Computer Graphics Research at the University of Auckland –

A Modular Direct Volume Rendering Framework for Multi-dimensional and Multi-field Data & Augmented Reality Interfaces for Robot Development

Burkhard Wuensche

Dept. of Computer Science, University of Auckland

School of EECS, Kyungpook National University

burkhard@cs.auckland.ac.nz

Welcome

Coverage

- University of Auckland
- Graphics Group
- A Modular Direct Volume Rendering Framework for Multi-dimensional and Multi-field Data
- Augmented Reaility Interfaces for Robot Development

The University of Auckland

1 University in New Zealand
15 in Asia
50 worldwide

Times Education Word University Ranking [Nov 2007]

5 Campuses
30000 Students
86 Degrees

Burkhard Wuensche University of Auckland http://www.cs.auckland.ac.nz/~burkhard

Graphics Group University of Auckland, NZ

- 1 academic staff
- 1 research programmer
- 3 PhD & 3 MSc Students
- Project students

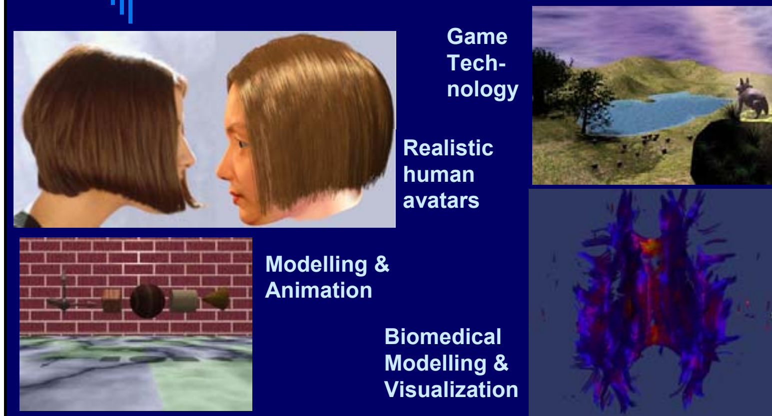
- > 70 international publications since 1998
- Participation in 10+ research grants (> 2 million NZ$) in the past 6 years
- > 20 student scholarships in the past six years

URL: http://www.cs.auckland.ac.nz/GG

Burkhard Wuensche University of Auckland http://www.cs.auckland.ac.nz/~burkhard GG

The Graphics Group - Some Research Interests

The Graphics Group - Some Research Interests



A Direct Volume Rendering Framework for the Interactive Exploration of Higher-Order and Multifield Data

Felix Manke and Burkhard Wünsche

GRAPHICS GROUP

Division for Biomedical Imaging & Visualization
Graphics Group, Department of Computer Science,
The University of Auckland

Outline

- Introduction
- Motivation
- Design Aspects of our DVR Framework
- Texture Transfer Functions
- Results
- Conclusion and Future Work
- References

Introduction

- Volumetric Data ↔ discrete 3D data array
- Obtained from measurements or simulations
- Visualisation is very powerful to analyse and explore huge data sets interactively
- Direct Volume Rendering allows the user to see all aspects of a volume data set.

(Engel et al., 2004)

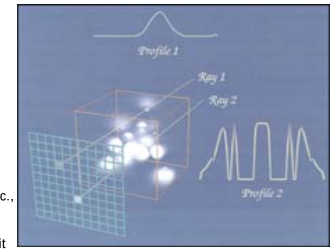
Direct Volume Rendering (DVR)

- Regard scalar field values as densities of a gas-like material
- Gas emits light, and also attenuates light coming from behind
- For each pixel in the view plane accumulate the light intensity along a ray through the volume: $C = \int_0^\infty c(t) e^{-\int_0^t \kappa(u)du} dt$

where

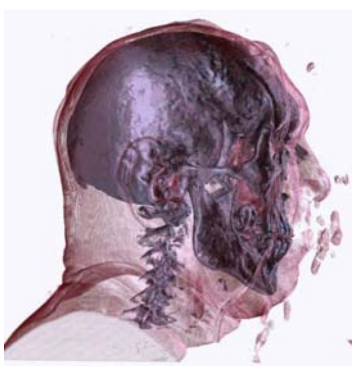
- $c_\lambda(t)$ is the emission per unit length wavelength λ
- κ_λ is the attenuation coefficient along the ray, defined by

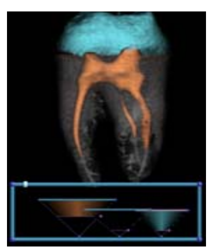
$$\frac{dI_\lambda}{dt} = -\beta_\lambda I_\lambda$$



© 2003 Kitware Inc., Schroeder, Martin, Lorensen. The Visualization Toolkit

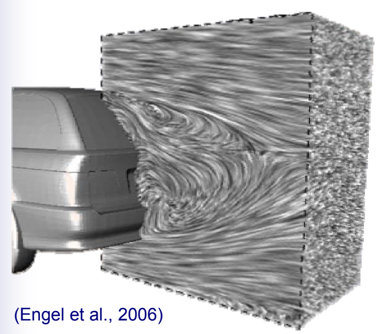
DVR Applications in Medicine





© 2006, Real-Time Volume Graphics, Daniel Weiskopf, Joe M.Kniss, Markus Hadwiger, Christof Rezk-Salama, Klaus Engel, AK Peters

DVR Applications in Science and Engineering



(Engel et al., 2006)

(Engel et al., 2006)

DVR Applications in Archeology

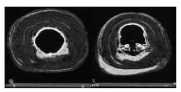
Non-destructive exploration and dissection of:

- prehistoric artifacts (dinosaur eggs, fossils in a chunk of soil)
- artifacts from ancient cultures (mummies, the 'frozen man')

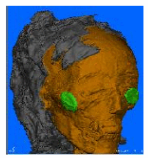
old procedure:
destructive unwrapping of the mummie

non-destructive procedure:

CT scan



CT slice reconstruction



volume rendering

artistic sketch based on volume rendering

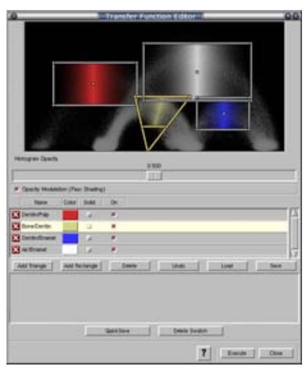
© Klaus Mueller, State University of New York at Stony Brook

Outline

- Introduction
- Motivation
- Design Aspects of our DVR Framework
- Texture Transfer Functions
- Results
- Conclusion and Future Work
- References

Motivation

- Shortcomings of today's volume visualisation tools:
  - Restricted to simple (scalar) data sets
  - Specialised for one type of complex data
  - Restricted in flexibility / controllability



© 2007, Scientific Computing and Imaging Institute, University of Utah, UT

Goals

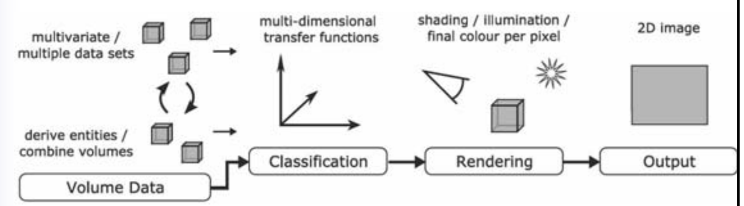
- Design of a volume rendering framework that
  - Is interactive
  - Renders in real-time using the GPU
  - Is extendable and allows to integrate arbitrary data
  - Gives full flexibility for the visualisation
  - Allows to derive whatever entities are needed
  - Allows to take arbitrary derived data as input to the various rendering stages

Outline

- Introduction
- Motivation
- Design Aspects of our DVR Framework
- Texture Transfer Functions
- Results
- Conclusion and Future Work
- References

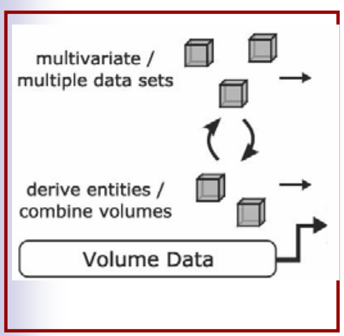
The DVR Process

- Flexibility is needed at each stage of the DVR Process:



The DVR Process

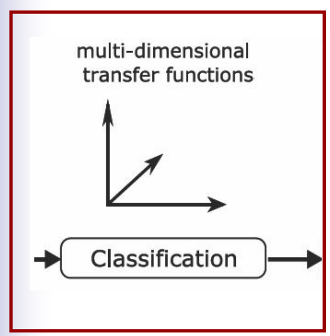
- Flexibility is needed at each stage of the DVR Process:



- Users must be able to
  - Load arbitrary data
  - Derive whatever entities are needed
  - Combine different volumes

The DVR Process

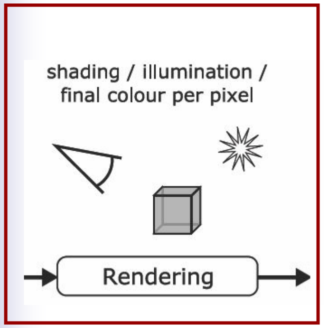
- Flexibility is needed at each stage of the DVR Process:



- Users must be able to
  - Use any input data for classification
  - Create new classification function, e.g. texture transfer functions

The DVR Process

- Flexibility is needed at each stage of the DVR Process:



  - Users must be able to
    - Define the rendering effects for the volume
    - Switch between different visualisations

Framework Extensibility

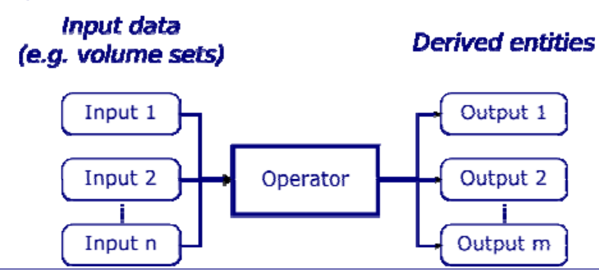
- (1) Creation of objects
  - Generic Singleton class serves as associative array (our "Generic Template Factory")
    - Maps identifiers to prototype objects (hash-table)
    - Template parameter defines common base-class of prototypes
    - Returns cloned copy of prototype on request
  - During initialisation, objects are created using a specified identifier
  - Registration of new sub-classes in a single line of code

Framework Extensibility

- (2) Initialisation of objects
  - Uses Adapter design pattern
    - Every Property is wrapped by a *Parameter* object
    - Parameter hides data type by using a unified string representation (→ Parameter is Adapter)
  - Objects hold list of available Parameters
  - On startup:
    - Parameters of objects are queried
    - `Set...()` method of Parameters is invoked using the string representation
    - Parameter performs conversion and sets Property

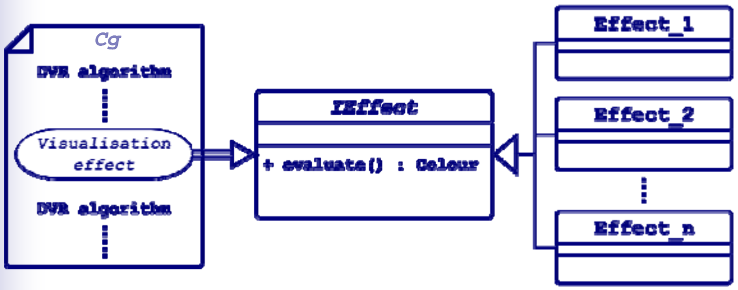
Framework Flexibility

- Derivation of entities on the GPU
  - Realised with custom kernels
  - Cg *annotations* specify operator configuration
  - *Up to 50x faster than on CPU!*



Framework Flexibility

- Visualisation effects on the GPU
  - Cg *interfaces* make switching possible
  - *Interfaces are exotic on GPUs!*



Outline

- Introduction
- Motivation
- Design Aspects of our DVR Framework
- Texture Transfer Functions
- Results
- Conclusion and Future Work
- References

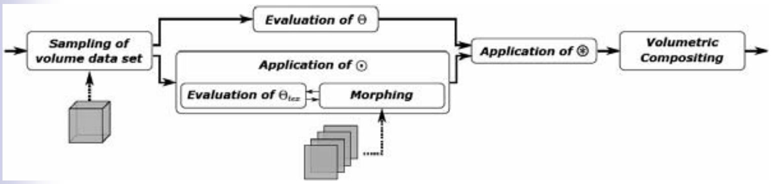
TE-DVR - Motivation

- DVR conventionally uses colours and opacities for conveying different aspects of the data
- Advances in data acquisition and simulation have resulted in increasingly complex and high-dimensional data sets → colour and opacity alone are often not sufficient for encoding all information.
- Textures are independent visual attributes
- IDEA: Use textures to encode additional information, such as material properties and additional data fields
- In order to correctly represent smooth

TE-DVR Framework

- Define function to map voxel positions into texture coordinates: $\Gamma : P \mapsto \Gamma(P) = Q$
- Define texture transfer function which associates (derived) field values with texture coordinates: $\Theta_{tex}(\triangleright(f)(P)) = \omega$
- Define operator which morphs textures in user-defined regions: $((C, \alpha)_{tex},\ \Lambda) = \odot[P,\ \Gamma_{1 \cdots L},\ \varsigma_{1 \cdots L},\ f,\ \triangleright,\ \Theta_{tex}]$
- Combine with results of colour and opacity transfer function: $(C, \alpha) = (C, \alpha)_{conv} \circledast (C, \alpha)_{tex}$

TE-DVR Algorithm Realisation



$(C_i, \alpha_i)_{conv}$ <- **evaluate** $\Theta(\triangleright(f)(P))$ // eval. of conv. transfer function

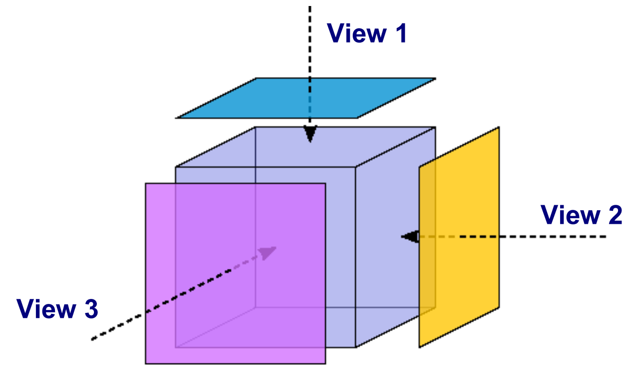
$((C, \alpha)_{tex}, \Lambda)$ <- $\odot[P, \Gamma_{1 \ldots L}, \varsigma_{1 \ldots L}, f, \triangleright, \Theta_{tex}]$ // morphing of textures

(C_i, α_i) <- $(C_i, \alpha_i)_{conv} \circledast (C_i, \alpha_i)_{tex}$ // combination of colours

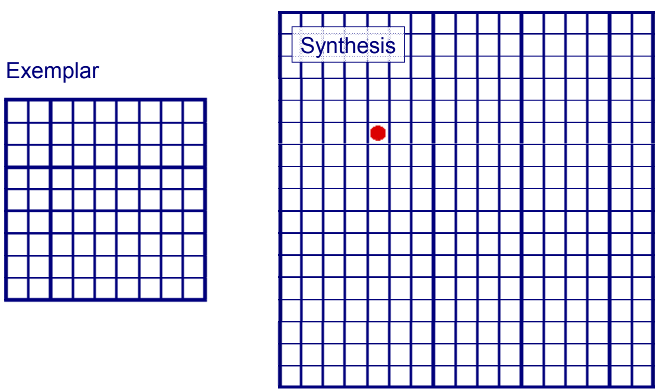
c'_i <- $(\alpha_i \cdot C_i) + (1 - \alpha_i) \cdot c'_{i-1}$ // volumetric compositing

3D Texture Synthesis and Morphing

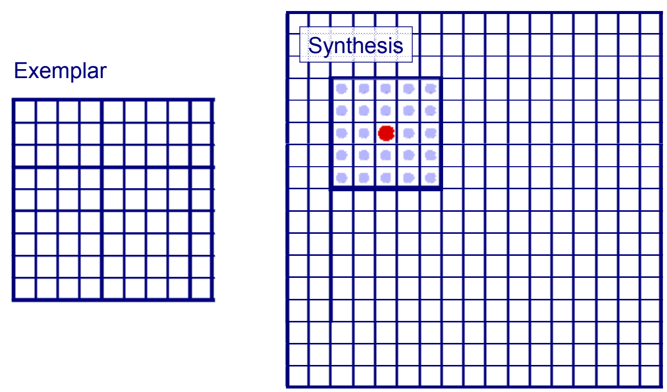
Reduce to 2D synthesis problem by using 2D "views" of the solid block



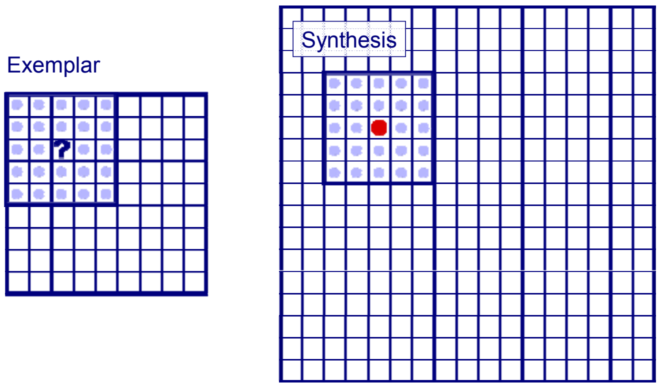
Neighbourhood-matching



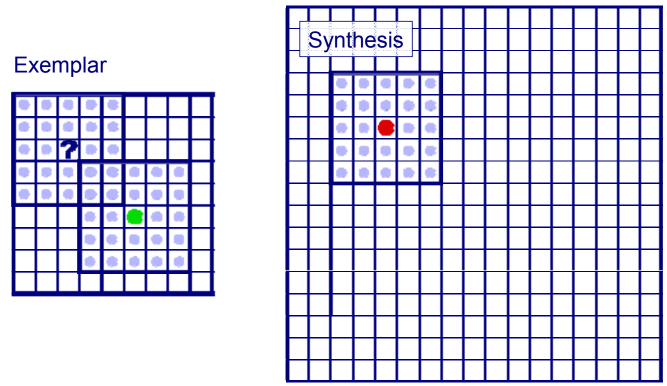
Neighbourhood-matching



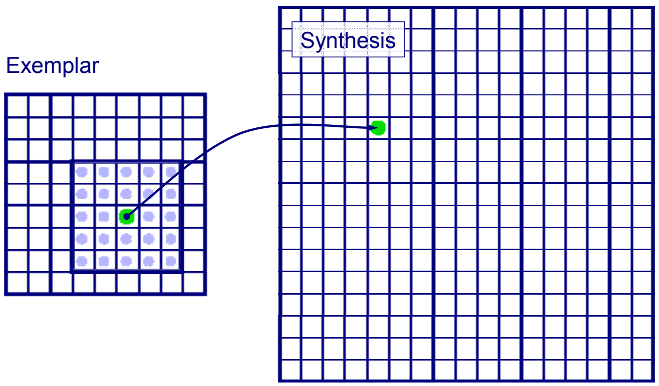
Neighbourhood-matching



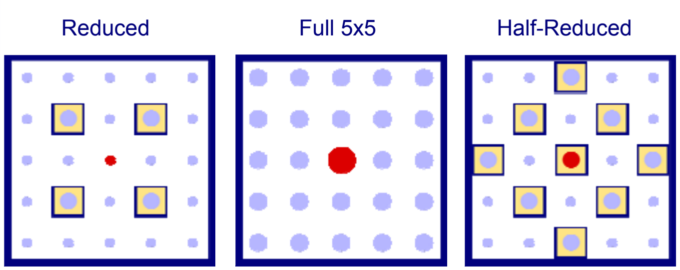
Neighbourhood-matching



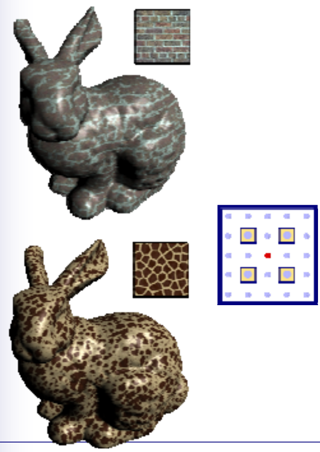
Neighbourhood-matching



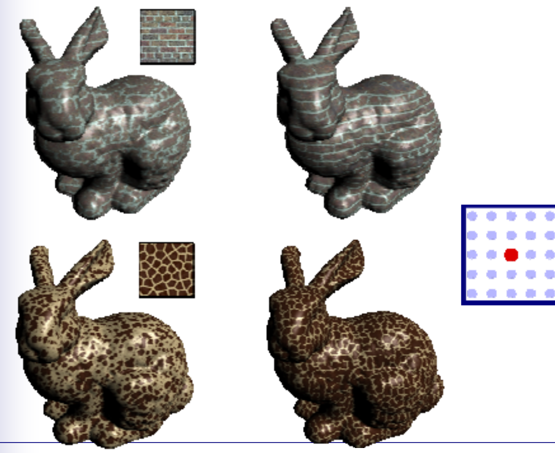
A New Neighbourhood



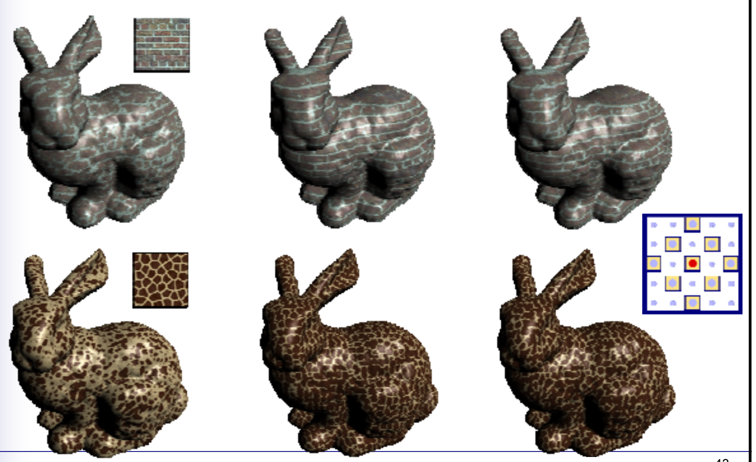
A New Neighbourhood



A New Neighbourhood

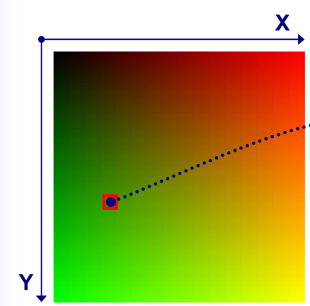


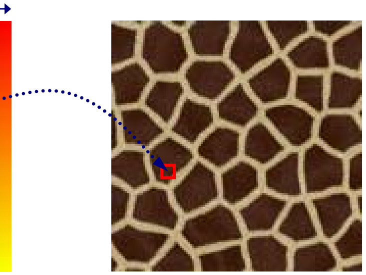
A New Neighbourhood



More Features

Synthesis with Texture Coordinates

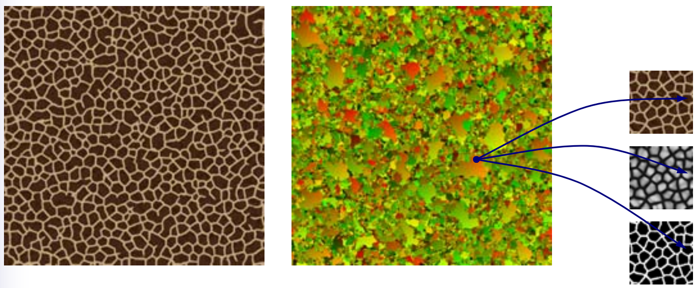




More Features

Arbitrary channels for free

Output Synthesis Exemplar



More Features

Arbitrary channels for free, e.g.

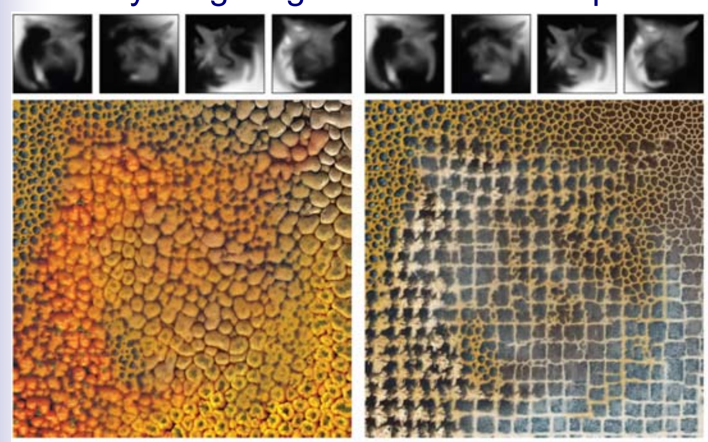
- transparencies, displacement maps,…

More Features

3D Morphing by using multiple synthesis pyramids

More Features

Arbitrary weighting functions and # input

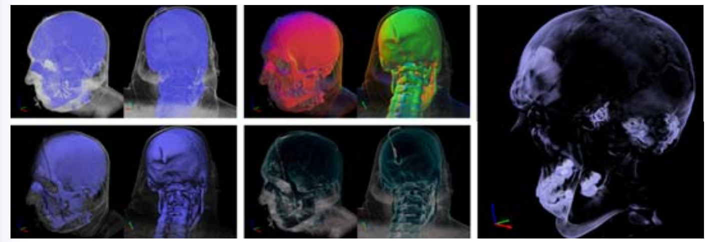


Outline

- Introduction
- Motivation
- Design Aspects of our DVR Framework
- Results
- Conclusion and Future Work
- References

Case study #1

- Different Effects for CT Data Set



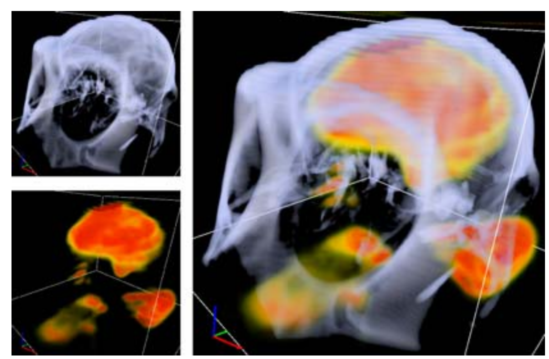
Results: Evaluators for CT Data Set

Different Effects for CT Data Set

```
// Simple colour-opacity evaluator
struct SimpleColourEvaluator : IEvaluator{
   float4 Evaluate(float3 texCoord){
      float density = tex3D(scalarVolumeSampler, texCoord).x;
      return tex1D(transferFunctionSampler, density);
   }
};
//  Gradient shading: gradient is mapped to RGB, opacity from transfer function
struct GradientEvaluator : IEvaluator{
   float4 Evaluate(float3 texCoord){
      float4 sample   = tex3D(gradAndVolSampler, texCoord);
      float  density  = sample.a;
      float  opacity  = tex1D(transferFunctionSampler, density).a;
      float3 gradient = normalize(2*(sample.xyz - 0.5));
      return float4(0.5 + 0.5*gradient, opacity);
   }
};
```

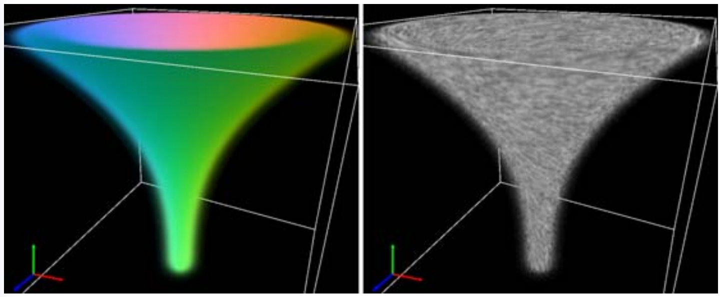

Case Study #2

- Combining 2 Data Sets



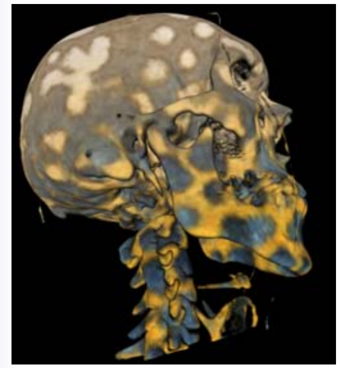
Case Study #3

- Procedural 3D Vector Field



Case Study #4

- Texture Transfer Functions



- Texture and colour
- Screendoor transparency
- Additional channels

Outline

- Introduction
- Motivation
- Design Aspects of our DVR Framework
- Results
- Conclusion and Future Work
- References

Conclusion

- We introduced a flexible DVR framework
  - It can easily be extended to support arbitrary volume data types and new DVR algorithms
  - Users have full control over the DVR process (as opposed to most conventional tools)
  - Helps us with current projects / research
  - The approach is new to the graphics community *(to our knowledge)*

Conclusion

- We introduced texture enhanced DVR
  - Mathematical framework for texture-enhanced DVR
  - The framework seamlessly integrates into the conventional DVR process, yet is extremely powerful and flexible, and enables entirely novel visualisations.
  - New texture synthesis and morphing algorithm
    - excellent trade-off between speed and quality
    - supports the morphing of arbitrarily many input textures
    - allows synthesising arbitrary channels (transparencies, displacement maps, or complex material properties) at almost no additional cost.
  - Other research groups are interested in the work

Future Work

- Additional functionality for the DVR
- Improved initialisation
  - String representation perhaps not perfect?
  - Support for abstract data types
- Improved usability
  - A GUI for selecting data sets, deriving entities and specifying visualisation effects
- Improve texture-transfer functions
  - Synthesise biomedical textures & improve morphing operator for these textures
  - Demonstrate method for real biomedical data sets
  - Implement morphing for multi-scale textures

Augmented Reality Interfaces for Robot Development

Bruce MacDonald, Toby Collett, Alex Kozlov, Ian Chen
Robotics and Intelligent Systems Laboratory
Department of Electrical and Computer Engineering

Burkhard Wünsche
Graphics Group, Department of Computer Science

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Outline

- UA Robotics and Intelligent Systems Lab
- Robotic software development
- Augmented Reality for:
  - Robot sensory data
  - Navigation algorithms (SLAM)
  - Mixed Reality simulation
  - Applications

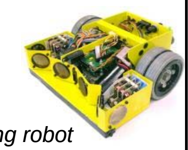
"Shuriken" Teaching robot

Robotics and Intelligent Systems Lab

- Long term goal: robot assistants for people
- Three main areas:
  - Robot Programming Systems
  - Human Robot Interaction
  - Applications: Healthcare and Agriculture
- Healthcare: NZ-Korean Centre for Aged Care
  - Link with South Korea: joint project with ETRI
  - UA researchers in robotics, IT, healthcare, health IT
  - NZ Health IT companies, US companies
  - ETRI, Korean Robot Companies

Multidisciplinary team

Bruce MacDonald,ECE	Robotics and Intelligent Systems
George Coghill,ECE	Artificial neural networks
Catherine Watson,ECE	Robotic Speech
Waleed Abdulla,ECE	Speech recognition
Karl Stol,Mech	Robot Navigation
Burkhard Wünsche,CS	Graphics and Visualisation
Liz Broadbent,Psych Med	Psychology in healthcare
Jim Warren,NIHI	Health Informatics
Karen Day,NIHI	Health Informatics
Martin Orr,NIHI	Health Informatics
Martin Connolly,Ger Med	Gerontology
Ngaire Kerse,Gen Practice	Gerontology
Mark Fisher,Middlemore	Geriatric Psychology
Gary Putt,UniServices	Business development
John Corey/John Hosking,CSI	Business development
Sarah Haydon/?,UniServices	Business development
Malcolm Pollock,NIHI	Business development



Vacuuming robot

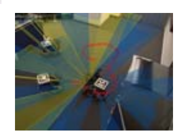
Capabilities

Robot face

- Mobile robot programming and control
- Robot programming systems
  - development environments
  - distributed programming
  - programming languages
- Programming by demonstration
- Emotional dimension of robotics (for speech and face)
- Perception augmentation using AR
- Visualisation
- Speech
- Navigation and coverage algorithms
- How are people's thoughts and feelings about robots influenced by the robot's behaviour? (Psychological studies)
- Evaluating robots in healthcare scenarios
- Applications in healthcare and agriculture

Facilities

- 2 large mobile robots
- 7 indoor pioneer robots
- 1 outdoor pioneer robot
- Helicopter robot
- Fiducial tracking system
- Debugging space
- Mechatronics testbed (Valeriy)



Robotic Software Development

- Currently developer tools are ad hoc
- Mobile robot environments:
  - Uncontrolled, dynamic, real world, unexpected variation
- Mobile robots
  - Change pose & relationship with environment and users
  - Large number of inputs and outputs
  - Variations in hardware and interfaces
- Tasks
  - Emphasize 3D geometry, complex data types, high dimension spaces and paths
  - May not be interruptible, multiple simultaneous activity

65

Robotic Software Development

- Programming languages
- Middleware & Libraries (Player)
- Tools

66

Programming languages for robotics (Geoff Biggs)

```
from time import sleep

event NearWall (sonar):
  for range in sonar.ranges:
    if range < 0.25~m:
      returnVal = range.index
      trigger

event HitWall (bumpers):
  for bumper in bumpers:
    if bumper == 1:
      trigger

```

67

Reactive programming
Dimensional analysis

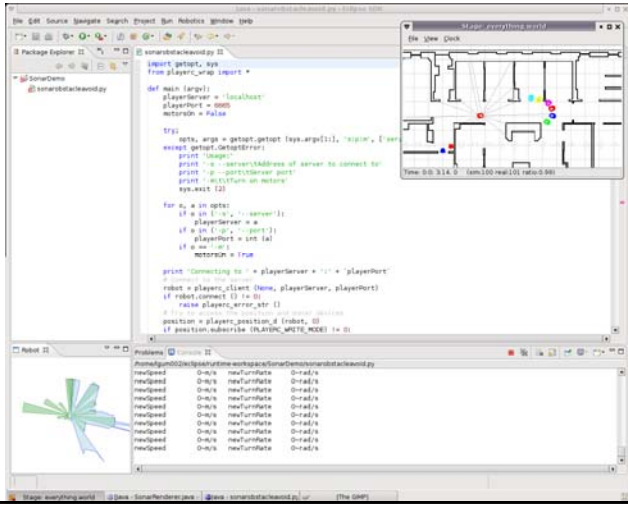
```
response UpdatePlayer (setSpeedFunc, speed):
  while True:
    setSpeedFunc (speed.getval ()[0], \
                  speed.getval ()[1])
    sleep (0.05~s)

response Drive (speed):
  speed.setval (0.5~m/s, 0~rad/s)
  while True:
    sleep (0.5~s)
    interrupt      # Check for interrupt @ 2Hz

...............
```

68

Eclipse based robotic software IDE (Luke Gumbley, Steve Hsiao)



69

Augmented Reality (Toby Collett)

- Debugging is a key process:

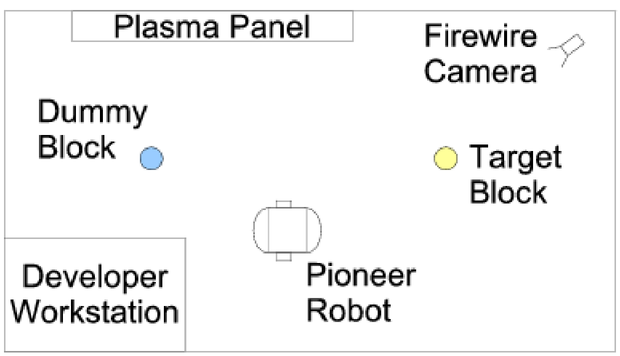
The programmer needs a clear understanding of the robot's world view

- Augmented reality for interacting with robots
- Targeted at developers
- Targeted at showing robot sensory data
- Increasing “perceptual overlap”
- Head mounted display OR large plasma display

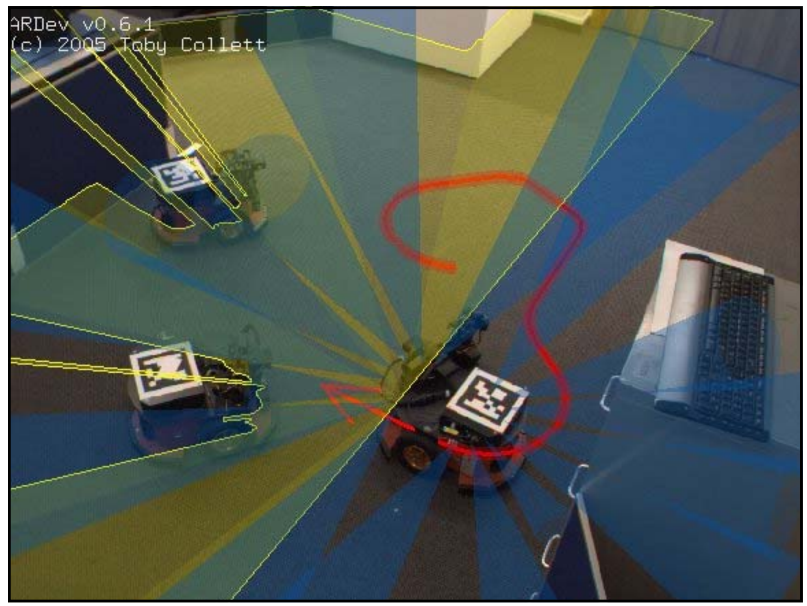
Presented at HRI2006, ICRA2006

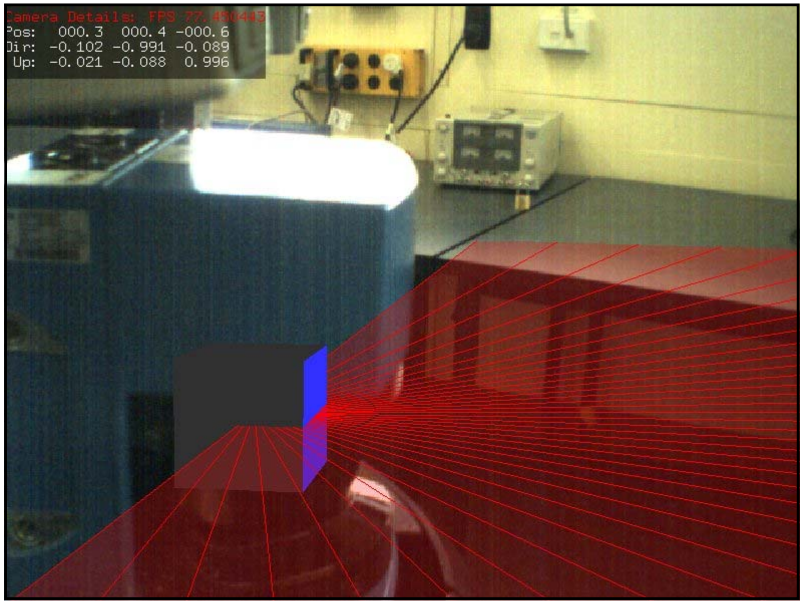
70

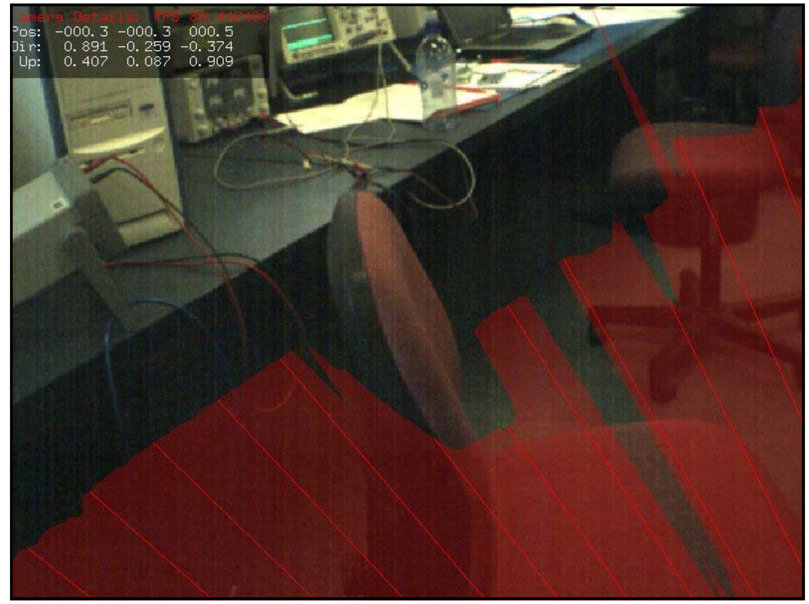
AR Debugging Space

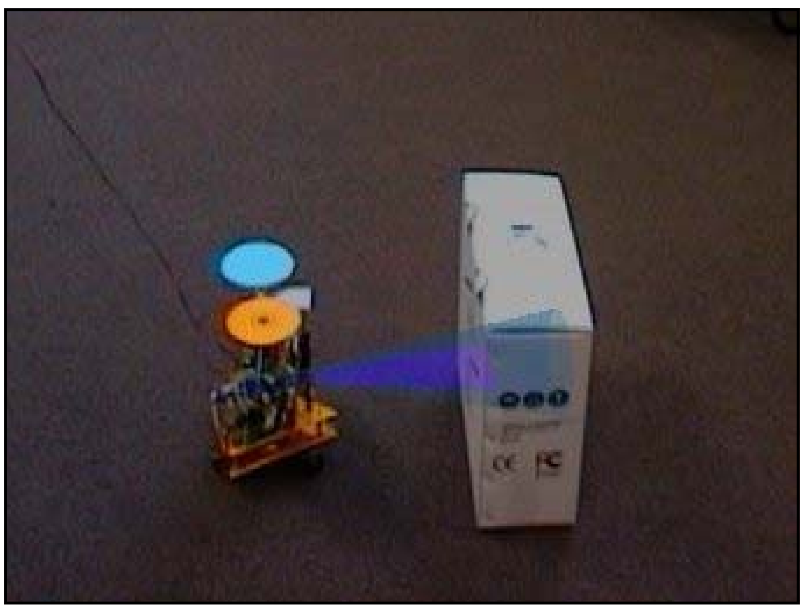


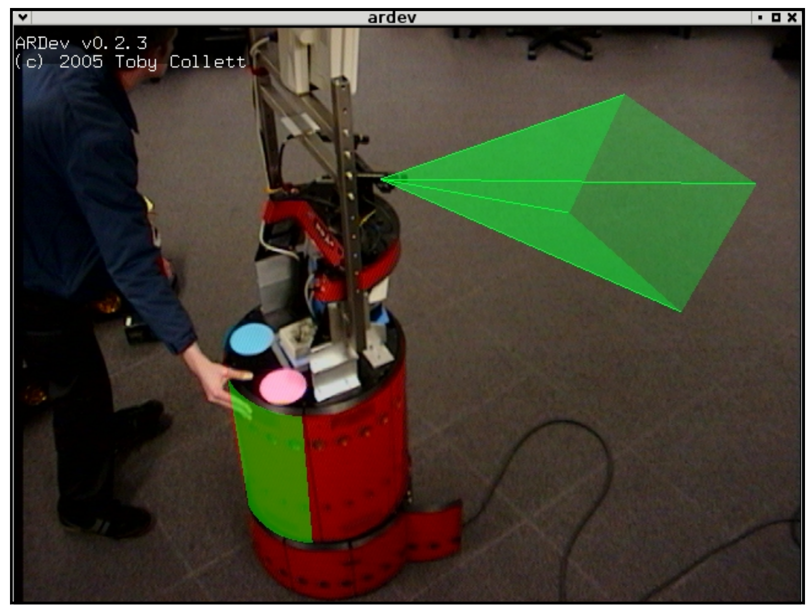
71

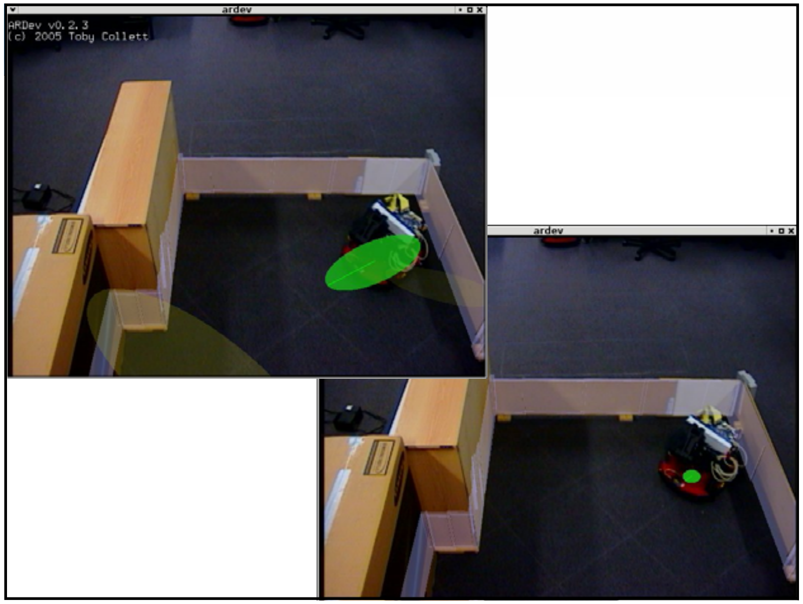



Camera Details:
Pos: 000.3 000.4 -000.6
Dir: -0.102 -0.991 -0.089
Up: -0.021 -0.088 0.996


Camera Details:
Pos: -000.3 -000.3 000.5
Dir: 0.891 -0.259 -0.374
Up: 0.407 0.087 0.909




ardev
ARDev v0.2.3
(c) 2005 Toby Collett



8 Stages for AR software

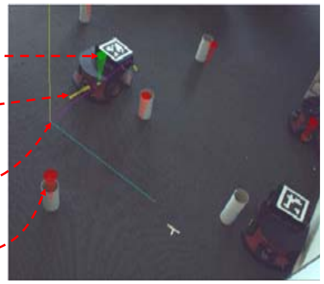
- *Capture*: background frame, orientation and camera position
- *Pre-processing*: e.g. blob tracking for robot registration
- *Render - Transformation*: view transforms are applied to the render object
- *Render - Base*: invisible models of known 3D objects are rendered into the depth buffer. Eg. so the robot obstructs the virtual data behind it
- *Render - Solid*: the solid virtual elements are drawn
- *Render - Transparent*: transparent render objects are drawn while writing to the depth buffer is disabled
- *Ray trace*: to aid stereo convergence, the distance to the virtual element in the centre of the view is estimated using ray tracing.
- *Post-processing*: eg encoding to a movie stream.

Augmented Reality for Simultaneous Localisation and Mapping (Alex Kozlov)

- We can also use AR to view the internal state of a robot
- For example, of a navigation algorithm
- Analysing SLAM estimation:
  - State vector: robot pose and feature map
  - Covariance matrix: uncertainties and correlations
  - Data association: sensed feature matches with map
  - Predicted maps, dynamic objects, pose trajectories, scan matching data

SLAM implementation: Alan Yang
Alex's Preliminary Results - *Prototype*

- Operation
  - Green marker - robot position
  - Yellow marker - robot orientation
  - The axes - origin of the map
  - Red marker - feature position



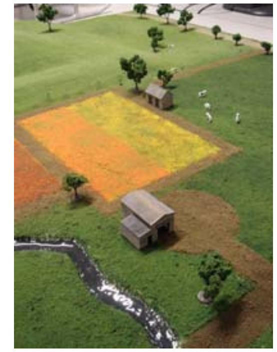
Preliminary Results - *Prototype*

- Expected behaviour
  - Green ellipse - robot position covariance
  - Yellow sector - robot orientation covariance
  - No features
    - High uncertainty

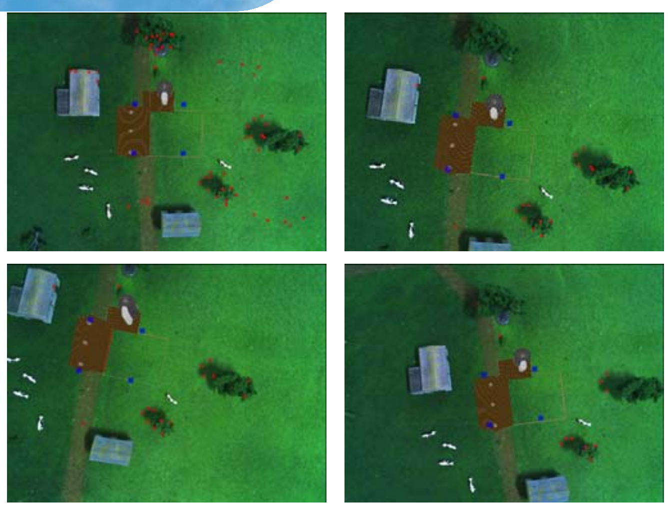
Presented at ACRA 2007

Mixed Reality Simulation for UAVs in Agriculture (Ian Chen)

- Objectives
  - To assist the current UAVs research by providing a 3D environment for real time visualisation and simulation
  - Effectively communicate useful feedback to robot application developers
  - Initially: Using AR to enable markerless tracking
    - KLT feature tracking
    - Projective reconstruction
    - Testing on a mockup farm



Prototype – Registration



Presented at Robot Vision 2008,LNCS, Springer

Application: Robotics in agriculture (Rick Chen)

- Helicopter project
- Tracking animals
- Monitoring fields and animals
- Interest from NZ IT companies in agriculture
- Mixed reality simulation and programming by demonstration

Rotomotion SR20 unmanned helicopter

UoA Healthcare Robotics Project (Tony Kuo)

- With Dr Liz Broadbent in Psychological Medicine
- Human reactions to good/bad robots: IROS 2007
- Student project in 2007, now a new PhD project
- Initially:
  - Taking blood pressure
  - Taking pulse
  - Taking temperature
  - Reminder service for medication
  - Networked communications to health services
- Shortly: taking blood samples, psychological evaluation
- AR for augmenting the interaction with carers and patients

85

Summary

- Robotic software development poses some difficult problems for developers
- Augmented Reality techniques can improve the developers' perception of the robot and its environment, and improve the programming process

86