277点）



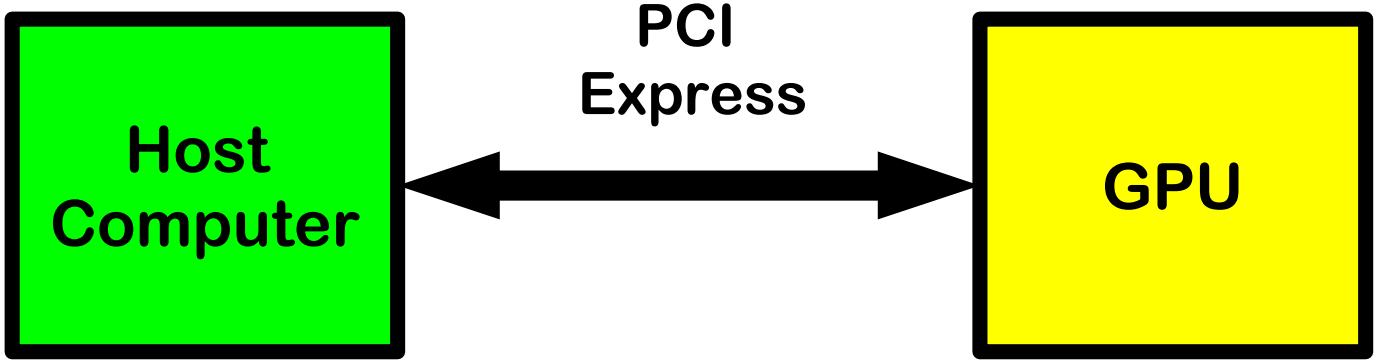
Sofa（52380点）



	従来手法・直接計算（CPU）		本手法（CPU）	従来手法・直接計算（GPU・GeForceGTX260）
***Emerald*（407点）**	13,941ms	6.3倍 →	2,216ms	17ms
Teapot（48,277点）	2,791,068ms	366倍 →	7,631ms	1,380ms
Sofa（52,380点）	3,025,730ms	302倍 →	10,024ms	1,497ms

※　CPU（Intel Core2 Quad Q6600）上の計算は1CPUスレッドで行ったもの計測した

GPUコンピューティング



http://www.elsa-jp.co.jp/products/hpc/tesla_c2075/index.html

- GPUをコプロセッサとして使用
- GPUの選択肢は2つ
  - NVIDIA社
    - 開発環境はCUDA，OpenCL，OpenACCなど
  - AMD社
    - 開発環境はCAL，Brook+，OpenCLなど

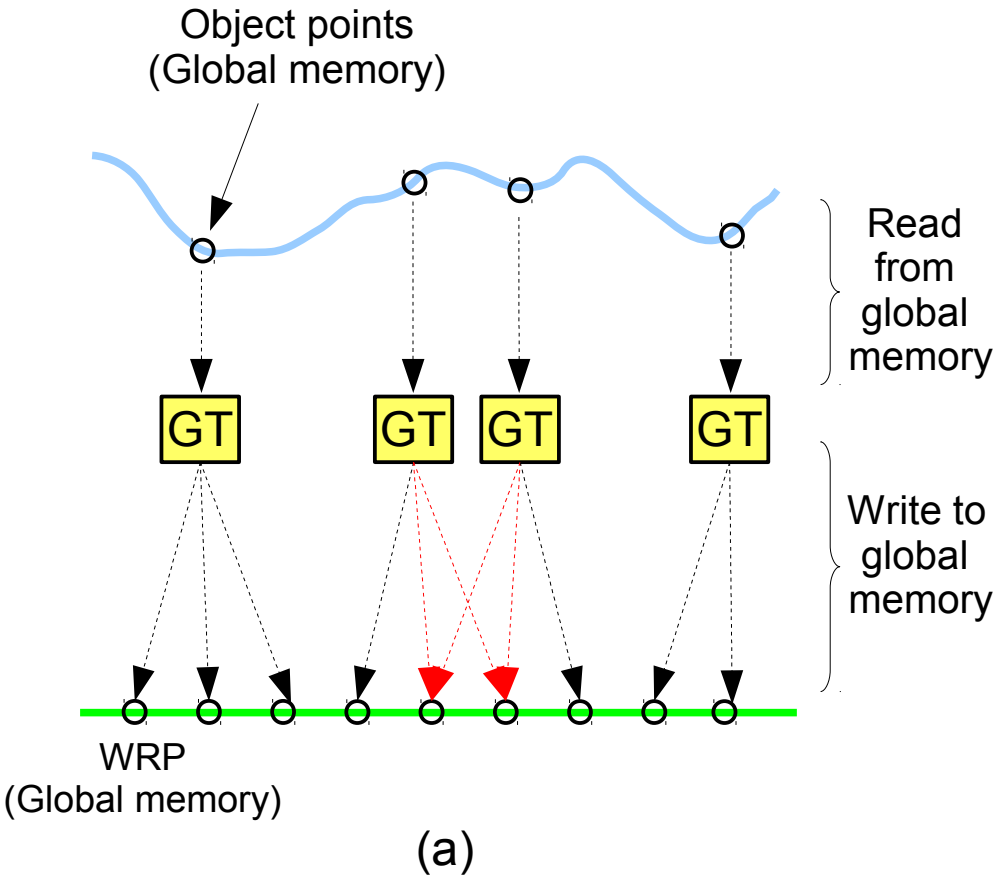
GPU上でのCGHの直接計算（2010年）

1920×1080画素CGHを直接計算した場合にNVIDIA GPUとAMD GPUの比較（2010年）

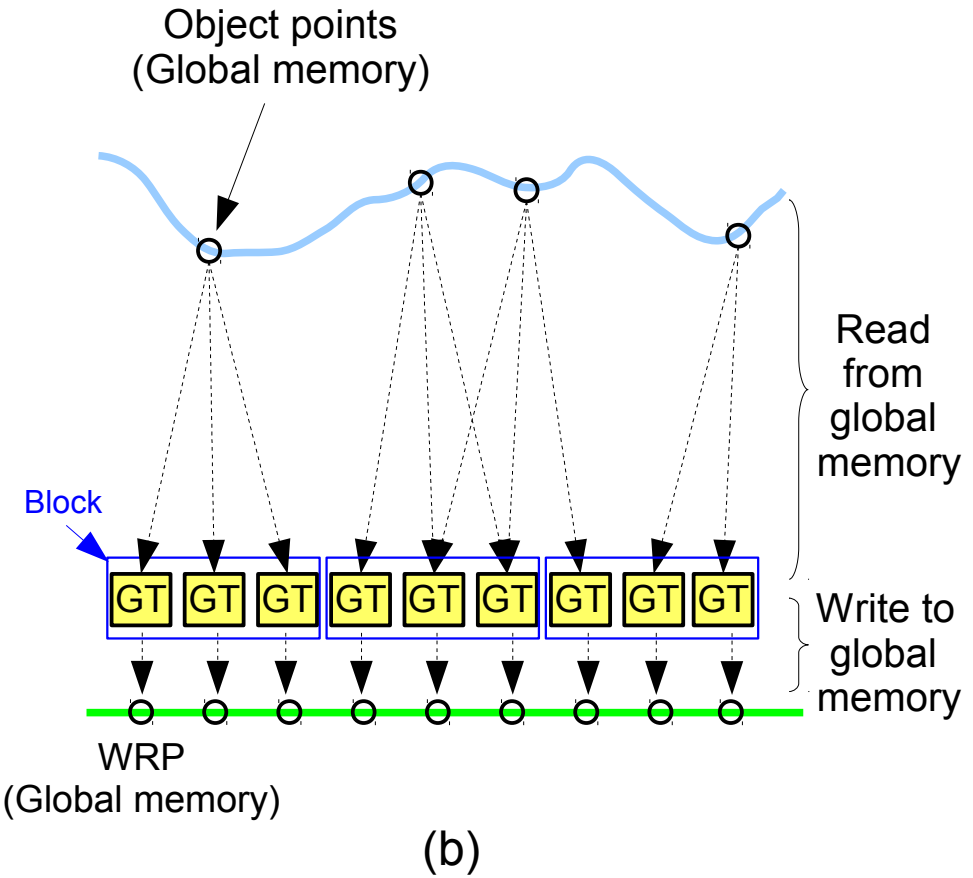
Number of object points	Time(ms)			
	CPU	NVIDIA GPU	AMD GPU	
			Without optimization	With optimization
512	30×10^3	33	215	19
1024	59×10^3	59	422	31
1536	88×10^3	85	630	44
2048	115×10^3	112	838	56
2560	146×10^3	139	1045	68
3072	174×10^3	165	1252	80
3584	204×10^3	192	1464	93

NVIDIA GPU : GeForce GTX260 (192 cores)
AMD GPU : RADEON HD5850 GPU (1440 cores)

波面記録法のGPUへの実装（2012年）



(a)



(b)

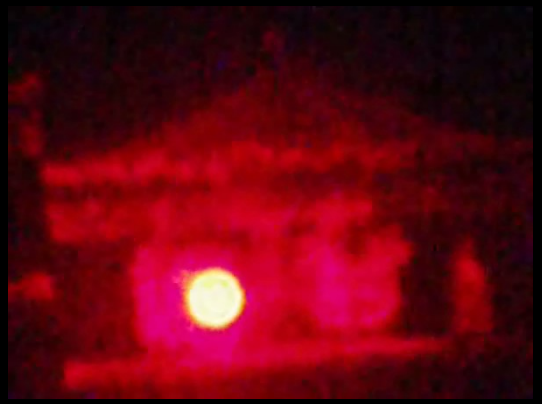
NVIDIA GeForce GTX580に実装

GT:GPUスレッド

再生像

Dinosaur
about 10,000 points

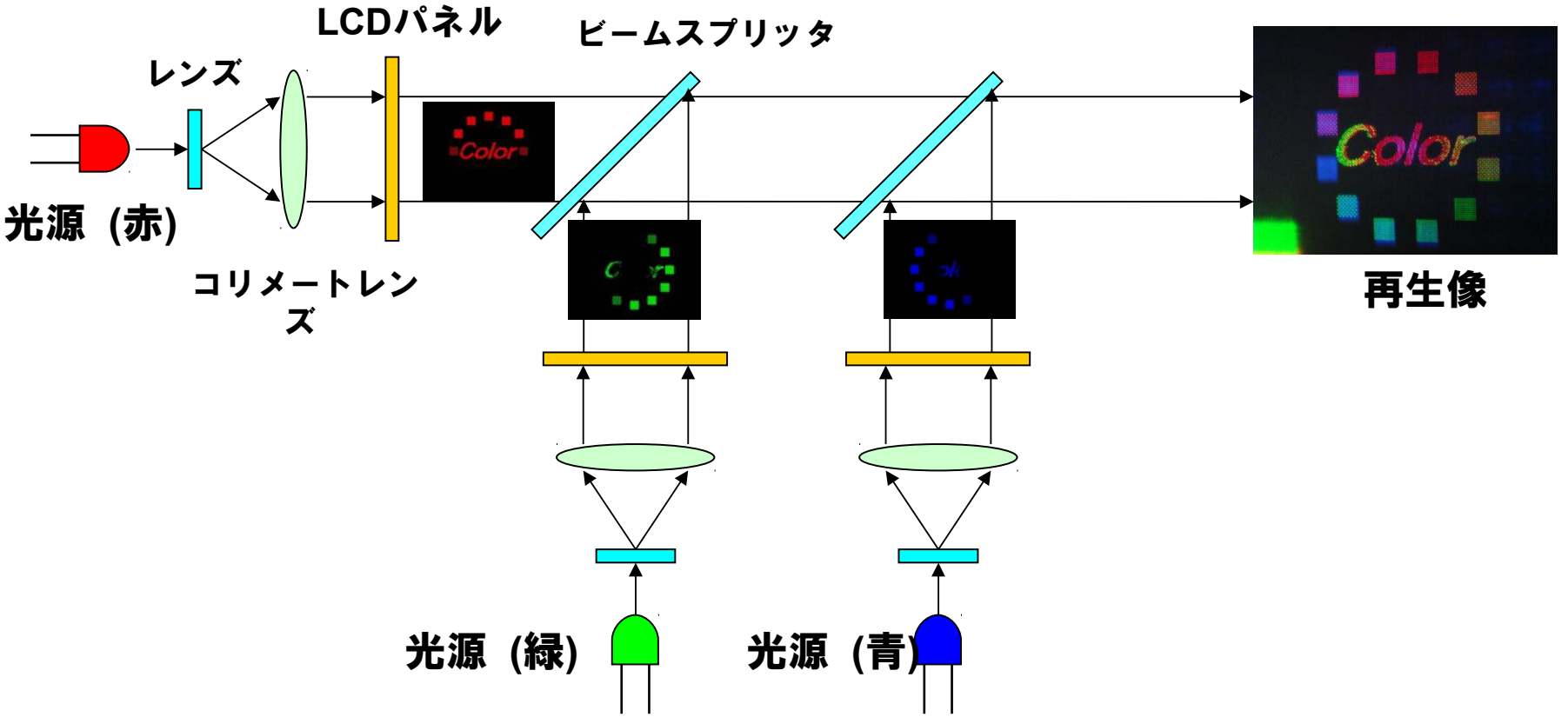
Earth
about 30,000 points
90 fps



Merry-go-round
about 100,000 points
23 fps

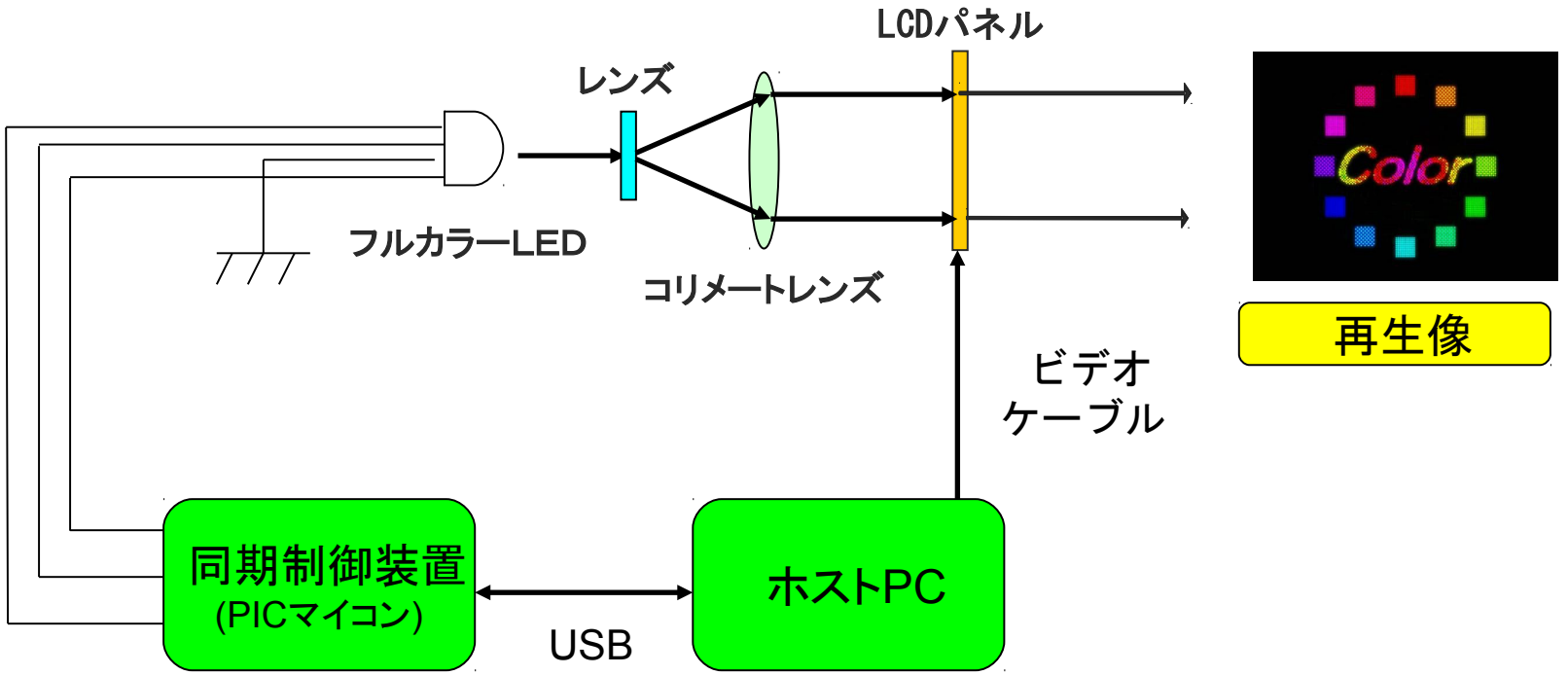
CGH size : 1920 x 1080　　Reference light : LED
GPU : NVIDIA GeForce GTX580

カラー化（3枚方式）



３枚のLCDパネル，ビームスプリッタが２枚必要

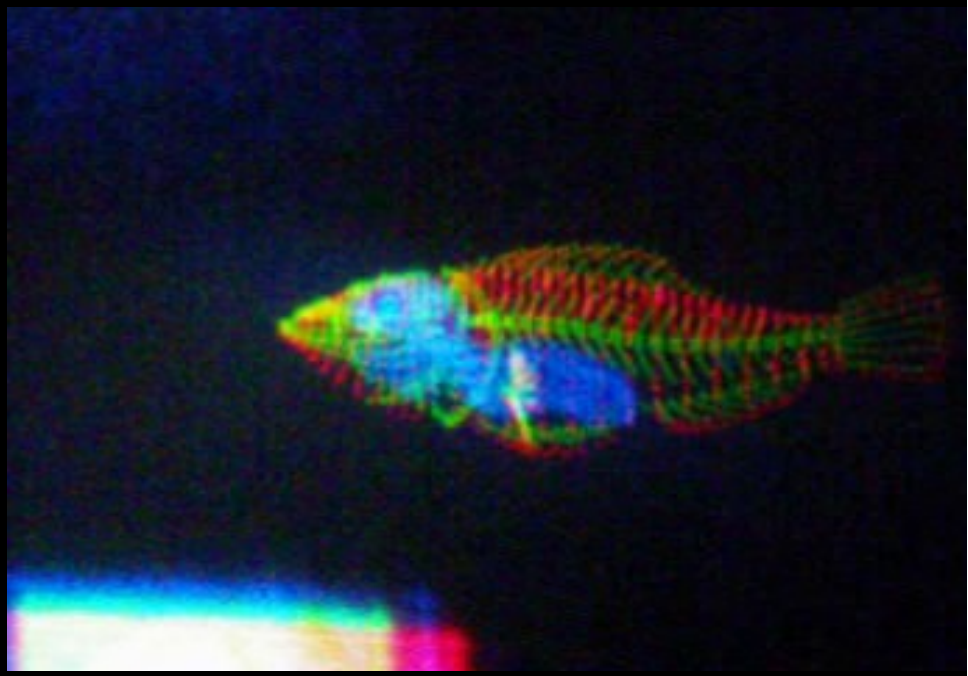
カラー化（時分割方式）



目の残像効果を利用

1光源1LCDのみでカラー電子ホログラフィ再生が可能

カラー再生像



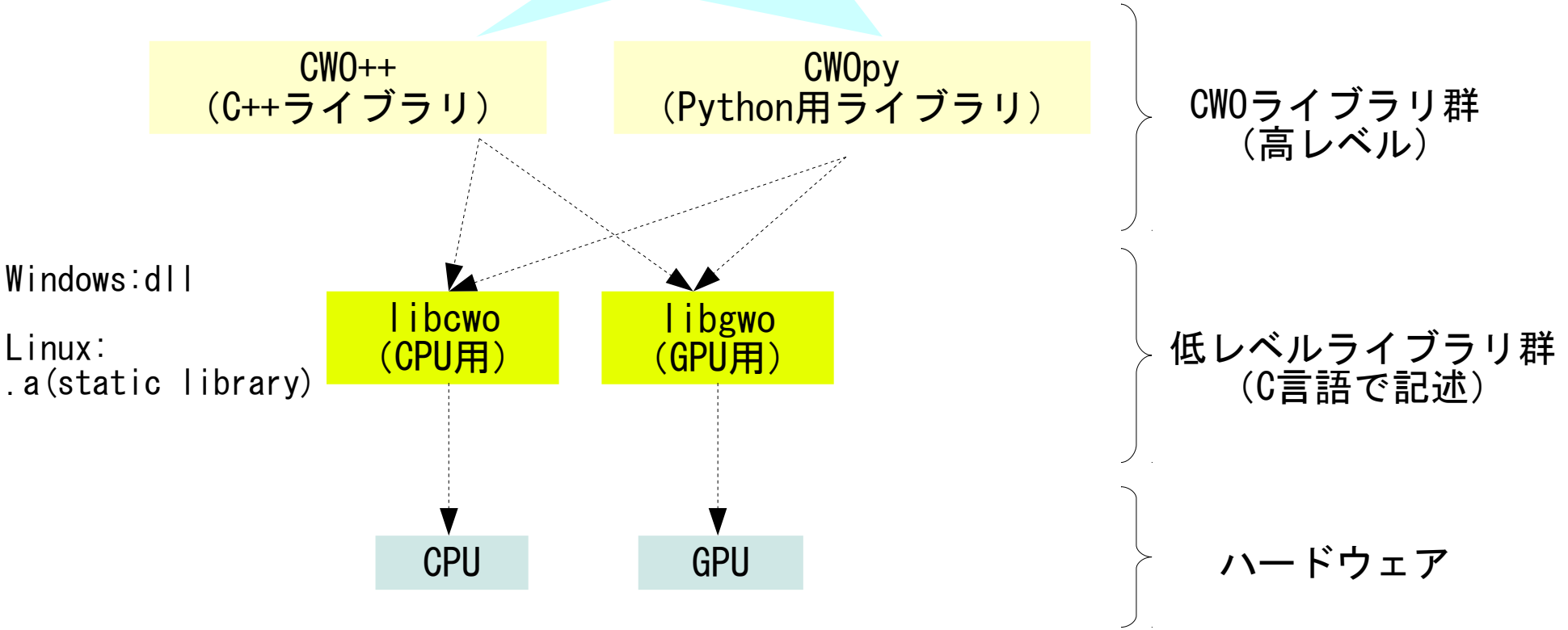
CWOライブラリ(Computational Wave Optics library)

- 波動光学を簡易に取り扱えるライブラリの開発
- GPUの知識のないユーザーでも，GPUの計算パワーを享受できる
- C++言語とPythonに対応
- 提供する主な機能
  - 2次元回折計算（8種類）
  - 3次元回折計算
  - CGH計算
- C++　と　Python
- CPUもしくはNVIDIA GPU（CUDA 4.0）
- Microsoft Windows XP以上で動作
- （準）オープンソース
- 下記URLからダウンロード可能

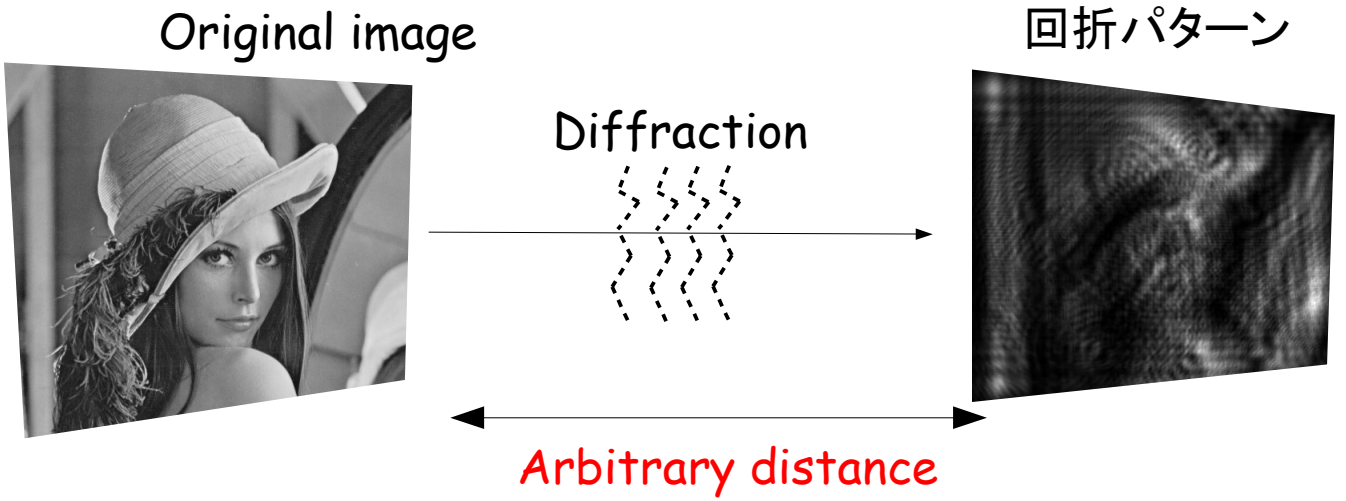
http://brains.te.chiba-u.jp/~shimo/cwo/

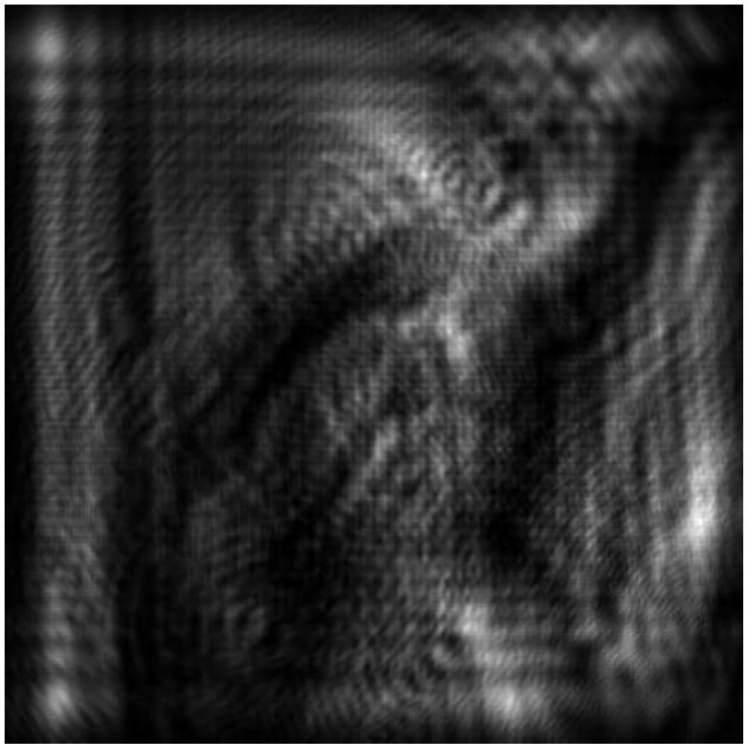
提供機能

光伝搬計算
・スカラー回折計算
・ベクトル回折計算
ホログラム計算
位相回復アルゴリズム
位相アンラッピング
他ライブラリ（OpenCVやKinect）とのインターフェース



フレネル回折計算の例





C++版 （CPU）

```
CWO c;
c.Load(“lena.bmp”);
c.Diffract(0.2,CWO_FRESNEL_CONV);
c.Intensity();
c.Scale(255);
c.Save(“lena_diffract.bmp”);
```

C++版 （GPU）

```
CWO c;
GWO g;
c.Load(“lena.bmp”);
g.Send(c);
g.Diffract(0.2,CWO_FRESNEL_CONV);
g.Intensity();
g.Scale(255);
g.Recv(c);
c.Save(“lena_diffract.bmp”);
```

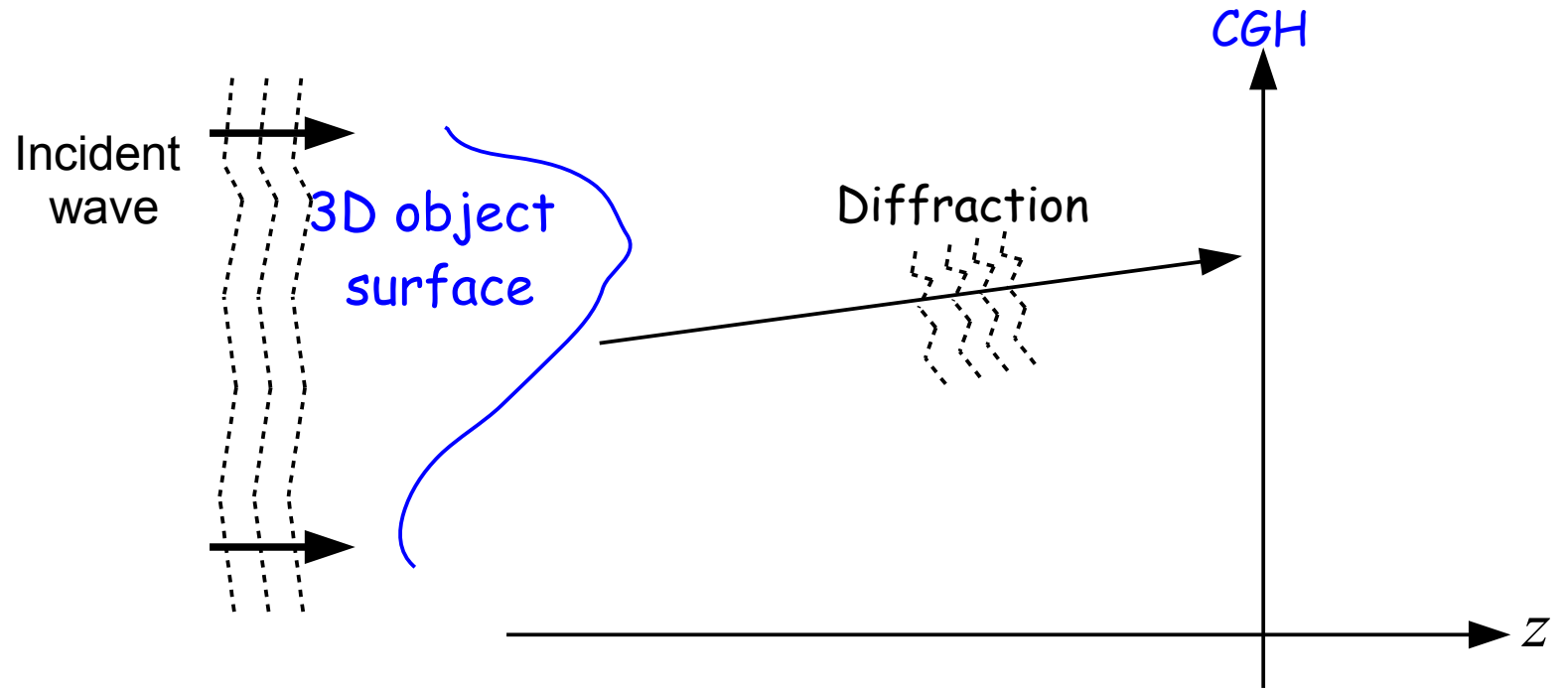
Python版 （CPU）

```
import CWO as c
c.Load(“lena.bmp”)
c.Diffract(0.2,CWO_FRESNEL_CONV)
c.Intensity()
c.Scale(255)
c.Save(“lena_diffract.bmp”)
```

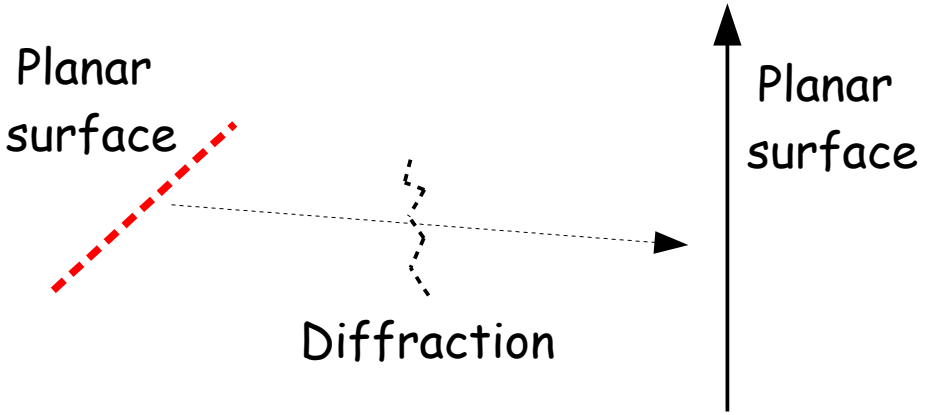
Python版 （GPU）

```
import CWO as c
import GWO as g
c.Load(“lena.bmp”)
g.Send(c)
g.Diffract(0.2,CWO_FRESNEL_CONV)
g.Intensity()
g.Scale(255)
g.Recv(c)
c.Save(“lena_diffract.bmp”)
```

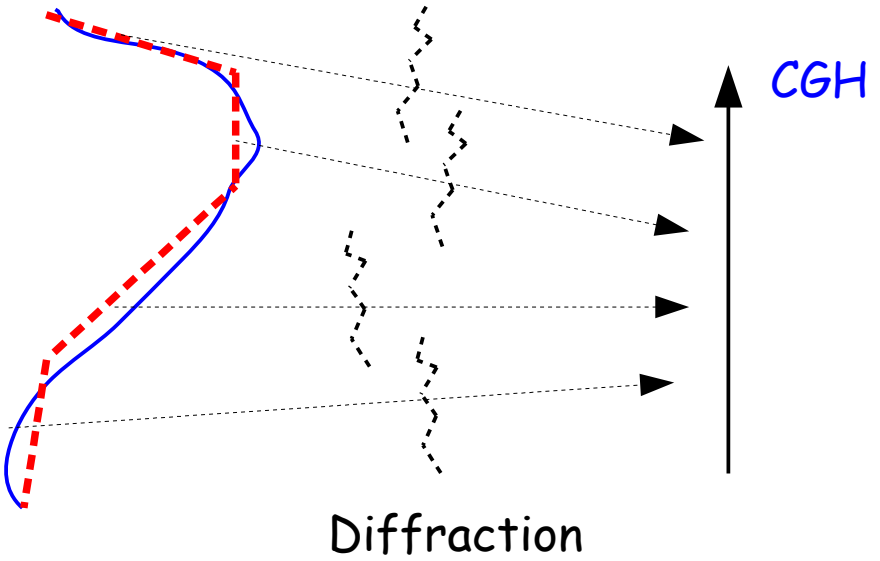
任意曲面フレネル回折



Normal Fresnel diffraction



Polygon approach



任意曲面フレネル回折

$$u(\mathbf{x}_2) = \frac{\exp(ik(z_0 + \frac{\mathbf{x}_2^2}{2z_0}))}{i\lambda z_0}$$

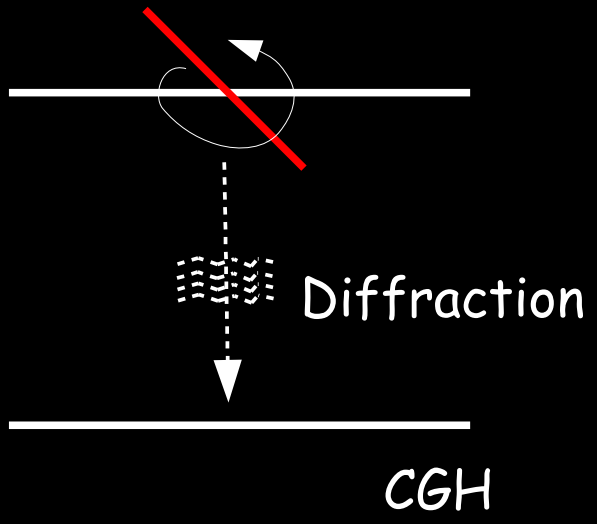
$$NUF\left[u_I(\mathbf{x}_1, d_1)u_1(\mathbf{x}_1, d_1)\exp(ik(-d_1 + \frac{\mathbf{x}_1^2}{2(z_0 - d_1)}))\right]$$

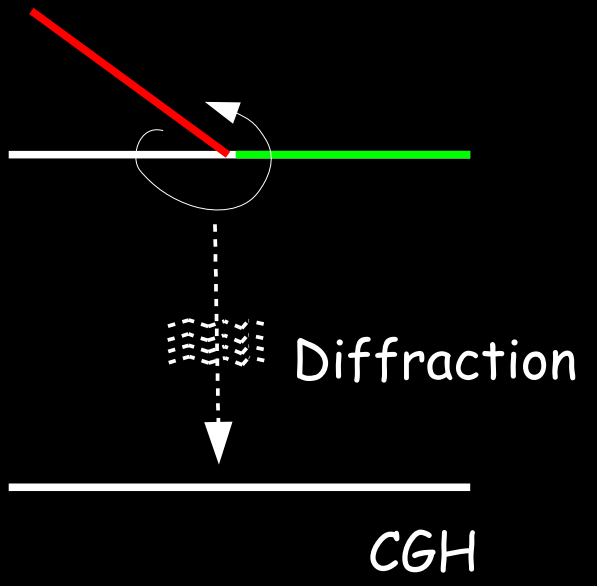
不等間隔高速フーリエ変換(NUFFT)

入射光

任意曲面

再生像





まとめ

- 複数枚のホログラム表示素子を用いた光学系の開発
  - 2400万画素ホログラムを表示可能，視域は約14°
- ホログラムを高速生成できるアルゴリズム（波面記録法）の開発
- 波面記録法のGPUへの実装
  - 10万点程度の複雑な3次元物体から2400万画素CGHを20fps以上で生成可能
- カラー再生手法の開発
  - 時分割法によるカラー再生
- 副次的な産物
  - 波動光学計算ライブラリ：CWOライブラリ
  - 任意曲面フレネル回折計算