3D Makeup Projection

Ng Hui Xian Lynnette
Supervisor: A/P Terence Sim

NUS National University of Singapore | School of Computing
Leading The World With Asia's Best

ABSTRACT

Imagine seeing a model's image and wondering how you look like with the same makeup colours on. 3D Face Makeup Projection implements the extraction of face makeup from a model's image and projects it to a user's face. The solution is developed using off-the-shelf equipment including a Kinect and a Projector in C++ using the Kinect API and DirectX11. The processing pipeline includes: Makeup Colour Extraction, Projector-Kinect Position Calibration, Colour Calibration, Face Detection with Kinect and Face Texture Rendering. The pipeline is implemented via a state machine. The results show a robust implementation and successful real-time makeup projection on to user's face. The user's face can be detected and tracked despite slight head movements, hair and facial accessories differing lighting conditions and distance to the Kinect. Detection of eye region allow the user to open his eyes.

PROCESSING PIPELINE

1 Makeup Colour Extraction

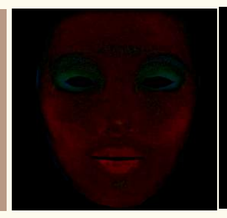
- *Modification and extension of Guo[1]'s method for 2D makeup transfer to extract only makeup colour.*
- *Performed in OpenCV with C++*
- *Extraction via Difference with Skintone, where the result can be used for Extraction via Masking for colours with red hues*
- *Extracted makeup image is warped to fit base UV image by Active Shape Model points*

Extraction via Difference with Skintone colour


Model Image


Skintone Image

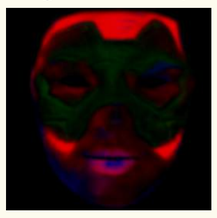

Skintone Makeup Image

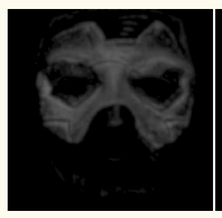

Difference Image

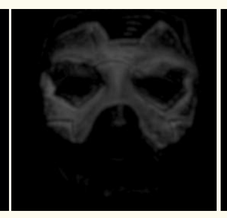

Denoised Image

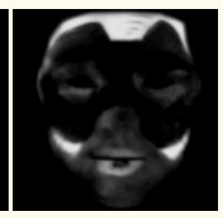
Extraction via Masking for makeup colours with red hues


Model Image


Denoised Image


Blue Channel


Green Channel


Red Channel

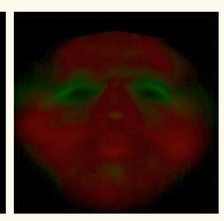

Blue Threshold


Green Threshold


Red Threshold

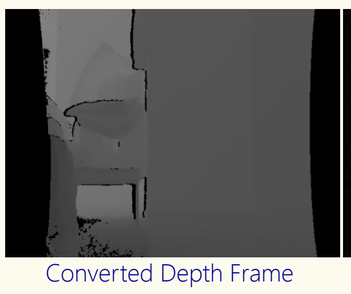

Combined Threshold

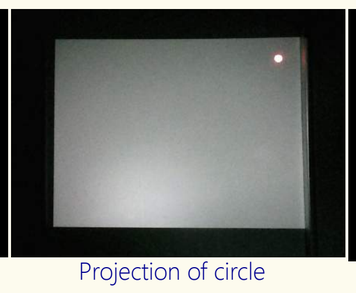

Final Mask Result

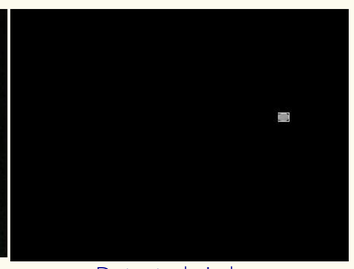

Warp Makeup to BaseUV mask

2 Projector-Kinect Position Calibration

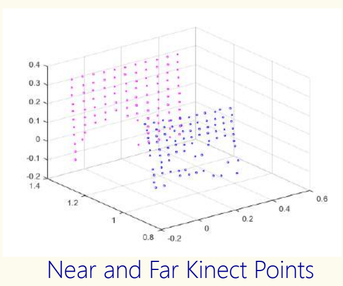
- *Kinect depth and colour frame matching*
- *Projection of 11x11 reddish-green circles on Near and Far Planes*
- *Scene captured with Kinect*
- *Detection of circle center (Kinect position) with OpenCV*

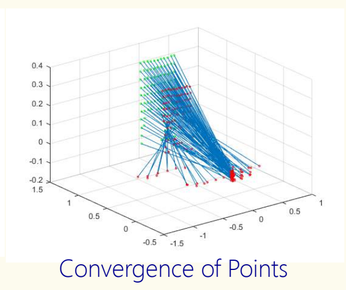

Converted Depth Frame


Projection of circle

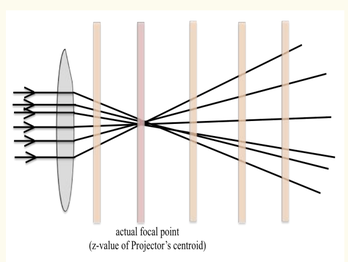

Detected circle

- *Multiple light rays in Projector converge at focal point*
- *Form vectors projected to infinity between Near and Far point pairs*
- *Mean vector as center of intersection*
- *Prune vectors more than 2 standard deviations away from centroid*
- *Centroid recalculated with new set of vectors*

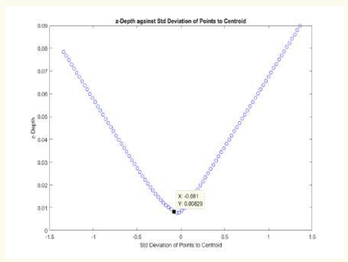

Near and Far Kinect Points


Convergence of Points

- *Find point of convergence with different z-planes by testing value of standard deviation of vectors projected to the z-value: z-value of Projector center*
- *Set Kinect's camera to obtained 3D point for alignment of image projection*



Finding z-value



Standard Deviation vs z-value

3 Colour Calibration

- *To account for Projector being off-coloured*
- *A white image projected to screen and center captured by Kinect*
- *RGB neutralisation, colour correction, gamma correction*

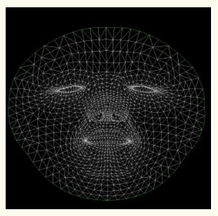
RGB Neutralisation

Gamma Correction

4 Face Detection with Kinect

- *Kinect face capture API captures 1347 HD Face Points*
- *Requires detection of at least half the user's body*
- *UI returns capture status such as 'Need: Right Views'*
- *Base UV mesh constructed with 3DS max to match new user's face points*


Kinect HD Face Points


Base UV unwrapped mesh

5 Face Texture Rendering with DirectX11

- *DirectX11 for fast and optimal rendering without overheads*
- *Point list obtained from Kinect HD points, triangle list generated*
- *Normal information for points calculated*
- *UV-coordinates from base UV map, vertices position, normal vectors merged into single graphic buffer for fast rendering*

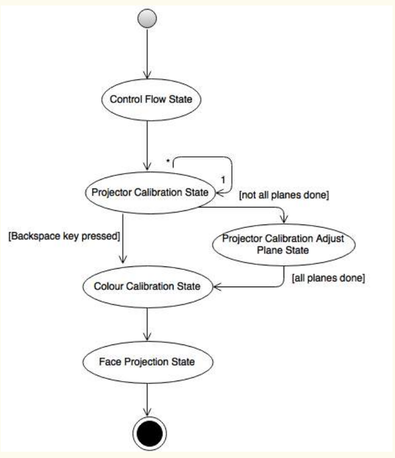

Point List


Normal Vectors


Base UV map

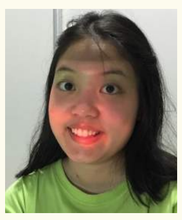
State Machine Implementation

State Machine for seamless transit through different steps of the pipeline, including repetition of states. The engineering of the system combines different inputs, outputs and system components such as Kinect, DirectX and Keyboard.



RESULTS

- *Time lag: ~4 frames*
- *Calibration time: ~2 min*



ACKNOWLEDGEMENTS

Reference: *[1]Guo, D & Sim, T. (2009) Digital Face Makeup by Example*
Special Thanks: *A/P Terence Sim, Dr. Alan Cheng, my family, Adrian Lim, my friends*