

Réalité augmentée

Bruce Lane et Christophe Vestri

Plan du cours

- 29 février : Réalité augmentée intro et Html5/JS - **CV**
- 14 mars: Integration ionic/angular - **BL**
- 21 mars: Présentation artmobilib/patrimoine CASA - **BL**
- 18 avril: RA avec openCV - **CV**
- 25 avril : Autres outils: unity/unreal - **BL**

Réalité augmentée

Introduction

Christophe Vestri

Le lundi 29 février 2016

Introduction

Christophe Vestri

vestri@3DVTech.com

3DVTech

- Bureau d'étude
- Développement
- Consulting



www.3DVTech.com

Plan Cours1

- Réalité augmentée
 - RA à partir de Tags
 - RA à partir d'image
- Projet ArtMobilib
 - 3 librairies JavaScript
 - 3 exercices
- Démo

Qu'est-ce que la Réalité augmentée

- Augmentée:
 - Amplifier
 - Rehausser
 - Améliorer
- [Wikipédia](#): La **réalité augmentée** désigne les systèmes informatiques qui rendent possible la superposition d'un modèle virtuel 2D ou 3D à la perception que nous avons naturellement de la réalité et ceci en temps réel.
- [RAPro](#) : Combiner le monde réel et des données virtuelles en temps réel

Continuum réalité-virtualité



Environnement
réel



Réalité
augmentée



Réalité
virtuelle



Environnement
virtuel



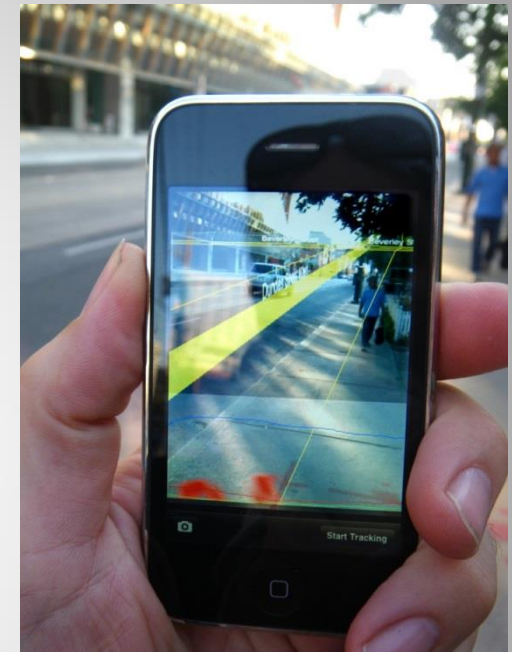
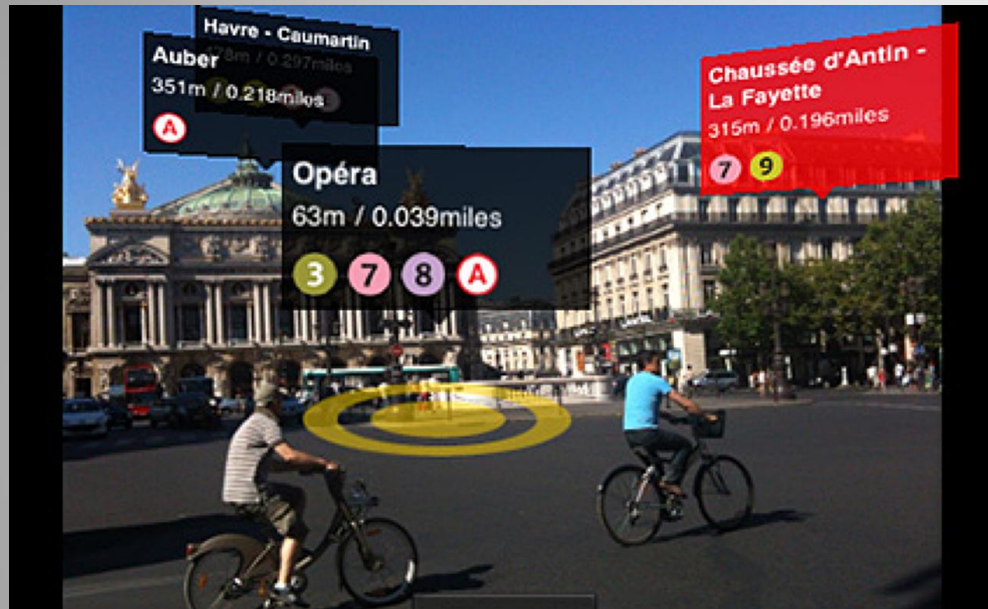
Qqs Demos

- RayBan: www.ray-ban.com/international/virtual-mirror
- webcamsocialshopper.com/
- Ford C-Max:
www.youtube.com/watch?v=bl8T9oYO5vY
- GoogleTranslate/Wordlens:
www.youtube.com/watch?v=06olHmcJjS0
- Domestic Violence
www.youtube.com/watch?v=ayPqRRCntVo

Exemples RA gustative

- [RAPro](#) : Combiner le monde réel et des données virtuelles en temps réel
- 5 sens:
 - Visuel: smartphone, lunettes...
 - Sonore: déficients visuels
 - Tactile/haptique: systèmes retour de force
 - Odorat: Cinema 4D
 - Goût:

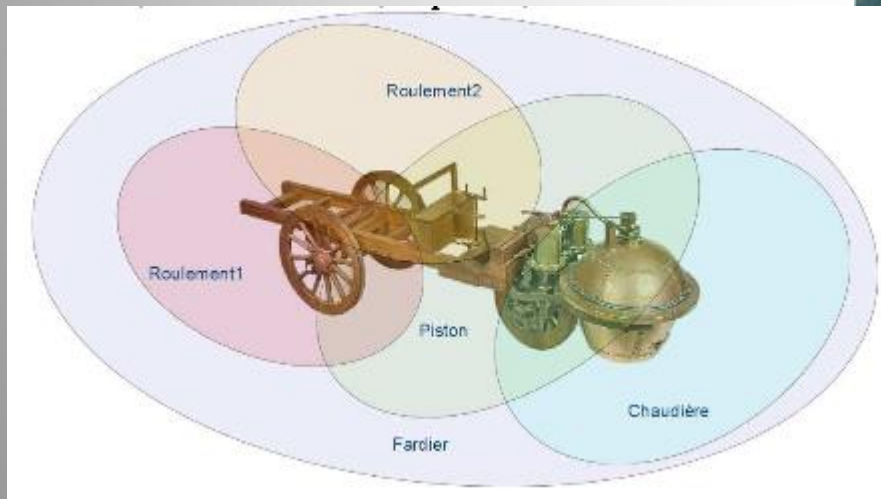
Exemples RA visuel



Exemples RA Sonore



Topophonie

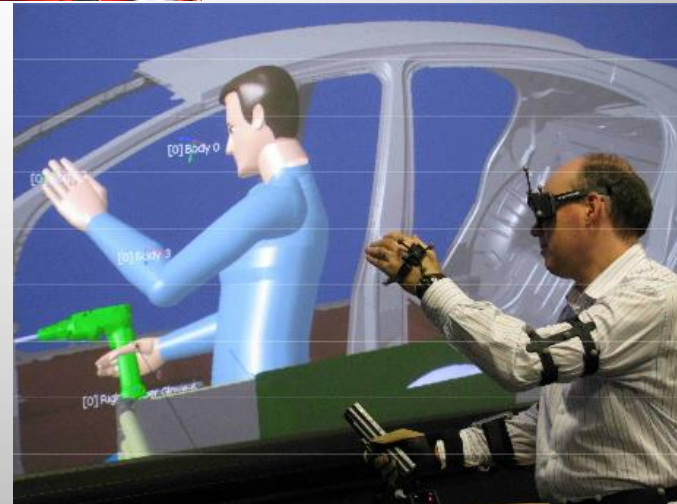


CNAM

Exemples RA Haptique



Sense-Roid



Haption

Examples RA Olfactive



AMBISCENT



Meta cookies

Examples RA gustative



TagCandy



Applications

- Augmentation de print



IKEA 2014



Idée3com : Application Brisach Vision



Applications

- Manuels augmentés



Applications

- Urbanisme



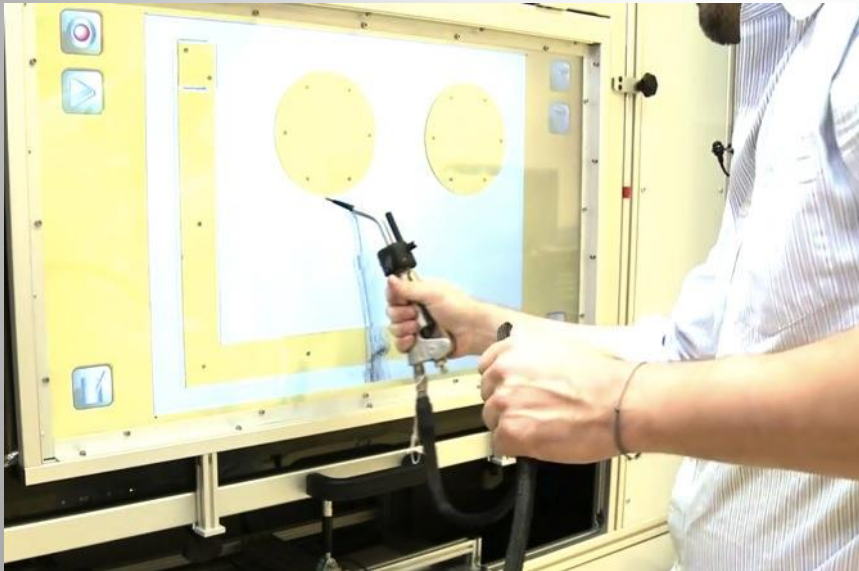
KRAKEN REALTIME



Métropole de Rennes

Applications

- Formation augmentée



CEA list & Renault : gestes techniques collage



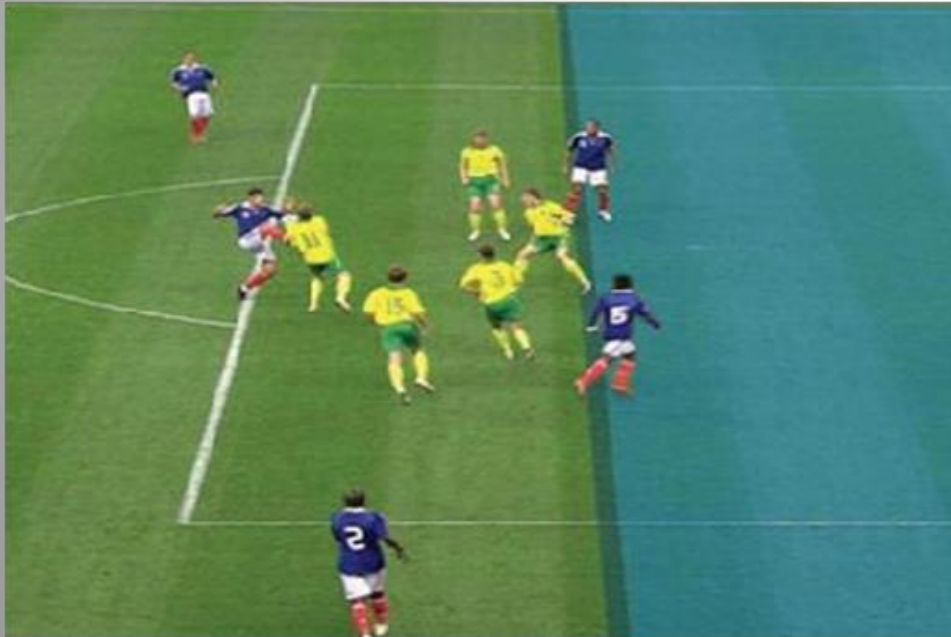
Institut de Soudure



Lincoln Electric

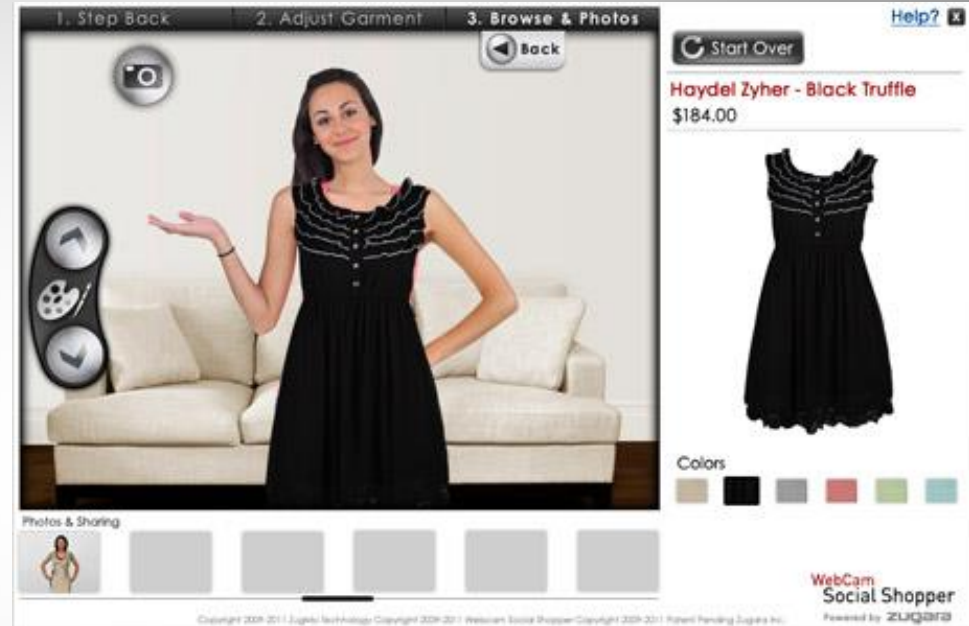
Applications

- TV



Applications

- Essayage sur internet



Applications

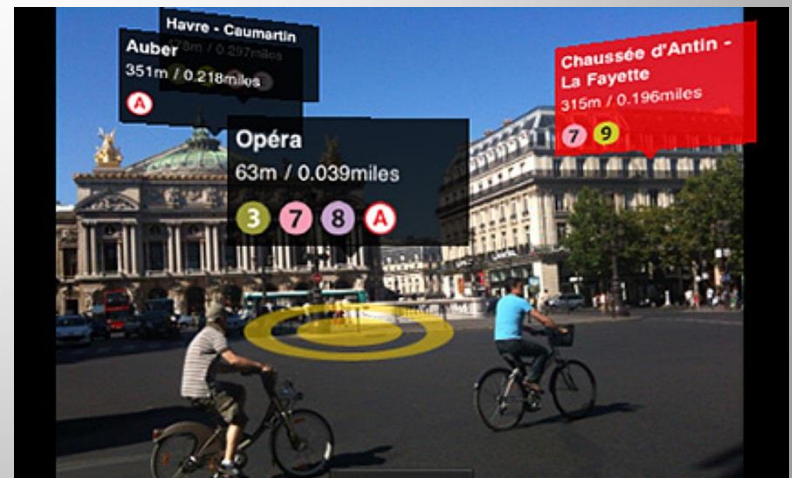
- Musées, art, tourisme



Museum d'histoires naturelles de Washington



MOMO urban art on the Williamsburg Art & Design Building in Brooklyn.



Applications

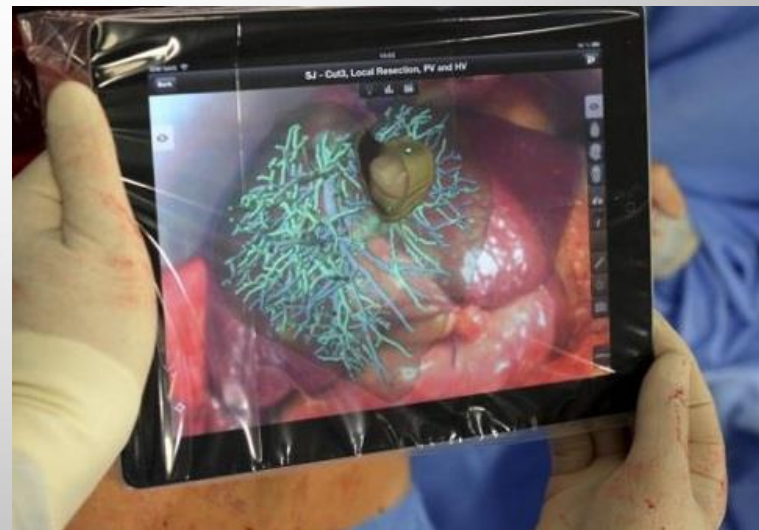
- Médical



VeinViewer



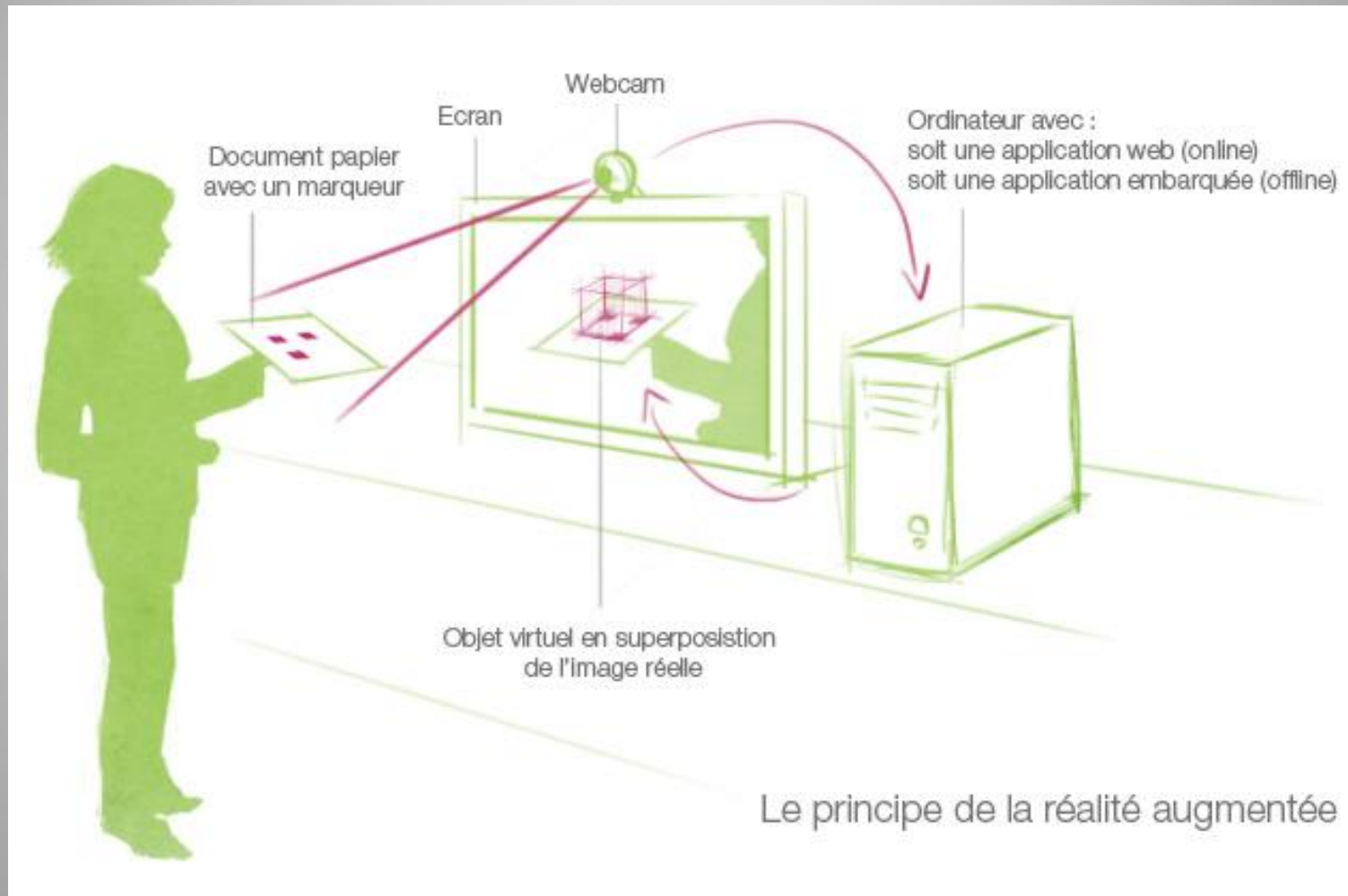
(cc) Grégory Maubon - 2012



Quelques entreprises 06

- **Robocortex**: SDK
- **Lm3labs**: interfaces interactives
- **Touchline Interactive**: Dev applis mobiles
- **Tokidev**: Dev applis mobiles
- **Wacan**: Dev applis mobiles
- **Avisto**: SSII

RA Fixe

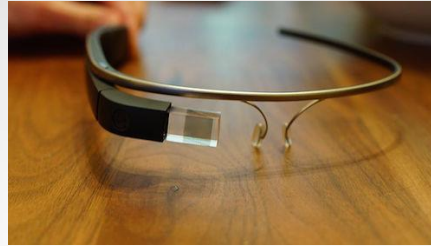


RA Mobile

- Smartphones, tout pour la RA
 - Camera – pour déterminer ce qui doit être vu
 - Donnée GPS– localisation
 - Compas – quelle direction on regarde
 - Accelerometre – orientation
 - Connection Internet – fournir des données utiles
- 58% des Français sont équipés d'un smartphone en 2015
- 90% des 18-24ans

Lunettes de RA

- Google



Glass



Glass 2 (A4R-GG1)

- VuViZ



M300

- Daqri



Smart Helmet

- Laster

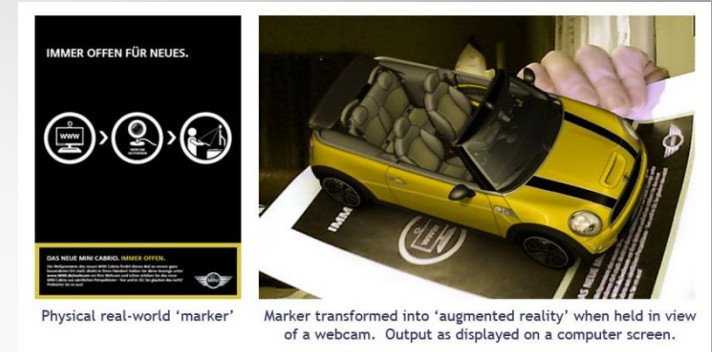


WAVE

Types de RA mobile

Marqueurs:

- Caméra pour détecter un marqueur dans le monde réel
- Calcul de sa position et orientation
- Augmente la réalité



GPS:

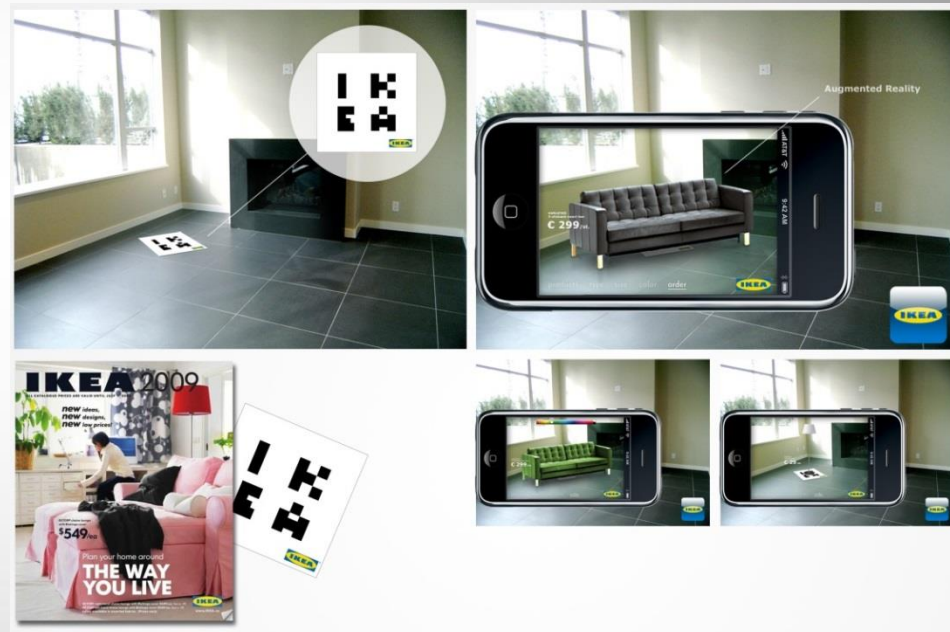
- GPS pour localiser son téléphone
- Recherche de Point d'intérêt proche de nous
- Mesure orientation (compas, accéléromètre)
- Augmente la réalité



Types de RA mobile

Utilisation de marqueurs spécifiques:

- Choix marqueur
- Détection
- Transformation 2D-3D
- Affichage 3D



Types de RA mobile

Utiliser les marqueurs image (objets naturels)

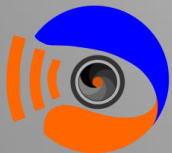
- Apprentissage
- Reconnaissance
- Transformation 2D-3D
- Affichage 3D



MOMO urban art on the Williamsburg Art & Design Building in Brooklyn.

Outils de RA

- Metaio (-> Apple)
- [Unity](#) et [Vuforia](#) (features)
- [Wikitude](#) ([features](#))
- Liste **SDK liste:** [Social Compare-AR-Sdk](#)
- **Lunettes RA:** [Social Compare-AR-lunettes](#)

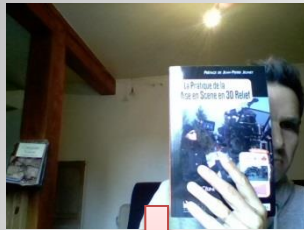


Vision par ordinateur et RA

- Camera -> vision par ordinateur
- Plusieurs technologies
 - Détection de marqueurs spécifiques: coins, primitives naturels, carrés, ronds
 - Mise en correspondance: primitives, images
 - Reconnaissance d'image: monument, façade, visage
 - Reconnaissance d'objets: tables, chaise....
 - Recalage caméra: calcule de la pose
 - Traitement d'image: contraste, segmentation
 - Mixer image et synthétique

Technologies nécessaires

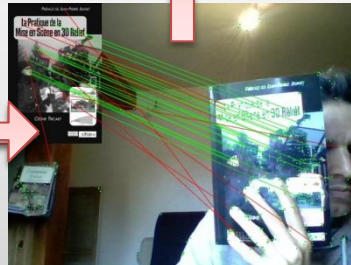
Références Acquisition vidéo



Détection coins et descripteurs



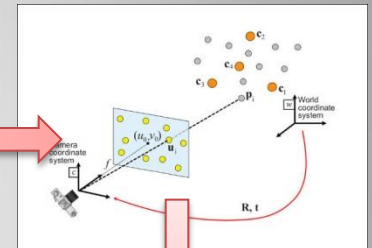
Matching



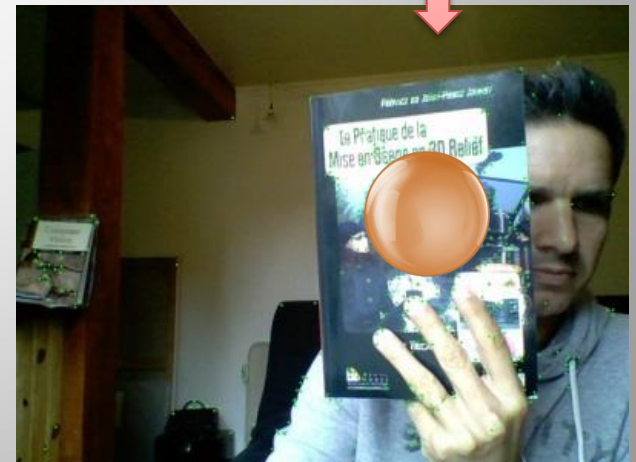
Localisation du Pattern



Calcul de la pose caméra



Affichage 3D



Detection et Appariement

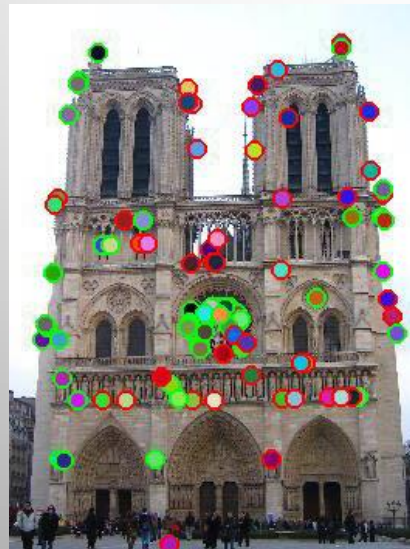
- Plusieurs méthodes existent pour décrire, détecter, et appairer les images
- Points, segments, régions, et droites des images peuvent être utilisées
- Quatre étapes sont nécessaires dans la détection et l'appariement des primitives
 - Détection de primitives
 - Description des primitives
 - Appariement des primitives
 - Tracking de primitives

Quelques termes

- **Marqueur** - utilisé pour spécifier où et quelle information ou contenu doit être placé (spécifiques ou image)
- **Primitives naturelles** – points/parties d'un objet visualisé
- **Detecteur** – utilisé pour rechercher dans les images les points spécifiques répétitifs
- **Descripteur** – utilisé pour caractériser les points ou région à partir de l'image. Ils sont utilisés dans la mise en correspondance
- **Canal** – association d'un marqueur à l'objet synthétique à afficher

Qu'est-ce qu'une primitive

- Une primitive c'est:
- Un élément spécifique de l'image
- Pixels/Point/coin unique de l'image
- Utilisé pour représenter/simplifier l'information contenue dans l'image



Points



Qu'est-ce qu'une primitive

Ca peut être aussi



Segments

Régions

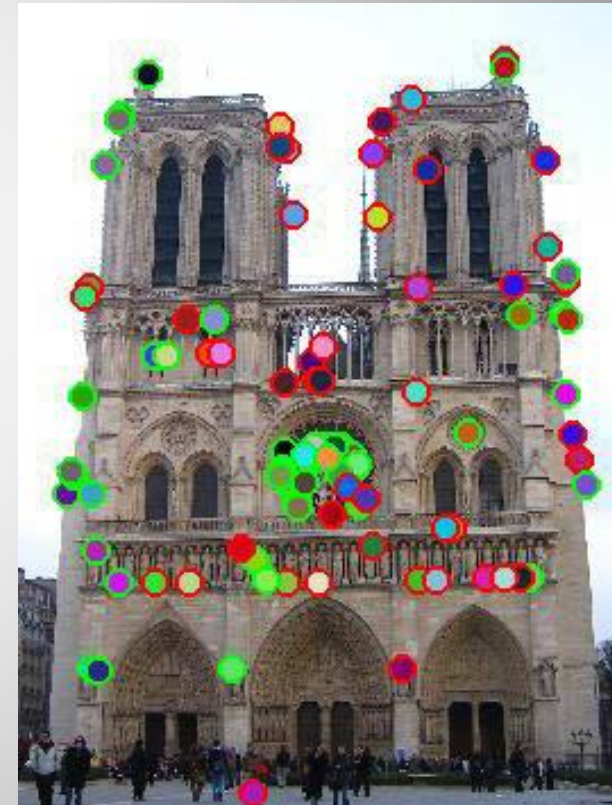


Contours



Détecteur de primitive

- Il va extraire/sélectionner les primitives de l'images
- Critères de qualité:
 - Caractérisables: distinctif, particularité, reconnaissable, précision
 - Répétabilité et invariance: échelle, rotation, illumination, point de vue, bruit



Détecteur de primitive

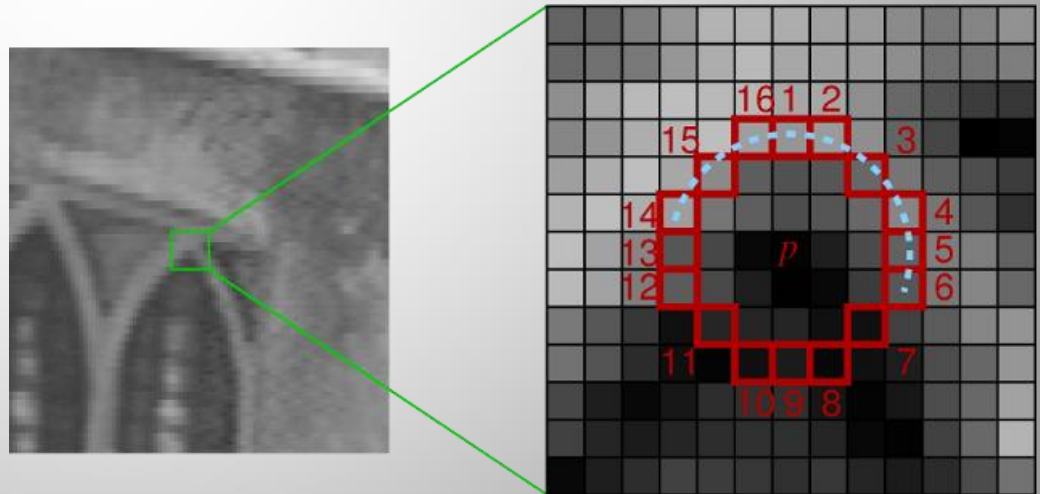


Détection de coins

FAST: Features from Accelerated Segment Test

<http://www.edwardrosten.com/work/fast.html>

- Cercle Bresenham 16 pixels autour du point analysé
- On détecte un coin en p si
l'intensité de N pixels
est $>$ ou $<$ de $X\%$ à I_p
- Rapide et robuste



Descripteur de points

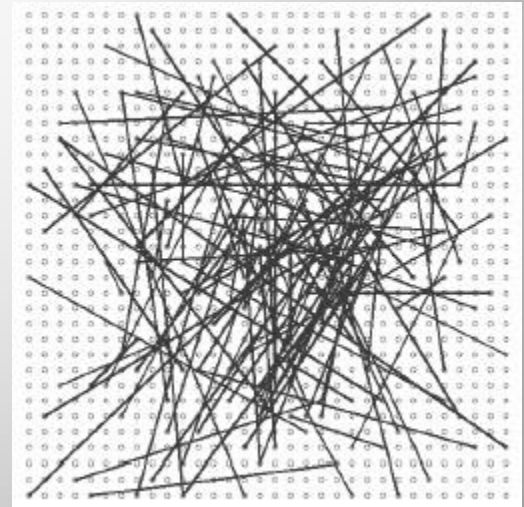
- Description du point à partir de l'image (locale)
- Utilisé pour l'appariement
 - Stockage des descriptions des marqueurs image
 - Comparer avec les primitives de l'image courante
- Critères de qualité:
 - Discriminant
 - Invariant : échelle, rotation, illumination, point de vue, bruit
 - Rapide et empreinte mémoire faible

Descripteur de points

BRIEF : Binary robust independent elementary features

<http://cvlab.epfl.ch/research/detect/brief>

- Vecteur de N paires de points sur un patch
- Comparaison pour chaque paire
 - Si $I_1 < I_2$ alors $c=1$
 - Sinon $c=0$
- Descripteur=100101001...
- Rapide et robuste

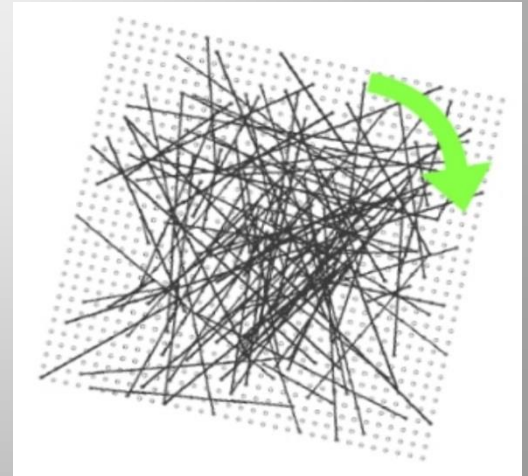
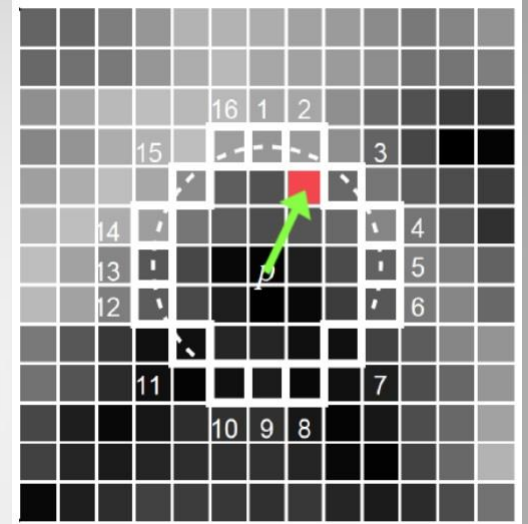


Descripteur de points

ORB (Oriented FAST and Rotated BRIEF)

http://docs.opencv.org/.../py_feature2d/py_orb/py_orb.html

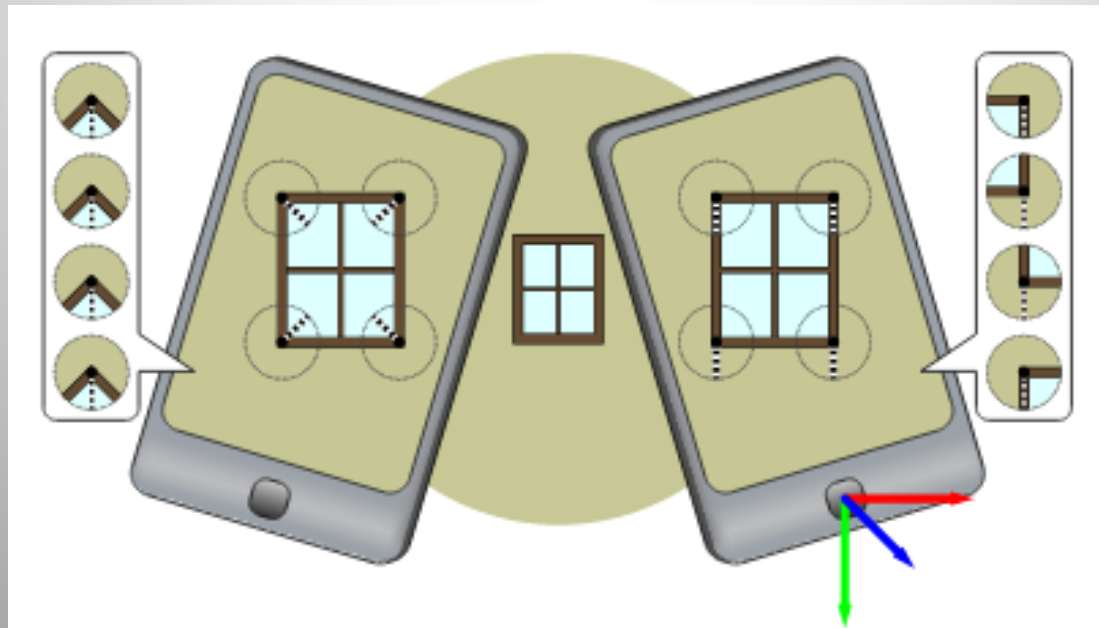
- Prise en compte rotation pour robustesse
- Direction=pixel avec variation la plus forte
- Rotated BRIEF pour aligner les descripteurs lors du matching



Descripteur de points

Autre exemple: GAFD Gravity Aligned Feature Descriptors

- Utilisé par Metaio (Apple)
- Utilise les capteur inertiel pour avoir des descripteurs alignés avec la gravité



Reconnaissance par matching

Appariement des coins

- Brute force matching, on teste toutes les paires
- Similarité= Distance de Hamming (nombre de bits différents)

A = 1 0 1 1 0 0 1 0 0 1 0 0

B = 1 0 0 1 0 0 0 0 1 1 1 1

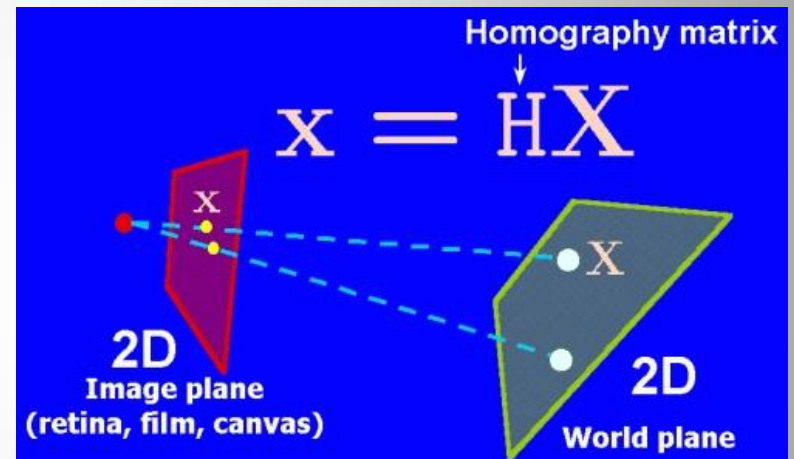
Distance de Hamming = 3

- Si on a un nombre de coins appariées suffisants, l'objet est retrouvé

Relocalisation 2D du pattern

Calcul de l'homographie du plan

- Système d'équation linéaire
- Estimation robuste (RANSAC)
- Filtrage des outliers
- Décomposition en VP

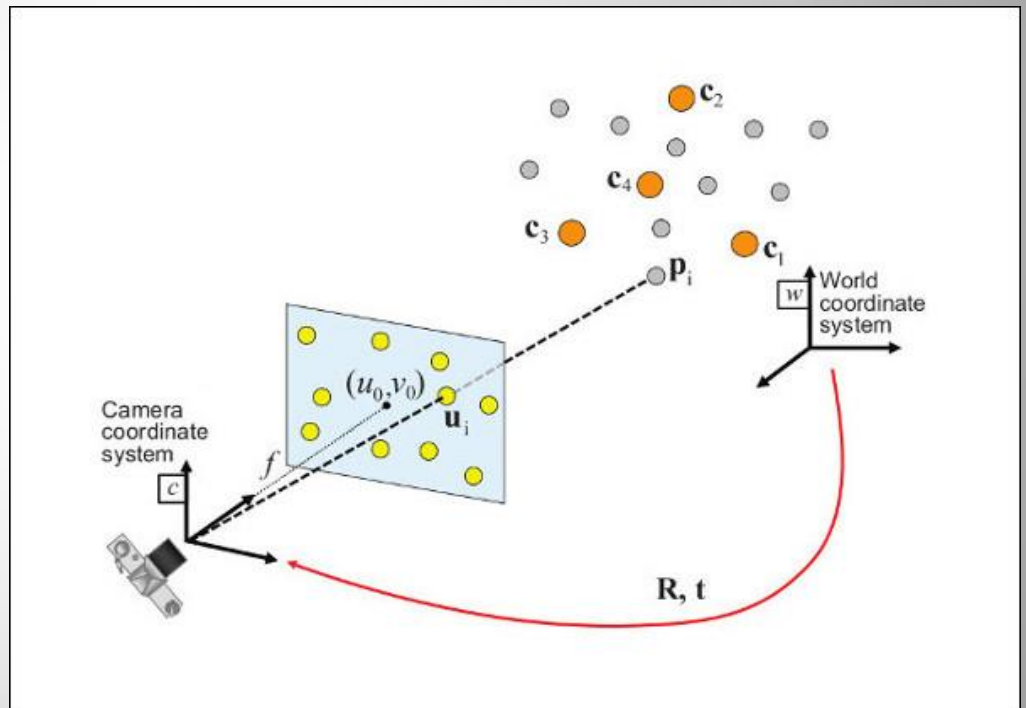


$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}}_{\text{homography } \mathbf{H}} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}.$$

Cacul de la Pose 3D

Calcul de la pose de la caméra par rapport à un objet 3D

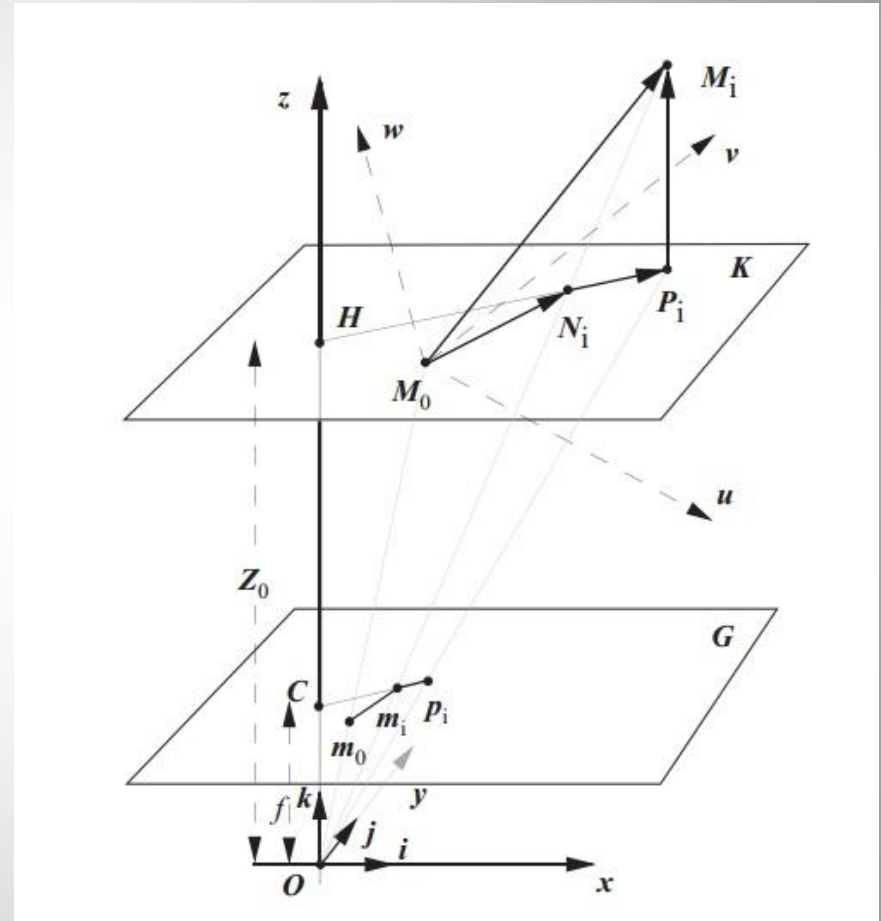
- General case:
 - 6DoF
 - Projection model
- Simplification
 - Calibration connue
 - Perspective-n-Point
 - Projection ortho
 - POSIT



POSIT

POSIT: **P**ose from **O**rthography and **S**caling with **I**terations

- Algorithme itératif pour résoudre PnP non coplanaires
- 4 points coplanaires:
Coplanar POSIT



More on Pose 3D

Calcul de la pose de la caméra par rapport à un objet 3D

- POSIT: [original publications](#), [3D pose estimation](#)
- [Real Time pose estimation](#) : OpenCV tutorial, C++
- [Eric Marchand](#): Article Complet Pose 3D AR
- [Caméra calibration](#): OpenCV tutorial, C++
- [posest](#): C++ opensource
- [Minimal problems](#) in Computer Vision: many links
- Moving camera = Kalman/SLAM


Objectif de ArtMobilis

Un parcours urbain en réalité augmentée

- Géolocalisation des points d'intérêts
- Tracking de la localisation des contenus augmentés
- Support mobile (android, IOS, tablettes)
- OpenSource: <https://github.com/artmobilis/>
- LabMobilis:
 - Implémentation orientée Web pour adaptabilité
 - Application HTML5, CSS3 et JavaScript

Navigateurs compatibles

- [CanIuse](#): 67% des navigateurs
- Compatible avec Firefox/chrome/AndroidBrowser/Edge

getUserMedia/Stream API  - WD

Global

9.66% + 57.74% = 67.4%

unprefixed:

1.03%

Method of accessing external device data (such as a webcam video stream). Formerly this was envisioned as the <device> element.

Current aligned		Usage relative		Show all							
IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android		
8			1 45					4.3			
9			1 46					4.4			
10		43	1 47			8.4		4.4.4			
11	13	44	1 48	9	1 34	9.2	8	1 47	1 47		
	14	45	1 49	9.1	1 35	9.3					
		46	1 50		1 36						
		47	1 51								

Librairies Javascript utilisées

- **Framework:**
 - Angularjs
 - Ionic
 - Cordova
- **AR Image demo:**
 - Js-ArUco: <https://github.com/jcmellado/js-aruco>
 - three.js : <https://github.com/mrdoob/three.js>
 - jsfeat : <https://github.com/inspirit/jsfeat>

Exercices

- **Chrome:**

- Bloque getUserMedia pour les fichiers locaux
- Lancer avec --disable-web-security pour du debug
- Navigator.getUserMedia plus supporté -> MediaDevices.getUserMedia()
- Il faudrait utiliser adapter.js
- Attention: exemples pas mis à jour -> utilisez Firefox

- **Firefox:**

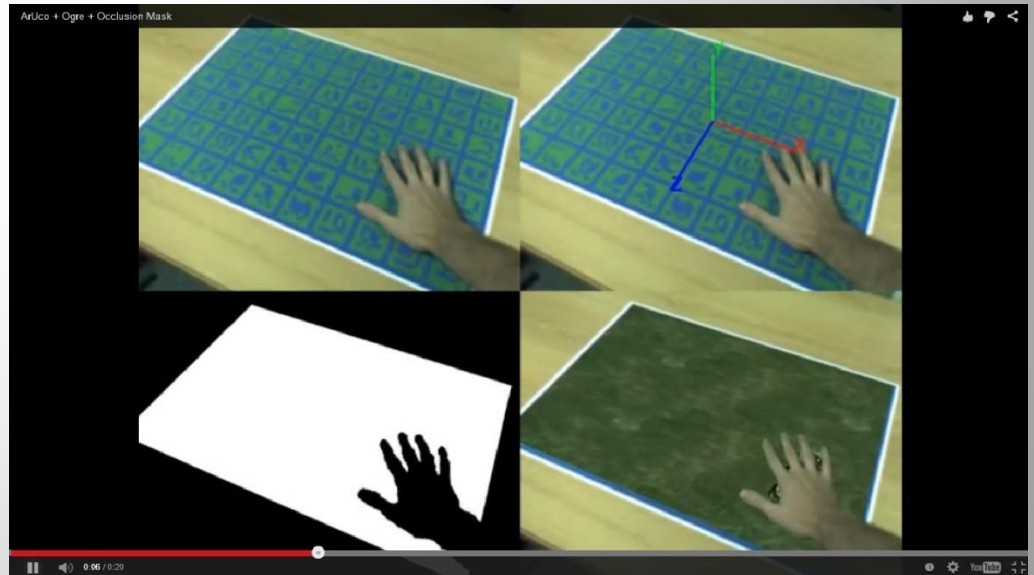
- Version 40 et +: pb avec les vieilles cartes graphique blacklistées
- Installer version 31 pour du debug (marche sur mon laptop)

Exercices

- <https://github.com/vestri/CoursAR1-exercices>
- **Forkez le repository sous github**
- **Téléchargez le Code**

Aruco

- [ArUco](#) est une librairie minimale pour la Réalité Augmentée à base de marqueurs (basée OpenCV)
- [js-aruco](#) est le portage en JavaScript d'ArUco
 - Image processing
 - Contours
 - Detection marqueurs
 - Calcul de pose



ArUco Exercice

- Code dans ImageProcessingAruco
- Faire **tourner la sphère**

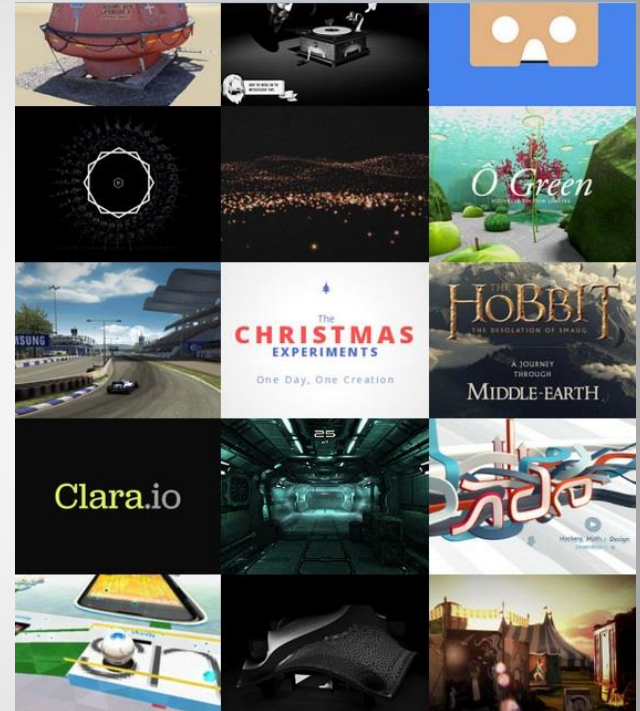
ArUco Solution

```
stat.start("Posit");  
pose = posit.pose(corners);  
stat.stop("Posit");  
  
stat.start("Update");  
updateObject(model, pose.bestRotation, pose.bestTranslation);  
stat.stop("Update");  
  
step += 0.025;  
model.rotation.z -= step;  
}  
};
```

Three.js

[Three.js](#) simplifie l'utilisation de WebGL

- Renderers: WebGL, <canvas>, <svg>...
- Scenes, Cameras, Geometry, Lights, Materials, Shaders, Particles, Animation, Math Utilities
- Loaders: Json compatible Blender, 3D max, Wavefront OBJ



Three.js Exercice

- Code dans testhtml/ImageProcessingThreeJS
- Combiner **l'image et la 3D**
- Indice: Faire des Layers

Three.js Solution

```
<title>JSFeat-ThreeJS - Exercice: Canvas 2D 3D</title>
<h1>JSFeat-ThreeJS - Exercice: Canvas 2D 3D</h1>

<video id="webcam" style="display:none;" height="480" width="640"></video>
<div style="width:640px;height:480px;margin: 10px auto;">
  <canvas id="canvas2d" width="640" height="480" class="overlapcanvas"></canvas>
  <canvas id="canvas3d" width="640" height="480" class="overlapcanvas"></canvas>
  <div id="log"></div>
</div>
```

```
.overlapcanvas{
  width:640;
  height:480;
  position: absolute;
  float: left;
  top: 0px;
  left: 0px;
}
```

```
function createRenderersScene() {
  renderer3d = new THREE.WebGLRenderer({ canvas: canvas3D, alpha: true });
  renderer3d.setClearColor(0xffffff, 0);
  renderer3d.setSize(canvas2d.width, canvas2d.height);

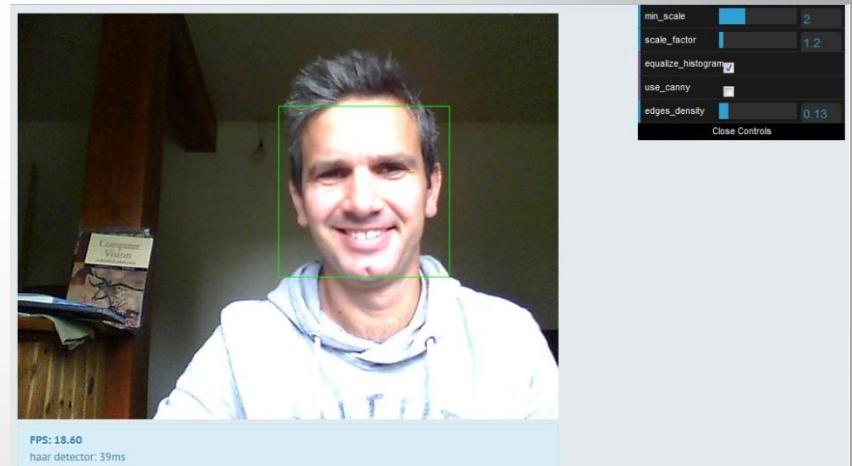
  // for 3d projection
  scene = new THREE.Scene();
  camera = new THREE.PerspectiveCamera(40, canvas2d.width / canvas2d.height, 1, 1000);
  scene.add(camera);

  model = createModel();
  scene.add(model);

  camera.position.z = 5;
};
```

Jsfeat

- Jsfeat: JavaScript Computer Vision library
- Algorithmes modernes de vision pour Html5
 - Custom data structures
 - Basic image processing
 - Linear Algebra and Multiview
 - Feature 2D
 - Optical flow
 - Object detection



Jsfeat Exercice

- Code dans testhtml/ImageProcessingJSfeat
 - Mettre l'image en **noir et blanc**
-
- **Exercice d'évaluation**

Jsfeat Solution

```
// process each acquired image
function tick() {
    compatibility.requestAnimationFrame(tick);
    stat.new_frame();
    if (video.readyState === video.HAVE_ENOUGH_DATA) {
        ctx.drawImage(video, 0, 0, 640, 480);
        var imageData = ctx.getImageData(0, 0, 640, 480);

        // greyscale conversion
        stat.start("grayscale");
        // I should put my code here
        jsfeat.imgproc.grayscale(imageData.data, 640, 480, img_u8);
        stat.stop("grayscale");

        // render result back to canvas (Warning: format is RGBA)
        stat.start("rewrite");
        // I should put my code here
        var data_u32 = new Uint32Array(imageData.data.buffer);
        var alpha = (0xff << 24); // opacity=1
        var i = img_u8.cols * img_u8.rows, pix = 0;
        while (--i >= 0) {
            pix = img_u8.data[i];
            // write 4 channels: RGBA with GreyGreyGreyAlpha
            data_u32[i] = alpha | (pix << 16) | (pix << 8) | pix;
        }
        stat.stop("rewrite");

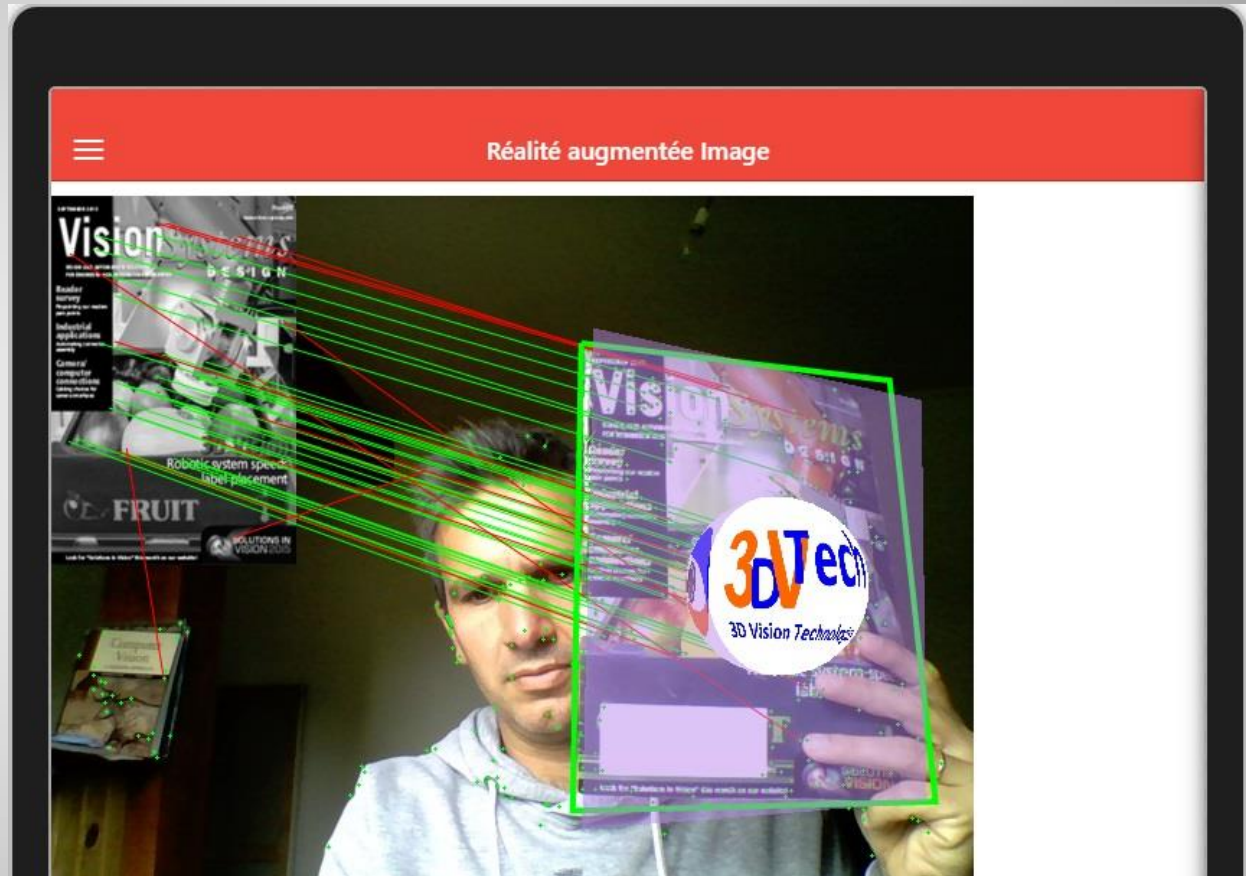
        ctx.putImageData(imageData, 0, 0);
        log.innerHTML = stat.log();
    }
}
```

Autres librairies intéressantes

- Computer Vision:
 - tracking.js: <https://github.com/eduardolundgren/tracking.js>
 - js-objectdetect: <https://github.com/mtschirs/js-objectdetect>
 - Convnetjs: <https://github.com/karpathy/convnetjs>
 - sgdSlam: <https://github.com/odestcj/sgd-slam>
- 3D:
 - Babylon.js: <https://github.com/BabylonJS/Babylon.js>

Prototype développé

- Demo
- Code



Futur de la RA

Display:

- MagicLeap
- Lentilles de contact pour RA
- Hololens

Techno:

- Unity et Vuforia
- Wikitude

Applications:

- Pour l'instant -> communication
- Shazam video? Google goggles?

Plus d'infos

- **Réalité Augmentée:**
 - RAPRO: <http://www.augmented-reality.fr/>
 - SDK liste: [Social Compare-AR-Sdk](#)
 - Lunettes RA: [Social Compare-AR-lunettes](#)
- **Projet**
 - <https://github.com/artmobilis/>
 - vestri@3DVTech.com