Mobile control interface for modular robots

Luc Girod
Semester project

Supervisors:
Stéphane Bonardi
Massimo Vespignani
Prof. Ijspeert
We want the current simulation on an augmented reality (AR)
Augmented reality techniques

• On camera sensors.
• Pattern detection.
• Simultaneous localization and mapping aka SLAM.
Why SLAM-like algorithm

- No need of extra markers.
- No need of sensors other than the camera.
- No need of any knowledge about the environment.
How SLAM works

- Initialization

- T is a supposed known translation. But in fact we don't have any idea of it.

- Give an idea of the real position of features.
How SLAM works

- A plane is computed relatively to the points.
- Points of interests and camera position are stored into key-frames.
How SLAM works

- Update camera position with known points.
- Add new points of interest.
- Add key-frame when good tracking.
- Adjust known mapping.
What is PTAM

- PTAM stands for Parallel tracking and mapping.
- Similar that SLAM but do tracking and mapping parts in separate threads.
- Currently works on PC (Linux/Windows) and iPhone.
- Actually in GPLv3 license.
PTAM tests

• Always draw something even when the camera doesn't catch any points of interest.
  - Solved by testing a tracking quality threshold.

• Designed to run in small area.
  - Using multiples maps can solve this problem.
  - A fork exists PTAMM.
PTAM tests

• AR scene dimension can vary according to the initialization phase.
  – A perfect solution isn't possible with only a single camera.
  – User should be aware of that.

• When the camera move fast, motion blur make points disappear, tracking quality decrease.
  – Motion blur direction and intensity can be detected. Missing points position can be estimated.
Motion blur
Adapt PTAM

- PTAM already compute the scenery quality.
  - Needed to add new key-frames and readjust the current known map.
- Here a Boolean was added to indicate if the threshold is enough high to draw or not the scenery.
Adapt PTAM

- Change the example scenery.
- Relatively easy to replace it by a similar class.
  - All it needs internally is a \texttt{Init}, a \texttt{Reset} and a \texttt{Draw} function.
- The camera position is given as an argument in the draw function.
- The transformation to place the origin is already done in the OpenGL projection matrix.
Adapt PTAM

• Some extra tools were added:
  – A mesh loader.
    The mesh format used is Wavefront obj.
  – A texture loader.
AR programming interface

class Bot
{
public:
    Bot * up;
    Bot * down;
    Vector<3> v3Position;
    int fixedPart;
    Vector<3> v3RotationAxis;
    float axisAngle;
    Vector<3> v3RotationSide[6];
    float sideAngle[6];
    Bot * connected[5];
    Bot();
    Bot(int x, int y);

    static Bot * createUnit();
    void SetPosition(Vector<3> pos);
    unsigned int NearestFace(Vector<3> v);
    void RotateOnSide(float angle, unsigned int side);
    void RotateOnAxis(float angle);
    void Attach(Bot * bot);
    void Attach(Bot * bot, unsigned int a, unsigned int b);
    void Attach(int x, int y);
    void Draw();
};
One side is always fixed.
What should be done now?

- Complete the 3D scenery.
- Improve the user interface.
  - Permit the user to place a Roombot in front of the camera, to choose between different scenery, ...
Thank you for your attention!

Questions?