Safe Human-Computer Interface Based on Efficient Image Processing Algorithm

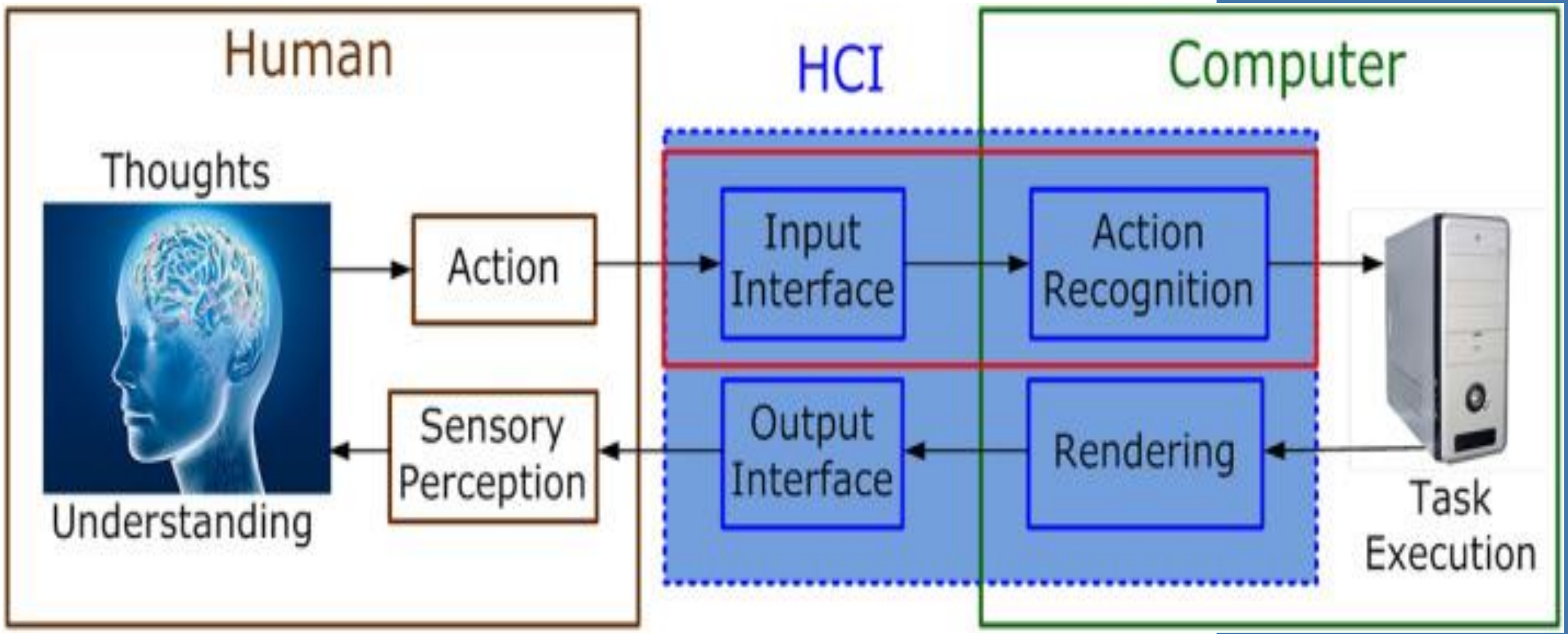
S. Wilding, P. Walker, S. Clinton, D. Williams, and J. Olszewska

Content

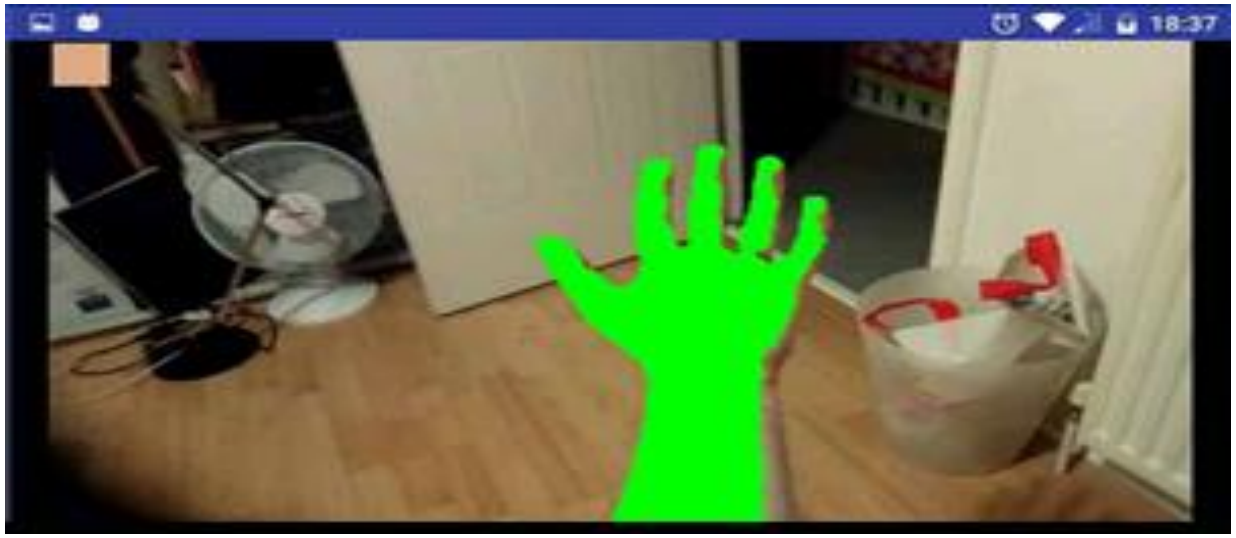
- The Context
  - Socially-Distanced Reality
  - Human-Computer Interface
  - Human-Centred Computer-Vision
- The Solution
  - Safe, Remote Human-Computer Interaction
  - New Image Processing Algorithm
  - Testing & Deployment

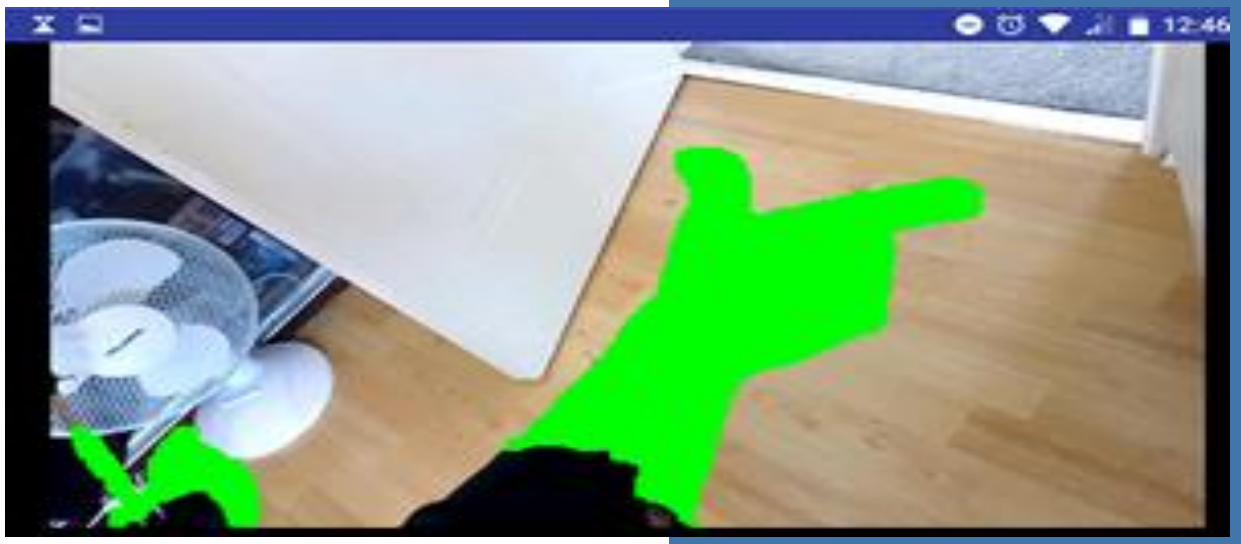
The Context

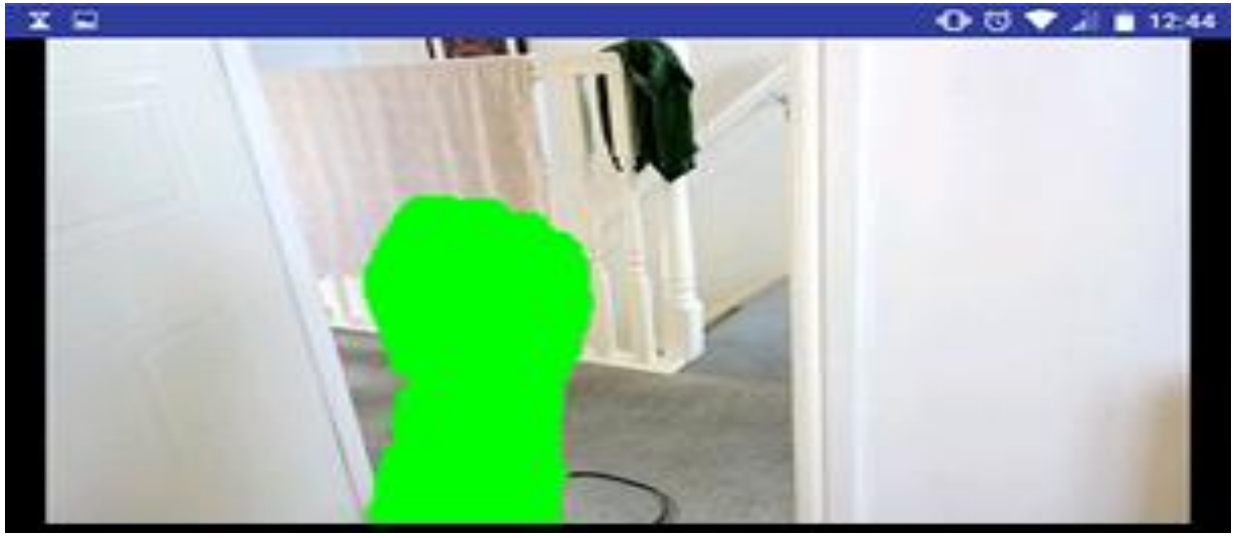
Human-Computer Interface

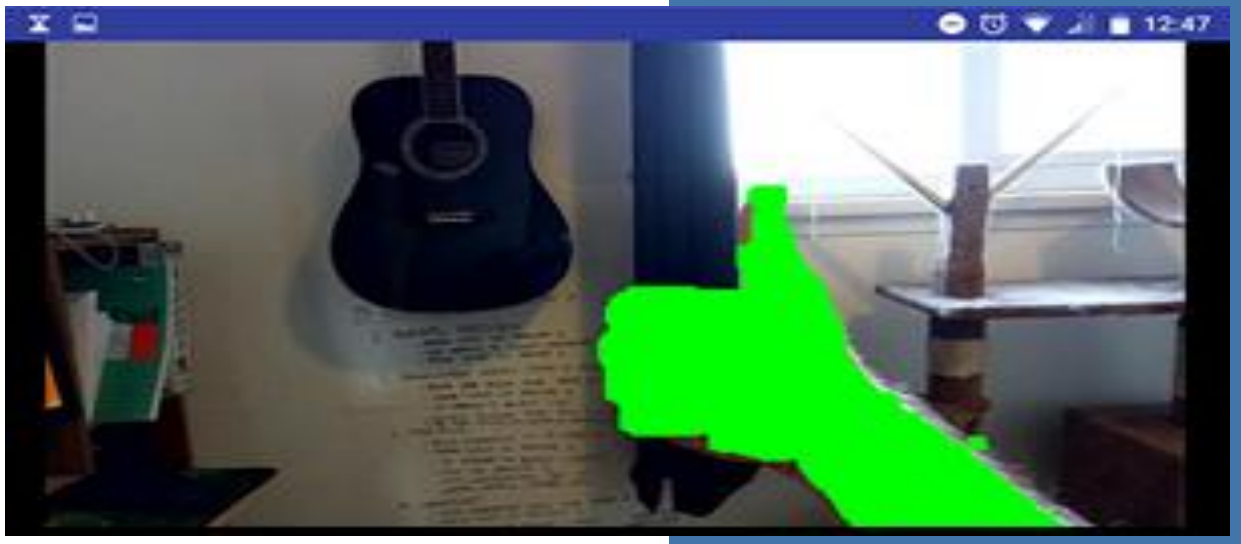


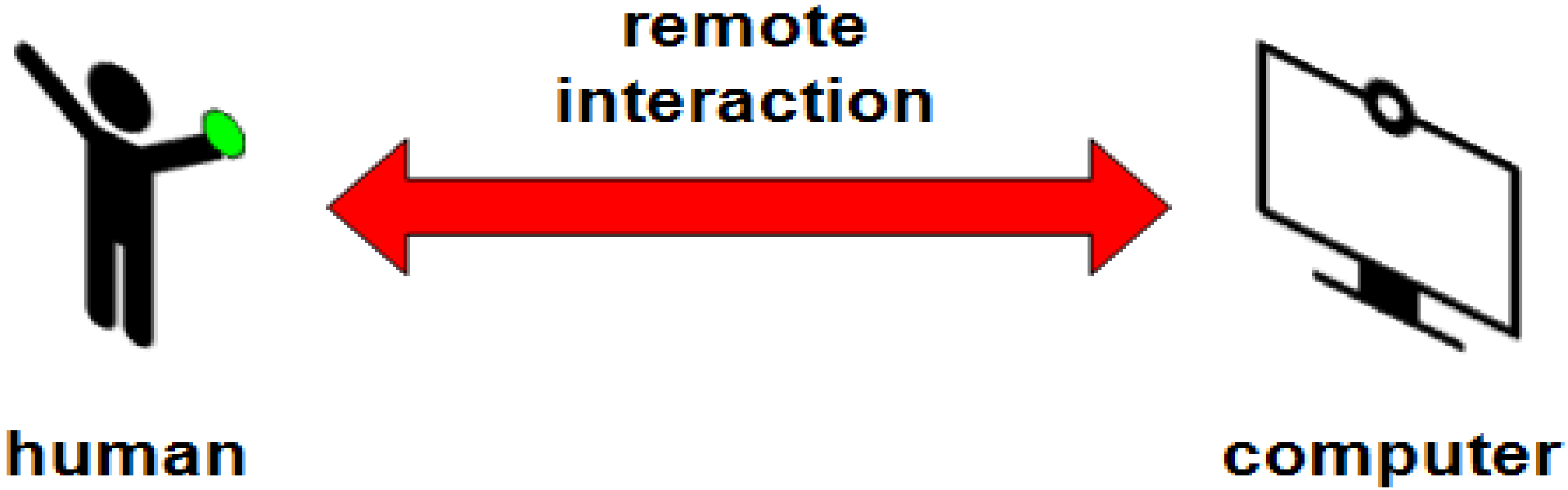
Human-Centred Computer Vision











The Solution

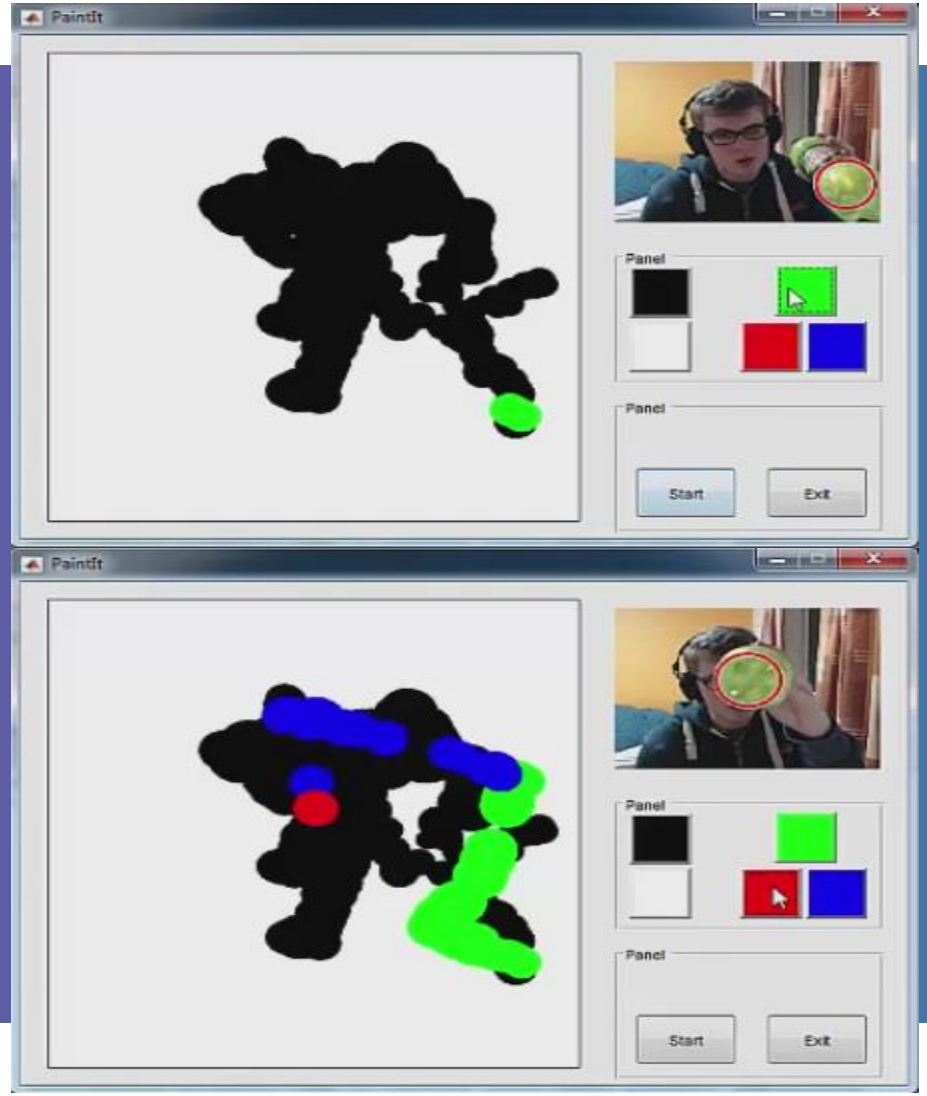
New Image Processing Algorithm

```
function findMarker(inputImage)
    %seperate the green channel from the ...
        other colour channels in the image
    inputImage = inputImage(:,:,2) - ...
        inputImage(:,:,1)/2 - ...
        inputImage(:,:,3)/2;
    %threshold the image so that green ...
        channel is above a specified intensity
    inputImage = inputImage > 50;
    %remove smaller objects from the image
    inputImage = bwareaopen(inputImage,30);
    %use region props to acquire the ...
        dimensions of any detected green object
    dimensions = regionprops(inputImage, ...
        'Centroid', 'MajorAxisLength', ...
        'MinorAxisLength', 'Area');
    %check if an object was detected
    if isempty(dimensions)
        %if no object detected, then set ...
            dimensions to zero
        center = [0,0]; radius = 0;
    else
        %if an object was detected, then ...
            isolate the dimensions of the ...
            object with the largest area
        [¬,id] = max([dimensions.Area]);
        %store the coordinates for the ...
            center of the object
        center = [dimensions(id).Centroid];
        %calculate and store the radius of ...
            the object
        radius = mean([dimensions(id). ...
            MajorAxisLength, dimensions(id). ...
            MinorAxisLength], 2) / 2;
    end
    %return the acquired dimensions of the ...
        object
    returnValue = [center, radius];
end
```

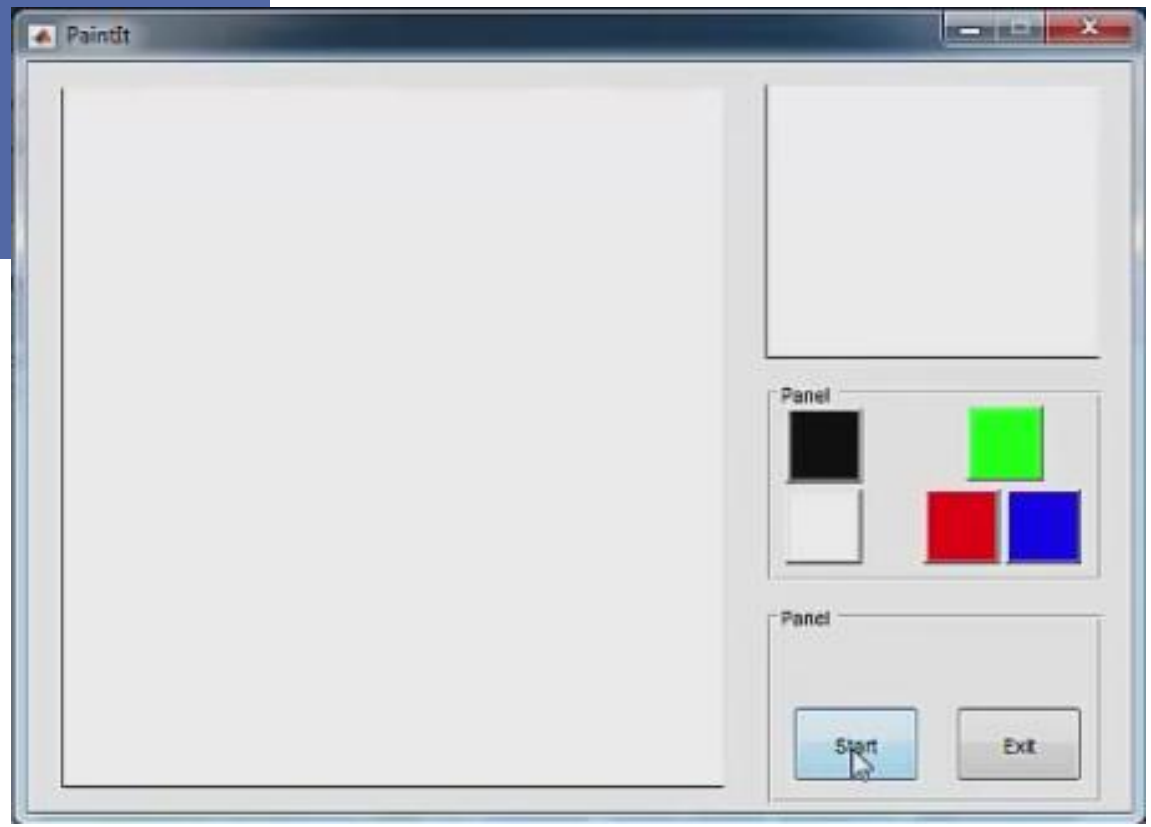
```
function ...
    start_Callback(hObject,eventdata,handles)
    %define the function global variables
    global started;
    global exit;

    if started ≠ true;
        %continue if the function is not ...
            already running
        started = true;
        %variable determining the cursor ...
            stroke colour
        global paintColour;
        paintColour = [0 0 0]; %default ...
            value is black

        %check if a webcam is connected
        if ¬isempty(webcamlist)
            %if a webcam is detected, then ...
                establish a connection to it
            cam = webcam;
            %variable determining the ...
                frame-per-second rate
            fps = 30;
            %acquires the resolution of the ...
                camera as a string and
            [xRes, yRes] = ...
                strtok(cam.resolution, 'x');
            %separate horizontal and ...
                vertical resolution by using ...
                strtok with 'x' as a delimiter
            yRes = strtok(yRes, 'x');
            %convert the obtained resolution ...
                values into numbers
            xRes = str2double(xRes);
            yRes = str2double(yRes);

            %loop while a webcam is detected ...
                or until the user exits
            while ¬isempty(webcamlist) && ...
                exit ≠ true;
                %direct output to cam ...
                    preview axes
                axes(handles.axes2);
                %obtain and store a mirrored ...
                    snapshot from the camera
                img = flip(snapshot(cam), 2);
                %search for a marker within ...
                    the image
                marker = findMarker(img);
                %display the snapshot in the ...
                    preview axes
                imshow(img);
                %check the radius of the ...
                    marker to determine if ...
                    one was present
                if marker(3) > 0
                    %draw a circle around ...
                        the ball in the preview
                    hold on;
                    viscircles([marker(1), ...
                        marker(2)], marker(3));
                    hold off;
                    %change the current axes ...
                        to the drawing canvas
                    axes(handles.axes1);
                    %plot to the canvas ...
                        using the marker ...
                        position, radius and ...
                        the current cursor ...
                        stroke colour
                    plot(marker(1), yRes- ...
                        marker(2), '.', ...
                        'color', ...
                        paintColour, ...
                        'MarkerSize', ...
                        marker(3));
                    %remove numbers and ...
                        markers from side of ...
                        the axes
                    set(handles.axes1, ...
                        'xTickLabel', [], ...
                        'yTickLabel', [], ...
                        'xTick', [], ...
                        'yTick', []);
                    %set the scale of the ...
                        axis to match the ...
                        webcam resolution
                    axis([0,xRes,0,yRes]);
                    %prevents changes from ...
                        being erased in the ...
                        next loop
                    hold on;
                end

                %pause before looping again ...
                    to acquire the desired ...
                    frame-per-second rate
                pause(1/fps);
            end
            %signal the program to close ...
                when the loop is exited
            started = false;
            close all;
        else
            %else abort the execution of the ...
                function
            started = false;
        end
    end
end
```

Safe, Remote HCI System

- Functional requirements of the HCI system are gesture detection/ identification/tracking.
- Non-functional requirements of the HCI system are latency, resolution, and stability.
- Other requirements include dependability, efficiency, and safety.

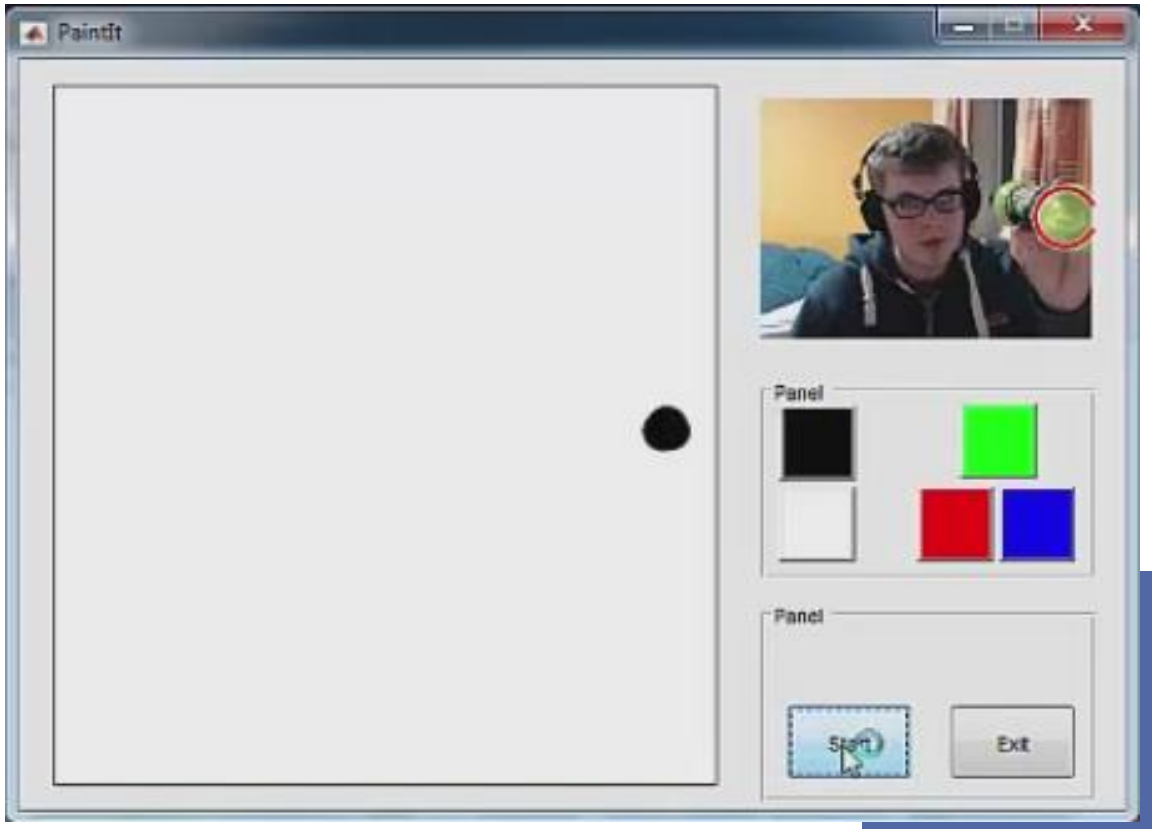


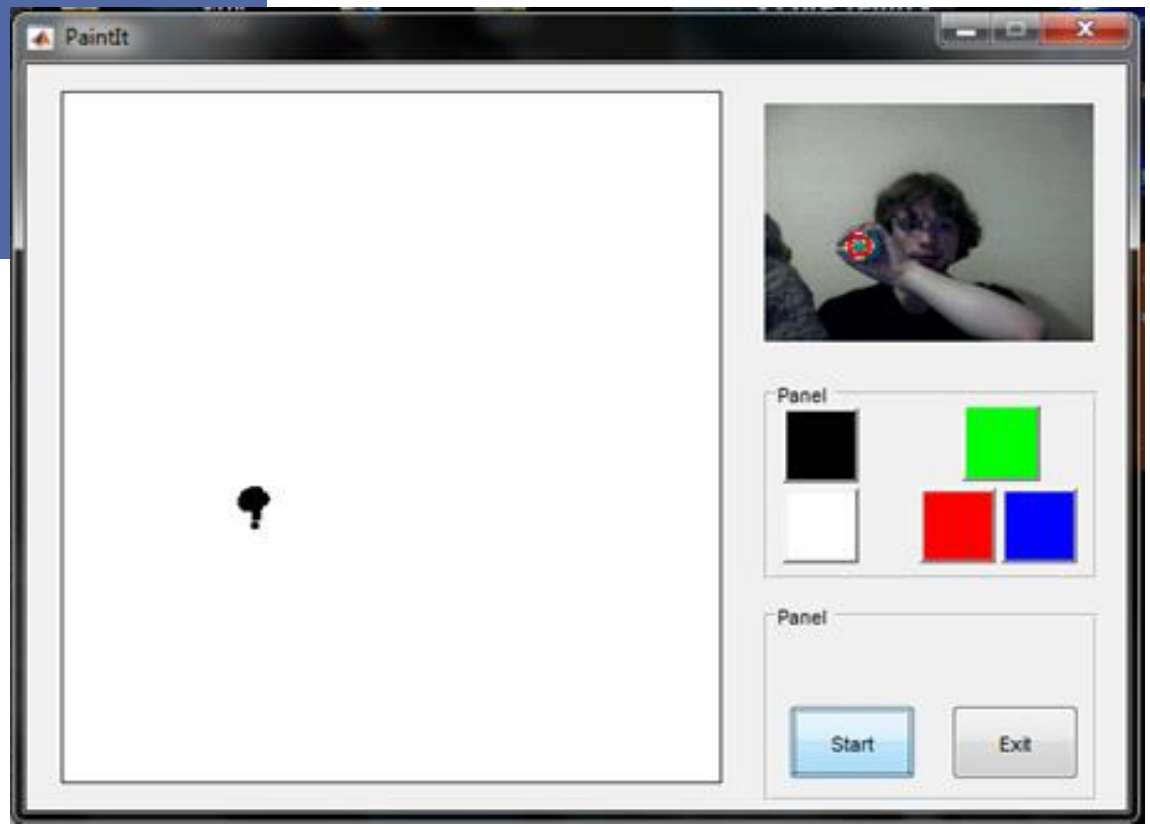
- A common set of usability requirements for HCI system consists of learnability, flexibility, robustness, predictability, synthesiability, familiarity, consistency, generalization, dialogue initiative, multithreading, task_migrability, substituability, customisability, observability, recoverability, responsiveness, and task conformance.



Menu selection

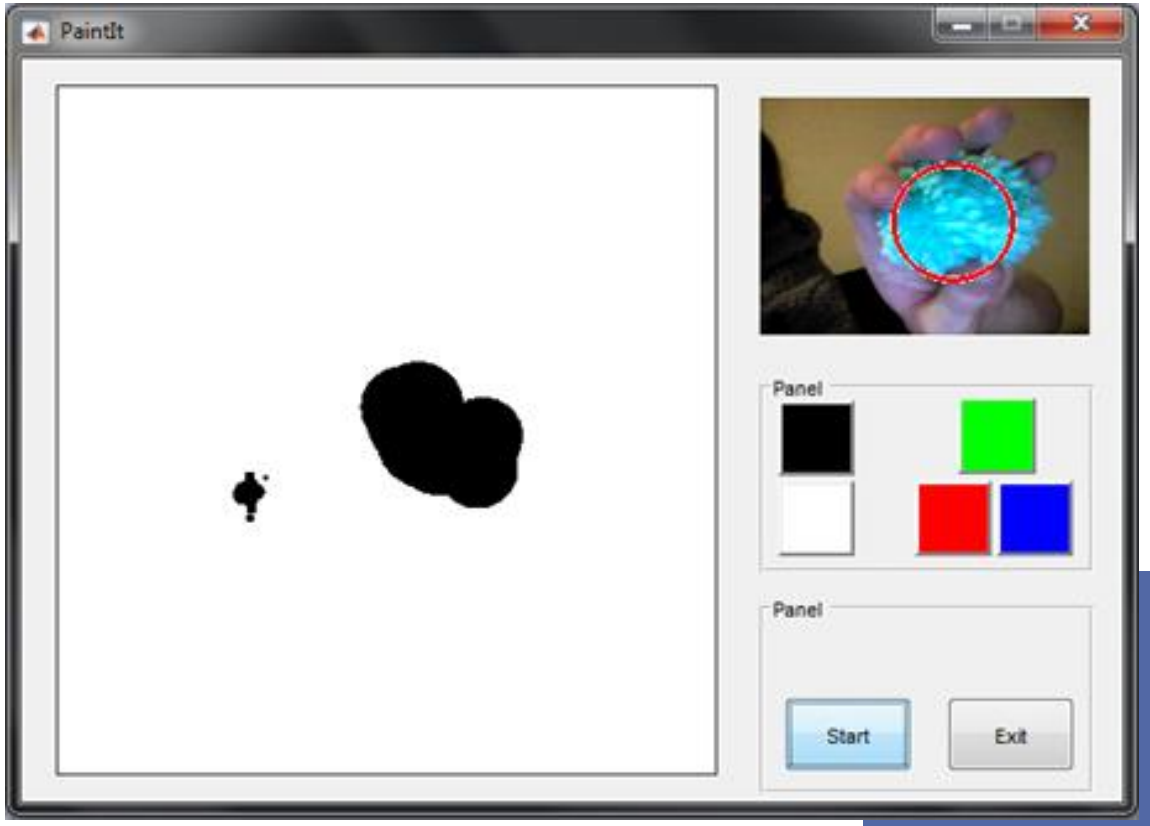
Direct manipulation





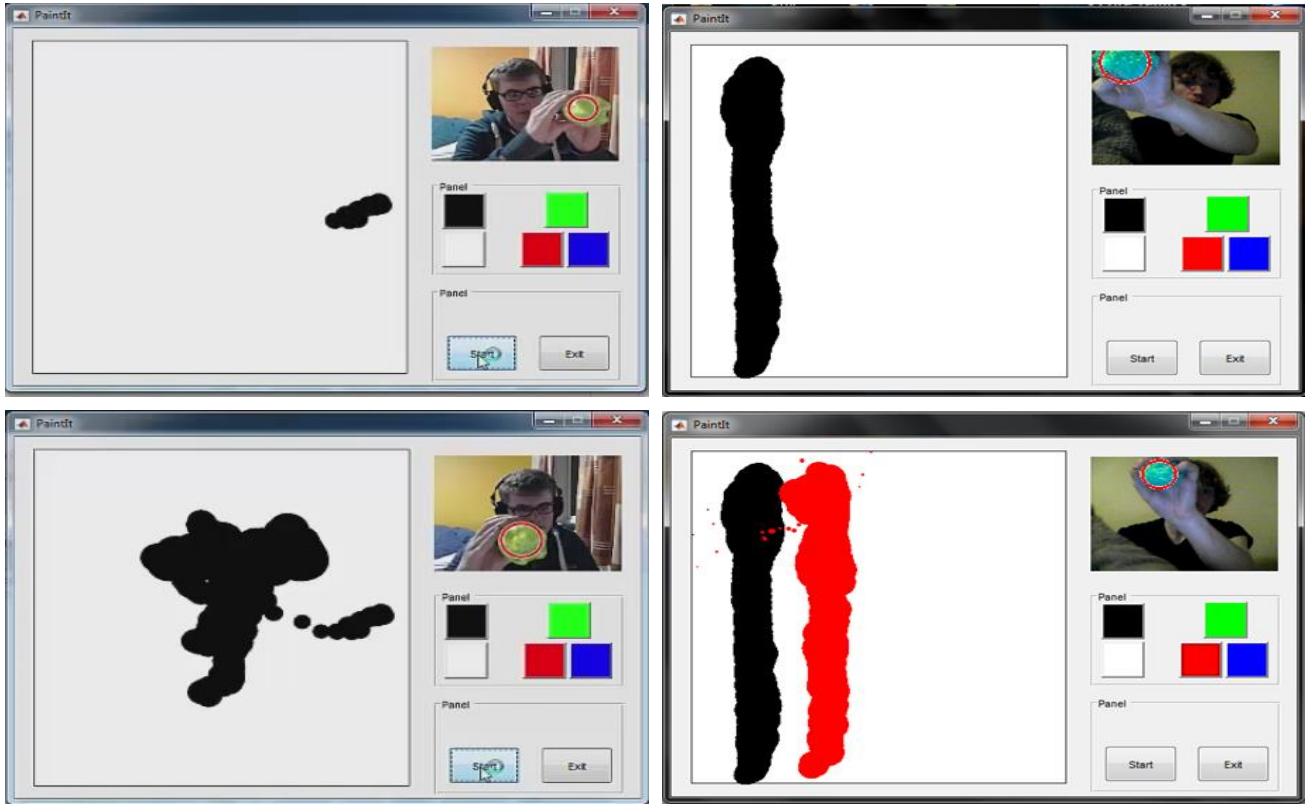
Zoomable interaction (zoom out)

Zoomable interaction (zoom in)



HCI System Performance

- We can observe in Table I that the latency of our system running on Computer 2 meets the requirements for a real-time HCI application, i.e. the latency is below the minimum acceptable latency which is 50ms.
- Moreover, the detection rate of our image-processing algorithm when run on Computer 2 achieves real-time performance, since it is equal or greater than 25Hz.



	With Computer 1	With Computer 2
HCI system latency	2 s	0.033 s
Marker detection rate	2 fps	30 fps

Conclusions

Human beings are presently surrounded by an increasing number of pervasive machines they have to interact with.

To help people with safe interactions, and thus social distancing with IT equipment or intelligent agents, we developed a safe, intuitive and stable HCI to allow touch-free interaction with indoor or outdoor machines.

Our HCI system consists in an accessible and non-invasive set-up which is made of a single camera mounted on a machine running a software which is based on an image-processing algorithm computing and interpreting the user's marker gestures in real time and real world environment.

The safety-by-design as well as transparency of our system together with its performance make it suitable for industrial and human-centered applications.