IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

INTRODUCTION À PSI

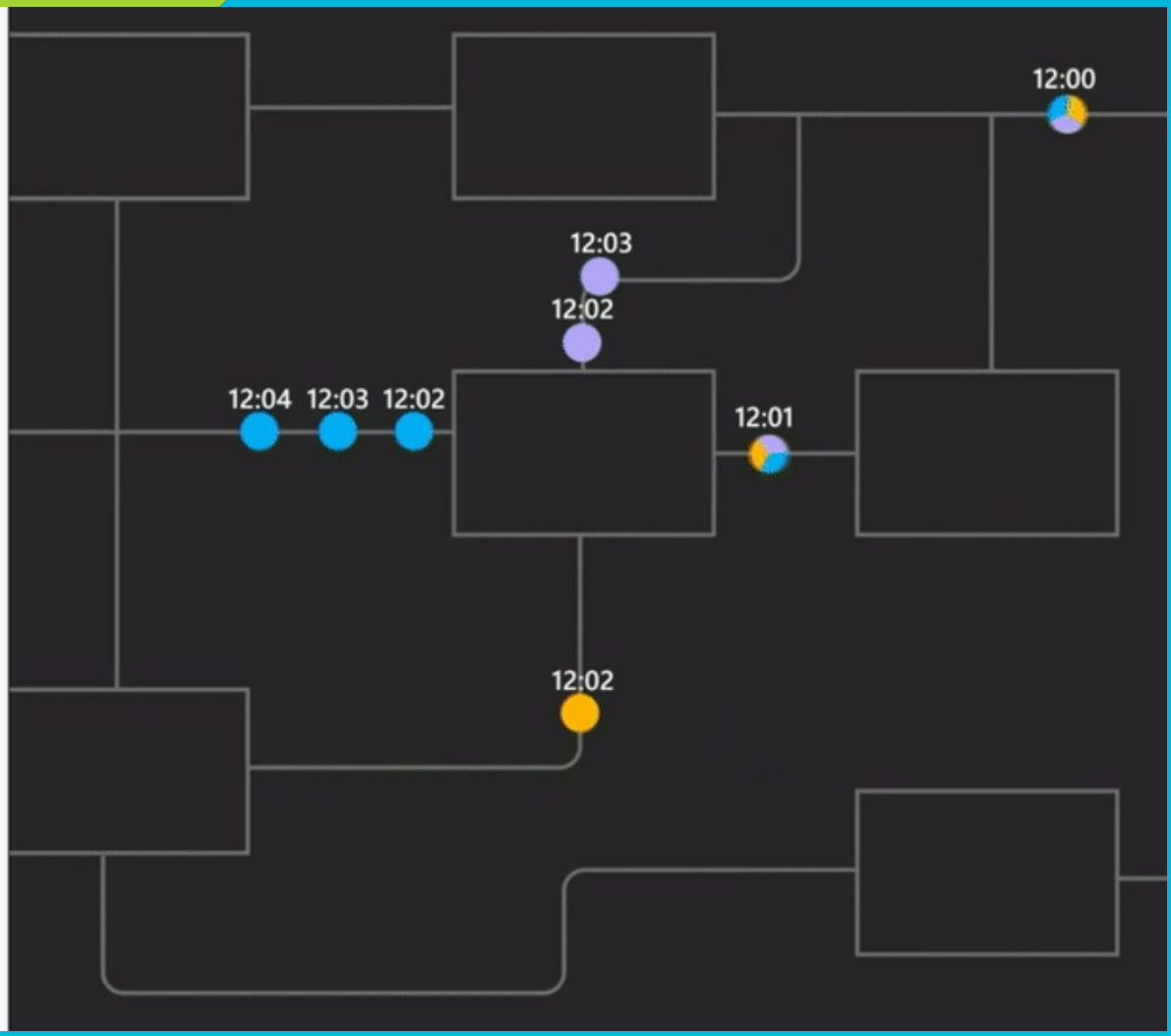
Cédric Dumas & Aurélien Milliat

IHM 2024 : La collaboration en présence mixte dans les espaces interactifs

SOMMAIRE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

INFRASTRUCTURE FOR SYNCHRONIZATION



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

https://camo.githubusercontent.com/1f2c3afc838e595a3d8738c25877b9cb108fd5576eef23e543e4a2985ab1ae11/68747470733a2f2f7777772e6d6963726f736f66742e636f6d2f656e2d75732f72657365617263682f75706c6f6164732f70726f642f323031382f30312f073692d476966322d3936302d436f727265637465642e676966

PSI - Présentation

https://youtu.be/Gfs808q8S0I

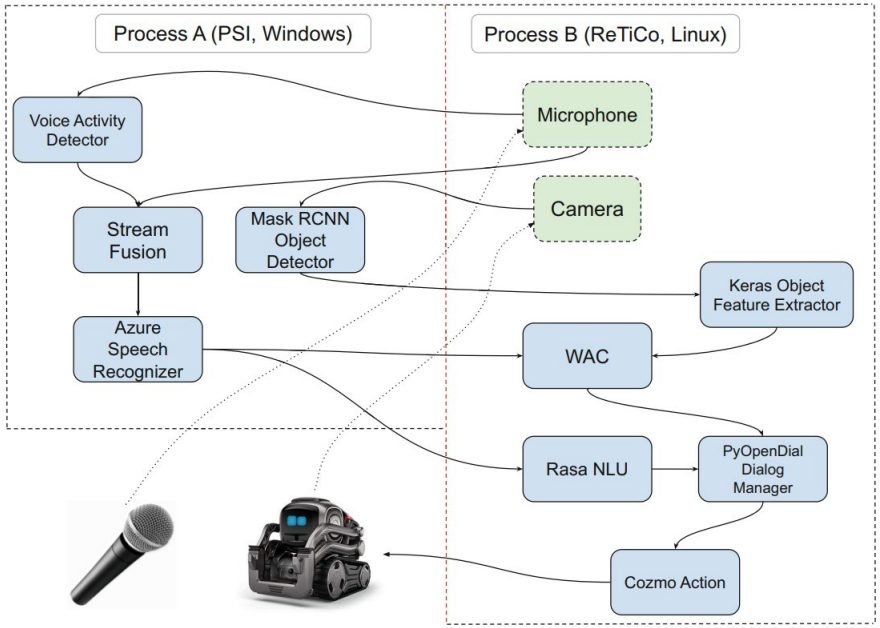
\psi

PSI - Présentation

- Plateforme de synchronisation de flux de données.
- Principe de composants / pipeline en open source (MIT License).
- Possibilité d'une utilisation réseau / remote (Hololens).
- Capacité d'enregistrement des flux.
- Possibilité de temps réel (ou de simuler via les enregistrements).
- Orienté Machine Learning
- Application modulaire de visualisation: Psi Studio.

PSI - Cas d'usages

- Human Robot Interaction (HRI) :

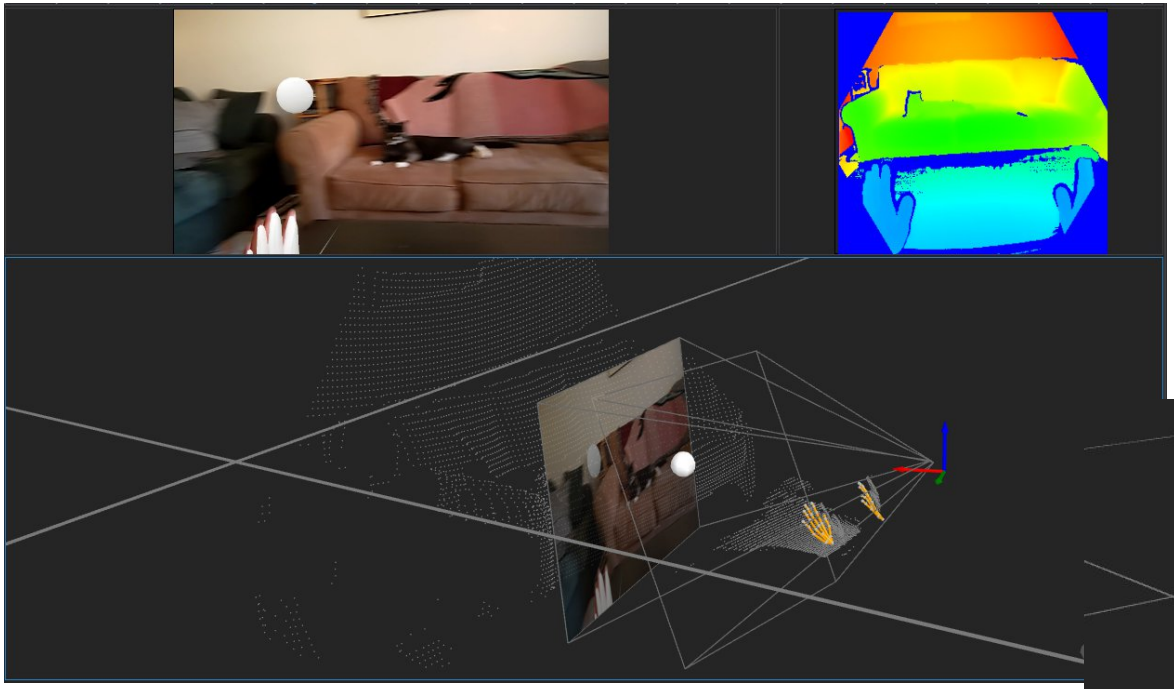


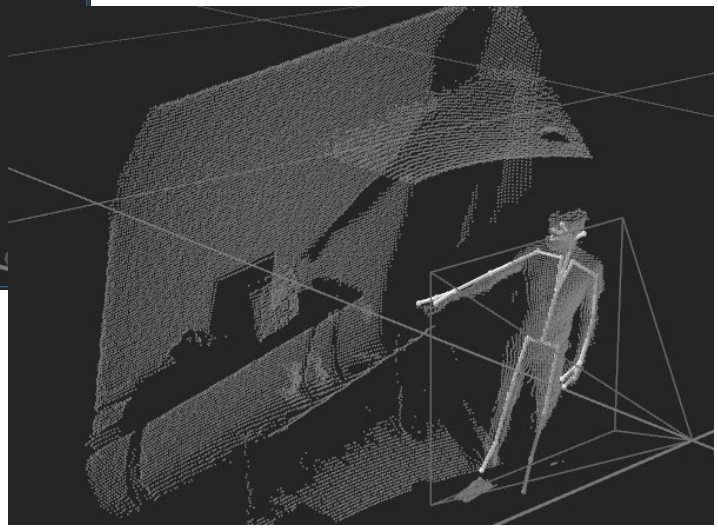
https://link.springer.com/content/pdf/10.1007/978-3-031-06018-2.pdf

- exemple : HARMONI (Composing HARMONI: An Open-source Tool for Human and Robot Modular OpeN Interaction)

PSI - Cas d'usages

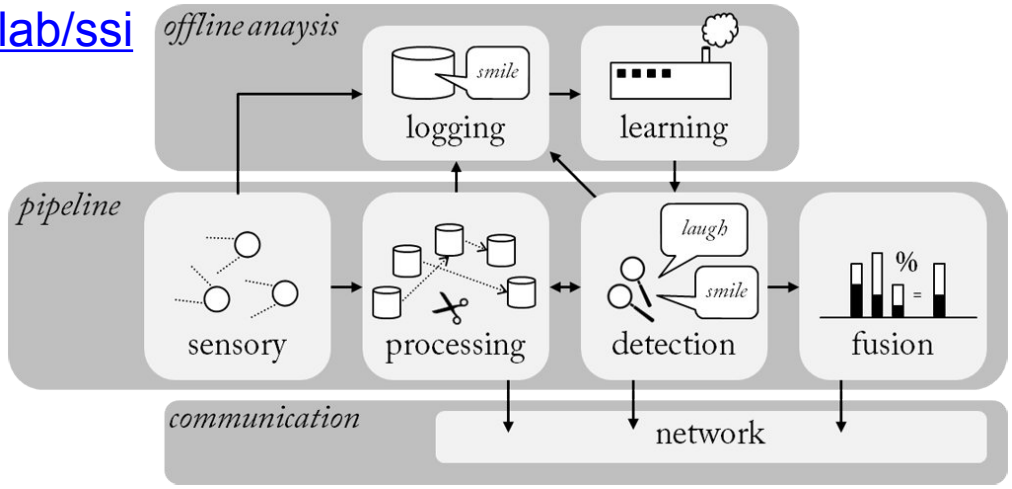
- Mixed Reality & Collaborative Interaction:





PSI - Autres solutions

- Social Signal Interpretation (SSI) Framework:
  - Human Centered Artifical Intelligence Lab of the Augsburg University
  - Beaucoup de composants implémentés : capteurs et libraires de process.
  - Principalement en C++
  - Latest Release v1.0.4 17 Apr 2018
  - https://github.com/hcmlab/ssi



\psi

PSI - Autres solutions

- OpenSense:
  - Intelligent Human Perception Lab, USC-ICT
  - Surcouche graphique à PSI.
  - Composants pour OpenFace, OpenPose, OpenSmile & Whisper.
  - Possibilité de constituer un pipeline de traitement en 'no code'.
  - Pas mal de bugs.
  - IHM pas très 'user friendly'.
  - https://github.com/intelligent-human-perception-laboratory/OpenSense/

PSI - Autres solutions

- MediaPipe:
  - Développé par Google
  - Orienté traitement d'image -> segmentation d'image.
  - Mais framework générique.
  - Disponible sur plusieurs plateformes
  - https://github.com/google/mediapipe

	Android	iOS	C++	Python	JS	Coral
Face Detection	✓	✓	✓	✓	✓	✓
Face Mesh	✓	✓	✓	✓	✓	
Iris	✓	✓	✓			
Hands	✓	✓	✓	✓	✓	
Pose	✓	✓	✓	✓	✓	
Holistic	✓	✓	✓	✓	✓	
Selfie Segmentation	✓	✓	✓	✓	✓	
Hair Segmentation	✓		✓			
Object Detection	✓	✓	✓			✓
Box Tracking	✓	✓	✓			
Instant Motion Tracking	✓					
Objectron	✓		✓	✓	✓	
KNIFT	✓					
AutoFlip			✓			
MediaSequence			✓			
YouTube 8M			✓			

PSI - Bilan

PROS:

- C#
- Relativement simple
- Store/Remote
- Microsoft
- Lien avec le cloud Azure mais ouvert
- Composants existants
- Compatible Unix
- Linq

CONS:

- Performance/ressource (bench?)
- In Dev (certaines limitations)
- Pas de plugin Unity/Unreal

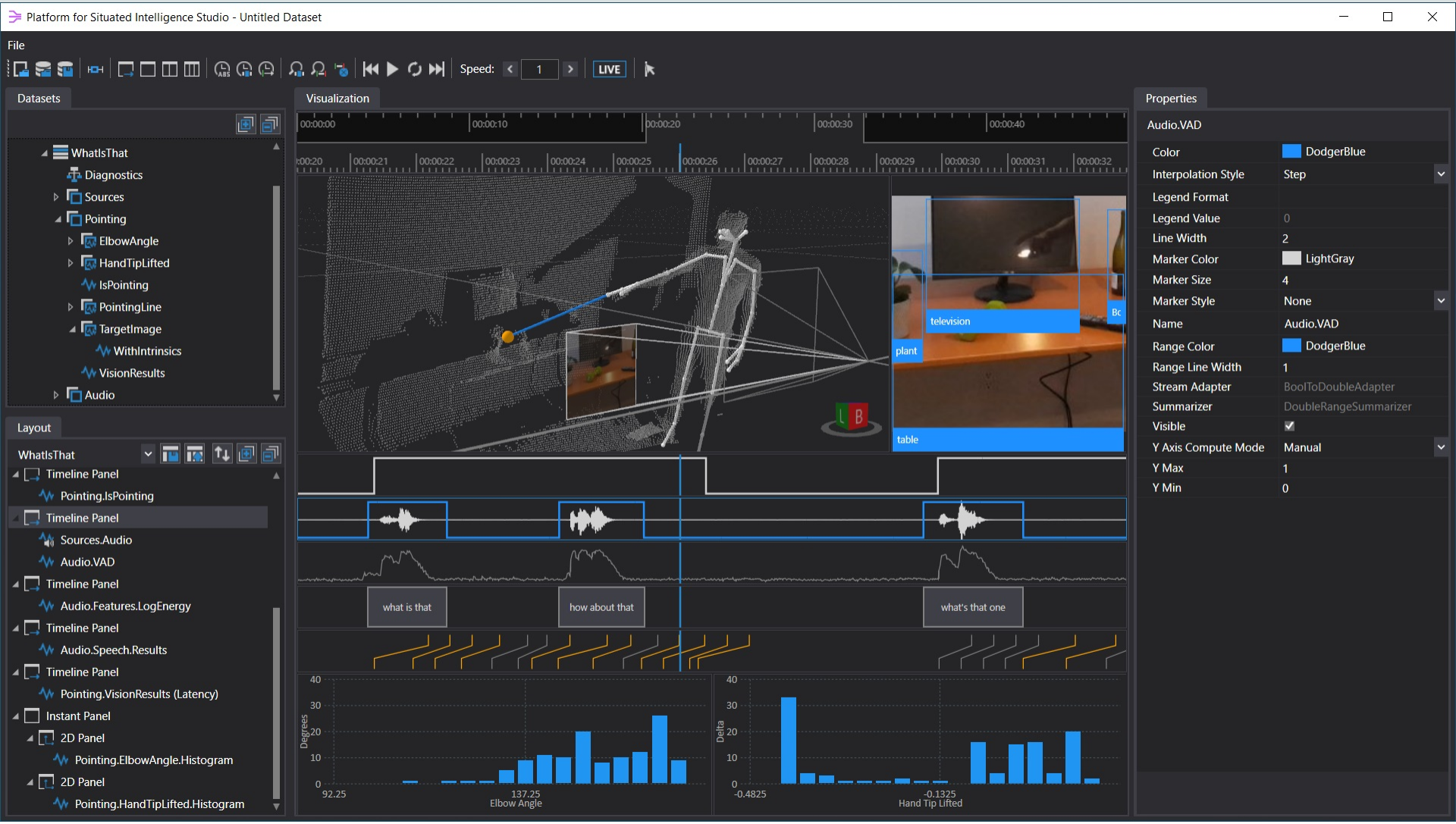
- Tarif des services Azure: https://azure.microsoft.com/fr-fr/pricing/details/cognitive-services/ (11-2023)

Produit	Fonctionnalités	Tarif
Compréhension du langage (LUIS) Standard jusqu'à 50 requêtes par seconde	Demandes de texte	**1,418 €** toutes les 1 000 transactions* de prédiction
	Demandes vocales	**5,199 €** toutes les 1 000 transactions
Langage Standard jusqu'à 1 000 requêtes par seconde et 1 000 requêtes par minute	Analyse des sentiments (et exploration des opinions) Extraction des expressions clés Détection de la langue Reconnaissance d'entité nommée, y compris les informations d'identification personnelle (non disponible en conteneur)	0millions-0.5millions d'enregistrements de texte - **0,9453 €** par 1 000 enregistrements texte 0.5millions-2.5millions d'enregistrements de texte - **0,7090 €** par 1 000 enregistrements texte 2.5millions-10.0millions d'enregistrements de texte - **0,2836 €** par 1 000 enregistrements texte Enregistrements de texte 10.0M+ - **0,2364 €** par 1 000 enregistrements texte
	Analyse de texte pour le secteur de la santé	0millions-0.005millions d'enregistrements de texte - Inclus 0.005millions-0.5millions d'enregistrements de texte - **23,6318 €** par 1 000 enregistrements texte 0.5millions-2.5millions d'enregistrements de texte - **14,1791 €** par 1 000 enregistrements texte Enregistrements de texte 2.5M+ - **9,4527 €** par 1 000 enregistrements texte
	Réponses aux questions*	Gratuit
Traducteur S1	Traduction de texte Personnalisation Détection de la langue Dictionnaire bilingue Translittération	**9,453 €** par million de caractères (Paiement à l'utilisation)

PSI - Bilan

Produit	Fonctionnalités	Tarif
Vision par ordinateur S1 30 transactions par seconde pour les opérations de lecture 15 transactions par seconde pour les opérations non-lecture	Balise Visage GetThumbnail Couleur Type d'image GetAreaOfInterest Détection de personnes* Cultures intelligentes	0-1 million de transactions - **0,946 €** toutes les 1 000 transactions 1-10 million de transactions - **0,615 €** toutes les 1 000 transactions 10-100 million de transactions - **0,568 €** toutes les 1 000 transactions +100 million de transactions - **0,379 €** toutes les 1 000 transactions
	OCR Adulte Célébrité Repère Détecter, Objets Marque	0-1 million de transactions - **0,946 €** toutes les 1 000 transactions 1-10 million de transactions - **0,615 €** toutes les 1 000 transactions 10-100 million de transactions - **0,568 €** toutes les 1 000 transactions +100 million de transactions - **0,379 €** toutes les 1 000 transactions
	Décrire* Lecture Sous-titre*	0-1 million de transactions - **1,418 €** toutes les 1 000 transactions +1 million de transactions - **0,568 €** toutes les 1 000 transactions
Content Moderator S0 jusqu'à 10 requêtes par seconde	Modérées, Révision	0-1 million de transactions - **0,946 €** toutes les 1 000 transactions 1-5 million de transactions - **0,709 €** toutes les 1 000 transactions 5-10 million de transactions - **0,568 €** toutes les 1 000 transactions +10 million de transactions - **0,379 €** toutes les 1 000 transactions
API Face Standard jusqu'à 10 requêtes par seconde	Détection de visage Vérification de visage Identification de visage Regroupement de visages Recherche de visages semblables	0-1 million de transactions - **0,946 €** toutes les 1 000 transactions 1-5 million de transactions - **0,757 €** toutes les 1 000 transactions 5-100 million de transactions - **0,568 €** toutes les 1 000 transactions +100 million de transactions - **0,379 €** toutes les 1 000 transactions
	Face Storage	**0,010 €** par 1 000 visages par mois

PSI - PSI Studio



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

\psi

https://github.com/microsoft/psi/wiki/PsiStudio.jpg

- Visualisation des données (temps réel ou enregistrement).
- Création de dataset à partir des stores.
- Annotation !
- Analyse de données via API :
  - Exécution d'algorithme spécifique
  - Processus d'extraction pour Machine Learning
  - Statistiques sur les données
- Possibilité de développer des modules de visualisation pour l'application.

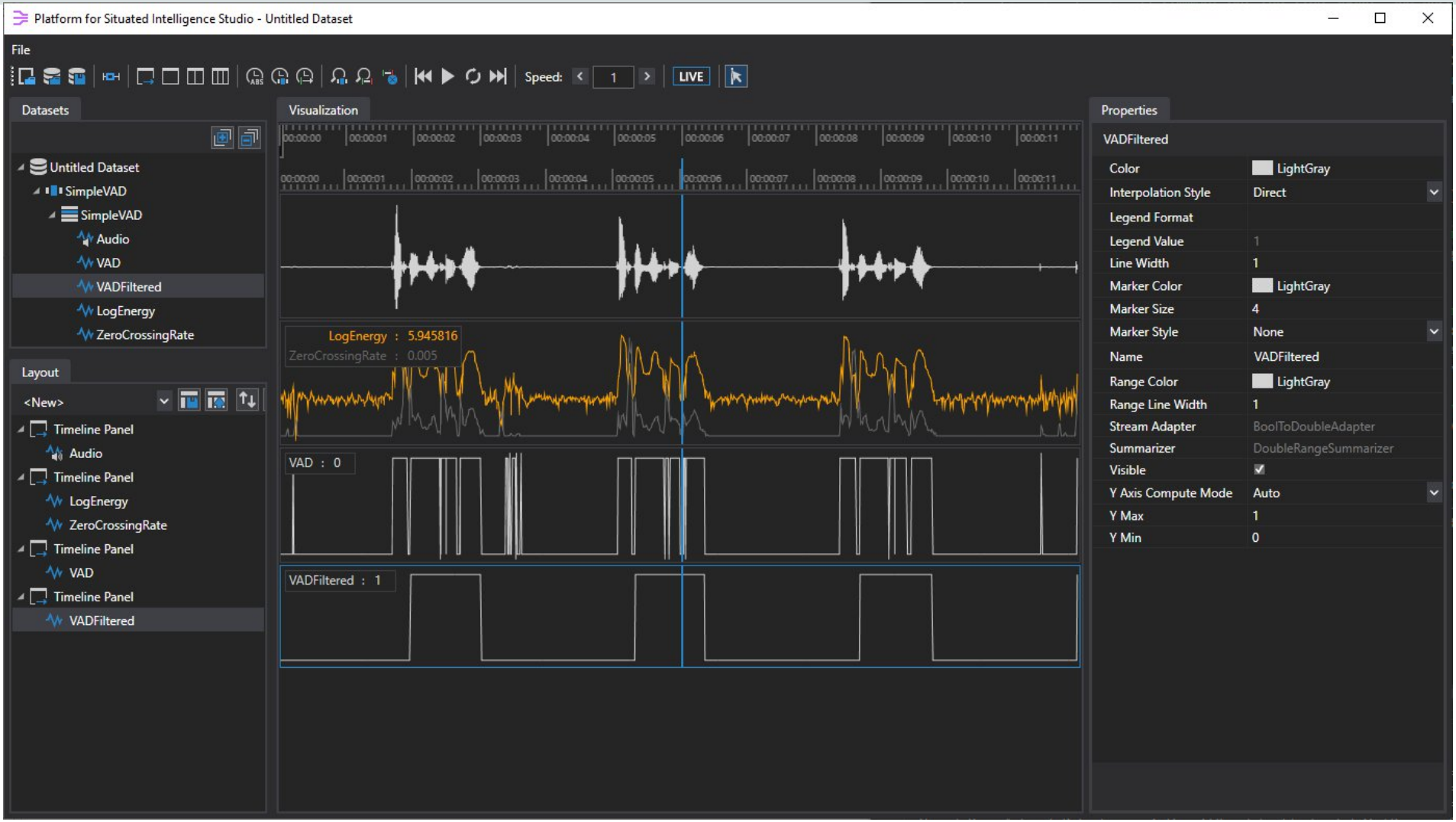
PSI - Programmation


```
using (var p = Pipeline.Create())
{
    // Create the microphone, acoustic feature extractor, and connect them
    //var microphone = new AudioCapture(p, WaveFormat.Create16kHz1Channel16BitPcm());
    var inputStore = PsiStore.Open(p, "SimpleVAD", Path.Combine(Directory.GetCurrentDirectory(), "Stores", "SimpleVAD.0000"));
    var microphone = inputStore.OpenStream<AudioBuffer>("Audio");
    var acousticFeaturesExtractor = new AcousticFeaturesExtractor(p);
    microphone.PipeTo(acousticFeaturesExtractor);

    // Display the log energy
    acousticFeaturesExtractor.LogEnergy
        .Sample(TimeSpan.FromSeconds(0.2))
        .Do(logEnergy => Console.WriteLine($"LogEnergy = {logEnergy}"));

    // Create a voice-activity stream by thresholding the log energy
    var vad = acousticFeaturesExtractor.LogEnergy
        .Select(l => l > 7);

    // Create filtered signal by aggregating over historical buffers
    var vadWithHistory = acousticFeaturesExtractor.LogEnergy
        .Window(RelativeTimeInterval.Future(TimeSpan.FromMilliseconds(300)))
        .Aggregate(false, (previous, buffer) => (!previous && buffer.All(v => v > 7)) || (previous && !buffer.All(v => v < 7)));

    // Write the microphone output, VAD streams, and some acoustic features to the store
    var store = PsiStore.Create(p, "SimpleVAD", Path.Combine(Directory.GetCurrentDirectory(), "Stores"));
    microphone.Write("Audio", store);
    vad.Write("VAD", store);
    vadWithHistory.Write("VADFiltered", store);
    acousticFeaturesExtractor.LogEnergy.Write("LogEnergy", store);
    acousticFeaturesExtractor.ZeroCrossingRate.Write("ZeroCrossingRate", store);

    p.RunAsync();
    Console.ReadKey();
}
```


\psi

https://github.com/Microsoft/psi-samples/tree/main/Samples/SimpleVoiceActivityDetector

PSI - Programmation



\psi

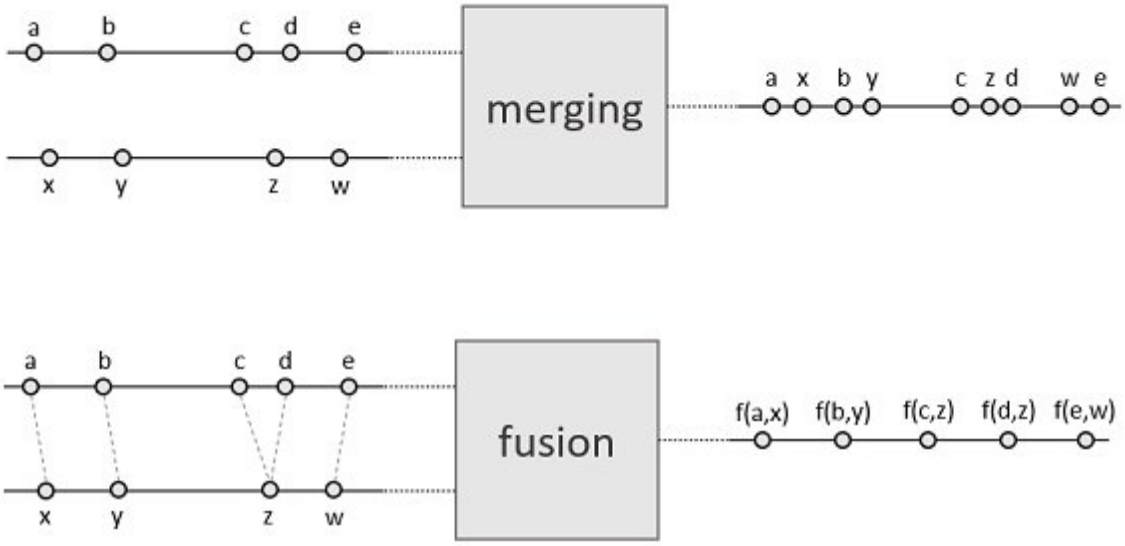
https://github.com/microsoft/psi-samples/blob/main/Samples/SimpleVoiceActivityDetector/Images/PsiStudio-4.png

PSI - Programmation


```
namespace ComponentNameSpace
{
    using System;
    using Microsoft.Psi;
    using Microsoft.Psi.Components;

    public class Component : Subpipeline
    {
        // Connector for the reciever
        protected Connector<TYPE_IN> InConnector;

        // Reciever that encapsulates the input.
        public Receiver<TYPE_IN> In => InConnector.In;

        // Emitter taht encapsulates the output.
        public Emitter<TYPE_OUT> Out { get; private set; }

        // ToDo : Add more/modify stuff if needed

        // Constructor
        public Component(Pipeline pipeline, string name = "Component", DeliveryPolicy policy = null)
            : base(parent, name, policy)
        {
            // Connections creations
            InConnector = pipeline.CreateConnector<TYPE_IN>(nameof(InConnector));
            Out = pipeline.CreateEmitter<TYPE_OUT>(this, nameof(Out));

            // Registering method that will be called for every input
            InConnector.Out.Do(Process);
        }

        private void Process(TYPE_IN data, Envelope envelope)
        {
            //TO DO : process
            lock (this)
            {
                //If out:
                Out.Post(TYPE_OUT, DateTime.Now);
            }
        }
    }
}
```

\psi

- Deux classes de réceptions *complexes*



https://github.com/microsoft/psi/wiki/Stream-Fusion-and-Merging

- Deux opérateurs de réceptions *Merging*:
  - Zip: concaténation basé sur le temps dans les données.
  - Merge: concaténation rapide incluant les messages originaux (temps).
- Deux opérateurs de réceptions *Fusion*:
  - Fuse: opérateur générique
  - Join: opérateur spécialisé pour les interpolations 'reproductible'.
  - Interpolations:
    - Reproducible.Exact()
    - Reproducible.Nearest<T>(...)
    - Reproducible.First<T>(...)
    - Reproducible.Last<T>(...)
    - Reproducible.Linear(...)
    - Available.Exact()
    - Available.Nearest<T>(...)
    - Available.First<T>(...)
    - Available.Last<T>(...)
    - AdjacentValuesInterpolator(...)

- Il y a d'autres opérateurs :
  - Ceux de Linq (Select, First, Where, Flip, Count...)
  - Ceux concernant les temps (Delay, Parallel, Latency..)
  - Ceux concernant les streams (BridgeTo, PipeTo...)
  - Pair: concaténation des entrées basée sur le temps des messages du premier stream.
  - ...

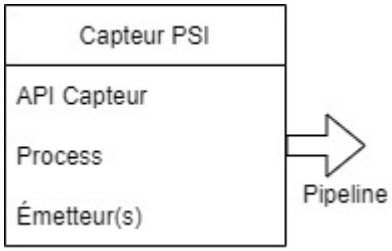
- Liaison entre les composants :

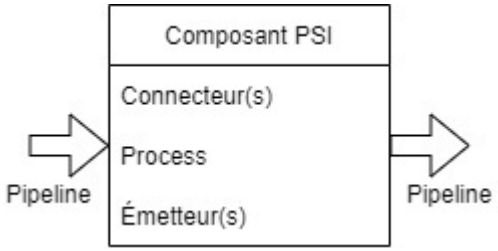
```
var mic = new AudioCapture(p);
var resampler = new AudioResampler(p);
// link : micro -> sampler
mic.PipeTo(resampler);

var voiceActivityDetector = new SystemVoiceActivityDetector(p);
// link : sampler -> voiceDetector
resampler.PipeTo(voiceActivityDetector);

var azureSpeechReco = new AzureSpeechRecognizer(p);
// link : sampler + voiceDetector -> SpeeachToText Azure
resampler.Join(voiceActivityDetector).PipeTo(azureSpeechReco);

// Process : complete results only
var finalResults = azureSpeechReco.Out.Where(result => result.IsFinal);
NLPBot nlp = new NLPBot(p, botDescription);
// link : SpeeachToText Azure -> NLP
finalResults.PipeTo(nlp);

var audioplayer = new AudioPlayer(p);
// link : NLP -> TextToSpeech
nlp.Voice.PipeTo(audioplayer);
```

- Exécution du pipeline:
  - Run(): exécute de façon bloquante le pipeline.
  - RunAsync(): non bloquant mais nécessite l'appel à *Dispose().*

```
// RunAsync the pipeline in non-blocking mode.
p.RunAsync(ReplayDescriptor.ReplayAllRealTime);
// Wainting for an out key
Console.WriteLine("Press any key to stop the application.");
Console.ReadLine();
// Stop correctly the pipeline.
p.Dispose();
```

- Comme tout framework de haut niveau, il est facile de faire n'importe quoi.
- La courbe d'apprentissage se situe surtout sur l'optimisation des applications
- Deux 'types':
  - Les capteurs: émission
  - Les composants: réception & émission

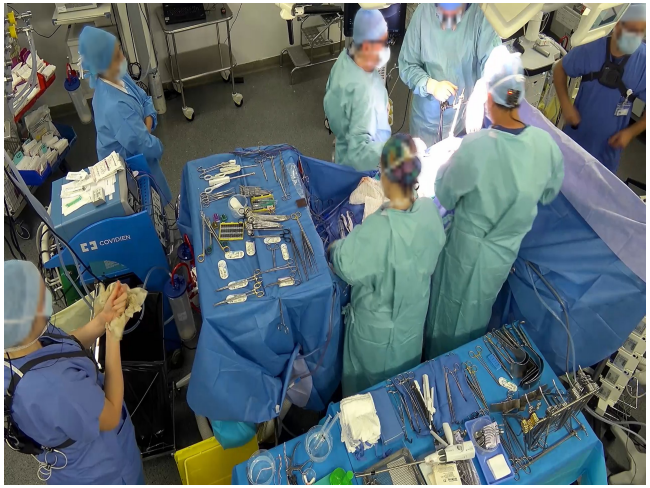




- Toujours de l'émission -> Store !
- De préférence des tâches unitaires.

PSI - Exemples

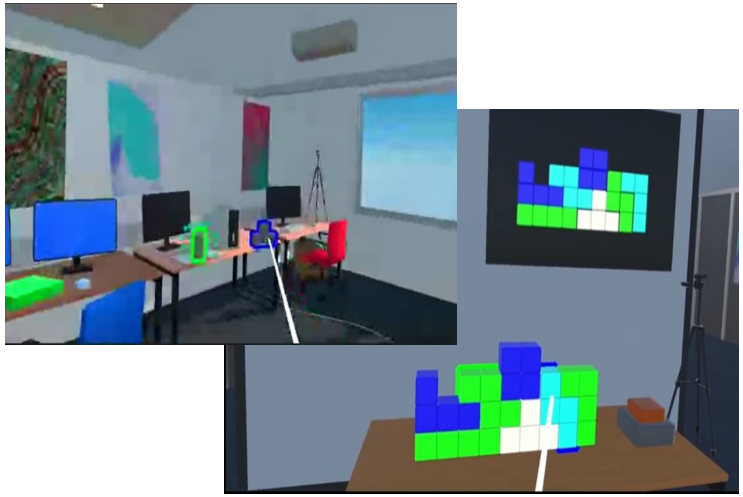
Surgery



OR nº3 Centre Hépato-Biliaire Paul Brousse

A. Allemang--Trivalle
ICMI 2023

VR task



A. Léchappé
ICMI 2023

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

MAX-PLANCK-GESELLSCHAFT

PSI - Exemple **Surgery**

- **Critical, complex and stressful situation**
- **Team effort:** Surgeons, anesthesiologists, and nurses collaborate actively

© [ihorvsn] /Adobe Stock

Processes to integrate information about the situation of each team member

Use multimodal sensors to capture these processes (the situation) and reduce errors

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

bop A bloc opératoire augmenté

Institut Mines-Télécom université PARIS-SACLAY
ASSISTANCE PUBLIQUE HÔPITAUX DE PARIS
INRIA CNAM

MAX-PLANCK-GESELLSCHAFT

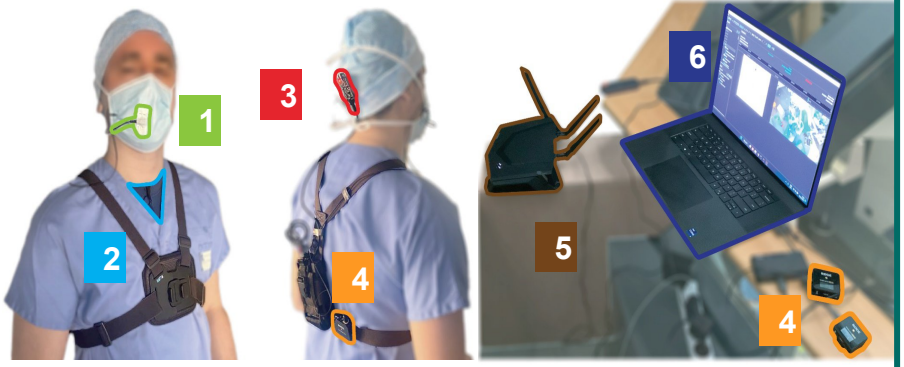
PSI - Exemple **Surgery**

- Collect precise behavioral data
- Enables fine control over the collaborative situation
- Deploy variety of environment augmentation in response to users' actions and collaborative states

PSI - Exemple Surgery

Machine-readable

Wearable sensors



1 **Lavalier microphone**

2 **Physiological garment**

3 **Tracking tag**

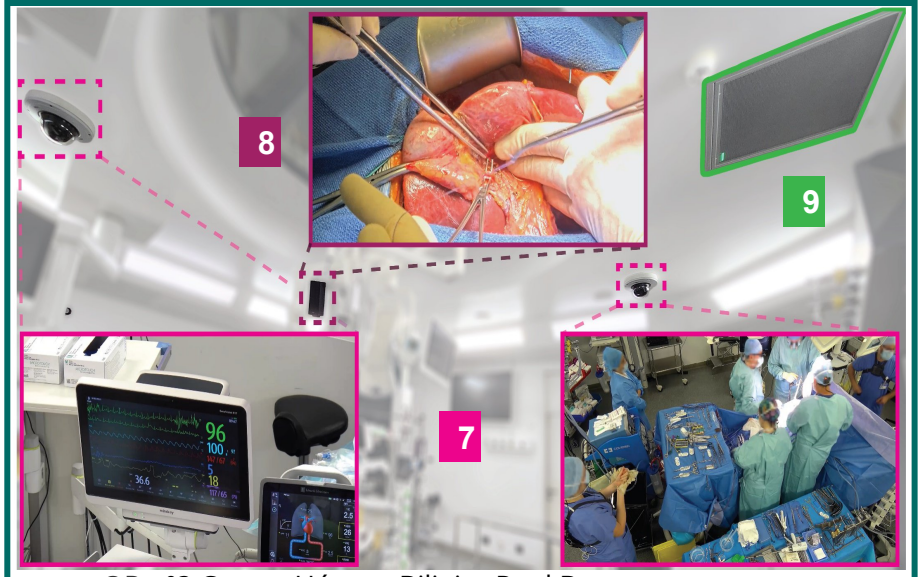
4 **Audio transmission system**

5 **Wi-Fi router**

6 **Computer** (Microsoft PSI)

Human-readable

Operating room sensors



OR n°3 Centre Hépato-Biliaire Paul Brousse

7 **PTZ cameras**

8 **Surgical camera**

9 **Ceiling microphone**

IMT Atlantique
Bretagne-Pays de la Loire
Ecole Mines-Télécom

bopA bloc opératoire augmenté

Institut Mines-Télécom université PARIS-SACLAY
ASSISTANCE PUBLIQUE HÔPITAUX DE PARIS
INRIA CNAM

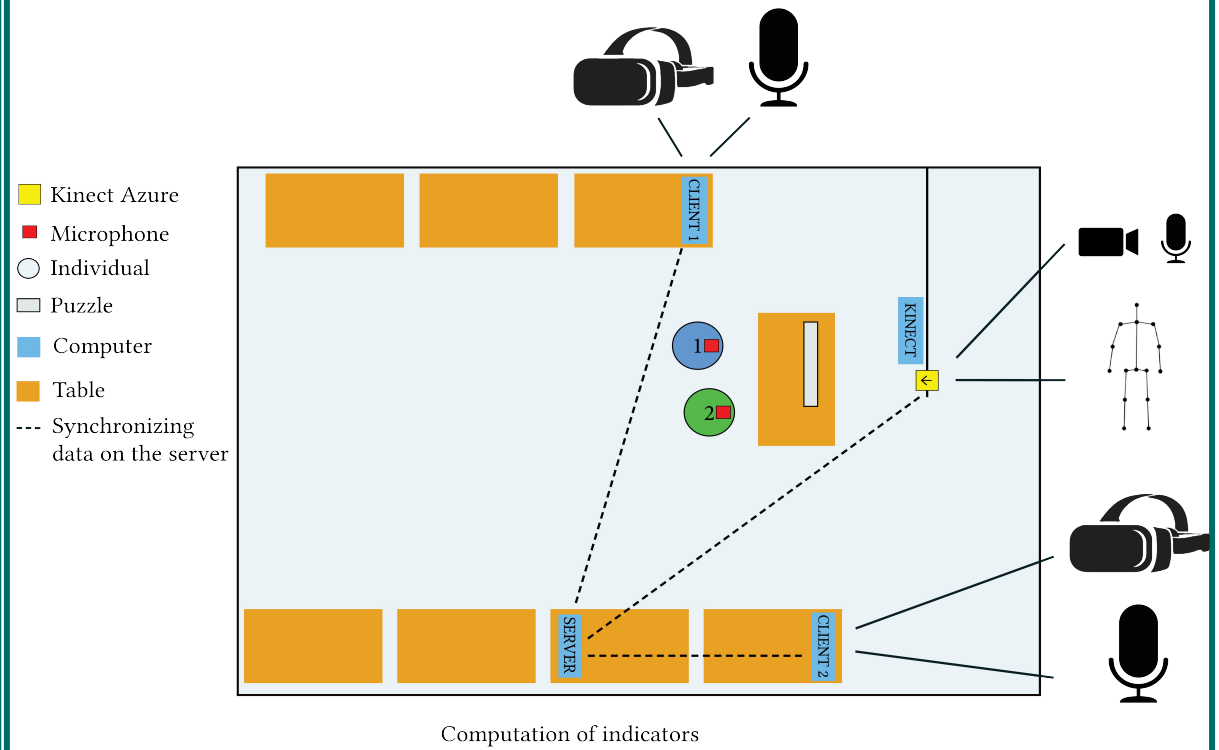
MAX-PLANCK-GESELLSCHAFT

PSI - Exemple VR task

Collaborative activity

- Dyads
- Puzzle-solving task
  - Exploration/search task
  - Solving task
- Individuals must coordinate to perform tasks

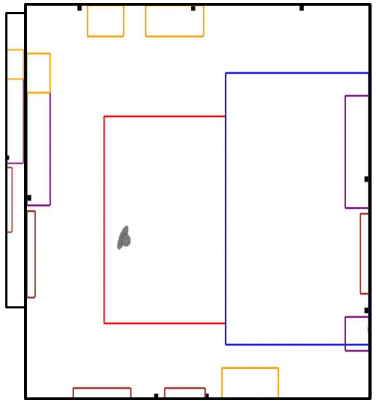
Sensors



PSI - Exemple VR task

Metrics

1. Speech
2. Physiological data
3. Localization
4. Postures
5. Gaze



User

- Speech flow
- Stress and mental workload
- Proximity to areas of interest
- Gaze on peers

Team

- Overlapping Speech
- Turn-taking
- Dominance
- Proximity to team members
- Physiological synchrony
- Joint visual attention
- Mutual gaze

Questionnaires

Get to know the people:

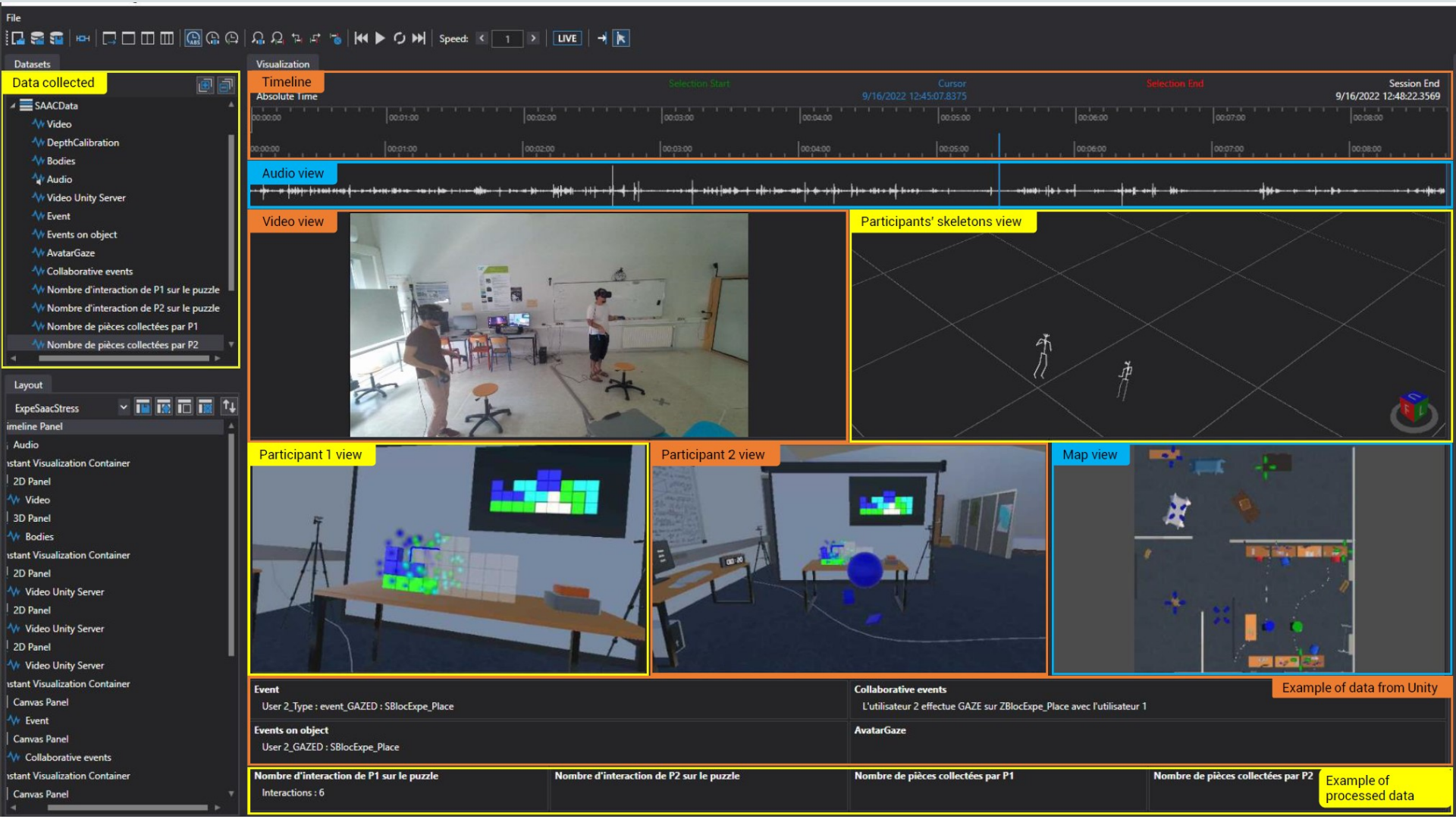
- **Background, Conditions, Relationships**

Context

Annotations made by expert:

- **Task steps and phases**
- **Errors**
- **Adverse events** (surgery)
- **Collaboration quality**

PSI - Exemple **VR task**



UN PEU DE PRATIQUE...

```
// Create a pipeline
var p = Pipeline.Create();

// Create a timer component that produces a message every second
var timer = Timers.Timer(p, TimeSpan.FromSeconds(1));

// For each message created by the timer, print "Hello world!"
// along with the message's originating time.
timer.Do((t, e) =>
{
    Console.WriteLine($"{e.OriginatingTime}: Hello world!");
});

// Run the pipeline, but don't block here
p.RunAsync();

// Wait for the user to hit a key before closing the pipeline
Console.ReadKey();

// Close the pipeline
p.Dispose();
```

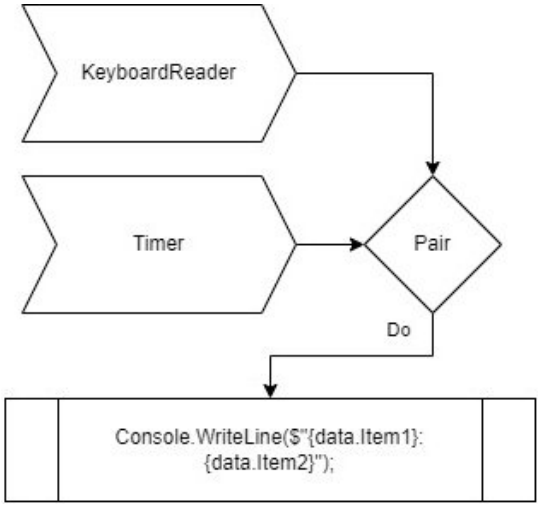
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

TP - Hello World

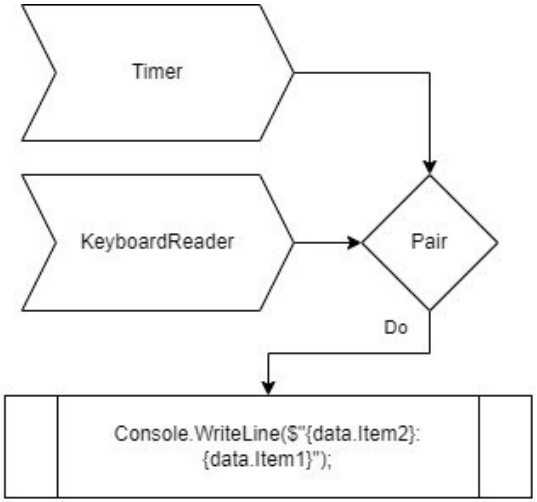
- Cloner https://github.com/microsoft/psi-samples et exécuter l'exemple HelloWorld.
- Intégrer le composant *KeyboardReader* fournit (lien) au projet.
- Coder les fonctionnalités suivantes:
  - Afficher la sortie du *KeyboardReader* avec timestamp du message (on n'utilise pas le timer du projet pour le moment) :

    Console.WriteLine($"Message @ {envelope.OriginatingTime}: {data}")

  - Coder des fonctions qui prennent en entrée la sortie du *KeyboardReader* et la sortie du timer de l'exemple (documentation).
    - Une fonction doit afficher & répéter le dernier message du *KeyboardReader* à chaque *tick* du timer.

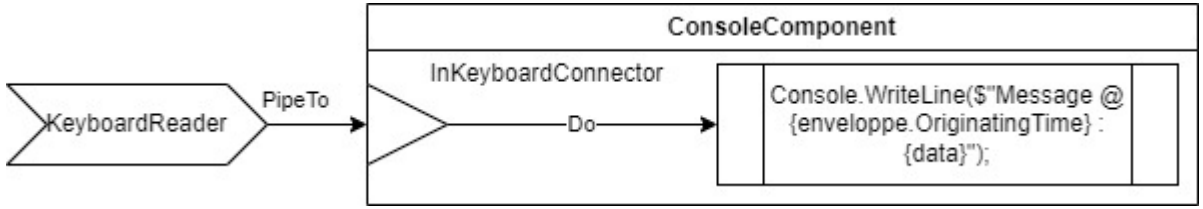


- L'autre doit afficher le dernier message du *KeyboardReader* avec le dernier *tick* du timer.

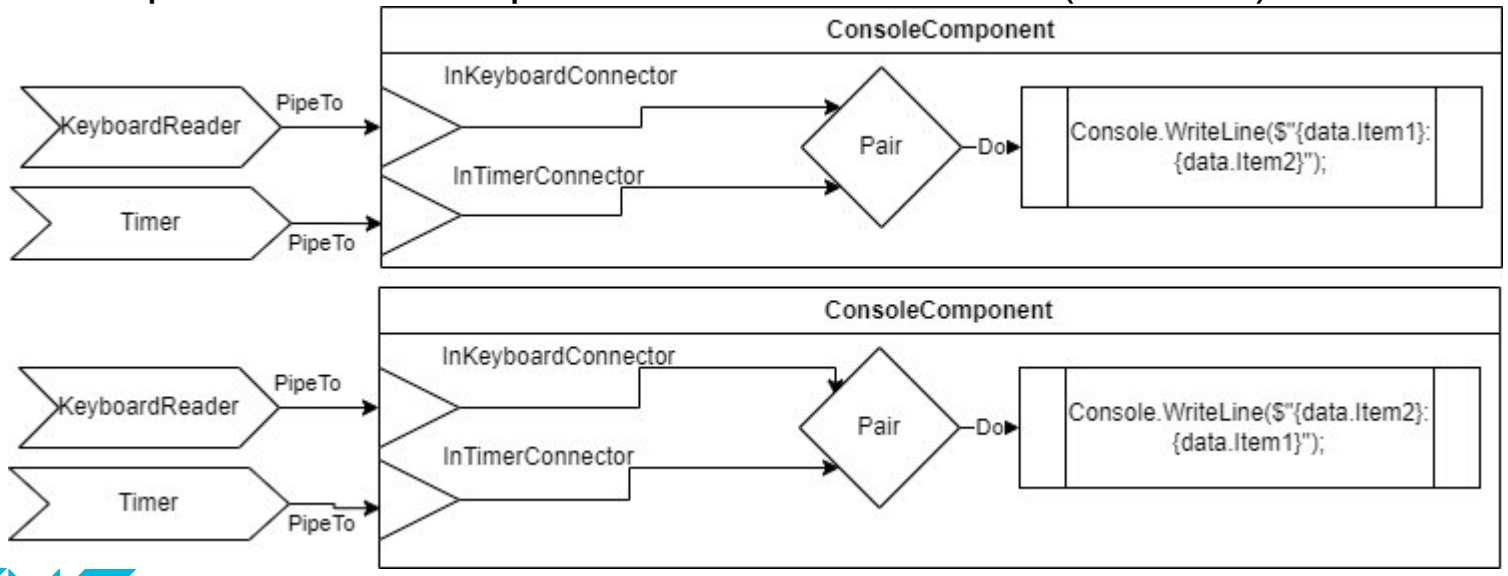


TP - Hello World

- Reprendre les fonctions précédemment réalisées sous formes de composants. *Un composant type est disponible dans les aides, documentation.*



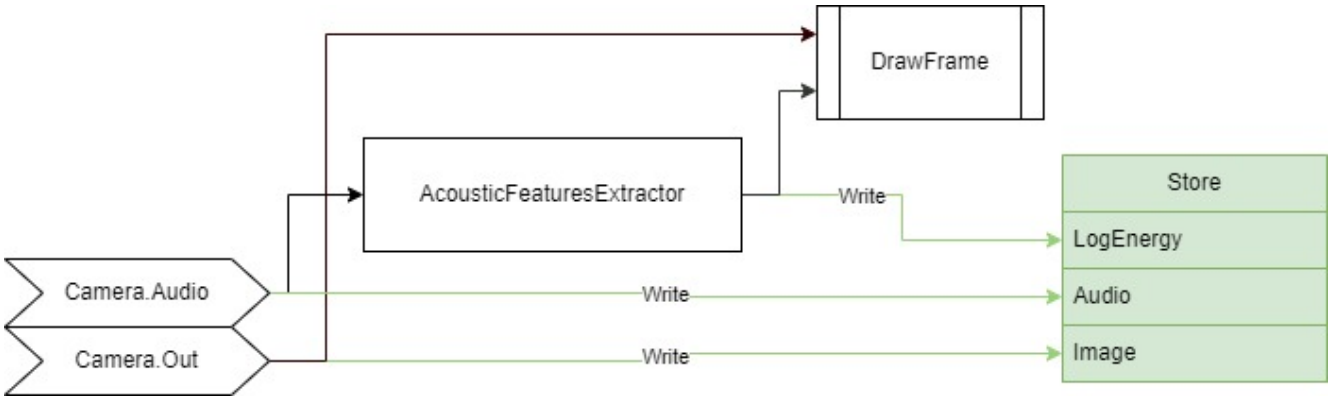
- Reprendre un exemple de fusion de données (au choix):



\psi

TP – WebcamWithAudio & Store

- Définir comme projet principal WebcamWithAudioSample
  - Exécuter et vérifier le bon fonctionnement.
  - Enregistrer les flux dans un(des) store(s) (documentation).



  - Optimiser l'enregistrement avec la fonction EncodeJpeg() :
    - Vérifiez que vous avez le package nuget dans le projet *Microsoft.Psi.Imaging.Windows*
  - Utiliser Psi Studio pour afficher en temps réel les informations stockées (documentation).

TP - Kinect Azure !

- Définir comme projet principal AzureKinectSample
- Intégrer l'enregistrement via un store de l'image couleur, de la profondeur et des squelettes.
- Afin d'afficher les squelettes, il est nécessaire d'ajouter le composant de visualisation de squelette (AzureKinect.Visualization.Windows.x64):

  https://github.com/microsoft/psi/wiki/3rd-Party-Visualizers

  - Celui-ci est disponible dans le dossier de PsiStudio fournit.
- Ouvrir dans Psi Studio et tester le mode *live*.

TP - Kinect Azure !

- De notre expérience, il est plus efficace d'effectuer un enregistrement par store, surtout quand il y a beaucoup de données (en fréquence et en taille).
- Effectuer un enregistrement où :
  - Deux personnes agissent en collaboration, on inclura des moments où les mains sont proches.
  - Un groupe de personne se rassemble autour d'un objet.
  - À votre imagination !

TP - Offline Processing

- Télécharger le projet : https://github.com/AuMilliat/AzureKinectPostures
- Développer une application :
  - Détection de groupes.
  - Détecter des positions d'intérêt : pointage, bras croisés, accroupie...
  - Détecter des actions de collaboration, comme le passage d'objet.
  - Lier ces détections à un enregistrement des images (couleur et profondeur).

- Les packages Unity sont disponible (Unity et WebRTC) :
  - https://github.com/SaacPSI/saac
- Les projets \psi fonctionnant avec Unity sont disponibles (PSIUnity & WebRTC) :
  - https://github.com/AuMilliat/PsiTPCorrection