Technological Educational Institute of Crete

Interdepartmental MSc in Advanced Production Systems, Automation and Robotics

Master Thesis

Model-based 3D hand tracking using the lighting information as a visual cue

Nelson Stefanos Mbatha

RN: MTH51

Supervisor

Dr Alexandros Makris

HERAKLION 2017

Abstract

The goal of this thesis is to exploit lighting as a visual cue for the development of a 3D hand tracking system.

The proposed method follows the hypothesize and testing paradigm. Under this paradigm, the hand pose estimation boils down to comparing the RGB hand observation with several rendered hand hypotheses. An objective function calculates the discrepancy between the real hand pose and a rendered hand pose. In this thesis, we propose to use lighting information and a simple RGB sensor to build the objective function. This approach compares favorably with previous state of the art methods that require depth sensors and RGB-D observations to build the objective function.

Under the assumption of a known single light source we are able to photo-realistically render the hand model. To do so we use the principles of the Phong reflection model usually used in computer graphics to develop a lighting model. Also, an object detection technique able to extract the foreground information of the RGB images is developed. Several objective functions with different properties are created and tested.

The experiments showed which lighting objective functions are the most efficient and that under certain lighting conditions the use of the lighting information can be used to build a robust 3D hand tracking system which overcomes depth ambiguities in complex cases were the real hand pose either rotates or the fingers are bending.

Acknowledgements

First and foremost, I would like to thank my thesis supervisor Dr. Alexandros Makris for his guidance and support during all this time of research and writing of this thesis.

Besides my supervisor, I would like to thank my professor Dr. Emmanuel Drakakis who granted me a place to work and a personal computer until the completion of my thesis.

I thank my professors Dr. Ioannis Fasoulas and Dr. Emmanuel Kavvousanos who provided me the Kinect sensor.

Finally, I would like to address my thanks to my colleague Evangelos Vardakis for the support.

Table of Contents

Abstract i
Acknowledgements ii
Table of Contents iii
Table of figures viii
List of tables xv
Chapter 1 - Introduction 1
1.1. Motivation and aim of the project 1
1.2. Related work 1
1.3. The Kinect sensor 3
Chapter 2 - Background 7
2.1. The 3D Hand Tracker 7
2.1.1. Particle filters 7
2.1.2. The skinned hand model 7
2.1.3. Hierarchical Model Fusion framework (HMF) 7
2.1.4. Observation likelihood 9
2.1.5. Our contribution 10
2.2. Lighting Models 10
2.2.1. Introduction 11
2.2.2. Light and matter 11
2.2.3. Light sources 12
2.2.3.1. Color Sources 12
2.2.3.2. Ambient light 12
2.2.3.3. Point Sources 12
2.2.3.4. Spotlights 13
2.2.3.5. Distant light sources 15
2.2.4. The Phong reflection model 15
2.2.4.1. Ambient reflection 16
2.2.4.2. Diffuse reflection 16
2.2.4.3. Specular reflection 17

2.2.4.4. The Phong reflection 18
2.2.5. Implementation of the Phong reflection model 19
2.2.5.1. Normals and positions 19
2.2.5.2. Lighting vectors and viewer vectors 20
2.2.5.3. Reflection vectors 21
2.2.5.4. Diffuse reflection θ angles and specular reflection ϕ angles 22
2.2.6. Experiments and results 23
2.2.6.1. Reflection coefficients 23
2.2.6.2. Distance term 25
2.2.6.3. Shininess coefficient 26
2.2.6.4. Moving light's position 26
2.2.6.5. Moving hand's model position 31
2.3. Objective functions 34
2.3.1. Introduction 34
2.3.2. RGB-D visual data 34
2.3.2.1. RGB to GRAY 35
2.3.2.2. Foreground extraction (or background subtraction) 36
2.3.2.3. Normalization 40
2.3.3. The objective functions and their properties 41
2.3.3.1. Sum of absolute ($d_{i,j}$) differences 42
2.3.3.2. Number of big absolute ($d_{i,j}$) differences 43
2.3.3.3. Percent parameter "r" 44
2.3.3.4. Masks and the mask parameter "m" 45
2.3.3.5. Intersected pixels, penalty parameter and lighting absence 47
2.3.3.6. The objective functions 49
Chapter 3 - Experiments 54
3.1. Offline experiments 54
3.1.1. Case 1: X-Y axis movement 56
3.1.1.1. X-Y axis movement 56
3.1.1.2. Simulated – X-Y axis movement 63
3.1.2. Case 2: Y axis rotation 68
3.1.2.1. Clockwise Rotation 68

3.1.2.2. Simulated - Clockwise Rotation 75
3.1.2.3. Counter Clockwise Rotation 79
3.1.2.4. Simulated – Counter Clockwise Rotation 85
3.1.3. Case 3: Y axis rotation 2 89
3.1.3.1. Counter Clockwise Rotation 90
3.1.3.2. Simulated – Counter Clockwise Rotation 95
3.1.4. Case 4: X axis rotation 100
3.1.4.1. Downward Rotation 100
3.1.4.2. Simulated – Downward Rotation 106
3.1.5. Case 5: Bending 110
3.1.5.1. Bending 110
3.1.5.2. Simulated – Bending 116
3.1.6. Conclusions of the offline experiments 120
3.2. Tracking experiments 122
3.2.1. Datasets 122
3.2.2. Simulated sequence tracking 124
3.2.2.1. Simulate sequence tracking – 128 particles 124
3.2.2.1.1. Sequence 1 124
3.2.2.1.1.1. Default objective function 125
3.2.2.1.1.2. MS objective function 125
3.2.2.1.1.3. MSPL objective function 126
3.2.2.1.1.4. MCL objective function 126
3.2.2.1.2. Sequence 2 127
3.2.2.1.2.1. Default objective function 128
3.2.2.1.2.2. MS objective function 128
3.2.2.1.2.3. MSPL objective function 129
3.2.2.1.2.4. MCL objective function 129
3.2.2.1.3. Sequence 3 130
3.2.2.1.3.1. Default objective function 131
3.2.2.1.3.2. MS objective function 131
3.2.2.1.3.3. MSPL objective function 132
3.2.2.1.3.4. MCL objective function 132

3.2.2.1.4. Sequence 4 133
3.2.2.1.4.1. Default objective function 134
3.2.2.1.4.2. MS objective function 134
3.2.2.1.4.3. MSPL objective function 135
3.2.2.1.4.4. MCL objective function 135
3.2.2.1.5. Sequence 5 136
3.2.2.1.5.1. Default objective function 137
3.2.2.1.5.2. MS objective function 137
3.2.2.1.5.3. MSPL objective function 138
3.2.2.1.5.4. MCL objective function 138
3.2.2.1.6. Sequence 6 139
3.2.2.1.6.1. Default objective function 140
3.2.2.1.6.2. MS objective function 140
3.2.2.1.6.3. MSPL objective function 141
3.2.2.1.6.4. MCL objective function 141
3.2.2.2. Simulate sequence tracking – 256 particles 142
3.2.2.2.3. Sequence 3 142
3.2.2.2.3.1. Default objective function 143
3.2.2.2.3.2. MS objective function 143
3.2.2.2.3.3. MSPL objective function 144
3.2.2.2.3.4. MCL objective function 144
3.2.2.2.4. Sequence 4 145
3.2.2.2.4.1. Default objective function 146
3.2.2.2.4.2. MS objective function 146
3.2.2.2.4.3. MSPL objective function 147
3.2.2.2.4.4. MCL objective function 147
3.2.2.2.5. Sequence 5 148
3.2.2.2.5.1. Default objective function 149
3.2.2.2.5.2. MS objective function 149
3.2.2.2.5.3. MSPL objective function 150
3.2.2.2.5.4. MCL objective function 150
3.2.2.2.6. Sequence 6 151

3.2.2.2.6.1. Default objective function 152
3.2.2.2.6.2. MS objective function 152
3.2.2.2.6.3. MSPL objective function 153
3.2.2.2.6.4. Objective function MCL 153
3.2.2.3. Quantitative results and conclusions of the simulated sequence tracking experiments 154
3.2.3. Real video tracking 155
3.2.3.1. Video 1 155
3.2.3.1.1. Default objective function 156
3.2.3.1.2. MS objective function 156
3.2.3.1.3. MSPL objective function 157
3.2.3.1.4. MCL objective function 157
3.2.3.2. Video 2 158
3.2.3.2.1. Default objective function 159
3.2.3.2.2. MS objective function 159
3.2.3.2.3. MSPL objective function 160
3.2.3.2.4. MCL objective function 160
3.2.3.3. Video 3 161
3.2.3.3.1. Default objective function 162
3.2.3.3.2. MS objective function 162
3.2.3.3.3. MSPL objective function 163
3.2.3.3.4. Objective function MCL 163
3.2.3.4. Video 4 164
3.2.3.4.1. Default objective function 165
3.2.3.4.2. MS objective function 165
3.2.3.4.3. MSPL objective function 166
3.2.3.4.4. Objective function MCL 166
3.2.3.5. Conclusions of the real tracking experiments 167
Chapter 4 – Conclusions and future work 168
4.1. Conclusions 168
4.2. Future work 168
References 171

Table of figures

Figure 1.1: A Kinect sensor. 3
Figure 1.2: A Kinect sensor unwrapped. 4
Figure 1.3: Infrared dots pattern. 4
Figure 1.4: RGB image of a sofa (Left), The same image as the Kinect sensor sees it (Right). 5
Figure 1.5: Infrared image (Left), The depth map from white "near" to blue "far" (Right). 5
Figure 1.6: Kinect horizontal Field of View (Left), Kinect vertical Field of View (Right). 6

Figure 2.1: The employed 3D skinned hand model: hand polygonal surface (left) and hand kinematics (right). 7
Figure 2.2: Light – material interactions: a) Specular surface. b) Diffuse surface. c) Translucent surface. 11
Figure 2.3: Point source illuminating a surface. 13
Figure 2.4: Spotlight. 14
Figure 2.5: A point P inside the angle limits of a spotlight 14
Figure 2.6: Example of a spotlight with consecutively larger value of e. 15
Figure 2.7: Parallel rays of a distant light source 15
Figure 2.8: Vectors used by the Phong model. 16
Figure 2.9: Illumination of a diffuse surface when the light direction is vertical to the surface (right) and when there is an angle between them (left). 17
Figure 2.10: Specular reflection vectors and angles. 18
Figure 2.11: A normal to a surface at a point is the same as a normal to the tangent plane to that surface at that point 19
Figure 2.12: The normal vector n passing through a point at (x_0,y_0,z_0) 20
Figure 2.13: Normals (Right image) and Positions (Left image). 20
Figure 2.14: Specular reflection vectors. 22
Figure 2.15: Ambient reflection (left), diffuse reflection (middle) and specular reflection (right). 24
Figure 2.16: The effect of the ambient reflection. 24
Figure 2.17: The effect of the diffuse and specular reflections. 25
Figure 2.18: The effect of the distance term. 25
Figure 2.19: The effect of the shininess coefficient. 26

Figure 2.20: Original size of the images (height = 480, width = 640) and the light's initial position (red dot). 27
Figure 2.21: Realistic representation of the Kinect camera center and the hand model. 28
Figure 2.22: Light source: X – axis movement. 29
Figure 2.23: Light source: X – axis movement. 30
Figure 2.24: Light source: Z – axis movement. 31
Figure 2.25: Finger bending. 32
Figure 2.26: X – axis rotation. 33
Figure 2.27: Y – axis rotation. 33
Figure 2.28: RGB image (Left), Depth image (Right). 35
Figure 2.29: Realistic representation of the Kinect camera center and the RGB image. 35
Figure 2.30: GRAY image. 36
Figure 2.31: Red – channel. 36
Figure 2.32:Object detection. 37
Figure 2.33: Result after object detection: original (Left), zoomed (Right). 38
Figure 2.34: Extraction using the Depth information (Left), extraction using object detection (right). 39
Figure 2.35: RGB image. 40
Figure 2.36: Extraction using the Depth information (Left), extraction using object detection (right). 40
Figure 2.37: The absolute intensity differences – d. 41
Figure 2.38: Examples of different hypotheses. 42
Figure 2. 39: Hypothesis examples 45
Figure 2.40: Observation mask (Left), Hypothesis mask (Right). 46
Figure 2.41: Mask differences (Left), Union mask (Right). 46
Figure 2.42: Only the intersected pixels are used. 48
Figure 2.43: The intensity differences – d with the penalties. 48
Figure 2.44: Lighting absence. 49

Figure 3.1: RGB image of Case 1. 56
Figure 3.2: Observation (Left), initial hypothesis (Middle), and the differences (Right). 56
Figure 3.3: Observation (Left), final hypothesis (Middle), and the differences (Right). 57
Figure 3.4: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent. 57
Figure 3.5: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty parameter is used. ... 58

Figure 3.6: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used. 59
Figure 3.7: N equations. 59
Figure 3.8: R equations. 60
Figure 3.9: M equations. 60
Figure 3.10: NSL, RSL, MSL equations comparison. 61
Figure 3.11: NCL, RCL, MCL equations comparison. 62
Figure 3.12: Problems display. 63
Figure 3.13: Observation (Left), initial hypothesis (Middle), and the differences (Right). 64
Figure 3.14: Observation (Left), final hypothesis (Middle), and the differences (Right). 64
Figure 3.15: N equations. 65
Figure 3.16: R equations. 65
Figure 3.17: M equations. 66
Figure 3.18: NSL, RSL, MSL equations comparison. 67
Figure 3.19: NCL, RCL, MCL equations comparison. 67
Figure 3.20: RGB image of Case 2. 68
Figure 3.21: Observation (Left), initial hypothesis (Middle), and the differences (Right). 69
Figure 3.22: Observation (Left), final hypothesis (Middle), and the differences (Right). 69
Figure 3.23: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent. 70
Figure 3.24: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty component is used. 70
Figure 3.25: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used. 71
Figure 3.26: N equations. 72
Figure 3.27: R equations. 72
Figure 3.28: M equations. 73
Figure 3.29: NSL, RSL, MSL equations comparison. 74
Figure 3.30: NCL, RCL, MCL equations comparison. 74
Figure 3.31: Observation (Left), initial hypothesis (Middle), and the differences (Right). 75
Figure 3.32: Observation (Left), final hypothesis (Middle), and the differences (Right). 75
Figure 3.33: N equations. 76
Figure 3.34: R equations. 76

Figure 3.35: M equations. 77
Figure 3.36: NSL, RSL, MSL equations comparison. 78
Figure 3.37: NCL, RCL, MCL equations comparison. 78
Figure 3.38: Observation (Left), initial hypothesis (Middle), and the differences (Right). 79
Figure 3.39: Observation (Left), final hypothesis (Middle), and the differences (Right). 79
Figure 3.40: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent. 80
Figure 3.41: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty component is used. 80
Figure 3.42: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used. 81
Figure 3.43: N equations. 82
Figure 3.44: R equations. 82
Figure 3.45: M equations. 83
Figure 3.46: NSL, RSL, MSL equations comparison. 84
Figure 3.47: NCL, RCL, MCL equations comparison. 84
Figure 3.48: Observation (Left), initial hypothesis (Middle), and the differences (Right). 85
Figure 3.49: Observation (Left), final hypothesis (Middle), and the differences (Right). 86
Figure 3.50: N equations. 86
Figure 3.51: R equations. 87
Figure 3.52: M equations. 87
Figure 3.53: NSL, RSL, MSL equations comparison. 88
Figure 3.54: NCL, RCL, MCL equations comparison. 89
Figure 3.55: Observation (Left), initial hypothesis (Middle), and the differences (Right). 90
Figure 3.56: Observation (Left), final hypothesis (Middle), and the differences (Right). 90
Figure 3.57: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent. 91
Figure 3.58: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty component is used. 91
Figure 3.59: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used. 92
Figure 3.60: N equations. 92
Figure 3.61: R equations. 93

Figure 3.62: M equations. 93
Figure 3.63: NCL, RCL, MCL equations comparison. 94
Figure 3.64: NSL, RSL, MSL equations comparison. 95
Figure 3.65: Observation (Left), initial hypothesis (Middle), and the differences (Right). 96
Figure 3.66: Observation (Left), final hypothesis (Middle), and the differences (Right). 96
Figure 3.67: N equations. 97
Figure 3.68: R equations. 97
Figure 3.69: M equations. 98
Figure 3.70: NCL, RCL, MCL equations comparison. 99
Figure 3.71: NSL, RSL, MSL equations comparison. 99
Figure 3.72: RGB image of Case 4. 100
Figure 3.73: Observation (Left), initial hypothesis (Middle), and the differences (Right). 100
Figure 3.74: Observation (Left), final hypothesis (Middle), and the differences (Right). 101
Figure 3.75: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent. 101
Figure 3.76: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty component is used. 102
Figure 3.77: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used. 102
Figure 3.78: N equations. 103
Figure 3.79: R equations. 103
Figure 3.80: M equations. 104
Figure 3.81: NSL, RSL, MSL equations comparison. 105
Figure 3.82: NCL, RCL, MCL equations comparison. 105
Figure 3.83: Observation (Left), initial hypothesis (Middle), and the differences (Right). 106
Figure 3.84: Observation (Left), final hypothesis (Middle), and the differences (Right). 106
Figure 3.85: N equations. 107
Figure 3.86: R equations. 107
Figure 3.87: M equations. 108
Figure 3.88: NCL, RCL, MCL equations comparison. 109
Figure 3.89: NSL, RSL, MSL equations comparison. 109
Figure 3.90: RGB image of Case 5. 110
Figure 3.91: Observation (Left), initial hypothesis (Middle), and the differences (Right). 110

Figure 3.92: Observation (Left), final hypothesis (Middle), and the differences (Right). 111
Figure 3.93: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent. 111
Figure 3.94: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty component is used. 112
Figure 3.95: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used. 112
Figure 3.96: N equations. 113
Figure 3.97: R equations. 113
Figure 3.98: M equations. 114
Figure 3.99: NSL, RSL, MSL equations comparison. 115
Figure 3.100: NCL, RCL, MCL equations comparison. 115
Figure 3.101: Observation (Left), initial hypothesis (Middle), and the differences (Right). 116
Figure 3.102: Observation (Left), final hypothesis (Middle), and the differences (Right). 116
Figure 3.103: N equations. 117
Figure 3.104: R equations. 117
Figure 3.105: M equations. 118
Figure 3.106: NCL, RCL, MCL equations comparison. 119
Figure 3.107: NSL, RSL, MSL equations comparison. 119
Figure 3.108: MSPL and MCL objective functions comparison. 121
Figure 3.109: Sequence 1 observation images. 124
Figure 3.110: Default objective function images. 125
Figure 3.111: MS objective function images. 125
Figure 3.112: MSPL objective function images. 126
Figure 3.113: MCL objective function images. 126
Figure 3.114: Sequence 2 observation images. 127
Figure 3.115: Default objective function images. 128
Figure 3.116: MS objective function images. 128
Figure 3.117: MSPL objective function images. 129
Figure 3.118: MCL objective function images. 129
Figure 3.119: Sequence 3 observation images. 130
Figure 3.120: Default objective function images. 131
Figure 3.121: MS objective function images. 131
Figure 3.122: MSPL objective function images. 132
Figure 3.123: MCL objective function images. 132
Figure 3.124: Sequence 4 observation images. 133
Figure 3.125: MCL objective function images. 134

Figure 3.126: MS objective function images. 134
Figure 3.127: MSPL objective function images. 135
Figure 3.128: MCL objective function images. 135
Figure 3.129: Sequence 5 observation images. 136
Figure 3.130: Default objective function images. 137
Figure 3.131: MS objective function images. 137
Figure 3.132: MSPL objective function images. 138
Figure 3.133: MCL objective function images. 138
Figure 3.134: Sequence 6 observation images. 139
Figure 3.135: Default objective function images. 140
Figure 3.136: MS objective function images. 140
Figure 3.137: MSPL objective function images. 141
Figure 3.138: MCL objective function images. 141
Figure 3.139: Sequence 3 observation images. 142
Figure 3.140: Default objective function images. 143
Figure 3.141: MS objective function images. 143
Figure 3.142: MSPL objective function images. 144
Figure 3.143: MCL objective function images. 144
Figure 3.144: Sequence 4 observation images. 145
Figure 3.145: Default objective function images. 146
Figure 3.146: MS objective function images. 146
Figure 3.147: MSPL objective function images. 147
Figure 3.148: MCL objective function images. 147
Figure 3.149: Sequence 5 observation images. 148
Figure 3.150: Default objective function images. 149
Figure 3.151: MS objective function images. 149
Figure 3.152: MSPL objective function images. 150
Figure 3.153: MCL objective function images. 150
Figure 3.154: Sequence 6 observation images. 151
Figure 3.155: Default objective function images. 152
Figure 3.156: MS objective function images. 152
Figure 3.157: MSPL objective function images. 153
Figure 3.158: MCL objective function images. 153
Figure 3.159: Video 1 RGB images. 155
Figure 3.160: Default objective function images. 156
Figure 3.161: MS objective function images. 156
Figure 3.162: MSPL objective function images. 157
Figure 3.163: MCL objective function images. 157
Figure 3.164: Video 2 RGB images. 158
Figure 3.165: Default objective function images. 159
Figure 3.166: MS objective function images. 159

Figure 3.167: MSPL objective function images. 160
Figure 3.168: MCL objective function images. 160
Figure 3.169: Video 3 RGB images. 161
Figure 3.170: Default objective function images. 162
Figure 3.171: MS objective function images. 162
Figure 3.172: MSPL objective function images. 163
Figure 3.173: MCL objective function images. 163
Figure 3.174: Video 4 RGB images. 164
Figure 3.175: Default objective function images. 165
Figure 3.176: MS objective function images. 165
Figure 3.177: MSPL objective function images. 166
Figure 3.178: MCL objective function images. 166

Figure 4.1: Shaded areas. 169
Figure 4.2: Foreground extraction fail points. 170

List of tables

Table 1: N equations. 50
Table 2: R equations 50
Table 3: M equations. 51
Table 4: Offline experiments cases. 54
Table 5: Simulated tracking sequences. 123
Table 6: Real tracking videos. 123
Table 7: Simulated sequence tracking average errors [mm] for 128 particles. 154
Table 8: Simulated sequence tracking average errors [mm] for 256 particles. 154

Chapter 1 - Introduction

1.1. Motivation and aim of the project

This project is based on visual tracking, the process of estimating the location of one or more objects in a video sequence over time using a camera. Visual tracking has a variety of uses, some of which are: human-computer interaction, security and surveillance, augmented reality, traffic control, medical imaging, and video editing. Tracking an object is a time consuming and complex process due to the big amount of data and the use of object detection techniques. Especially when the target object is moving fast and changes orientation, tracking is very difficult. However, many methods have been used to solve this problem. Each of them has its strengths and weaknesses.

The object in this project is the human hand. Tracking articulated objects in 3D can be a challenging problem. The high dimensionality of the problem, the chromatically uniform appearance of the hand and often self-occlusions are some of the factors that complicate the process. A specialized hardware or the use of visual markers would facilitate the process but at the same time the process would be more complex. For this reason, there are many approaches proposed for 3D hand tracking considering markerless visual data only. These approaches can be categorized in two major components. The appearance based (bottom-up) approaches and the model based (top-down) approaches. There are more approaches that combine and adapt more elements to improve the performance of tracking like the approach used in this project.

A software that tracks the 3D position, orientation, and full articulation of the human hand from markerless visual observations has been created by the Computer Vision and Robotics Lab at ICS/FORTH *[5]*. The visual observations come from an RGB-D camera, the Kinect sensor. A model based approach using a Bayesian Hierarchical Model Fusion framework (HMF), first proposed by Makris *[7]*, addresses the problem of 3D hand tracking. The HMF decomposes the initial problem into smaller and simpler problems and efficiently addresses the implications of the high dimensionality. We use this HMF approach due to the increased speed and robustness in comparison with other methods. The most computationally demanding parts of the process have been implemented to run efficiently on a GPU.

The hand tracker implementation relies mostly on depth information. The goal of this thesis is to develop an algorithm that will be able to estimate the pose of a human hand but focus on the incorporation of visual information to the system like the **light intensities** in an RGB image, to help recognize the hand pose in cases where depth information usually fails.

1.2. Related work

The work on hand tracking can be divided into two main approaches, the **model-**based approaches and the **appearance-based** approaches. The model-based approaches [6*, 8, 20, 21,31*] use an articulated 3D hand model which is projected into the image and an error function measures the quality of the match. The model projection is synthesized on-line and the registration of the model to the image can

be done with local optimization around the previous estimate or by filtering. These approaches require model initialization and do not recover easily from a track failure but provide accurate estimation of the hand's pose. Appearance-based approaches [*26, 27, 28*] work directly on the image data and the problem is formulated as a classification or regression problem. A set of hand features is labeled with a particular hand pose, and a classifier is learnt from training data that is generated off-line with a synthetic model, or acquired by a camera from a small set of known poses. The advantage of these approaches is that no hand initialization is required. **Hybrid** methods that combine the model-based and appearance-based approaches have also been proposed [*22, 23, 24,25*].

In model based approaches the objective (error) function has a determining role in the quality of the obtained solutions and the optimization process. The objective function in [*21*] combines fitting terms that measure how well the parameters of the hand model represent the observation with prior terms that regularize the solution to ensure realistic hand poses. The fitting terms include a 3D point cloud alignment that measures the distance between every point in the observation's 3D point cloud to the track model, a 2D silhouette alignment and a wrist alignment. Again in [*31*] the first term of the error function aligns the point cloud of the observation (distances) to the sphere hand model, the second term forces the model to lie inside the point cloud using penalizing factors as the third term which penalizes model self-collision. A sum of Gaussians (SoG) image approximation and a 3D SoG hand model based on colors are used in [*22*] and after projecting the 3D SoGs to 2D a color similarity function measures the Euclidean distance between their color values. In [*23*] a GPU-based rendering module renders a synthetic depth image and a scoring function measures the L^1 discrepancy between the synthetic and the input depth images. A fixed truncation value contributes to the discrepancy value for each pixel of the observation and the hand model that don't intersect while pixels that are considered background in both images have zero contribution. In [*24*] an objective function measures the discrepancy between the positions of the observation's and the hand model joints. Also, a clamping distance is used for the large distances which produce false high penalties due to noise existence. Similarly, in [6*,20*] the depth differences are clamped within a predetermined range and the objective function measures the discrepancy between the depth and the skin-colored maps. A penalty is added to the function for invalid hand configurations.

As it is shown in the above paragraph the depth information is usually used to evaluate the discrepancy between the observation's data and the hand model in model-based tracking cases. Also, the objective functions apart from the depth term they may consist of penalization terms or terms responsible for 2D silhouette matching to improve the tracking accuracy. However, using the depth information includes many ambiguities when estimating the hand pose. These ambiguities usually occur from depth holes incorporated in the depth data. Efforts have been made to reduce depth ambiguity by exploiting visual information. The use of texture, illumination and shading parameters has been used in few cases like [*29*] where the objective function embodies texture and illumination (shading) elements. The illumination it is often seen as a problem but this paper [*30*] shows how the lighting can be used to make pose and shape estimation more robust. This thesis follows the

same idea of exploiting the image information and the illumination to develop an objective function which uses the light (intensity) differences instead of the depth differences to evaluate the discrepancy. The model-based 3D hand tracking system in [*8*] and the principles of its depth based objective function were used to build and test our alternative lighting objective function in the effort to eliminate depth holes.

1.3. The Kinect sensor

Microsoft Kinect (*Figure 1.1*) it's a highly innovative combination of cameras, microphones and software that turns your body into a video game controller. It was first announced and demonstrated as Project Natal in June 2009 and later in November 2010 released as a new addition to Microsoft's Xbox 360 product line. In comparison with Nintendo's Wii there is no need for any physical controller and that was a revolutionary difference. Microsoft released the Kinect software development kit (SDK) for Windows on June 2011 to allow developers to write Kinecting apps in C++, C# or Visual Basic.NET. The SDK provides a set of libraries that someone can add to his own programs and games so they can use the sensor. The SDK also contains all the drivers needed to link the Kinect to a computer.

Figure 1.1: A Kinect sensor.

The Kinect contains three hardware innovations that work together to enable body and voice recognition (*Figure 1.2*): a color VGA video camera, a depth sensor, and a multi-array microphone.

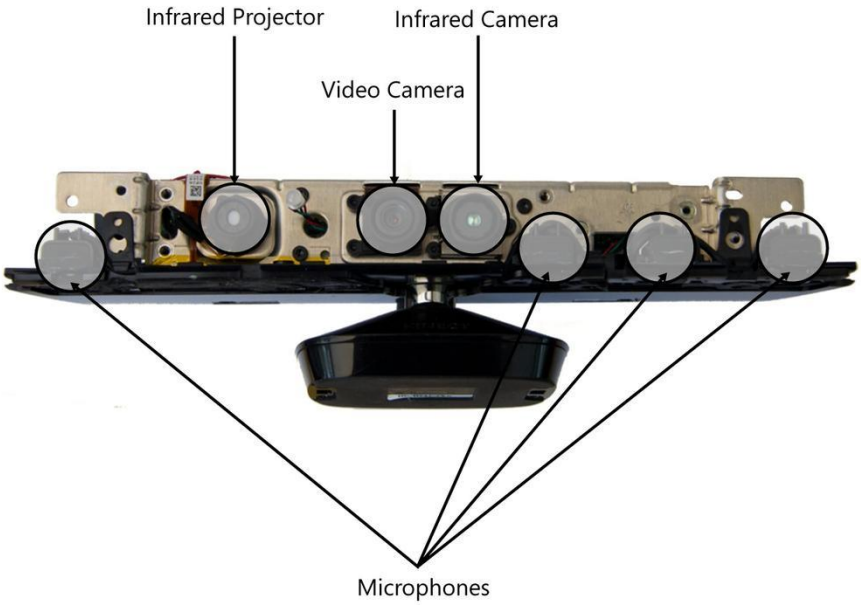

Figure 1.2: A Kinect sensor unwrapped.

The color VGA video camera detects the red, green, and blue components as well as body-type and facial features. Microsoft calls this an "RGB camera" referring to the color components it detects. The depth sensor consists of an infrared projector and a monochrome CMOS sensor with an IR filter. Working together they can see the room in 3D regardless of the lighting conditions. Both the video and depth sensor cameras have a 640 x 480-pixel resolution and run at 30 FPS.

The Kinect sensor constantly projects a pattern of infrared dots (*Figure 1.3*). It “knows” what the pattern looks like and how it is drawn. The pattern will change upon objects as the dots will change size and position based on how far the objects are. Then by analyzing the dot’s reflection (via triangulation) can calculate the distance of each point from the sensor and build a 3D map of the room (*Figures 1.4 and 1.5*).

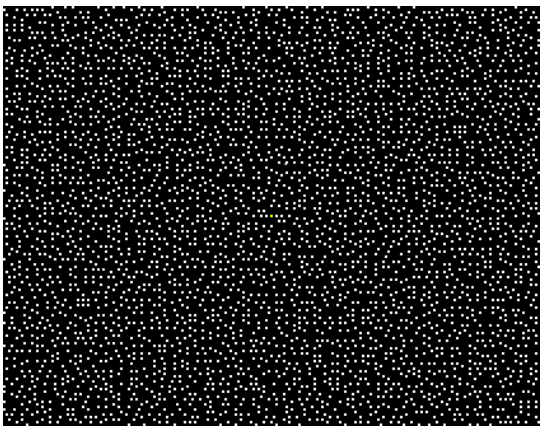

Figure 1.3: Infrared dots pattern.

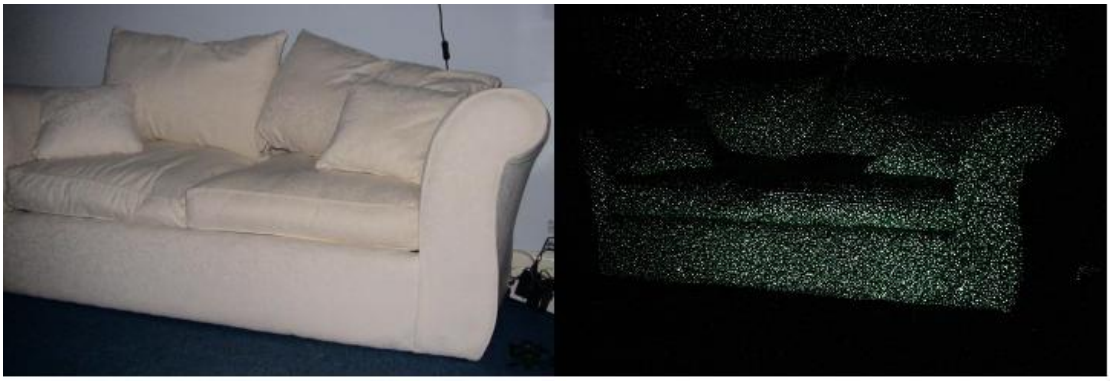

Figure 1.4: RGB image of a sofa (Left), The same image as the Kinect sensor sees it (Right).

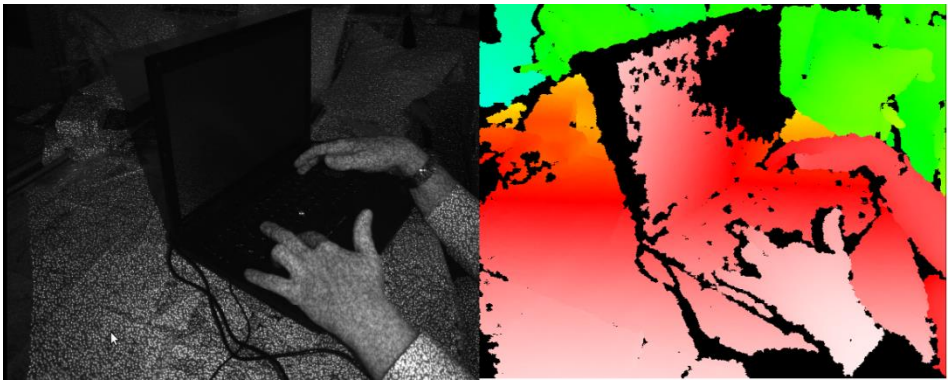

Figure 1.5: Infrared image (Left), The depth map from white "near" to blue "far" (Right).

There are also four microphones arranged along the bottom of the bar. The Kinect uses these microphones to help determine from where in a room a voice is coming. When the user speaks to the Kinect sensor, his voice will arrive at each microphone at different times, because each microphone is a slightly different distance away from the sound source. The voice waveform of the user is extracted from the sound signal produced by each microphone (using the timing information) and the location of the source in the room is calculated. The Kinect can even work out the direction from which the voice is coming for several users. It can then remove the unwanted sounds from the signal to make it easier to understand the speech content.

In (*Figure 1.6*) are displayed, the horizontal limits of the Kinect sensor from 0.8m to 4.0m with a practical range of 1.2m to 3.5m and the vertical limit of 1.8m. The horizontal field of view is about 57° and the vertical field of view is 43.5°. A tilt motor inside the Kinect sensor provides a vertical tilt range of ±27°.

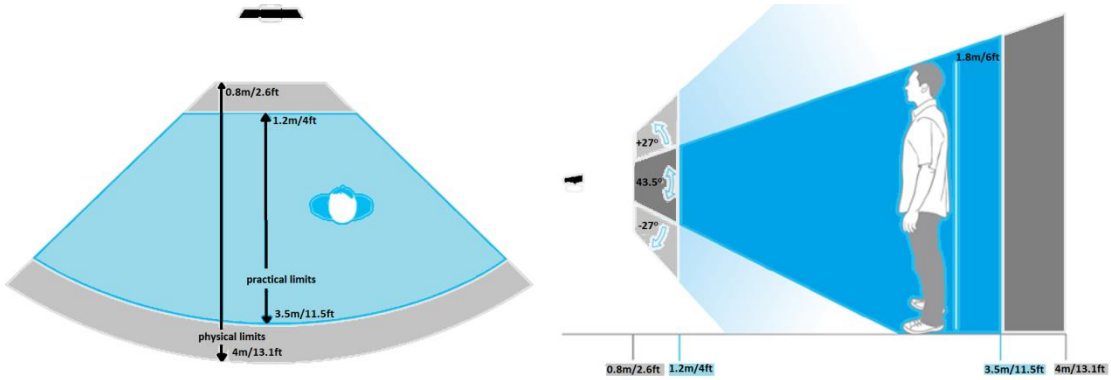

Figure 1.6: Kinect horizontal Field of View (Left), Kinect vertical Field of View (Right).

The Kinect sensor has numerous applications in many field of science. It can be used in education through active learning techniques. Healthcare projects use the sensor in physical therapy and rehabilitation (e.g. after a stroke), helping the blind to navigate (object detection) and the deaf to communicate (sign language translation). It can also be used in robotic applications, virtual reality, surveillance or as a 3D scanner.

Chapter 2 - Background

2.1. The 3D Hand Tracker

In this section, the background and operation of the 3D hand tracker used in this thesis will be presented.

2.1.1. Particle filters

Bayes filter is a general probabilistic approach for estimating an unknown probability density function (PDF) recursively over time using incoming measurements and a mathematical process model *[14]*. Particle filter is a type of Bayes filter and a sequential Monte Carlo (SMC) based technique, which models the PDF using a set of discrete points. Particle filters are useful tools for a variety of situations like tracking, signal and image processing, smart environments, machine learning, artificial intelligence, and other fields *[11][12]*.

2.1.2. The skinned hand model

The hand model is a polygon model with a kinematic skeleton based on linear blend skinning which is a well-known algorithm for direct skeletal shape deformation *[19]*. This method gives a detailed and highly accurate reproduction of an actual hand shape.

The kinematics of each finger is modeled using four parameters, two for the base angles and two for the remaining joints. Bounds on the values of these parameters are set based on anatomical studies. The global position of the hand is represented by a fixed point on the palm and the global orientation by a quaternion representation. The resulting parameterization encodes a 26-DOF hand model with a representation of 27 parameters.

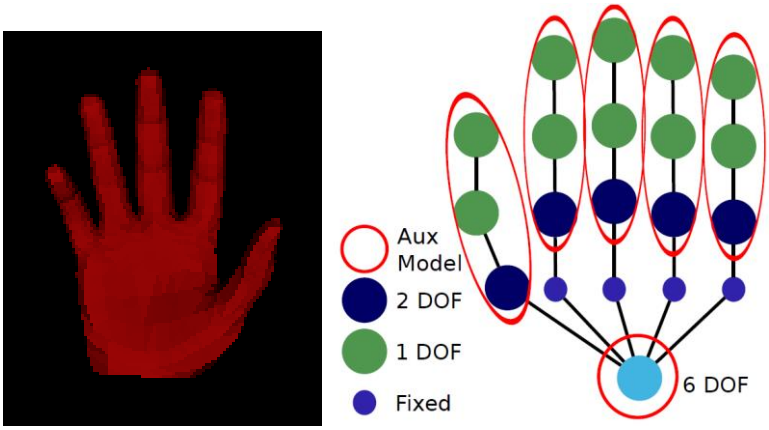

Figure 2.1: The employed 3D skinned hand model: hand polygonal surface (left) and hand kinematics (right).

2.1.3. Hierarchical Model Fusion framework (HMF)

The hand tracking problem is tackled with a model-based approach using the HMF tracking framework *[7]*. The HMF uses several auxiliary models that can provide information for the state of the main model which is to be estimated. In the hand

tracking problem, the main model is a full 26-DOF model of the hand configuration. Each of the auxiliary models tracks a distinct part of the hand; one is used for the palm with 6-DOF for its 3D position and orientation and one for each finger with 4-DOF for the joint angles. The auxiliary models that we selected have at most 6-DOF which is much less than the original 26-DOF. Of course, the main model still has 26-DOF but since it exploits the information from the auxiliary models the search in its high dimensional space is significantly narrowed.

The framework follows the Bayesian approach for tracking. The goal is to recursively estimate the unobserved state x_t (at time t) of the model given all the observations $z_{1:t}$ up to that point. This way the received data can be processed sequentially and there is no need to store the complete data set or to reprocess existing data if new observations become available. Thus, it is required to construct the posterior PDF $p(x_{0:t} \mid z_{1:t})$ at every step. It is assumed that the initial PDF (prior probability) $p(x_0 \mid z_0) = p(x_0)$. Using the Bayes rule and probability theory the solution is

$$p(x_{0:t} \mid z_{1:t}) = p(x_{0:t-1} \mid z_{1:t-1}) \frac{p(z_t \mid x_{0:t}, z_{1:t-1}) p(x_t \mid x_{0:t-1}, z_{1:t-1})}{p(z_t \mid z_{1:t-1})}$$

Two assumptions were made

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}) = p(x_t \mid x_{t-1}) \text{ and } p(z_t \mid x_{0:t}, z_{1:t-1}) = p(z_t \mid x_t)$$

and the solution becomes

$$p(x_{0:t} \mid z_{1:t}) = p(x_{0:t-1} \mid z_{1:t-1}) \frac{p(z_t \mid x_t) p(x_t \mid x_{t-1})}{p(z_t \mid z_{1:t-1})}$$

The PDF is obtained recursively in two stages: prediction and update. The update step uses the latest observations to modify the prediction PDF and is written

$$p(x_t \mid z_{1:t}) = \frac{p(z_t \mid x_t) p(x_t \mid z_{1:t-1})}{p(z_t \mid z_{1:t-1})}$$

The prediction step uses the system model to predict the state PDF forward from one measurement time to the next and is written

$$p(x_t \mid z_{1:t-1}) = \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid z_{1:t-1}) dx_{t-1}$$

The HMF uses M auxiliary models and the main model so the solution is finally expressed

$$p(x_{0:t} \mid z_{1:t}) \propto p(x_{0:t-1} \mid z_{1:t-1}) \prod_i p(z_t \mid x_{[i]t})\, p(x_{[i]t} \mid Pa(x_{[i]t}))$$

where the observation (measurement) likelihood is proportional to the product of individual model likelihoods

$$p(z_t \mid x_t) \propto \prod_i p(z_t \mid x_{[i]t})$$

and the state transition probability is decomposed as

$$p(x_t \mid x_{t-1}) \propto \prod_i p(x_{[i]t} \mid Pa(x_{[i]t}))$$

where $Pa(x_{[i]t})$ denotes the parent nodes of $x_{[i]t}$.

In such models, the PDF is difficult to be calculated. A particle filter is used to efficiently approximate the posterior. The posterior is approximated by propagating a set of N hypotheses (particles) with associated weights based on the importance sampling method. The larger the number of the weighted particles the better the representation of the PDF.

The particles are drawn from the proposal distribution

$$q(x_{0:t} \mid z_{1:t}) = p(x_{0:t-1} \mid z_{1:t-1}) \prod_i p(x_{[i]t} \mid Pa(x_{[i]t}))$$

and are weighted by

$$w_t = \frac{p(x_{0:t} \mid z_{1:t})}{q(x_{0:t} \mid z_{1:t})} = w_{t-1} \prod_i p(z_t \mid x_{[i]t})$$

The weights are normalized to sum up to one. A resampling step is added so that the high weighted particles to be chosen instead of the low weighted particles. Thus, given an input of the set of N weighted particles from the previous time step $\{x^{(n)}_{[0:M]t-1}, w^{(n)}_{t-1}\}^N_{n=1}$, the output of the algorithm is the current weighted particle set $\{x^{(n)}_{[0:M]t}, w^{(n)}_t\}^N_{n=1}$.In the end, the weighted average of the main model particles corresponds to the track estimate at each step.

2.1.4. Observation likelihood

The observation likelihood measures the degree of matching between a hypothesis and the observations. The **observations z** correspond to the RGB-D data acquired from the Kinect sensor and the **hypotheses x** (particles) correspond to 3D rendered hand model poses. Each hypothesis is a representation of 27 parameters and includes a rendered depth map like the observations. A set of pixels that corresponds to the pixels of the hand in the RGB is extracted using image segmentation techniques. The set of pixels that are labeled as foreground in both (intersection) the observation and a hypothesis is denoted as Pi and the set of pixels

that are labeled as foreground in either the observation or the hypothesis is denoted as Pu. The ratio of the number of elements of these two sets is denoted as λ:

$$\lambda = \frac{Pi}{Pu}$$

The following objective function D(z,x) is then used to evaluate the discrepancy between a hypothesis x and the observation z.

$$\mathrm{D}(\mathrm{z},\mathrm{x}) = \lambda \frac{\sum_{p \in Pi} \min(\left|z_{d,p} - x_{d,p}\right|, d_M)}{d_M \left|Pi\right|} + (1-\lambda)$$

where z_d and x_d are the observation depth map and the hypothesis depth map respectively. The result of D(z,x) ranges from 0 for a perfect match to 1 for a mismatch. The sum of the absolute depth differences is divided by the number of the pixels that intersect (Pi) and the rest of the pixels influence negatively the result. Pixels with depth differences above the threshold d_M are considered mismatch so that they don't influence an otherwise reasonable fit. They penalize D(z,x) with the maximum value the same way a pixel that doesn't belong in Pi does.

The observation likelihood function depends on the discrepancy value and is defined

$$p(\mathrm{z} \mid \mathrm{x}) = \exp\left\{\frac{-D^2(z,x)}{2\sigma_i^2}\right\}$$

2.1.5. Our contribution

The thesis contribution to the 3D hand tracker is the creation of a **lighting** objective function that will compute the discrepancy between the rendered 3D model poses and the RGB-D observations. Instead of distances we focus on the light intensities that the RGB observations involve. In other words, an alternative way of calculating the observation likelihood is proposed based on the lighting information in the acquired RGB data. An illumination model is developed to find the intensity values at each point of a rendered 3D pose. This model follows the principles of the Phong reflection model usually used in computer graphics. Then the discrepancy is evaluated in a similar way as when the default distance objective function was used. Several functions with different properties are tested.

2.2. Lighting Models

This section refers to the lighting models, how we use them and their relationship with the lighting objective functions. A brief discussion about the lighting of a scene, light-matter interactions and the most common light sources used in graphic systems is made.

2.2.1. Introduction

A realistic illuminated scene in graphic systems generally depends on two main parameters. The accuracy of the lighting system used for rendering the scene and the accuracy of the material properties of the objects in the scene.

The lighting in graphic systems is a fundamental process and aims to accurately calculate the observed brightness of a point in the scene, which is illuminated by a set of light sources. A lighting model is based on physics, where principles like the conservation of energy and optics are used to produce equations that describe how the light is reflected from surfaces [9].

2.2.2. Light and matter

Light or visible light is the portion of the electromagnetic spectrum that is visible to the human eye and is responsible for the sense of sight. The human eye and brain together translate light into color. Everything from the cloths we wear, to the pictures we paint revolves around color. The color that we see at a point on an object is the result of multiple interactions among light sources and reflective surfaces.

When light hits a surface, it may be absorbed by the material, reflected, or transmitted through it. The main physical properties of the light and matter interaction in graphic systems can be categorized into three groups: **specular surfaces**, **diffuse surfaces,** and **translucent surfaces** as show in (*Figure 2.2)*.

a) Specular surfaces are mirror like smooth surfaces where the incident light ray and the reflected light ray make the same or almost the same angle $\theta_i \approx \theta_r$ with respect to the surface normal. For this reason, specular surfaces appear shiny.

b) Diffuse surfaces are rough surfaces where the reflected light is scattered evenly in all directions. Paper, clothes, skin are all examples of rough surfaces and appear dull.

c) Translucent surfaces allow light to pass through the surface and to emerge from another location of the object like water or windows.

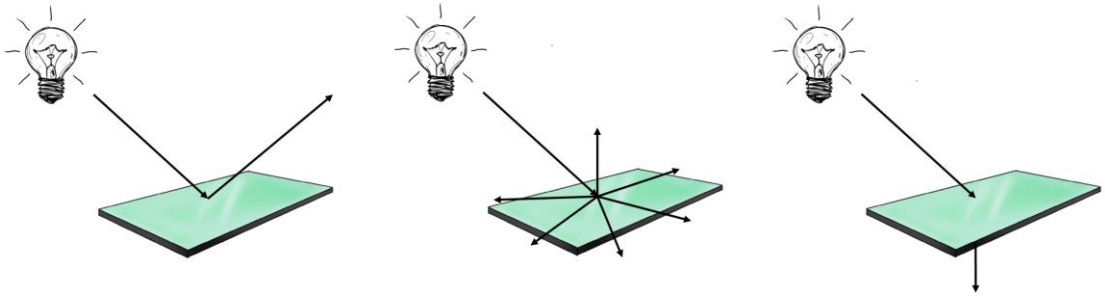

Figure 2.2: Light – material interactions: a) Specular surface. b) Diffuse surface. c) Translucent surface.

2.2.3. Light sources

Light source is any object that emits light and affects the lighting effects of the objects in the scene. A light source may be of any shape and emit light of any color. Furthermore, a light source can be an object (e.g. light bulb) that not only emits light but reflects light too, as occurs in natural environments. In practice, however, it is considered that the light source doesn't reflect the incident light to keep the geometric representation of the light sources simple and the calculation complexity low. There are five basic types of sources commonly used: color sources, ambient light, point sources, spotlights, and distant light sources.

2.2.3.1. Color Sources

Color sources usually use the additive primary colors red, green, and blue (commonly called RGB) to produce the color components that a human eye can see. The RGB color model is very common in computer graphics and each of the three colors corresponds to the intensity of red, green, and blue respectively. Light sources represented by this color model can be described by the equation

$$I = \begin{bmatrix} I_R \\ I_G \\ I_B \end{bmatrix}.$$

2.2.3.2. Ambient light

Ambient light refers to light sources that provide a uniform illumination in a scene. They could be natural sources like the sun and the moon or artificial sources used to light a room. Modeling a set of these sources and rendering a scene it's not an easy work to do. To avoid a complex illumination system, we consider a uniform lighting where every point in the scene has the same intensity value I_A. If the RGB color model is embodied to the ambient source, we can derive the equation

$$I_A = \begin{bmatrix} I_{AR} \\ I_{AG} \\ I_{AB} \end{bmatrix}.$$

2.2.3.3. Point Sources

Point sources have negligible extent in comparison with other light sources because they have the geometry of a mathematical point in the environment. They emit light evenly in all directions and if characterized by the RGB color model then

$$I_{P_0} = \begin{bmatrix} I_{P_0 R} \\ I_{P_0 G} \\ I_{P_0 B} \end{bmatrix}.$$

The intensity of a point source decreases in proportion to the inverse square of the distance from the source. Therefore, the intensity of light received at a point P from the point source P_0 (*Figure 2.3*) is given by the equation

$$I_P = \frac{1}{d^2} I_{P_0},$$

where d is the distance between the point source P_0 and a point P on a surface.

Although, in practice this distance term is usually replaced by $(k_0 + k_1 d + k_2 d^2)^{-1}$ where k_0, k_1, k_2 are constants that can be chosen to modify the attenuation of the lighting. So,

$$I_P = \frac{1}{k_0 + k_1 d + k_2 d^2} I_{P_0}.$$

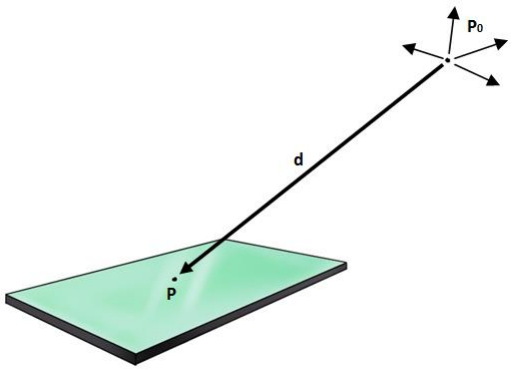

Figure 2.3: Point source illuminating a surface.

2.2.3.4. Spotlights

Spotlights are very interesting light sources that can simulate many realistic effects. If we limit the light of a point source in a narrow range of angles, a cone is formed where the light source is mounted on the top of the cone and only the points of the scene inside the angle limits are illuminated. (*Figure 2.4*) shows a cone whose top is at P_0, pointing in the direction I_s and his width depends on the value of an angle θ. A spotlight with θ = 180 it's an equivalent of a point source.

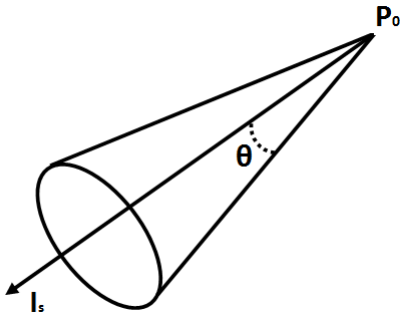

Figure 2.4: Spotlight.

A point in position P and a unit vector v_p with starting point the top of the cone P_0 and terminal point the position P forms an angle ω with a unit vector v_s of the same direction as I_s (*Figure 2.5*). To determine if the point P is illuminated from the spotlight we must compute the cosine of the angle ω between the two unit vectors with the dot product. If ω is smaller than the angle θ then it is illuminated. If not the light intensity is equal to zero. The equation that describes how spotlights illumination system works is

$$I_s = \begin{Bmatrix} I_{P_0}, if\ \omega < \theta \\ 0, if\ \omega \geq \theta \end{Bmatrix},$$

where $\omega = \cos^{-1}(\overrightarrow{v_p} \cdot \overrightarrow{v_s})$.

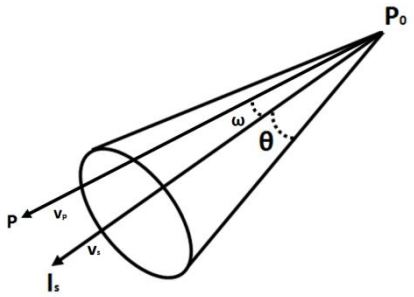

Figure 2.5: A point P inside the angle limits of a spotlight

A realistic spotlight wouldn't have the same intensity value at each point inside the cone. The cosine of an angle ϕ characterizes the light distribution of the cone with the highest intensity value being at the center of it and gradually decreasing as we move towards points with larger angles. Also, if we want the intensity to depend on the distance we can add the previous distance term. The final equation that describes the light intensity from a spotlight at a given point P as before is

$$I_s = \begin{Bmatrix} \dfrac{\cos^e \varphi}{k_0 + k_1 d + k_2 d^2} I_{P_0}, if\ \omega < \theta \\ 0, if\ \omega \geq \theta \end{Bmatrix},$$

where exponent **e** determines how rapidly the light intensity drops off (*Figure 2.6*).

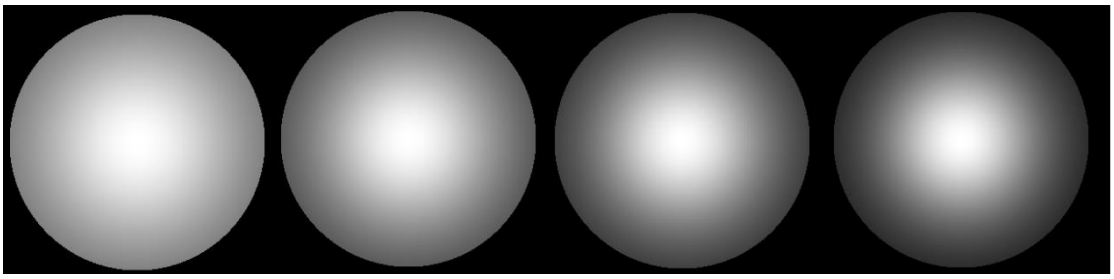

Figure 2.6: Example of a spotlight with consecutively larger value of e.

2.2.3.5. Distant light sources

Distant light sources refer to light sources that are far from the objects in the scene. The sun is a good example in cases like these where the distance between the light source and a surface is so large that the light rays from the source can be considered parallel to each other as the light rays from the sun to a finite area on the earth (*Figure 2.7*).

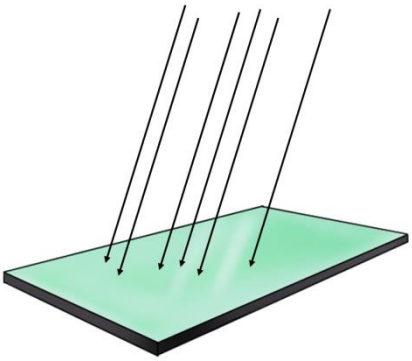

Figure 2.7: Parallel rays of a distant light source

The light source we chose to use is a point source. It is simpler than a spotlight and it can be described by a point in the 3D space. Thus, the computational cost is reduced.

2.2.4. The Phong reflection model

The Phong reflection model is an approximate local lighting model that efficiently renders the reflections in a scene with a very small computational effort. A realistic model that describes the way a surface reflects the light is very complex in graphic systems but this model has proved to be good alternative of the physical reality.

There are four vectors that Phong model uses to calculate the intensity at each point in the environment. These vectors change their directions depending on

where the light rays hit the surface and the shape of the surface. In (*Figure 2.8*) the four vectors can be shown. The vector **n** is the normal at the point P. The **l** vector is in the direction from the point P to the light source (i.e. point source). The vector **v** is in the direction from the point P to the viewer (i.e. the center of the Kinect camera). If a light ray from a point source in direction l hits a surface at a point P, the perfectly reflected result would be the vector **r**.

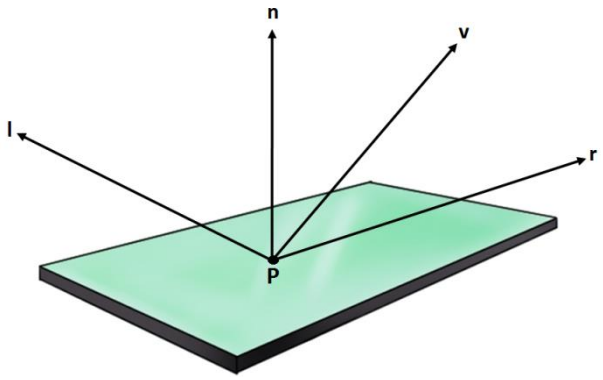

Figure 2.8: Vectors used by the Phong model.

In the Phong reflection model the illumination of an object is described defining three distinct reflection parameters: **ambient reflection**, **diffuse reflection,** and **specular reflection**.

2.2.4.1. Ambient reflection

Ambient light has the same intensity everywhere in the scene and consequently at every point on a surface. A portion of the incident ambient light $\mathbf{L_a}$ is reflected and the rest of it is absorbed. The amount reflected depends on the ambient reflection coefficient $\mathbf{k_a}$. So, the light is given by

$$I_{AR} = k_a L_a \text{, where } 0 \le k_a \le 1$$

2.2.4.2. Diffuse reflection

Diffuse reflection is the reflection of light from a surface at many angles and it depends both on the material properties and on the position of the light source relative to the surface. Diffuse reflection characterizes rough surfaces. An ideal (but extremely unlikely) diffuse reflecting surface would reflect light equally in all directions. Such a surface is called **Lambertian surface** and obeys **Lambert's cosine law**. In computer graphics, Lambertian reflection is often used as a model for diffuse reflection due to the low computational cost. The Phong reflection model assumes such surfaces when calculating the diffuse reflection.

Figure 2.9: Illumination of a diffuse surface when the light direction is vertical to the surface (right) and when there is an angle between them (left).

An illuminated diffuse surface as in (*Figure 2.9*), is brightest when there is no angle between the light rays and the surface. If there is an angle **θ** between them, the surface will appear darker because the same amount of light will be spread over a larger area. Combining the Lambert's cosine law with the unit length vectors n (surface normal) and l (direction of light from the point source) we have

$$I_{DR} = k_d L_d \cos\theta = k_d (\vec{l} \cdot \vec{n}) L_d ,$$

where $\mathbf{k_d}$ $(0 \le k_d \le 1)$ is the diffuse reflection coefficient representing the fraction of the incoming diffuse light that is reflected and $\mathbf{L_d}$ is the intensity of the diffuse component of the light source. If we add the distance term, then

$$I_{DR} = \frac{k_d (\vec{l} \cdot \vec{n})}{k_0 + k_1 d + k_2 d^2} L_d$$

2.2.4.3. Specular reflection

To have a more realistic representation of the lighting on a surface ambient and diffuse reflections are not enough. Specular reflection gives the highlights that we see on smooth and shiny surfaces when they reflect light. An ideal (mirror like) specular reflecting surface would reflect light in the same angle as the incidence angle of a light ray. However, in practice, modeling the specular reflection it's more complex because light is not scattered in a symmetric way.

The Phong reflection model assumes, like in diffuse reflection, a simplified approximate specular reflection model which is described by the following equation

$$I_{SR} = k_s L_s \cos^a \varphi = k_s (\vec{r} \cdot \vec{v})^a L_s ,$$

where **k_s** $(0 \le k_s \le 1)$ is the specular reflection coefficient representing the fraction of the incoming specular light that is reflected and **L_s** is the intensity of the specular component of the light source. As is shown in (*Figure 2.10*) , **r** is the unit vector that describes the direction of a perfect reflection, **v** is the unit vector that describes the direction of the viewer (i.e. the center of the Kinect camera) and **ϕ** is the angle between vectors r and v and describes the amount of light that the viewer sees. The exponent **a** is the highlight (shininess) coefficient that describes the extent to which the reflected light is focused in the direction of a perfect reflection.

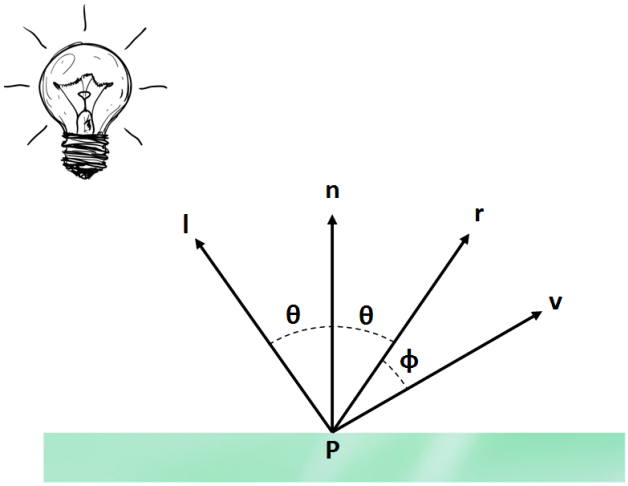

Figure 2.10: Specular reflection vectors and angles.

If we add the distance term then

$$I_{SR} = \frac{k_s(\vec{r}\cdot\vec{v})^a}{k_0 + k_1 d + k_2 d^2} L_s$$

2.2.4.4. The Phong reflection

The Phong reflection model adds all the individual components I_{AR}, I_{DR}, I_{SR} to calculate the final reflectance value

$$I_{PHONG} = \frac{1}{k_0 + k_1 d + k_2 d^2}(k_d(\vec{l}\cdot\vec{n})L_d + k_s(\vec{r}\cdot\vec{v})^a L_s) + k_a L_a$$

In case of trichromatic lighting (e.g. RGB color model), each chromatic component has its own light intensity and light coefficients apart from the k_s specular coefficient which is the same for every chromatic component. This happens because in specular reflection the reflected light is the light of the source and it doesn't depend on surface.

$$I_{PHONG,R} = \frac{1}{k_0 + k_1 d + k_2 d^2}(k_{dR}(\vec{l}\cdot\vec{n})L_{dR} + k_s(\vec{r}\cdot\vec{v})^a L_{sR}) + k_{aR}L_{aR}$$

$$I_{PHONG,G} = \frac{1}{k_0 + k_1 d + k_2 d^2}(k_{dG}(\vec{l}\cdot\vec{n})L_{dG} + k_s(\vec{r}\cdot\vec{v})^a L_{sG}) + k_{aG}L_{aG}$$

$$I_{PHONG,B} = \frac{1}{k_0 + k_1 d + k_2 d^2}(k_{dB}(\vec{l}\cdot\vec{n})L_{dB} + k_s(\vec{r}\cdot\vec{v})^a L_{sB}) + k_{aB}L_{aB}$$

2.2.5. Implementation of the Phong reflection model

The computation of the I_{PHONG} intensity premises the computation of some vectors. In this section, it will be shown how these vectors are calculated, so that they can be used in our reflection model.

2.2.5.1. Normals and positions

The rendered 3D hand models include two very important elements which also are the main inputs of our lighting model. These elements are: an array with the **normals** and an array with the **positions** in the 3D space of the rendered hand model.

In geometry, in the three dimensional case a surface normal or simply normal, to a surface at a point P is a unit vector that is perpendicular to the tangent plane to that surface at P as shown in (*Figure 2.11*).

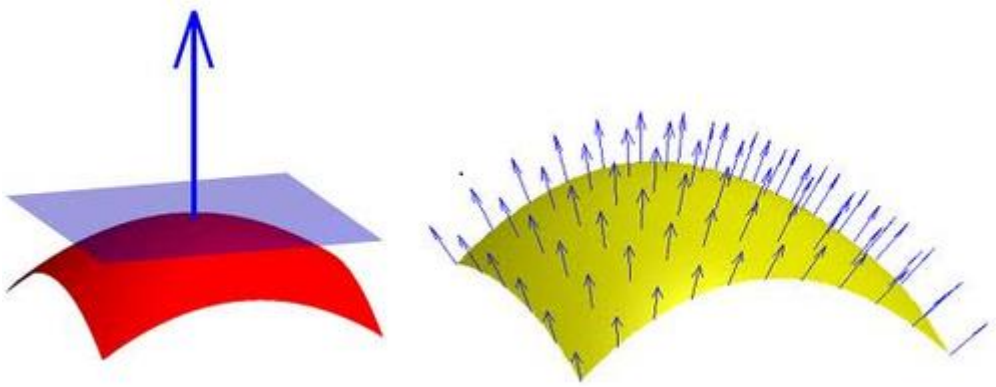

Figure 2.11: A normal to a surface at a point is the same as a normal to the tangent plane to that surface at that point.

A normal vector to a plane specified by

$$f(x, y, z) = ax + by + cz + d = 0$$

is given by

$$\vec{n} = \nabla f = \begin{vmatrix} a \\ b \\ c \end{vmatrix}$$

where ∇f denotes the gradient.

The positions refer to the position that each point of the hand model has in the 3D space relative to the center of the Kinect camera.

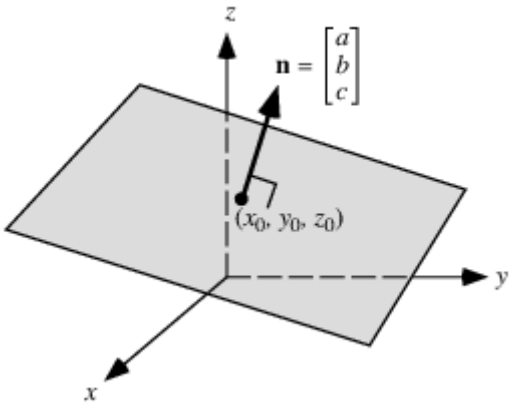

Figure 2.12: The normal vector n passing through a point at (x_0,y_0,z_0)

Thus, we know every normal at each point (position) of the rendered hand model (*Figure 2.12*). The two arrays can be seen in (*Figure 2.13*).

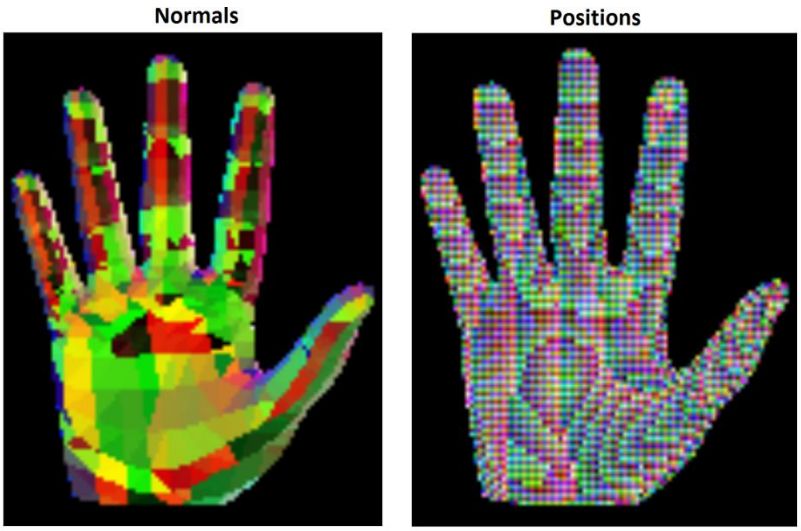

Figure 2.13: Normals (Right image) and Positions (Left image).

2.2.5.2. Lighting vectors and viewer vectors

We need two more default values that there is no need to be computed. These are the viewer's location in the 3D space which is the center of the Kinect camera, **viewer = [0,0,0]** and the light's (point) source location in the 3D space relative to the center of the Kinect camera, **light = [x,y,z].** Now, since we have the

position of every point of the hand model, it's possible to compute the **lighting vectors** for every illuminated point of the hand model.

$$\vec{l} = light(x, y, z) - positions(x_i, y_i, z_i)$$

Likewise, we compute the **viewer vectors** for every illuminated point of the hand model.

$$\vec{v} = viewer(0, 0, 0) - positions(x_i, y_i, z_i)$$

2.2.5.3. Reflection vectors

We assume that the angle of incidence equals the angle of reflection as in (*Figure 2.14*). Thus,

$$\cos\theta_i = \vec{l} \cdot \vec{n} = \cos\theta_r = \vec{n} \cdot \vec{r}$$

We can write **r** as a linear combination of **l** and **n** using the coplanar condition

$$\vec{r} = \alpha\vec{l} + \beta\vec{n}$$

From the dot product with **n**, we have

$$\vec{n} \cdot \vec{r} = \alpha\vec{l} \cdot \vec{n} + \beta = \vec{l} \cdot \vec{n}$$

If all vectors are unit vectors, we get a second condition

$$1 = \vec{r} \cdot \vec{r} = \alpha^2 + 2\alpha\beta\vec{l} \cdot \vec{n} + \beta^2$$

Solving the two equations we get

$$\vec{r} = 2(\vec{l} \cdot \vec{n})\vec{n} - \vec{l}$$

The final equation gives us the reflection vectors for every illuminated point of the hand model.

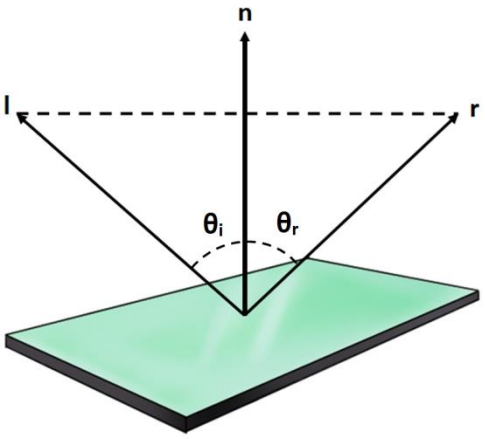

Figure 2.14: Specular reflection vectors.

2.2.5.4. Diffuse reflection θ angles and specular reflection φ angles

All the necessary vectors are computed so now what's left is the computation of the angles from which depends the final intensity value I_{PHONG}. The best way to find these angles is to use the theory of the dot product for two vectors in the 3D space. So, the two equations that have as a result the **θ** angles needed to compute the diffuse reflection term and the **ϕ** angles needed to compute the specular reflection term are

$$\theta = \cos^{-1}\left(\frac{\vec{l}\cdot\vec{n}}{|l||n|}\right) = \cos^{-1}(\vec{l}\cdot\vec{n}) \text{ and } \varphi = \cos^{-1}\left(\frac{\vec{r}\cdot\vec{v}}{|r||v|}\right) = \cos^{-1}(\vec{r}\cdot\vec{v})$$

A brief algorithm that shows the intensity value computation of the Phong reflection model is shown below. The input data *h* and *w* refer to the height and width of the image respectively. Also, the vectors that are not unit vectors are converted to unit vectors.

$$|n| = |l| = |r| = |v| = 1$$

Algorithm 1: Brief Phong reflection model algorithm.

Input Data: $\{$normals,positions,h,w$\}$

Light source coords (x, y, z)

1. *Diffuse reflection*

Light vectors $\vec{l}$ calculation.

θ angles between light vectors and normals using the dot product.

2. *Specular reflection*

Reflection vectors $\vec{r}$ calculation.

Viewer coords (0,0,0) and viewer vectors $\vec{v}$ calculations.

ϕ angles between reflection vectors and viewer vectors using the dot product.

3. *Phong reflection*

Point source intensity $P_0 = 1$ in order to the final values be in a range of 0 to 1.

Ambient reflection term I_{AR}.

Radial intensity attenduation term RIA or the distance term

Diffuse reflection term I_{DR}.

Specular reflection term I_{SR}.

$I_{PHONG} = I_{AR} + I_{DR} + I_{SR}$.

Output Data: $\{I_{PHONG}\}$

2.2.6. Experiments and results

To verify the efficiency of the Phong reflection model algorithm various tests have been made.

We show again the equation of the Phong reflection model.

$$I_{PHONG} = \frac{1}{k_0 + k_1 d + k_2 d^2}(k_d(\vec{l}\cdot\vec{n})L_d + k_s(\vec{r}\cdot\vec{v})^a L_s) + k_a L_a$$

2.2.6.1. Reflection coefficients

In (*Figure 2.15*) each image shows the result of the hand model using one kind of reflection. From left to right we have the results of the ambient reflection, the diffuse reflection, and the specular reflection respectively. For example, we set

$$k_d = 1, k_a = k_s = 0$$

if we want only to the diffuse reflection participate. The distance term coefficients are set

$$k_0 = 1, k_1 = k_2 = 0$$

in order, not to influence the result at this point.

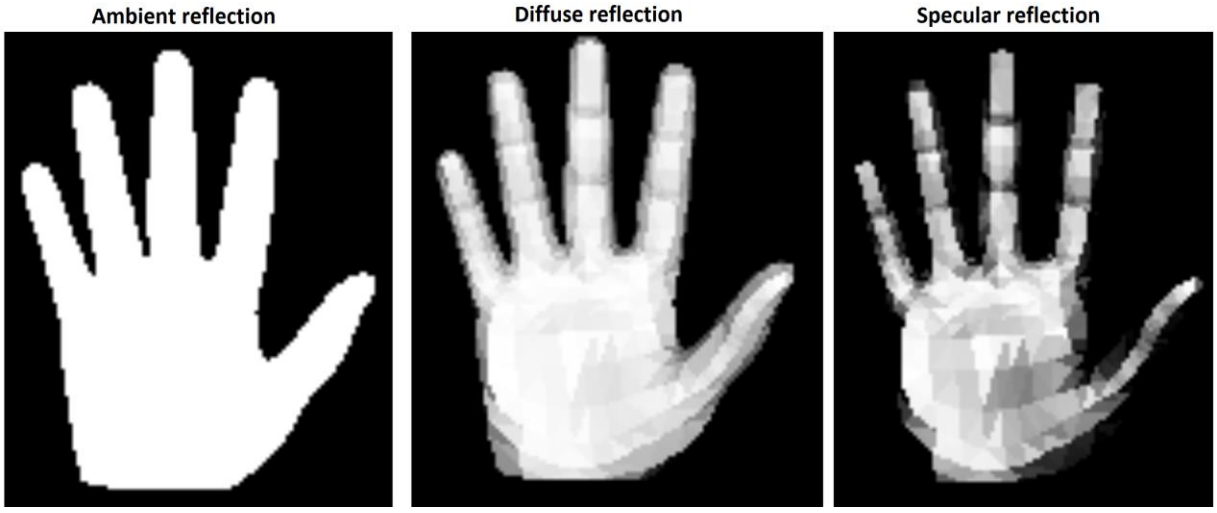

Figure 2.15: Ambient reflection (left), diffuse reflection (middle) and specular reflection (right).

A combination of the three reflection coefficients is shown in (*Figure 2.16*) and (*Figure 2.17*). In (*Figure 2.16*) as the ambient reflection coefficient gets larger the average intensity gets larger and the hand model seems to be duller.

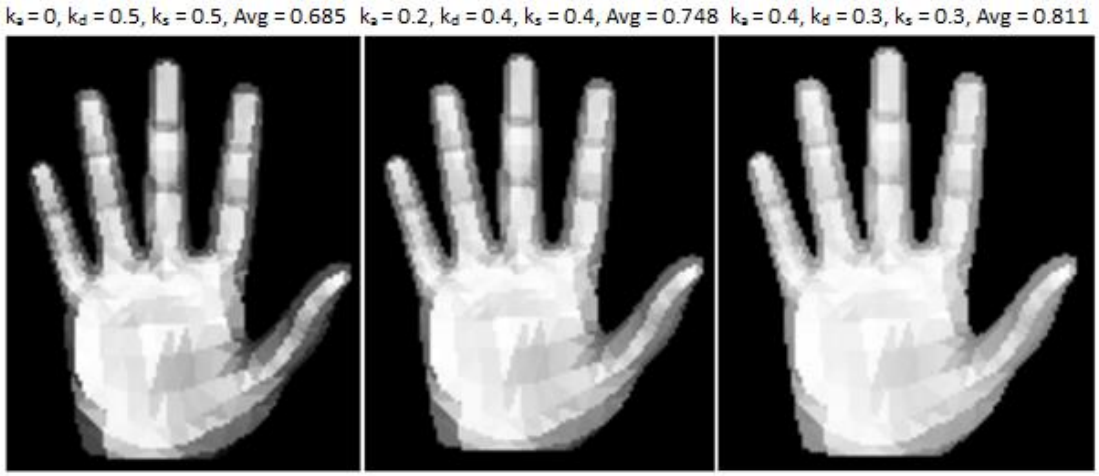

Figure 2.16: The effect of the ambient reflection.

In (*Figure 2.17*), when the amount of the diffuse reflection is larger than the amount of the specular reflection the lighting seems to be smoother. If the opposite happens characteristics of the hand model become sharper and the contrast is stronger.

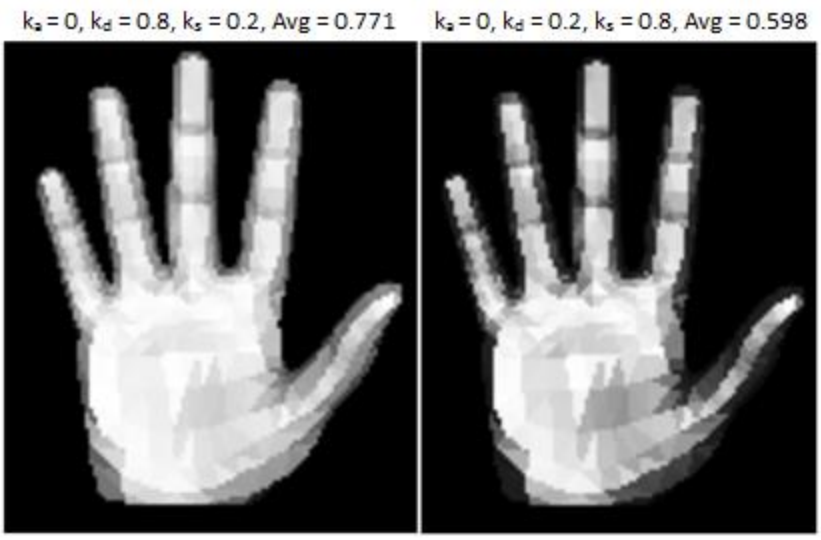

Figure 2.17: The effect of the diffuse and specular reflections.

2.2.6.2. Distance term

The distance term or the radial intensity attenuation term $(k_0 + k_1 d + k_2 d^2)^{-1}$ is responsible for the intensity reduction while the light travels from the point source to a point of the hand model. The three distance constants $\mathbf{k_0, k_1, k_2}$ determine how rapidly the light's intensity will drop off. Although, in (*Figure 2.18*) for different constant values the average intensity has a tremendous difference but there is no change in the distribution of light on the hand model.

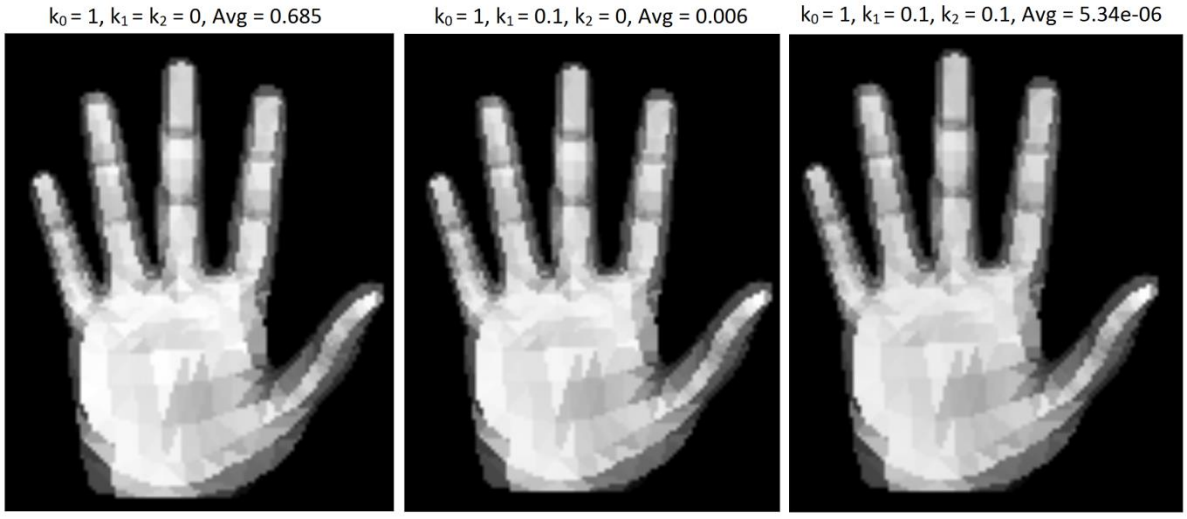

Figure 2.18: The effect of the distance term.

In other chapters, the illumination of the hand model will be compared to the illumination of the hand in real images.

2.2.6.3. Shininess coefficient

The shininess coefficient **a** describes the extent to which the reflected light is focused in the direction of a perfect reflection. It is clear (*Figure 2.19*) that the value of this coefficient affects the distribution of light on the hand model as well as the average intensity value. The default value where there is no effect is a = 1. If this value becomes smaller than the default value, the reflected light is spread and the average intensity value is higher. On the other hand, if the value becomes larger the reflected light is more focused and the average intensity value is lower.

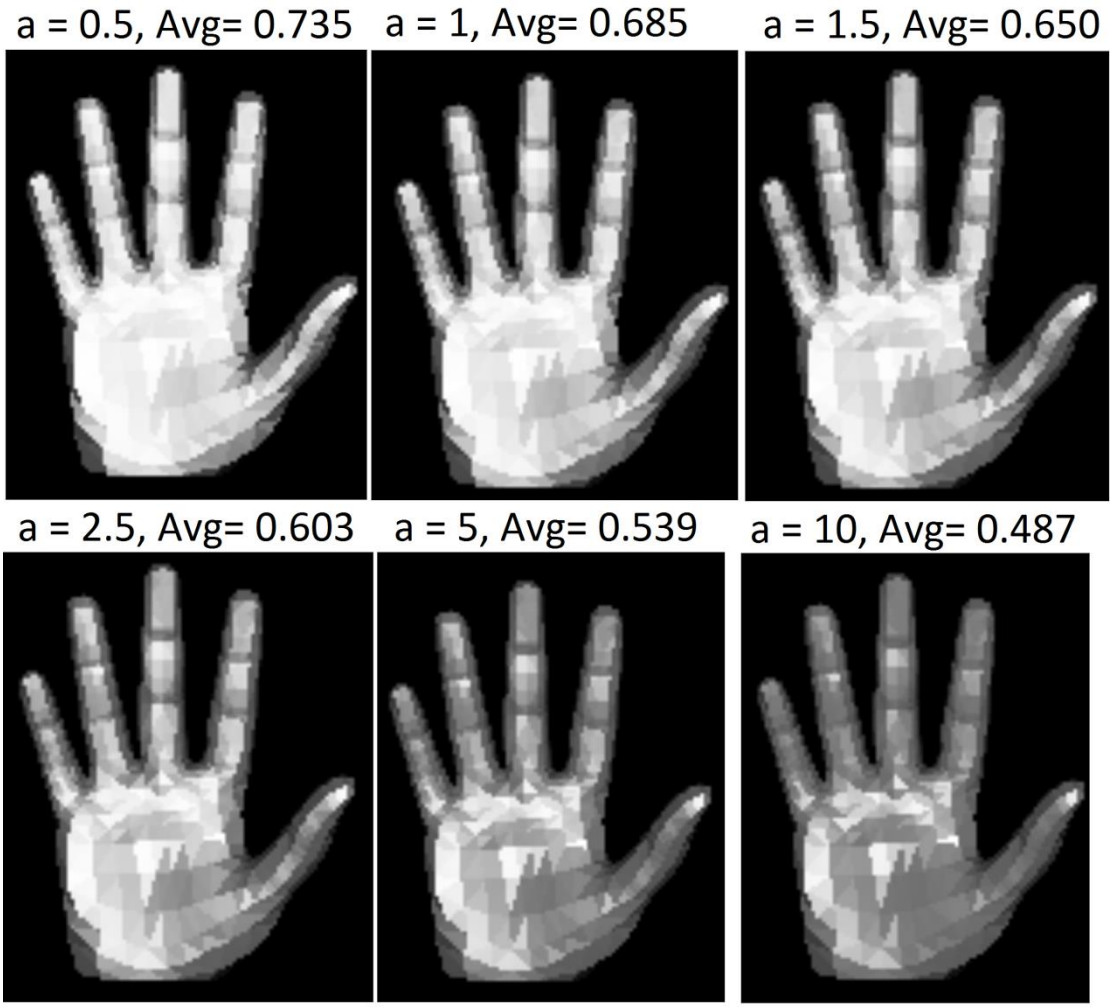

Figure 2.19: The effect of the shininess coefficient.

2.2.6.4. Moving light's position

The values of the diffuse and specular reflection coefficients as well as the shininess coefficient were arbitrary selected to be $k_d = k_s = 0.5$, $a = 3$. In the experiments of this section the illumination of a hand model will be tested while the light's position is changing in the 3D space.

The original size of the images is: height = 480 and width = 640 but the images we provide are zoomed to be able to see the details clearer. The initial image can be seen in (*Figure 2.20*) and the red dot represents the initial position of the light (point) source. The red dot is placed there to aesthetically understand the light's position. In the 3D space, this red dot has the coordinates of the center of the Kinect camera (0,0,0) (*Figure 2.21*) where we arbitrary selected to place the modeled point source. Also, the hand model's coordinates are approximately 900mm in front of the point source. All the distance values are measured in millimeters.

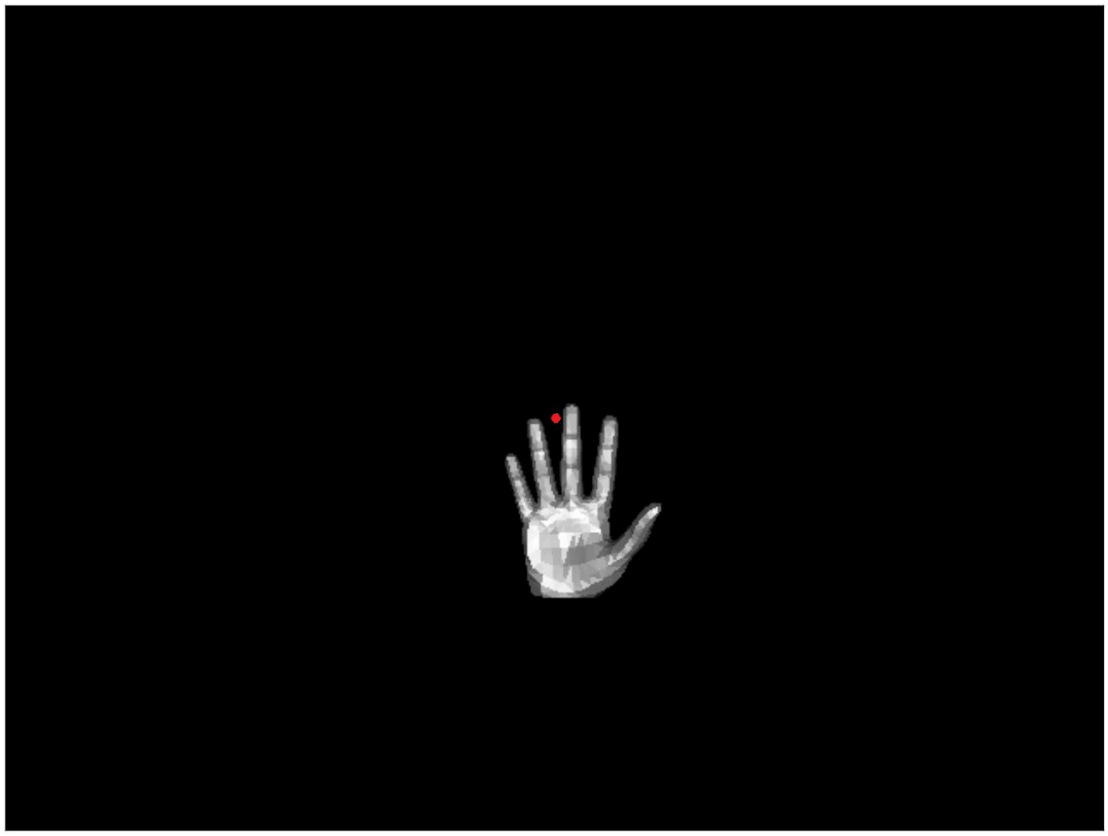

Figure 2.20: Original size of the images (height = 480, width = 640) and the light's initial position (red dot).

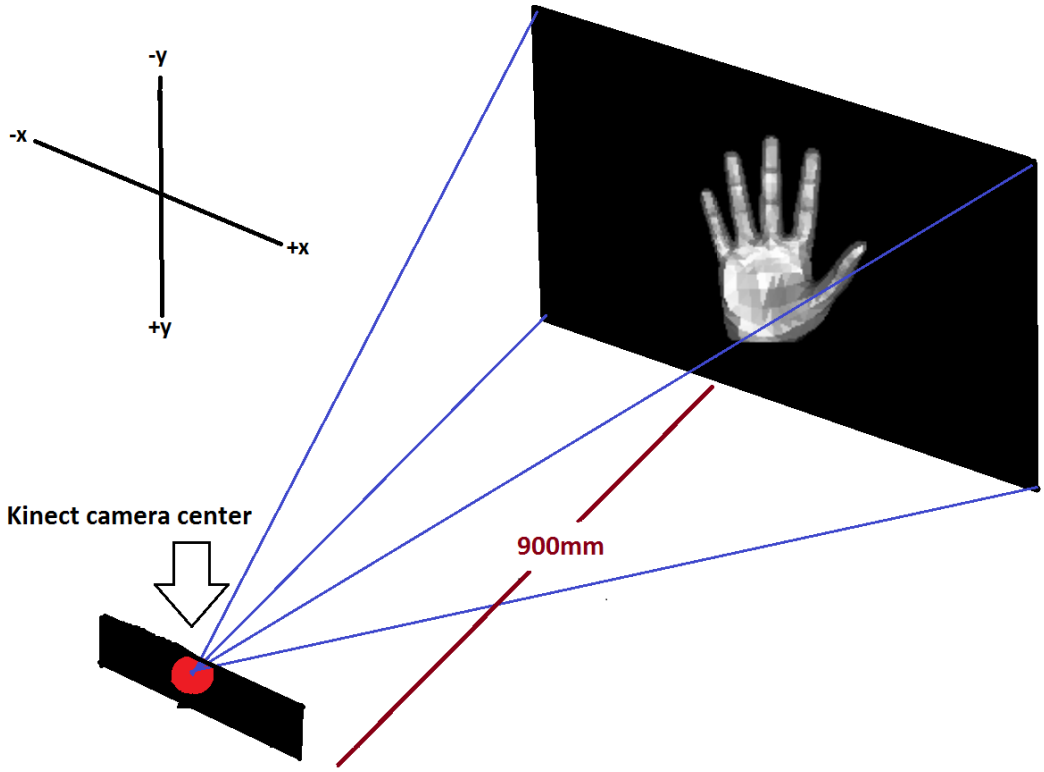

Figure 2.21: Realistic representation of the Kinect camera center and the hand model.

The point source (red dot) is moving from x = -2400mm to x = +2400mm only on the x – axis and from y = -2400mm to y = +2400mm only on the y – axis to take the results shown in (*Figure 2.22*) and (*Figure 2.23*). When x takes negative values (moving left) the dark points on the fingers are formed at the right side of the hand model while the left side appears brighter and when x takes positive values (moving right) the dark points on the fingers are formed at the left side of the hand model while the right side appears brighter.

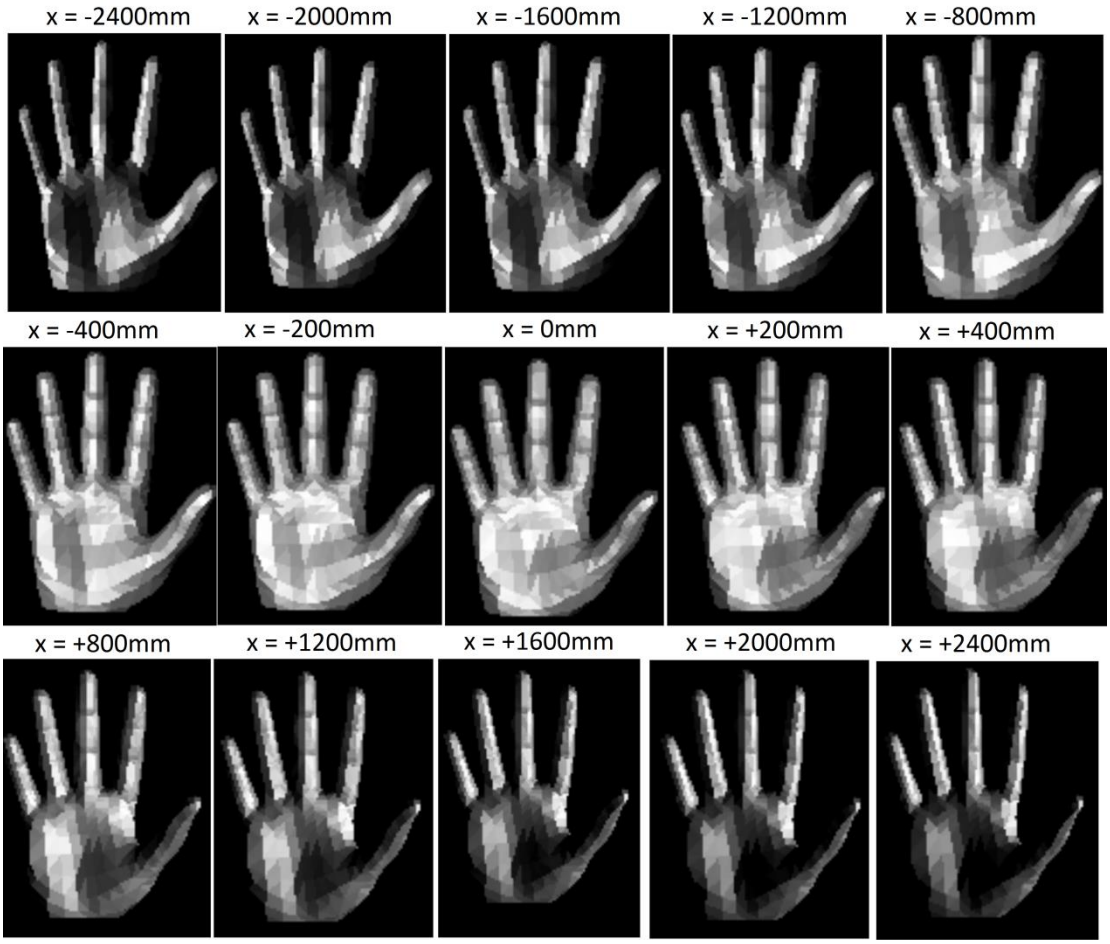

Figure 2.22: Light source: X – axis movement.

In a similar way, when y takes negative values (moving up) we can see that the dark points on the fingers are formed at the bottom of the hand model, the inside of the palm is also darker and the tips of the fingers appear brighter. When y takes positive values (moving down) the bottom of the hand model appears much brighter than the inside of the palm. In both cases, it is very clear how much brighter the palm seems to be when the light source forms small angles relative to the hand model.

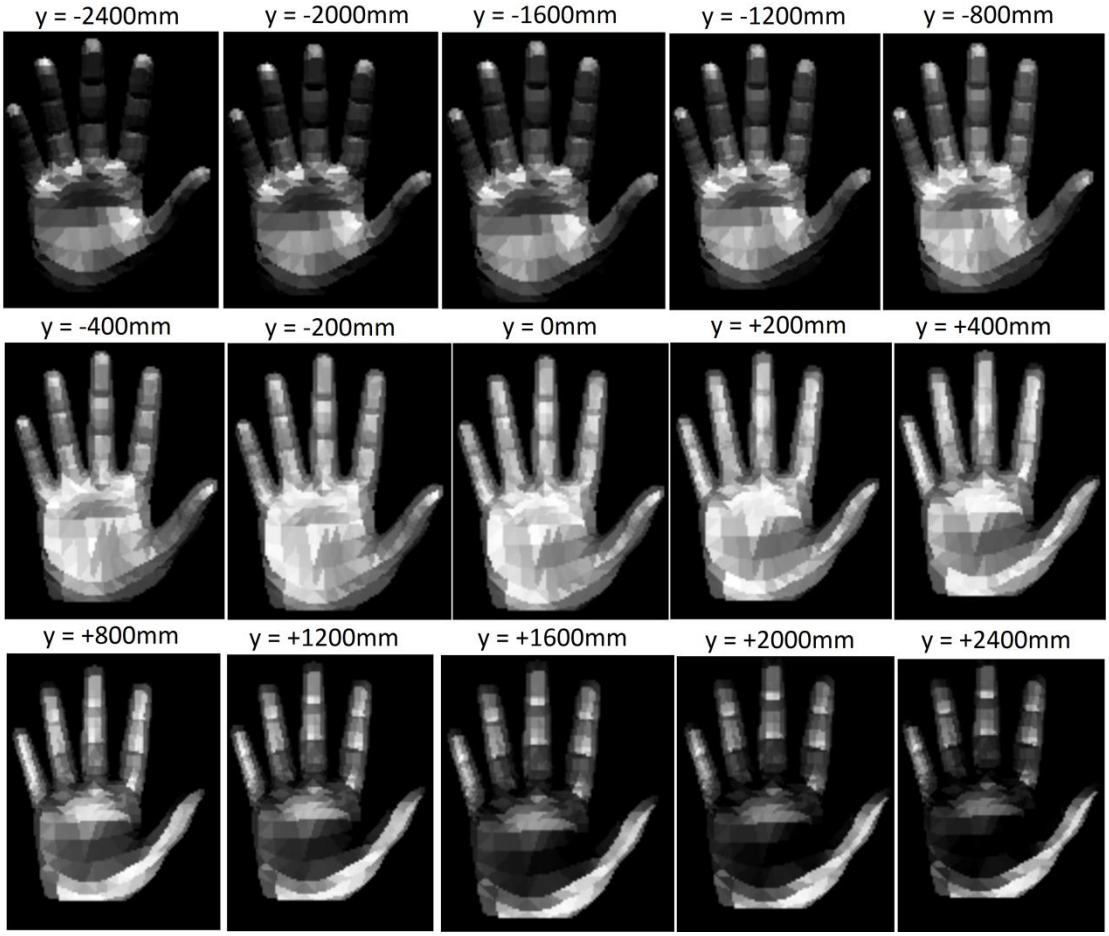

Figure 2.23: Light source: X – axis movement.

An interesting case is when the point source is moving forward to the hand model. (*Figure 2.24*) shows that at some point the illumination on the hand model suddenly drops off. What happens is that the light source is very close to the z value of +900mm, overtakes the hand model and then is behind of it.

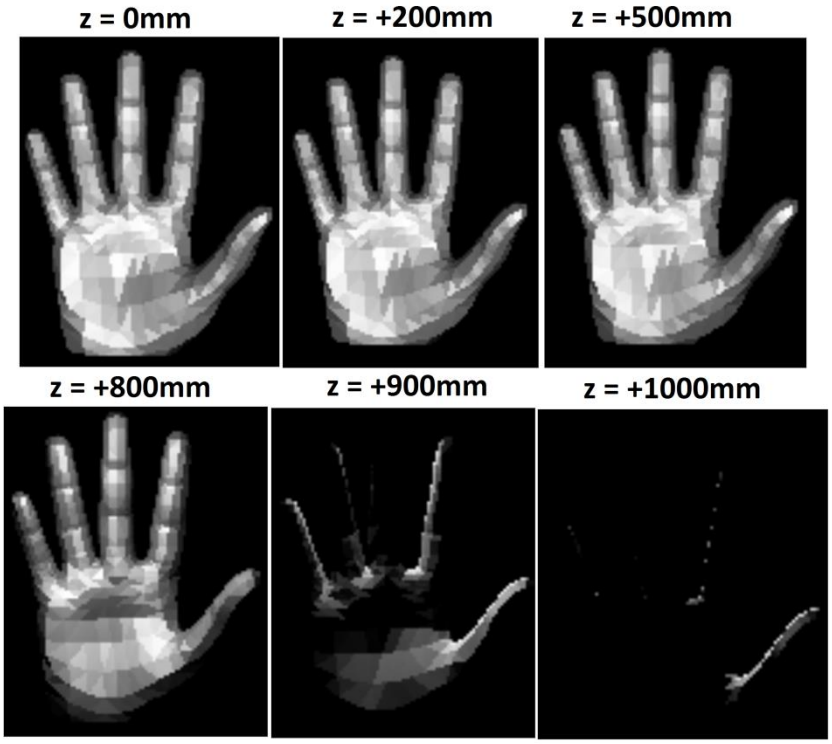

Figure 2.24: Light source: Z – axis movement.

2.2.6.5. Moving hand's model position

In the experiments of this section the illumination of a hand model will be tested while the hand's model position is changing relative to the light's position which is fixed at (0,0,0).

In (*Figure 2.25*) the fingers of the hand model have been bended. The darker spots are formed at the points where the fingers are bended because the angles relative to the point source are bigger than before. The tips of the fingers are highlighted as they tend to come in the same direction with the light rays. Since the fingers are bended enough, the highlights appear on the top of them.

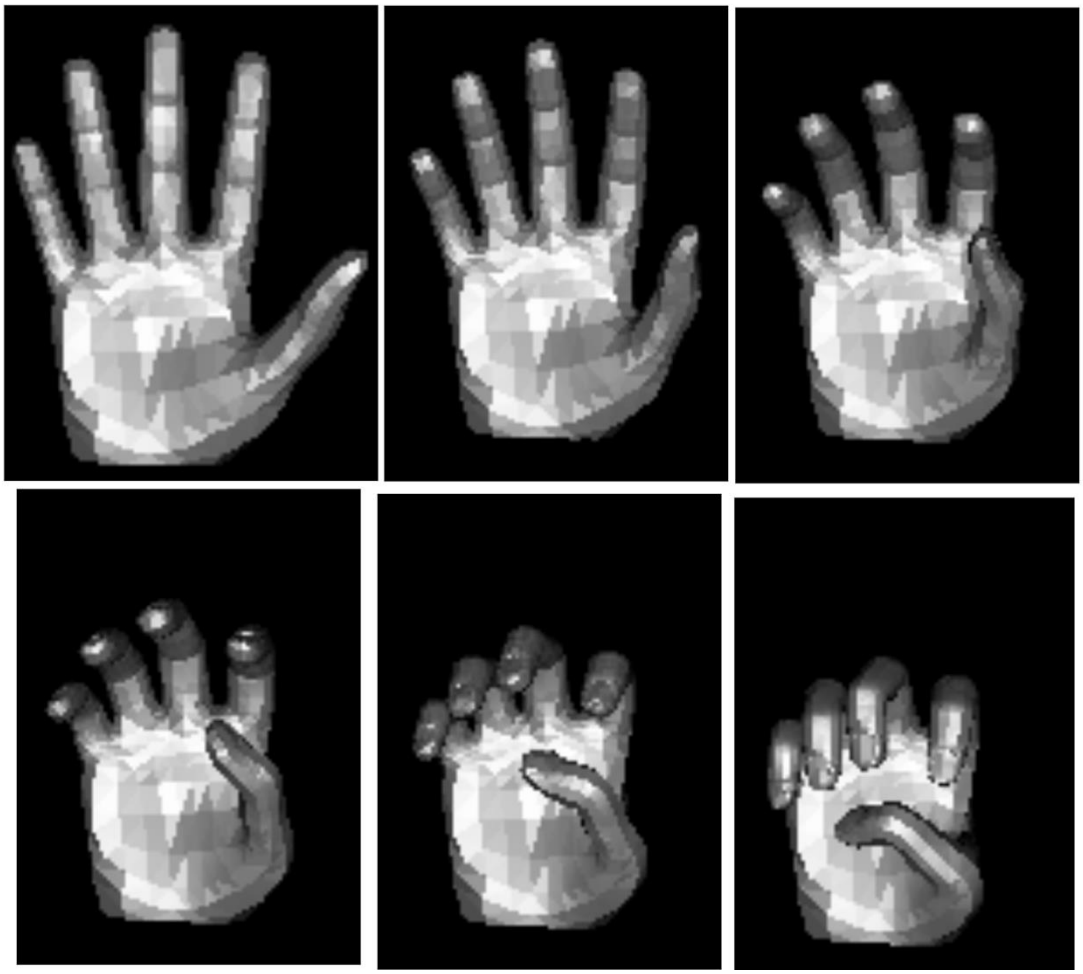

Figure 2.25: Finger bending.

The hand model is rotated on the X – axis and Y – axis to get the results in (*Figure 2.26*) and (*Figure 2.7* respectively. The points of the hand model that correspond to large θ angles appear darker than the others. The highlights are also clear where ϕ angle is small.

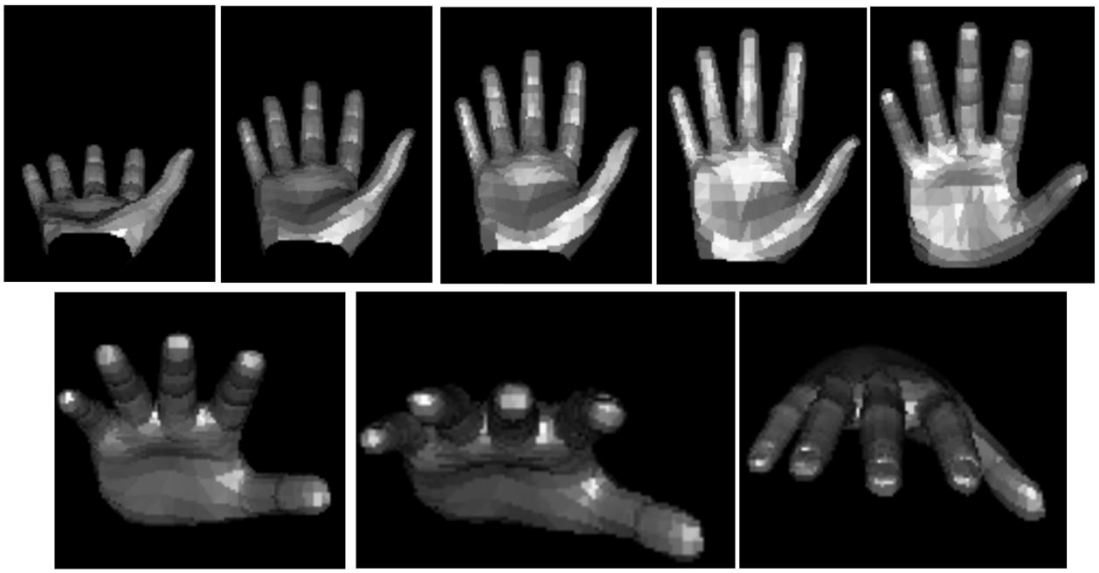

Figure 2.26: X – axis rotation.

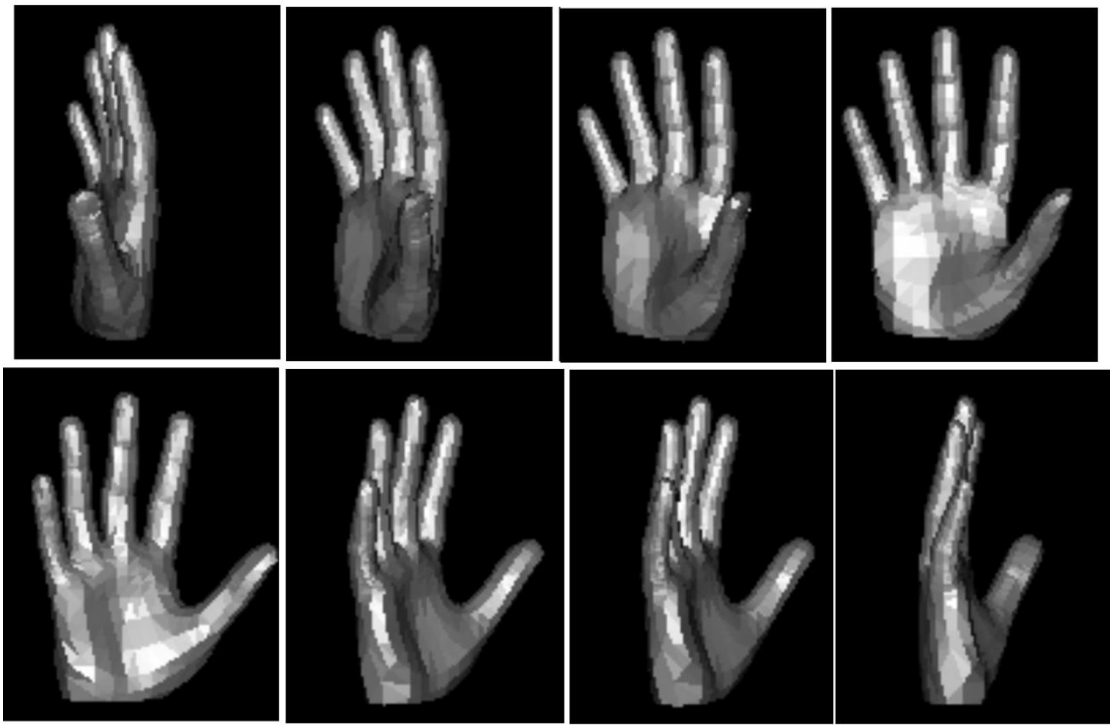

Figure 2.27: Y – axis rotation.

The Phong reflection model which we developed has the desirable illumination results but its application has yet to be tested.

The next section refers to the objective functions used to compare the illumination of the rendered 3D hand model with the illumination of the RGB observation and give as a result value of discrepancy (or dissimilarity) V(O,H) between them.

2.3. Objective functions

In the previous section, an algorithm was developed that describes the illumination of a rendered 3D hand pose. The information of the **normals** and the **positions** in the 3D space are used to find the intensity value of every point of the rendered 3D hand pose, based on the theory of the Phong reflection model which is usually used in computer graphics.

During the hand tracking process, visual RGB-D data are acquired from the Kinect sensor. Each frame includes an RGB image and a Depth image. By extension, it includes the information of the intensity and depth values in these images and we refer to them as **observations.** Several rendered 3D hand model instances (represented as a vector of 27 parameters), that we call **hypotheses,** are generated from the 3D hand tracker and the best hypothesis that matches the observation's hand must be selected. Several objective functions were developed to compare the intensities between each hypothesis and the observation and compute their discrepancy value V(O,H) which ranges from 0 (for a perfect match) to 1 (for a mismatch).

2.3.1. Introduction

The observations are acquired from the Kinect camera, the RGB image is converted to GRAY and the foreground (pixels of the hand) is extracted using an object detection technique. The intensity values of the observation are normalized to the same value range with the hypothesis. That would be zero for the lowest intensity value and one for highest intensity value.

Several hypotheses are generated to be compared with the observation. An objective function evaluates their discrepancy value V(O,H) and the hypothesis with the lowest discrepancy value should be the one which matches better the observation. We developed several objective functions, each one with different parameters that will be described in the following sections.

2.3.2. RGB-D visual data

The Kinect sensor provides us with the RGB and Depth images as seen in (*Figure 2.28*). The only light source in the environment is a light bulb placed next to the Kinect sensor.

Figure 2.28: RGB image (Left), Depth image (Right).

The exact point of the light bulb for this image is (+200mm,+150mm,0mm) relative to the center of the Kinect sensor (*Figure 2.29*). When we simulate the illumination on the hypothesis we set the point source at the same position as in reality's light bulb.

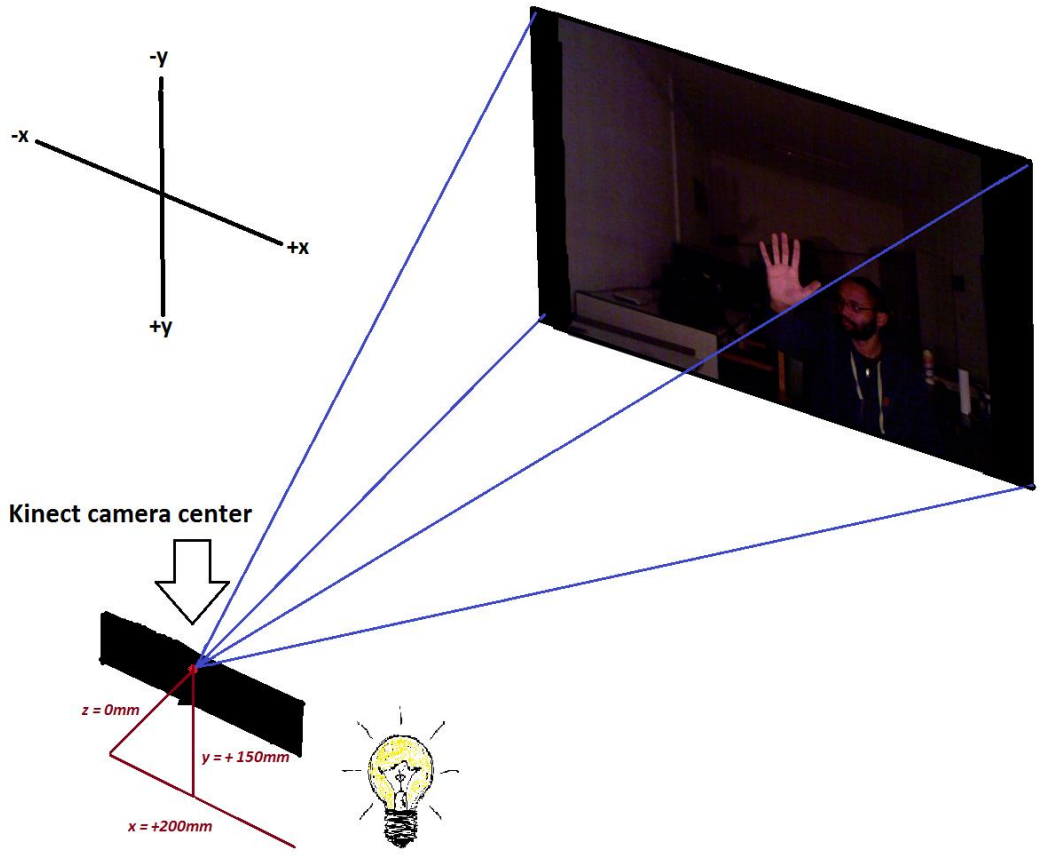

Figure 2.29: Realistic representation of the Kinect camera center and the RGB image.

2.3.2.1. RGB to GRAY

The RGB image is a 3-channel image. That means that at each point we know the intensities of Red, Green, and Blue. An illuminated hypothesis is displayed as a 1-

channel image so the RGB images need to be converted to GRAY, 1-channel images so that each observation point corresponds to one intensity value (*Figure 2.30*).

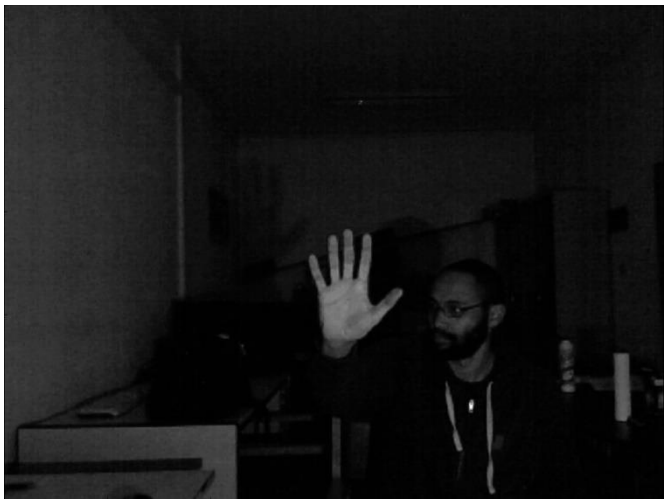

Figure 2.30: GRAY image.

2.3.2.2. Foreground extraction (or background subtraction)

An object detection technique is used to extract the foreground information. The RGB visual data are used only. Our goal here is to detect the pixels that correspond to the human hand.

The human skin has much amount of redness that's why we chose the red channel from the 3-channel RGB image to be used for the foreground extraction. In (*Figure 2.31*) the Red channel is visualized as a RGB image where bluer pixels denote that there isn't much redness at those points and redder pixels denote that there is high amount of redness at those points. Also, points that are closer to the Kinect sensor may appear even redder.

Figure 2.31: Red – channel.

Since our hand appeared in the observations is one of the closest objects to the Kinect sensor and skinned colored, it appears reddish. An **intensity threshold** is used below which every pixel's intensity is changed to zero. The remaining pixels with intensity above this threshold are labeled with a non-zero value and zero values are considered background. The objects in the image are the connected components of this image and the 4-Connectivity structure is the structure that we use to identify them. All the pixels of an identified object are labeled with the same non-zero value but every object is denoted with a different non-zero value.

The image below explains the thought. The pixels with intensity below the given threshold are considered background and they are labeled with a zero value. The pixels with intensity above the given threshold are considered foreground. The pixels of every different object have the same non-zero value. The yellow object is composed from two different objects which are connected (4-Connectivity structure) so the pixels of both objects have the same non-zero value.

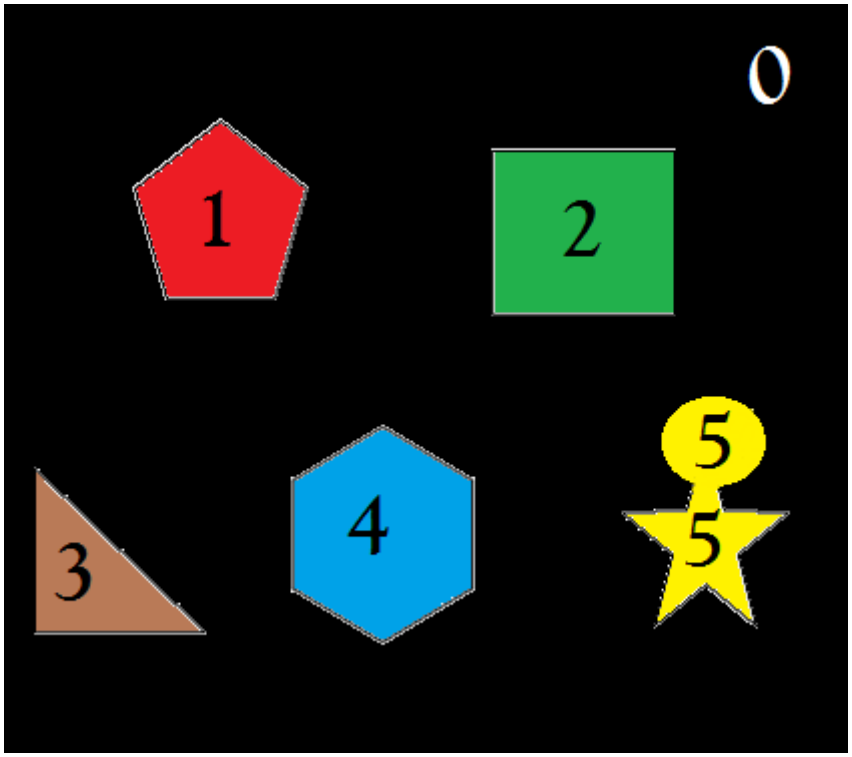

Figure 2.32:Object detection.

The object with the highest probability to consist of the largest number of pixels is the human hand. So, we identify the hand in the image by selecting the labeled object with the largest number of pixels.

The result it can be seen in the image below.

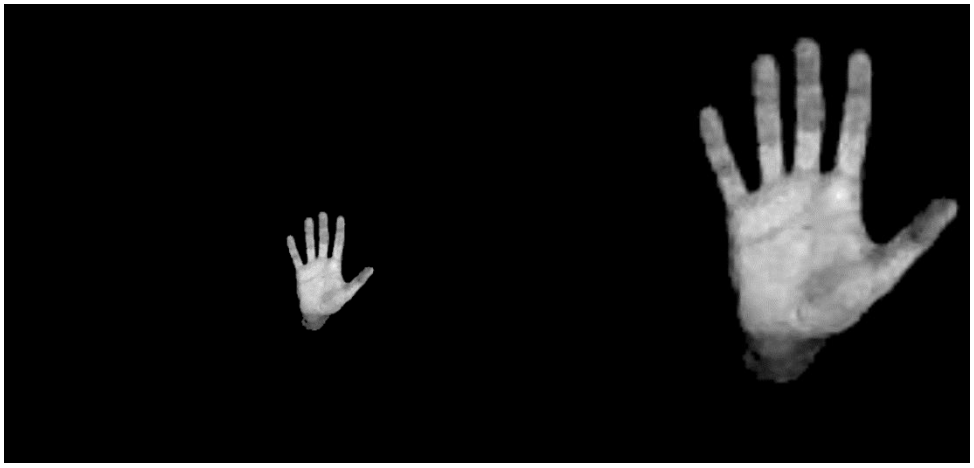

Figure 2.33: Result after object detection: original (Left), zoomed (Right).

This foreground extraction method in many cases overweighs previous used extraction methods based on the depth (distances) information like the default extraction method of the 3D hand tracker. The results of both the extraction methods are shown in the images below.

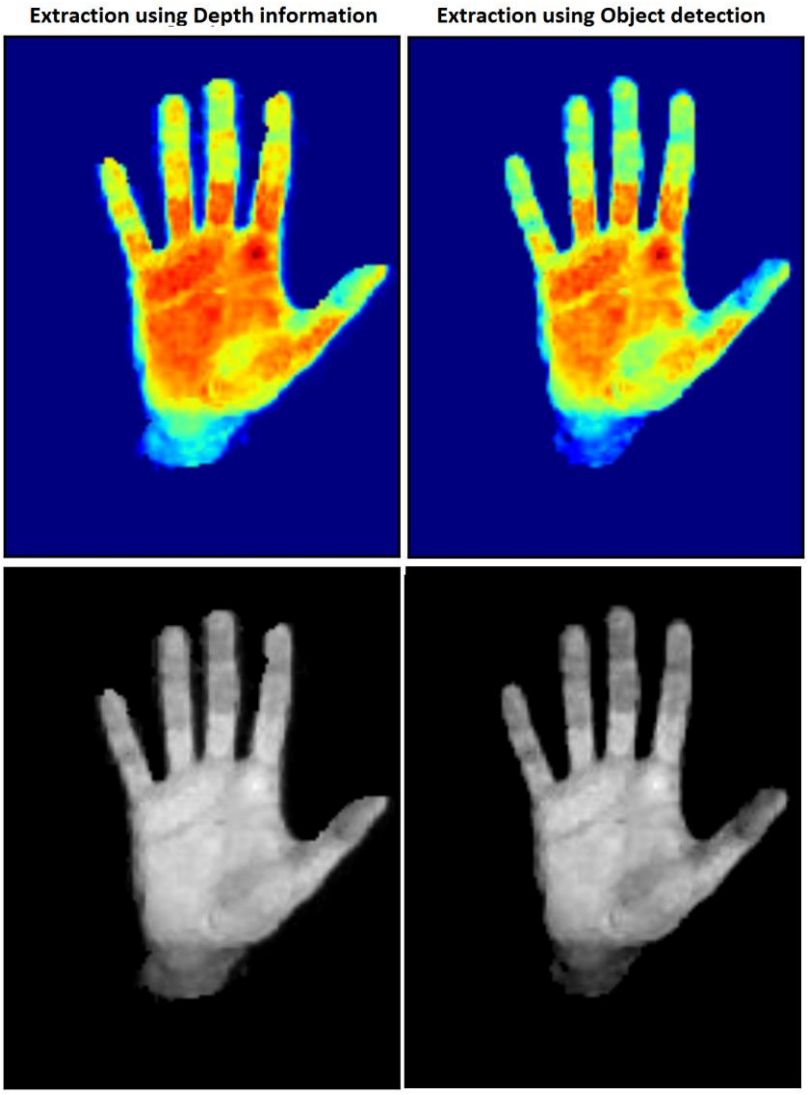

Figure 2.34: Extraction using the Depth information (Left), extraction using object detection (right).

Extraction using this proposed object detection technique was proved to be less noisy than the extraction method based on depths.

Even though the extraction using the depth information gives noisier results, there is another reason that justifies our choice to use the proposed object detection technique. Due to limitations that the Kinect sensor has, it is possible in many occasions to lose important depth information that corresponds to an inevitably loss of useful RGB information during the foreground extraction.

To be more specific, in a case like the one in (*Figure 2.35*), the Kinect sensor is unable to record the necessary depth information that we need to extract the hand pixels from RGB image. That leaves us with a result like the left image of the (*Figure 2.36*) where some points of the hand are missing. This doesn't happen when the proposed object detection technique is used like in the right image of the (*Figure 2.36*).

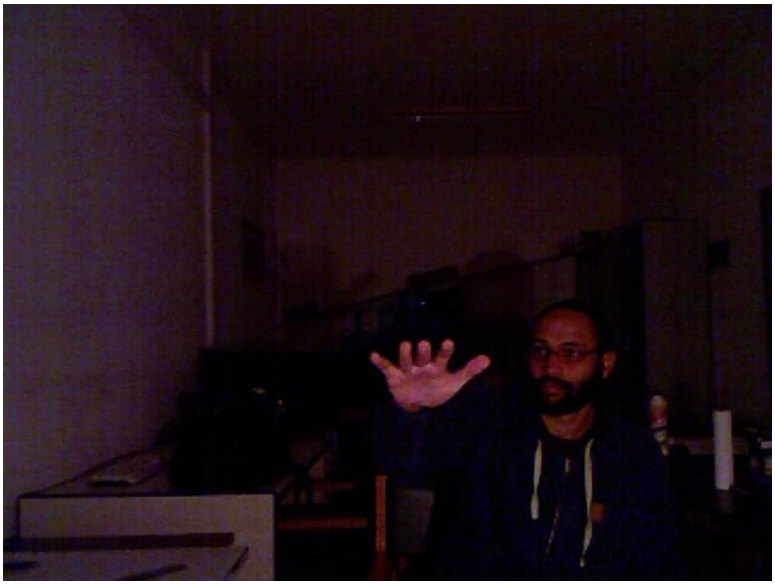

Figure 2.35: RGB image.

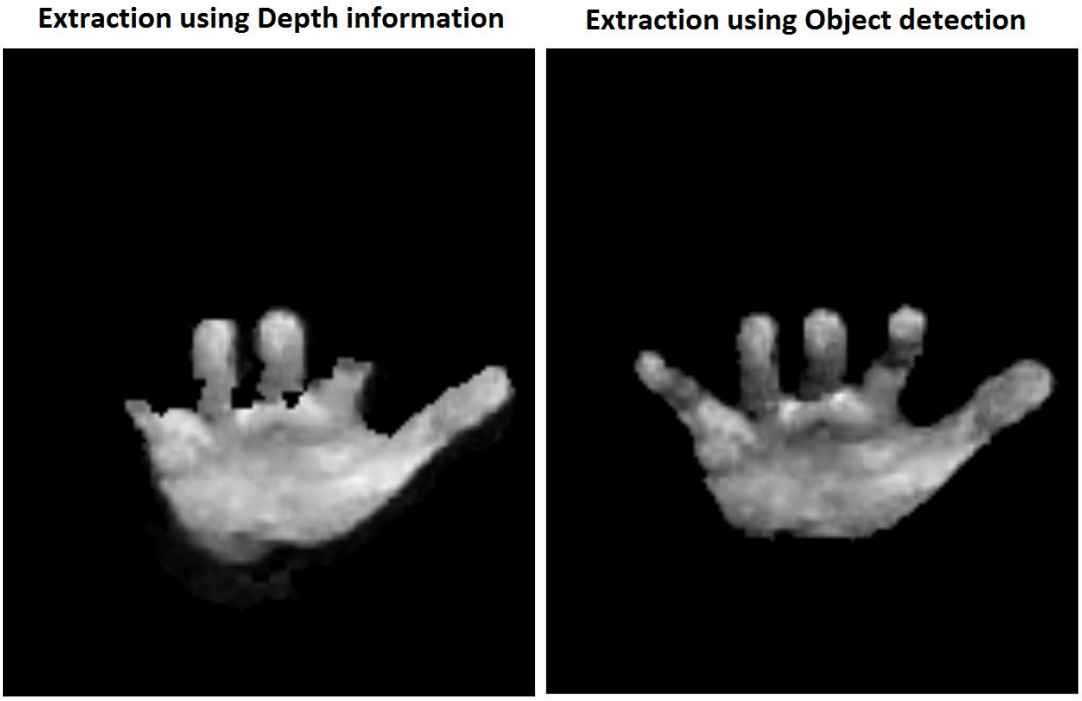

Figure 2.36: Extraction using the Depth information (Left), extraction using object detection (right).

2.3.2.3. Normalization

The intensity values of the observation are ranged from 0 to 255. To compare them with the intensity values of a hypothesis ranged from 0 to 1, they are first normalized to that same range based on the equation below.

$$Gray(i,j) = \frac{Gray(i,j) - \min(Gray)}{\max(Gray) - \min(Gray)}$$

2.3.3. The objective functions and their properties

Until now the lighting model algorithm, that interprets the illumination on a hypothesis, has been presented as well as the way the RGB data are handled to extract the foreground information of the hand in an observation.

The comparison between observation and hypothesis is done pixel-wise. The intensisty value of a pixel in the hypothesis's image is subtracted from the intensity value of a pixel with the same coordinates in the observation's image. The absolute difference between them is taken

$$d_{i,j} = \left| o_{i,j} - h_{i,j} \right|$$

where **i** and **j** correspond to the coordinates of a pixel. The intensity differences after the subtraction of all the pixels can be seen in (*Figure 2.36*). Small differences appear darker while big differences appear brighter.

Figure 2.37: The absolute intensity differences – d**.**

Two more hypothesis examples can be seen in (Figure 2.38).

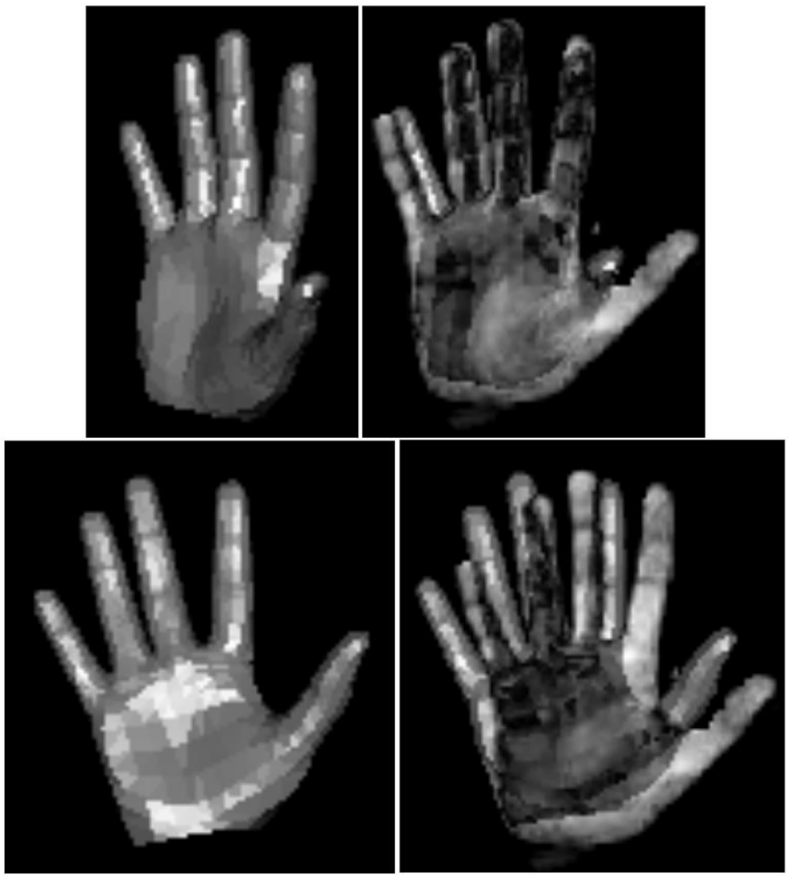

Figure 2.38: Examples of different hypotheses.

To evaluate the discrepancy value between an observation **O** and a hypothesis **H** several objective functions were formed each one with different parameters and properties. These properties are related to some auxiliary parameters which an objective function has with the goal to improve its efficiency. The objective functions are separated in three main categories: the category where **no parameter** is used denoted with the capital letter **N**, the category where the **percent parameter "r"** is used denoted with the capital letter **R** and the category where the **mask parameter "m"** is used denoted with the capital letter **M.** Each of these categories consists of 7 objective functions. The characteristics of all the objective functions are presented next.

2.3.3.1. Sum of absolute ($d_{i,j}$) differences

In set theory, given two sets A and B, the union is the set that contains elements or objects that belong to either A or to B or to both. In the same way, the union of an observation with a hypothesis is an image (array) that contains elements to either observation or to hypothesis or to both. The number of the pixels of the union is denoted as **Pu**. The discrepancy value V(O,H) is computed when the sum of all the absolute differences $d_{i,j}$ is divided by the number Pu.

$$V(O,H) = \frac{\sum_{i,j=0}^{h,w} \left| o_{i,j} - h_{i,j} \right|}{Pu}$$

Even if the hypothesis in (*Figures 2.37*) is a very good match, there are some points with big intensity difference $d_{i,j}$ that may influence negatively the discrepancy value V(O,H). For this reason, a **maximum discrepancy threshold "d_{max}"** is set. When the $d_{i,j}$ value is higher than the d_{max} then $d_{i,j} = d_{\max}$. That also means that the highest difference value is set to d_{max}. The discrepancy value is given by

$$V(O,H) = \frac{\sum_{i,j=0}^{h,w} \min(\left| o_{i,j} - h_{i,j} \right|, d_{\max})}{d_{\max} * \mathrm{Pu}}$$

This function ranges from 0 for a perfect match to 1 for a mismatch.

2.3.3.2. Number of big absolute ($d_{i,j}$) differences

A small $d_{i,j}$ difference means that a point with coordinates (i,j) in both the observation and the hypothesis has similar intensity value. As the number of these differences gets larger the total discrepancy between observation and hypothesis gets lower and the V(O,H) tends to zero. Contrariwise, as the number of big differences gets larger the V(O,H) tends to one.

An **intensity threshold "d_{big}"** above which an intensity difference value is considered big is established. The number of these big absolute differences denoted as **Pb** is computed**,** and divided by the Pu. So, if all the differences are above d_{big} the V(O,H) value is 1. On the other hand, if all the differences are below this d_{big} the V(O,H) value is 0. A good d_{big} value was found efficient around 0.4.

The discrepancy value is now given by

$$V(O,H) = \frac{Pb}{Pu}$$

where $V(O,H) = \begin{cases} 0 \text{ if } Pb = 0 \text{ (All } d_{i,j} < d_{big}) \\ 1 \text{ if } Pb = Pu \text{ (All } d_{i,j} \geq d_{big}) \end{cases}$

The proposed objective functions either use the sum of absolute ($d_{i,j}$) differences or the number of big absolute ($d_{i,j}$) differences "Pb".

During the tracking the 3D hand tracker generates several hypotheses. This number depends on the user. A large number of hypotheses (e.g. 800) means that the chance for a hypothesis to perfectly match the observation gets higher but the algorithm's execution time is slower. A small number of hypotheses (e.g. 200) means that the chance for a perfectly matched hypothesis is lower but the algorithm's execution time is faster. Whatever this number, there is a chance, a hypothesis that does not match the observation, to have very similar illumination with the observation. As the number of these hypotheses gets bigger, there are more chances for a wrong hypothesis to be chosen as a match. To avoid problems like this, the objective function must be also sensitive to margin matching changes between the observation and the hypothesis. This way a hypothesis with different orientation but with similar illumination will again have a worse V(O,H) value than a hypothesis with an orientation similar to the observation.

The next two parts refer to the problem of margin matching between the observation and the hypothesis.

2.3.3.3. Percent parameter "r"

The **percent parameter "r"** is a fraction with a numerator the number of the hand pixels of the hypothesis **Ph** and denominator the number of the hand pixels of the observation **Po**.

$$r = \frac{Ph}{Po}$$

The Po number in each frame cannot change so the percent parameter for each hypothesis depends only on the Ph number. When the Ph number is bigger than the Po number, numerator and denominator must switch positions so that the percent parameter always tends to 1. So

$$r = \begin{cases} \frac{Ph}{Po} & \text{if } Ph < Po \\ \frac{Po}{Ph} & \text{if } Ph > Po \end{cases}$$

The advantage of this parameter is that the generated hypotheses tend to consist of a similar number of pixels as the observation

$$(\text{i.e. } Ph = Po \Rightarrow r = 1)\,.$$

However, this parameter does not assure that the hypothesis will match the observation. It is sensitive when the orientation or the size of the hand is different

but when the orientation is very similar, the V(O,H) value mainly depends on the luminance difference. This can be explained better while looking the images in (*Figure 2.39*).

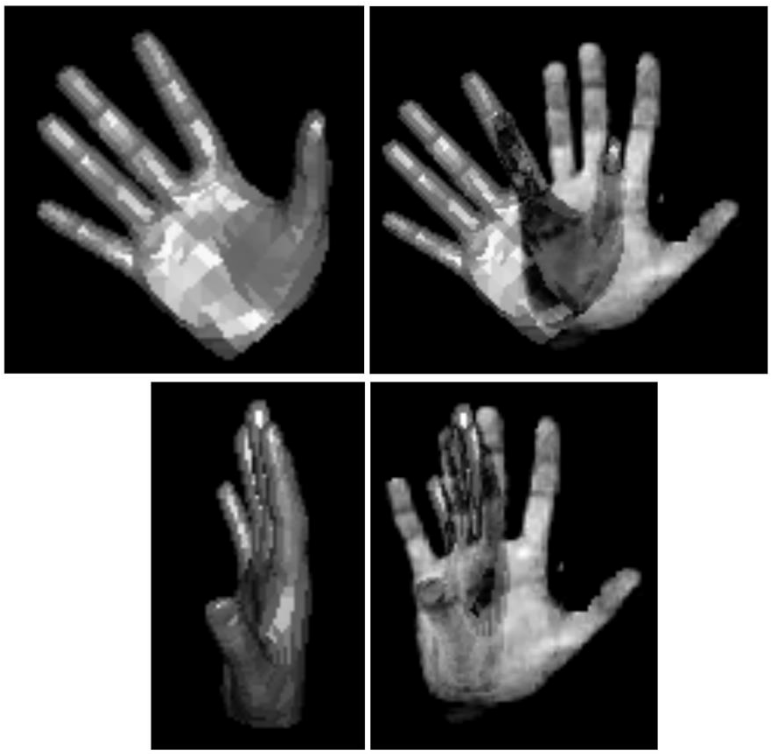

Figure 2. 39: Hypothesis examples

The percent value of the image in (*Figure 2.37*), is $r \approx 0.93$. In the two upper images of (*Figure 2.39*) it is clear how different the orientation of the hypothesis is but despite that the percent value is $r \approx 0.91$. This happens because the hypothesis has a very similar orientation with the observation. Now in the two lower images of (*Figure 2.39*) again the difference is obvious but the percent value is $r \approx 0.48$ because the Ph number was reduced significantly when the orientation changed.

2.3.3.4. Masks and the mask parameter "m"

A more efficient way to marginally match the hypothesis and the observation is by using masks. A mask image is a binary image. The mask image of the observation and a hypothesis can be seen in (*Figure 2.40*). Every hand pixel in both images has value equal to 1 and every other pixel's value is equal to 0.

Figure 2.40: Observation mask (Left), Hypothesis mask (Right).

To continue, the hypothesis and the observation image (arrays) are joined to find the union pixels, the intersection pixels, and the difference pixels. In left image of (*Figure 2.41*) the red pixels represent the observation pixels that don't match with the hypothesis and the blue pixels represent the hypothesis pixels that don't match with the observation. Summing the number of the red pixels with the blue ones we find the total number of the difference pixels **Pd**. In the right image of (*Figure 2.41*) the red pixels represent the intersection pixels **Pi** between observation and hypothesis. The green pixels represent the difference pixels Pd and summing the green pixels with the red ones we find the total number of the union pixels **Pu** that was discussed previously in 2.3.3.1. Sum of absolute ($di_{,j}$) differences

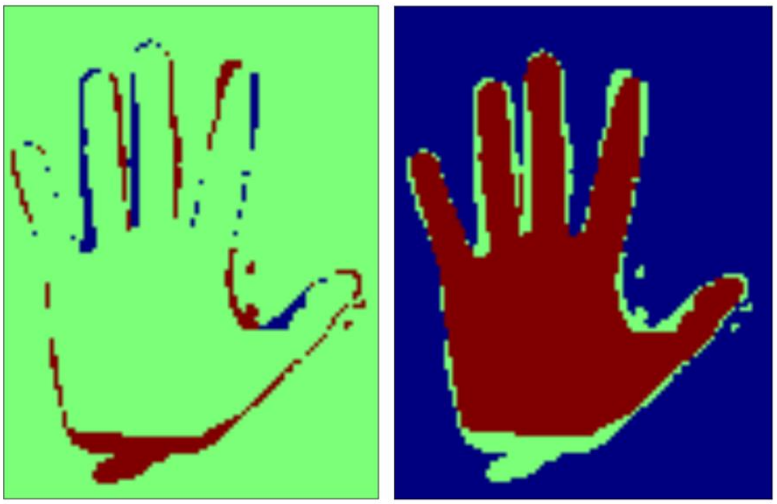

Figure 2.41: Mask differences (Left), Union mask (Right).

The **mask parameter** that arises from the use of the masks and the three numbers Pd, Pi and Pu is symbolized with the letter **m** and represents the

percentage of the intersection pixels Pi in the union of the two masks. The number of the intersection pixels is computed by

$$Pi = Pu - Pd$$

and the percentage m

$$m = \frac{Pi}{Pu}$$

As the number Pi approaches the number Pu the percentage m tends to be 1. That means that the hypothesis with the bigger number of intersection pixels fits better to the observation.

The advantage of this parameter is that adds an error to the discrepancy value when the hypothesis doesn't marginally match the orientation of the observation. As their orientation's difference gets bigger the error gets higher too. Also, this parameter was found to be more robust than the percent parameter "r".

2.3.3.5. Intersected pixels, penalty parameter and lighting absence

So far, the use of the auxiliary parameters "r" and "m" as long as the way an objective function computes the discrepancy value were described (either using the sum of the differences or the number of big differences). In this section, some further properties of the objective functions are discussed.

An objective function until now, uses all the pixels (**Pu**) to evaluate the discrepancy value V(O,H). An alternative objective function is tested, where only the pixels that intersect (**Pi**) are used in the evaluation of the discrepancy value V(O,H). Thus, we have

$$V(O,H) = \frac{\sum_{i,j=0}^{h,w} \min(\left|o_{i,j} - h_{i,j}\right|, d_{\max})}{d_{\max} * \mathrm{Pi}} \text{ and } V(O,H) = \frac{Pb}{Pi}$$

when the sum of absolute differences and the number of big differences are used respectively.

Figure 2.42: Only the intersected pixels are used.

Moreover, an alternative objective function is tested where a **penalty** for the pixels of the difference **Pd** is placed. More specifically, we place a penalty equal to the value of the $\mathbf{d_{max}}$ to the pixels that don't intersect (Pd). This way we penalize our objective function for every unmatched pixel between the observation and a hypothesis. The image below shows the thinking.

$$V(O,H) = \frac{\sum_{i,j=0}^{h,w} \min(\left|o_{i,j} - h_{i,j}\right|, d_{\max})}{d_{\max} * Pu} \quad \text{if } d_{i,j} \in Pd \Rightarrow d_{i,j} = d_{\max}$$

Figure 2.43: The intensity differences – d **with the penalties.**

To determine how useful the use of the lighting information really is, the objective functions have to be tested and compared with an alternative version of

them where the lighting is absent. Two more equations are generated; the first corresponds to the objective functions where all the pixels are used (Pu) and the second to the objective functions where only the intersected pixels are used (Pi).

$$V(O,H)=\frac{Pd}{Pu} \text{ and } V(O,H)=\frac{Pd}{Pi}$$

Since the lighting is absent, the discrepancy value V(O,H) is a subtraction result between the two binary masks of the observation and the hypothesis.

Figure 2.44: Lighting absence.

2.3.3.6. The objective functions

To avoid confusion, the generated objective functions that arise from the combination of all the elements mentioned in the previous sections, are separated in 3 categories: N, R, M and their versions are presented below with a unique label. The total number of the constructed objective functions is 21.

N	**No parameter is used**
R	**Percent parameter "r" is used**
M	**Mask parameter "m" is used**

S = Sum	**C = Count**	**L = Lighting**	**P = Penalty**	**I = Intersection**

N	Equations
NS	No Lighting
NSL	Lighting
NSPL	Lighting + Penalty
NSI	Intersection No Lighting
NSIL	Intersection Lighting
NCL	Lighting
NCIL	Intersection Lighting

Table 1: N equations.

$$V(O,H) = \frac{Pd}{Pu} \quad \{Eq.NS\}$$

$$V(O,H) = \frac{\sum_{i,j=0}^{h,w} \min(|o_{i,j} - h_{i,j}|, d_{\max})}{d_{\max} * Pu} \quad \{Eq.NSL\}$$

$$V(O,H) = \frac{\sum_{i,j=0}^{h,w} \min(|o_{i,j} - h_{i,j}|, d_{\max})}{d_{\max} * Pu} \text{ if } d_{i,j} \in Pd \Rightarrow d_{i,j} = d_{\max} \{Eq.NSPL\}$$

$$V(O,H) = \frac{Pd}{Pi} = 0 \quad \{Eq.NSI\}$$

$$V(O,H) = \frac{\sum_{i,j=0}^{h,w} \min(|o_{i,j} - h_{i,j}|, d_{\max})}{d_{\max} * Pi} \quad \{Eq.NSIL\}$$

$$V(O,H) = \frac{Pb}{Pu} \quad \{Eq.NCL\}$$

$$V(O,H) = \frac{Pb}{Pi} \quad \{Eq.NCIL\}$$

R	Equations
RS	No Lighting
RSL	Lighting
RSPL	Lighting + Penalty
RSI	Intersection No Lighting
RSIL	Intersection Lighting
RCL	Lighting
RCIL	Intersection Lighting

Table 2: R equations.

$$V(O,H) = r\frac{Pd}{Pu} + 1 - r \quad \{Eq.RS\}$$

$$V(O,H) = r\frac{\sum_{i,j=0}^{h,w} \min(|o_{i,j} - h_{i,j}|, d_{\max})}{d_{\max} * \mathrm{Pu}} + 1 - r \quad \{Eq.RSL\}$$

$$V(O,H) = r\frac{\sum_{i,j=0}^{h,w} \min(|o_{i,j} - h_{i,j}|, d_{\max})}{d_{\max} * \mathrm{Pu}} + 1 - r \; if \; d_{i,j} \in Pd \Rightarrow d_{i,j} = d_{\max} \quad \{Eq.RSPL\}$$

$$V(O,H) = r\frac{Pd}{Pi} + 1 - r = 1 - r \quad \{Eq.RSI\}$$

$$V(O,H) = r\frac{\sum_{i,j=0}^{h,w} \min(|o_{i,j} - h_{i,j}|, d_{\max})}{d_{\max} * \mathrm{P}\,i} + 1 - r \quad \{Eq.RSIL\}$$

$$V(O,H) = r\frac{Pb}{Pu} + 1 - r \quad \{Eq.RCL\}$$

$$V(Oi,Hi) = r\frac{Pb}{Pi} + 1 - r \quad \{Eq.RCIL\}$$

M	Equations
MS	No Lighting
MSL	Lighting
MSPL	Lighting + Penalty
MSI	Intersection No Lighting
MSIL	Intersection Lighting
MCL	Lighting
MCIL	Intersection Lighting

Table 3: M equations.

$$V(O,H) = m\frac{Pd}{Pu} + 1 - m \quad \{\text{Eq.MS}\}$$

$$V(O,H) = m\frac{\sum_{i,j=0}^{h,w} \min(\left|o_{i,j} - h_{i,j}\right|, d_{\max})}{d_{\max} * \mathrm{Pu}} + 1 - m \quad \{\text{Eq.MSL}\}$$

$$V(O,H) = m\frac{\sum_{i,j=0}^{h,w} \min(\left|o_{i,j} - h_{i,j}\right|, d_{\max})}{d_{\max} * \mathrm{Pu}} + 1 - m \; if \; d_{i,j} \in Pd \Rightarrow d_{i,j} = d_{\max} \quad \{\text{Eq.MSPL}\}$$

$$V(O,H) = m\frac{Pd}{Pi} + 1 - m = 1 - m \quad \{\text{Eq.MSI}\}$$

$$V(O,H) = m\frac{\sum_{i,j=0}^{h,w} \min(\left|o_{i,j} - h_{i,j}\right|, d_{\max})}{d_{\max} * \mathrm{P}i} + 1 - m \quad \{\text{Eq.MSIL}\}$$

$$V(O,H) = m\frac{Pb}{Pu} + 1 - m \quad \{\text{Eq.MCL}\}$$

$$V(Oi,Hi) = m\frac{Pb}{Pi} + 1 - m \quad \{\text{Eq.MCIL}\}$$

Algorithm 2: Brief discrepancy value calculation algorithm.

Input Data: $\{$Observation (GRAY image),Hypothesis (PHONG image),h,w$\}$

1.*Masks* and *mask parameter* "m"

GRAY mask.

PHONG mask.

Pu,Pd,Pi and m calculation.

2. *Percent parameter* "r"

Po,Ph and r calculation.

3. *Sum of absolute differences*

Intensity absolute differences (d array) calculation $d_{i,j}$

Threshold $d_{\max}$

4. *Number of absolute big differences*

Threshold d_{big}

Number Pb

5. *The equations*

No parameter equations : $NS,\ NSL,\ NSPL,\ NSI,\ NSIL,\ NCL,\ NCIL$

Percent parameter "r" equations : $RS,\ RSL,\ RSPL,\ RSI,\ RSIL,\ RCL,\ RCIL$

Mask parameter "m" equations : $MS,\ MSL,\ MSPL,\ MSI,\ MSIL,\ MCL,\ MCIL$

Output Data: $\{$V(O,H),d$\}$

Chapter 3 - Experiments

3.1. Offline experiments

In this section, the efficiency of the objective functions is tested using the following setup. An initial rendered hand pose is matched with the segmented hand pose of an observation (frame) and follows a specific move consisted of several hypotheses. Each hypothesis is compared with the observation using an objective function. The discrepancy value V(O,H) of each hypothesis is recorded and a graph is created (Discrepancy – Hypothesis) which shows how the discrepancy value changes in each hypothesis. The same move is used to create the graph for all the objective functions. A single case is characterized by a specific move and includes the graphs of all the objective functions for a given observation.

Then, a simulated hand pose is used as an observation's hand pose and the same move is used to test all the objective functions again. This way we compare the hypotheses with an observation which has the same light conditions and foreground. Thus, for each case we have the offline experiments using a real observation and the offline experiments using a simulated hand pose as an observation. Under the same logic, we tested the objective functions in various cases before proceeding to the tracking experiments.

Case 1	X-Y axis movement
Case 2	Y axis rotation (clockwise + counterclockwise)
Case 3	Y axis rotation 2 (counterclockwise)
Case 4	X axis rotation (downward)
Case 5	Bending

Table 4: Offline experiments cases.

In the last section of the previous chapter we categorized the objective functions in six categories. In each following case, a figure displays the performance of every objective function in each category. The performance of an objective function in a figure is visualized with a graph and has a specific color.

- **Equations N: NS, NSL, NSPL, NSI, NSIL, NCL, NCIL**
- **Equations R: RS, RSL, RSPL, RSI, RSIL, RCL, RCIL**
- **Equations M: MS, MSL, MSPL, MSI, MSIL, MCL, MCIL**

Objective functions using the "sum of absolute differences" method: the **blue colored** graphs refer to the functions (NS, RS, MS) where the lighting is absent and all the pixels are used, the **green colored** graphs refer to the functions (NSL, RSL, MSL)

where the lighting is present and all the pixels are used, the **red colored** graphs refer to the functions (NSPL, RSPL, MSPL) where the penalty parameter and all the pixels are used, the **black colored** graphs refer to the functions (NSI, RSI, MSI) where the lighting is absent and only the intersected pixels are used, the **magenta colored** graphs refer to the functions (NSIL, RSIL, MSIL) where the lighting is present and only the intersected pixels are used.

Objective functions using the "number of big absolute differences" method: the **yellow colored** graphs refer to the functions (NCI, RCI, MCI) where the lighting is absent and only the intersected pixels are used, the **cyan colored** graphs refer to the functions (NCIL, RCIL, MCIL) where the lighting is present and only the intersected pixels are used.

Also, when the **sum of differences** is used $d_{\max} = 1$ and when the **number of big differences** is used $d_{big} = 0.4$.

The constants of the equation of the Phong reflection model that was mentioned in (*2.2.4.4.* The Phong reflection) can be seen below.

$$I_{PHONG} = I_{AR} + I_{DR} + I_{SR} = \frac{1}{k_0 + k_1 d + k_2 d^2}(k_d(\vec{l}\cdot\vec{n})L_d + k_s(\vec{r}\cdot\vec{v})^{\mathrm{a}} L_s) + k_{\mathrm{a}} L_{\mathrm{a}}$$

are set

$$k_0 = 1, k_1 = k_2 = 0$$
$$k_{\mathrm{a}} = 0.05, L_{\mathrm{a}} = 1$$
$$k_d = 0.7, L_d = 1$$
$$k_s = 0.25, L_s = 1, \mathrm{a} = 1$$

3.1.1. Case 1: X-Y axis movement

The RGB image used in the first case can be seen in the image below.

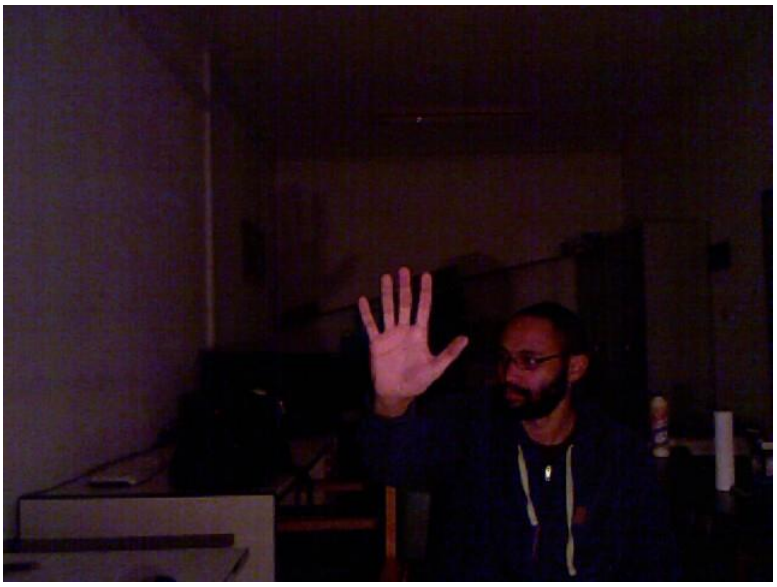

Figure 3.1: RGB image of Case 1.

3.1.1.1. X-Y axis movement

The result after the foreground extraction, the initial hypothesis, the final hypothesis, and their differences are shown in the figures below.

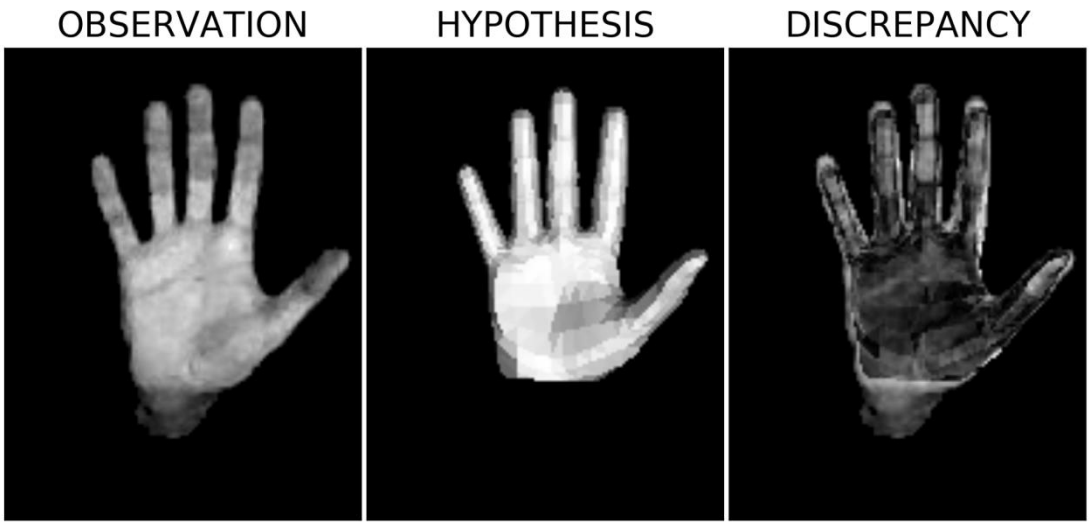

Figure 3.2: Observation (Left), initial hypothesis (Middle), and the differences (Right).

The initial hypothesis (pose) is then moved diagonally, in a down - left direction in 49 steps.

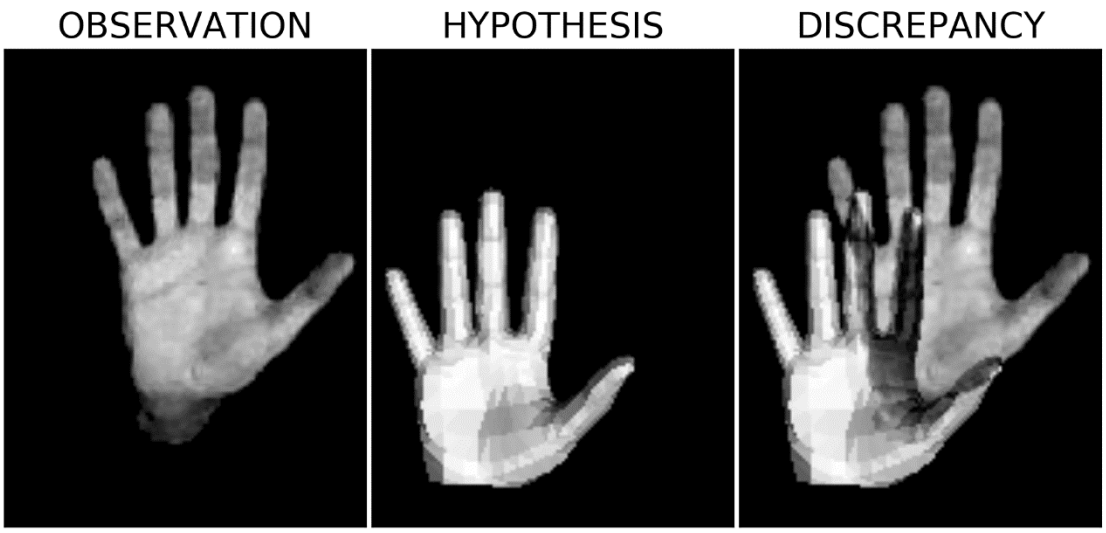

Figure 3.3: Observation (Left), final hypothesis (Middle), and the differences (Right).

When the lighting information is absent, the comparison is between the binary foreground mask of the observation and the binary foreground mask of each hypothesis.

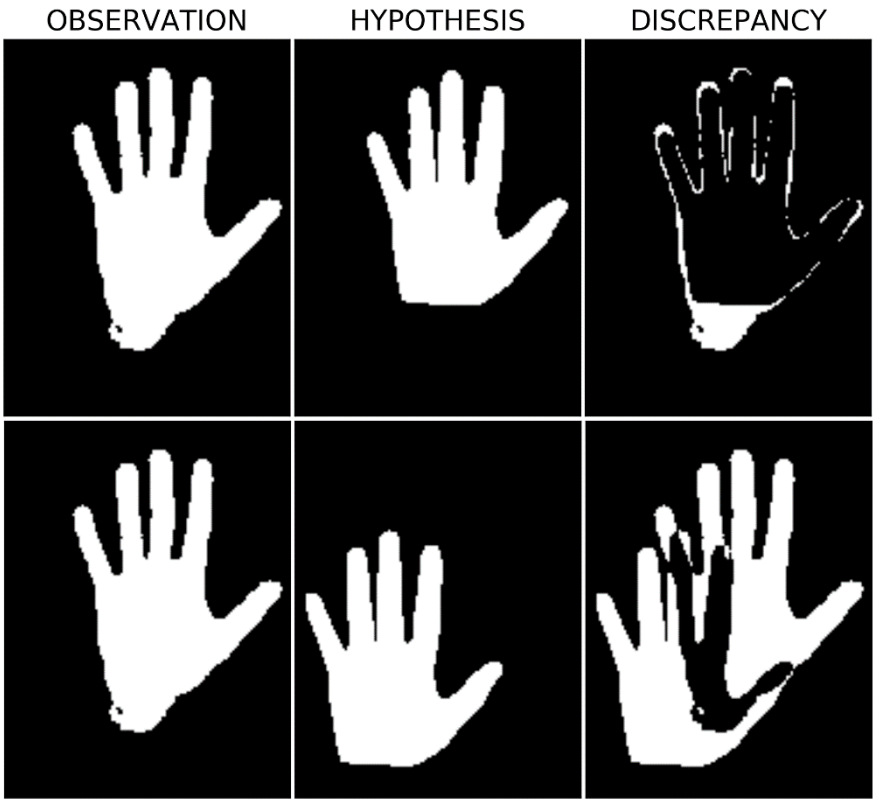

Figure 3.4: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent.

When the penalty component is used, the pixels that don't belong to either the observation or the hypothesis take the maximum default discrepancy value $\mathbf{d_{max}}$.

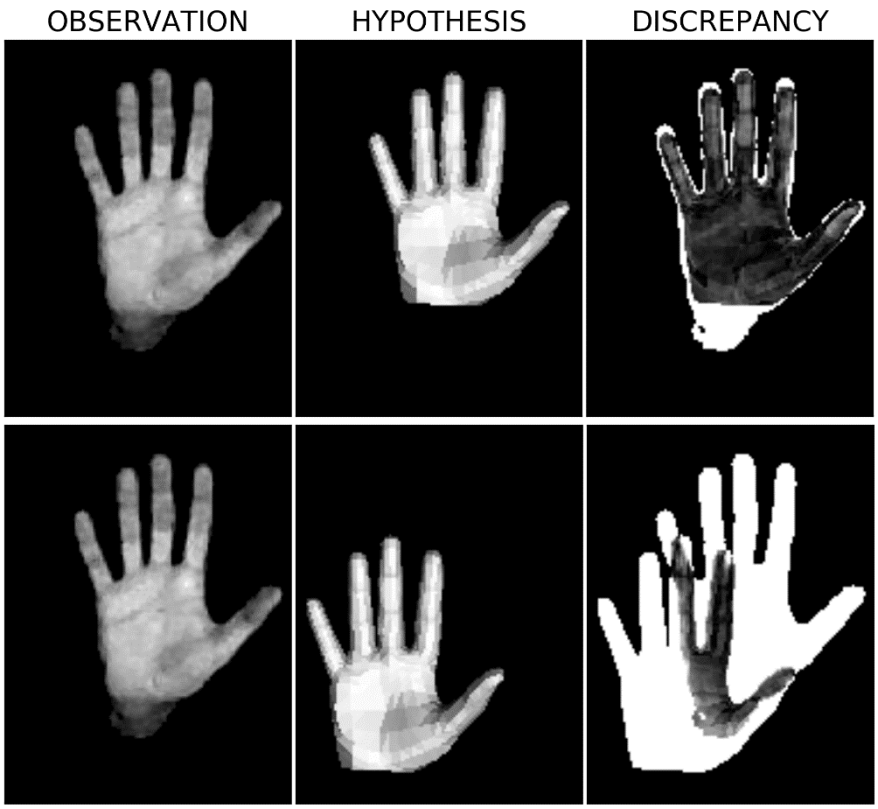

Figure 3.5: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty parameter is used.

The images below show the case where only the intersected pixels between the observation and the hypothesis contribute to the computation of the discrepancy value V(O,H).

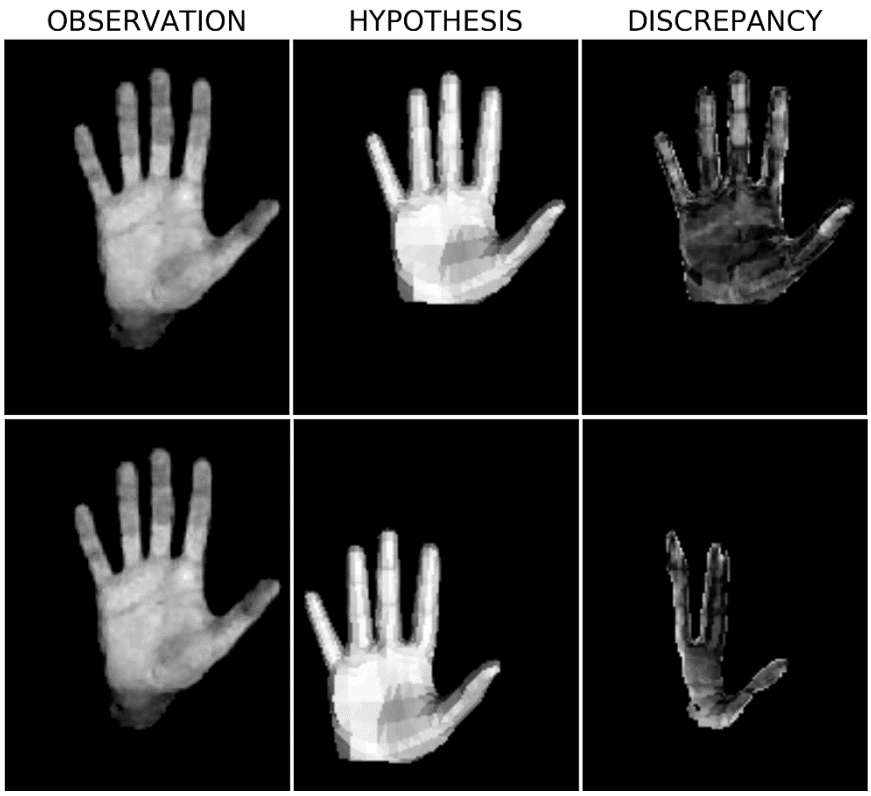

Figure 3.6: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used.

The graph results of the functions can be seen in the following figures.

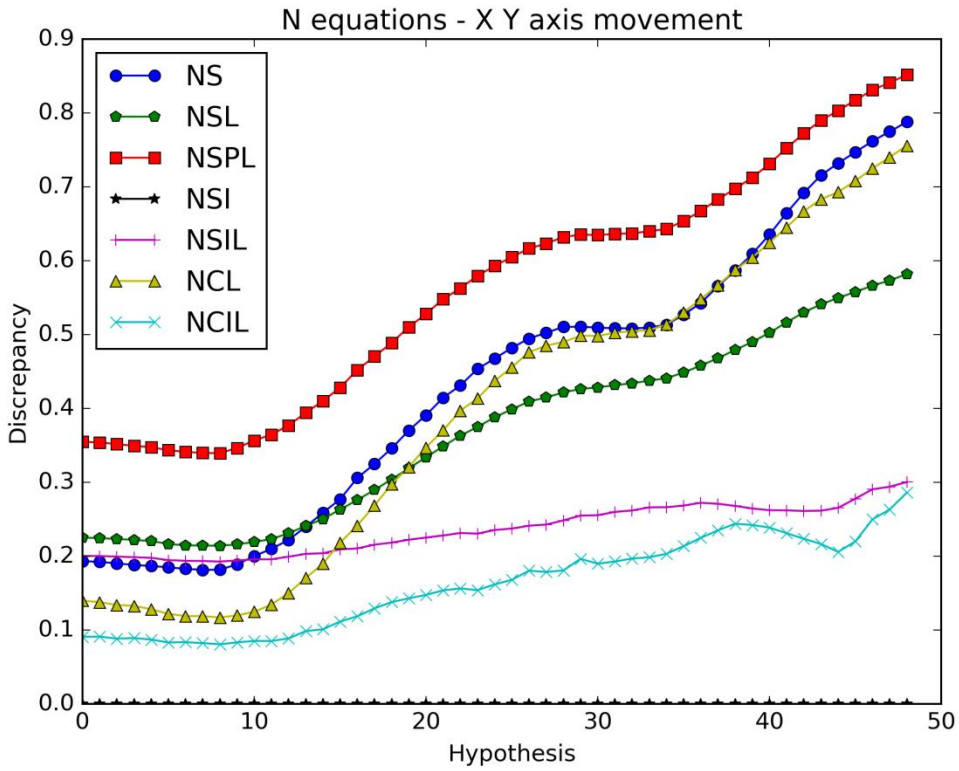

Figure 3.7: N equations.

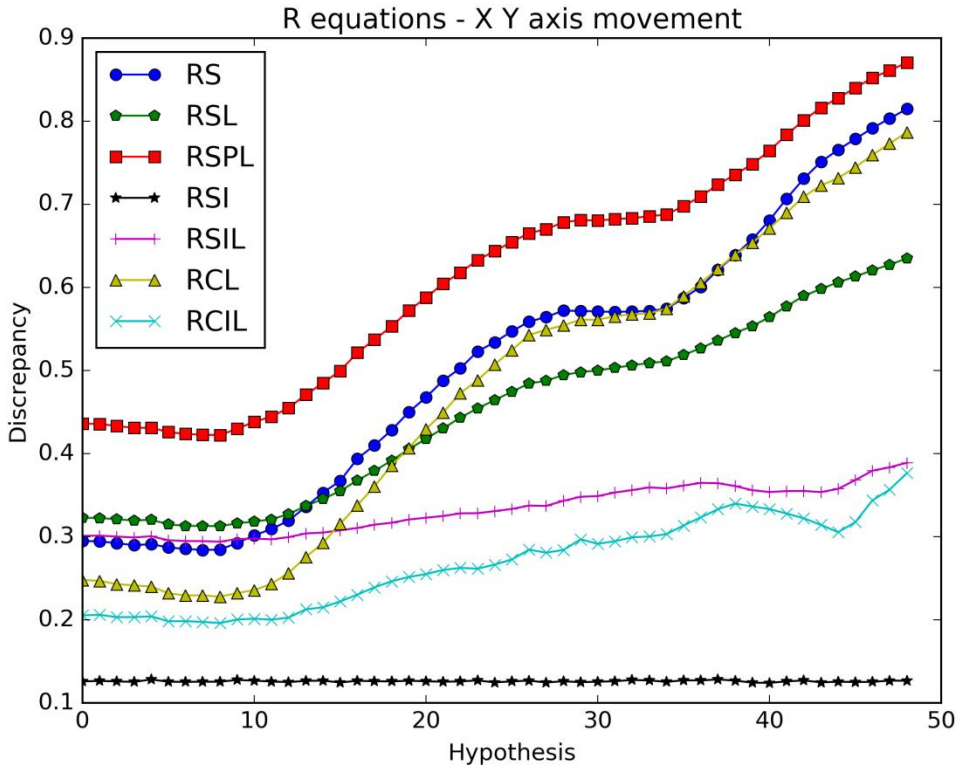

Figure 3.8: R equations.

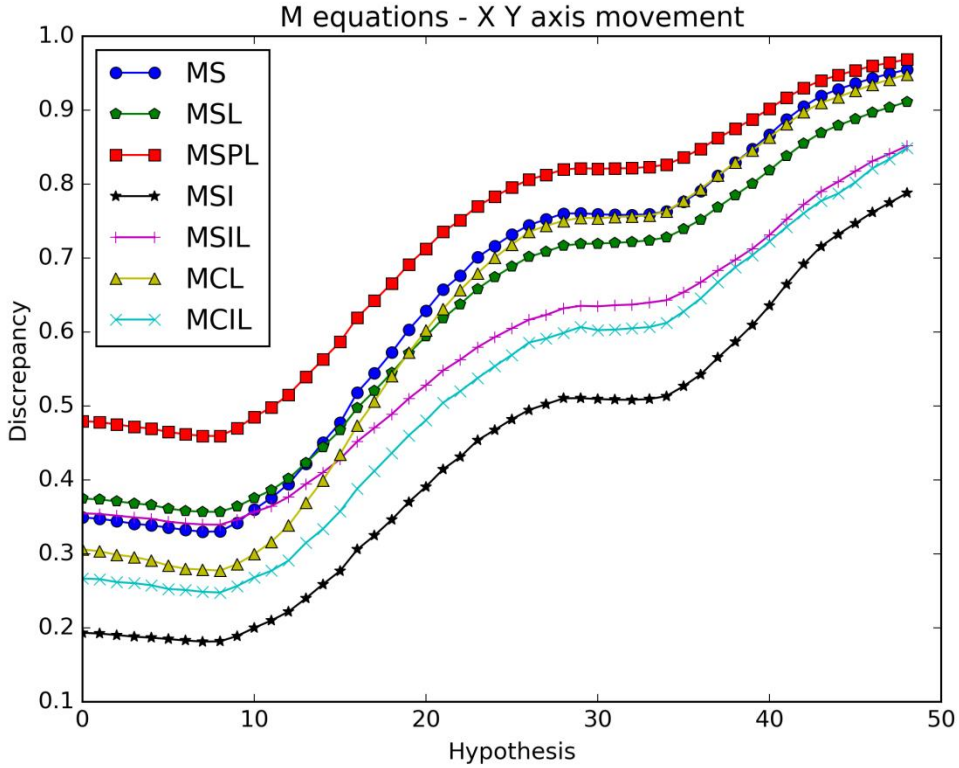

Figure 3.9: M equations.

The responses of the objective functions suggest that using only the intersected pixels don't improve the efficiency of the objective functions. The NCL, RCL, MCL objective functions rise faster than any other response in the corresponding figures. This shows that the lighting information may improve the efficiency of the objective functions.

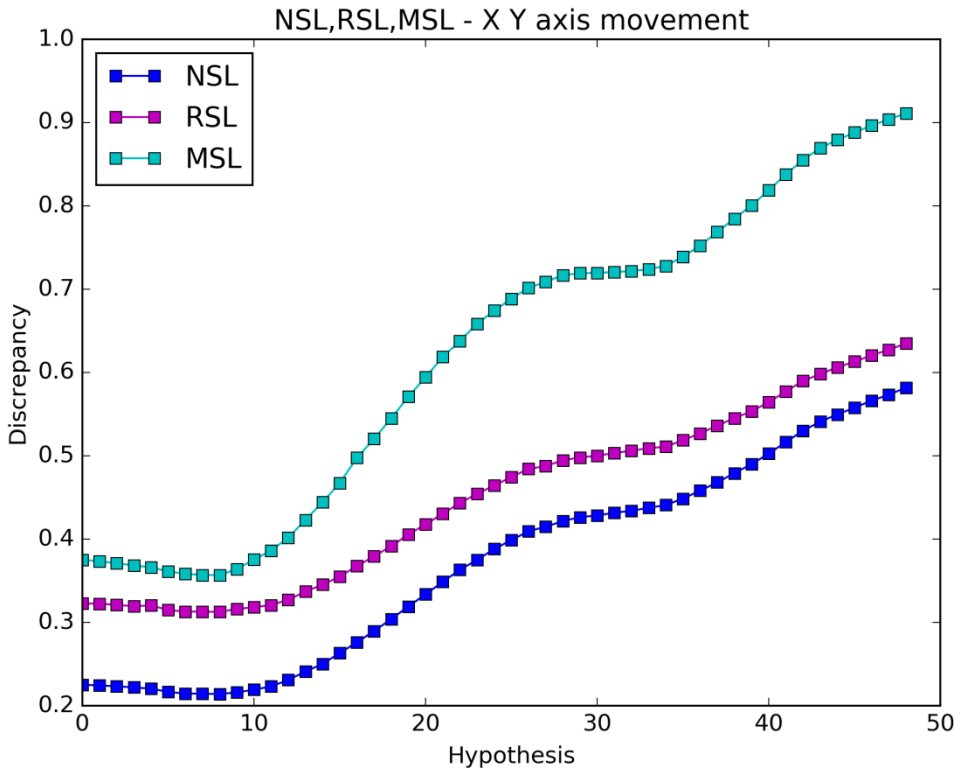

Figure 3.10: NSL, RSL, MSL equations comparison.

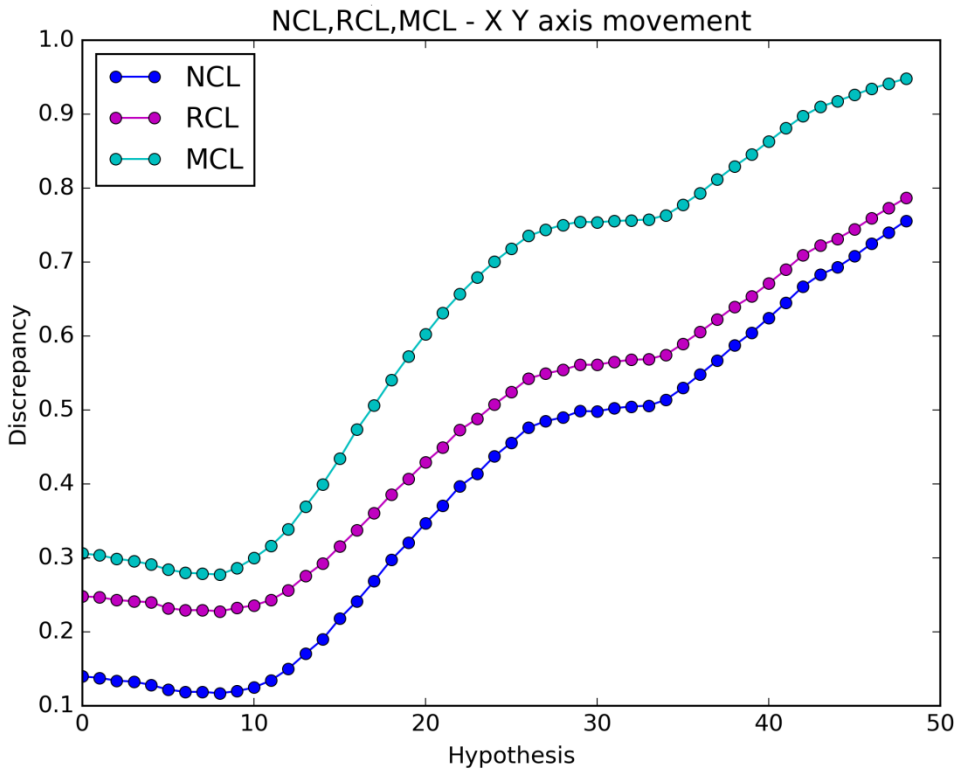

Figure 3.11: NCL, RCL, MCL equations comparison.

The percent parameter "**r**" and the mask parameter "m" are auxiliary parameters. The purpose of their use is to give an additional error to the discrepancy value when a hypothesis doesn't marginally match with the observation. So, what we expect is that the two auxiliary parameters will give a sharper increase in the graphs while the hypothesis is moving away from its initial state.

From the two figures above, the mask parameter seems to have the best results.

If the light conditions and the foreground where the same for both the observation and the hypotheses the graphs would have as a beginning the axis origin. There are two reasons why this isn't happening. The first reason is that the observation foreground doesn't marginally match with the hypothesis foreground. There are many pixels around and below the hand that we can't avoid completely. This has a significant impact on the result and also gives additional error to the discrepancy value V(O,H). The second reason is that the lighting distribution on the observation's hand pose differs from the lighting distribution on the hypothesis hand pose. In the figure below the intensities of the hypothesis are a lot higher than those of the observation.

These are the two major problems that we must front without being able to completely avoid. Because of them the objective functions give higher values than they should and the discrepancy value contains an error rate.

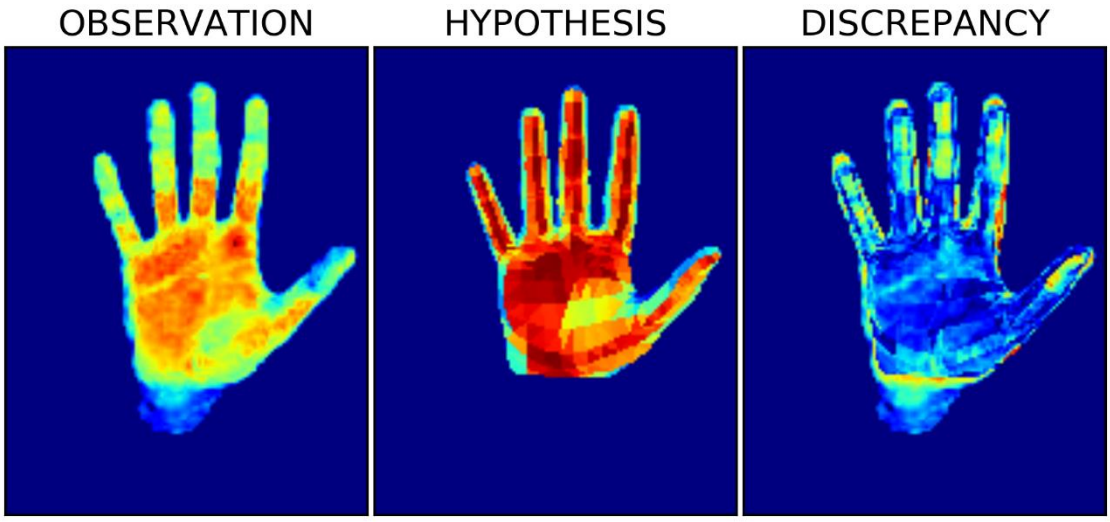

Figure 3.12: Problems display.

To have more trustworthy results about the efficiency of the objective functions we replaced the observation's hand pose with the initial simulated hand pose (hypothesis). The move doesn't change, the same hypotheses are generated but now the light conditions and the foregrounds are similar for both the observation and the hypotheses and there is no additional error to the discrepancy value V(O,H).

3.1.1.2. Simulated – X-Y axis movement

The initial rendered hand pose replaces the observation to see how the objective functions operate when the light conditions and the foregrounds are similar for both the observation and the hypothesis. The hypothesis is moved again in 49 steps as in the previous section. Due to the perfect match, the discrepancy image at the hypothesis initial state is black.

The result after the foreground extraction, the initial hypothesis, the final hypothesis and their differences are shown in the figures below.

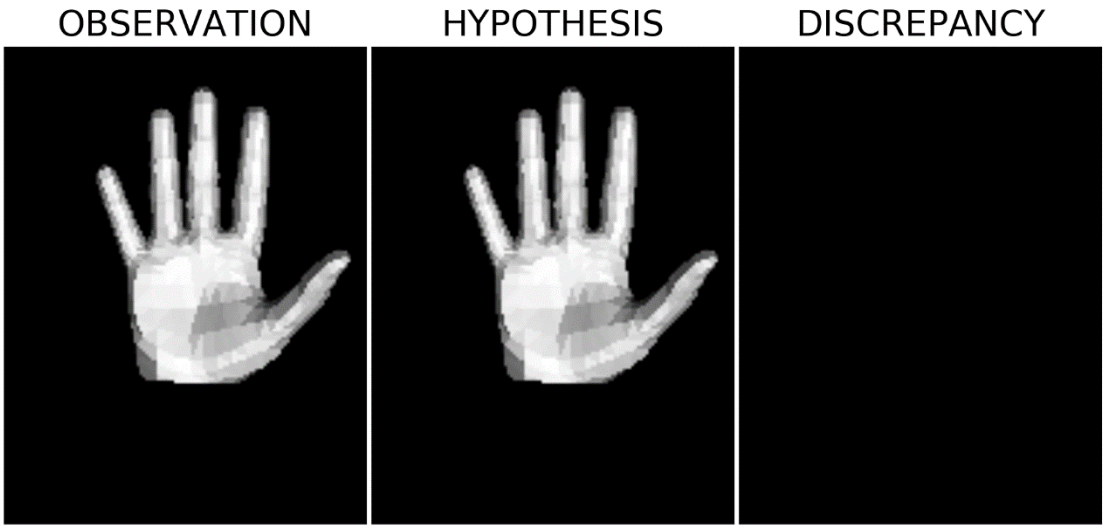

Figure 3.13: Observation (Left), initial hypothesis (Middle), and the differences (Right).

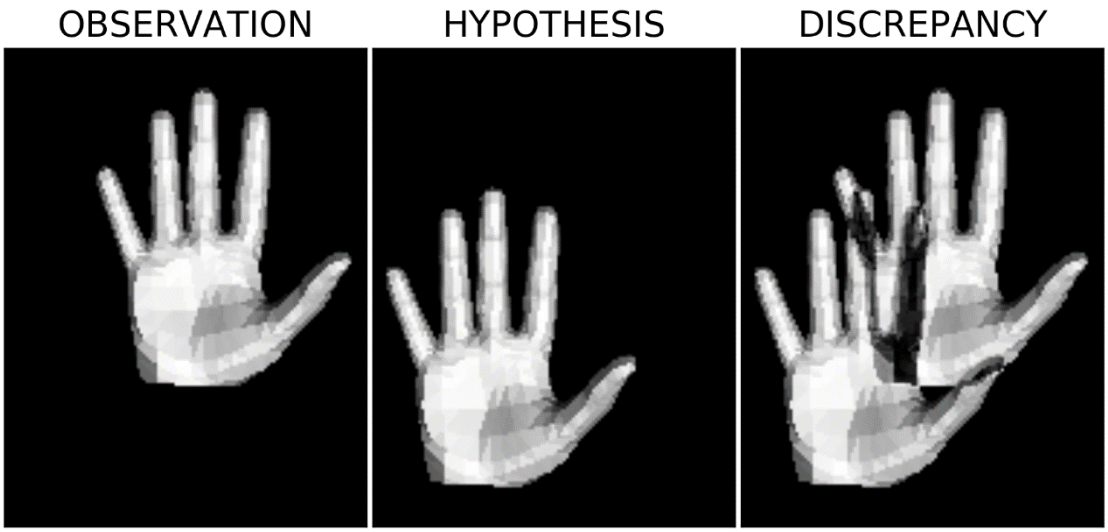

Figure 3.14: Observation (Left), final hypothesis (Middle), and the differences (Right).

The graph results of the objective functions can be seen in the following figures.

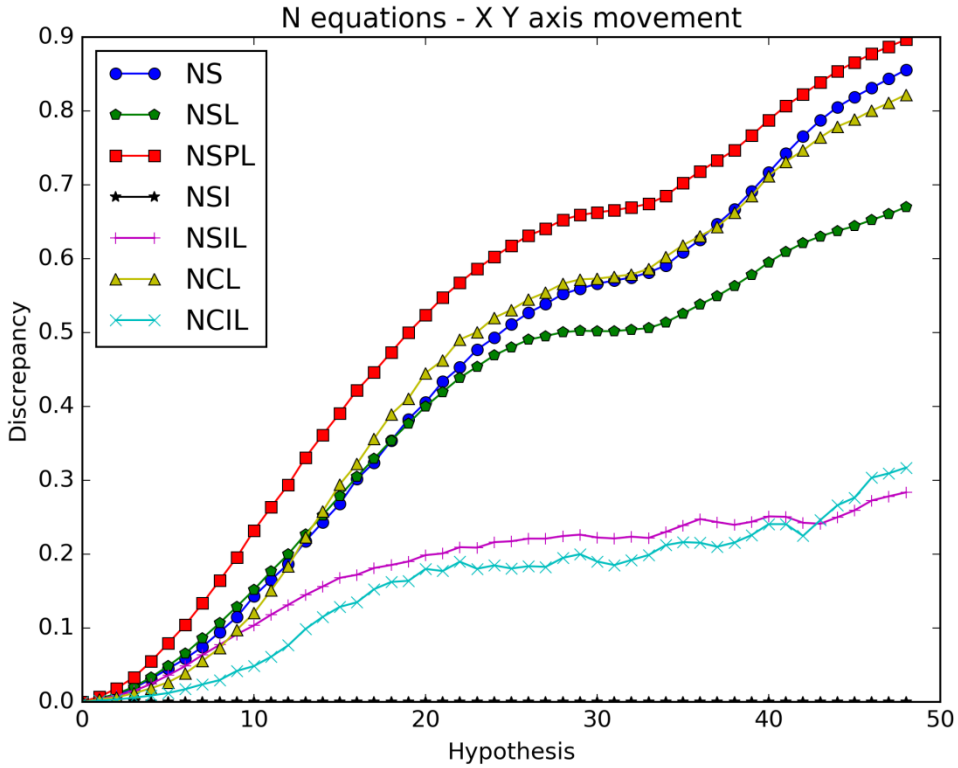

Figure 3.15: N equations.

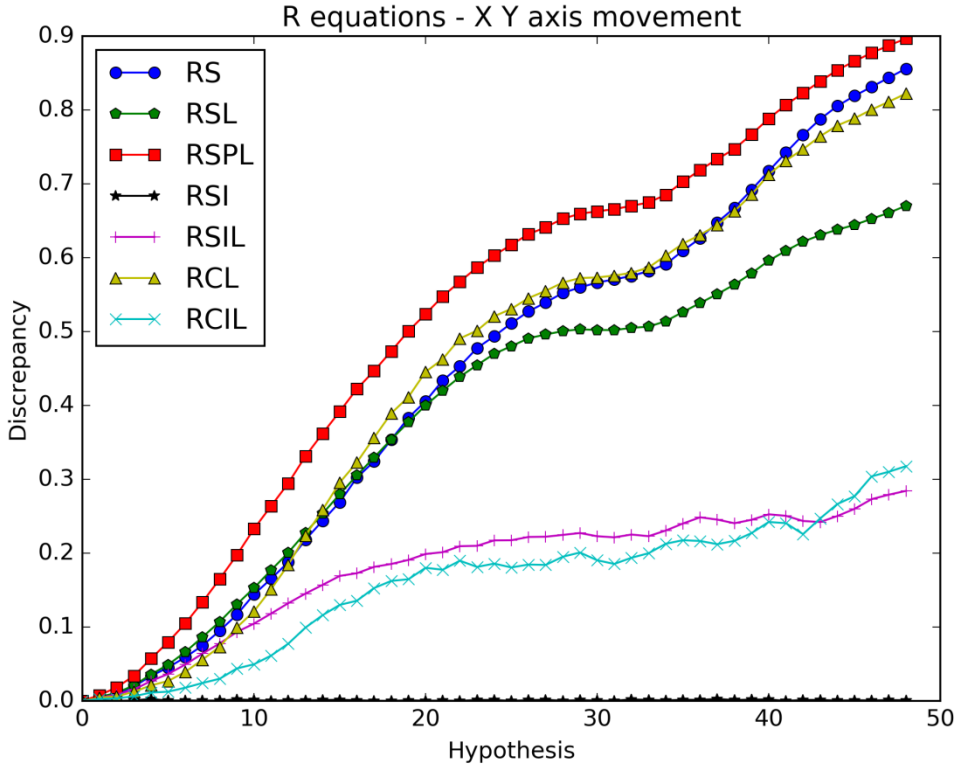

Figure 3.16: R equations.

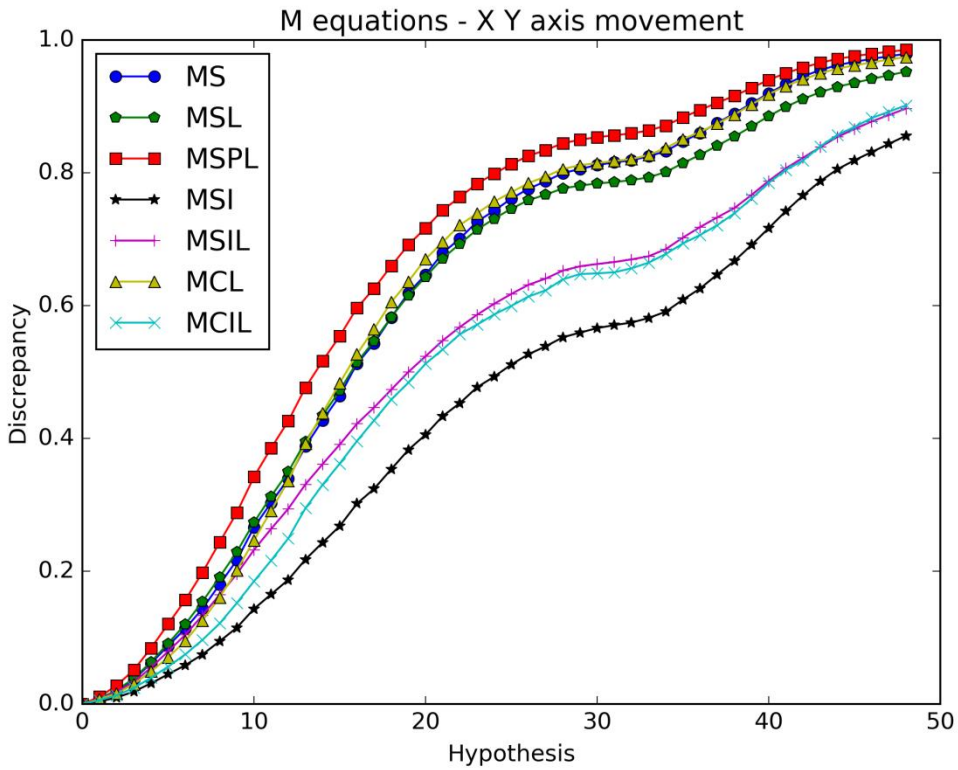

Figure 3.17: M equations.

All the responses of the objective functions now start from zero and the comparison between them is easier and more accurate.

Using only the intersected pixels doesn't improve the efficiency of the objective functions either the lighting is present or absent. In contrast with the previous results, when the penalty parameter is included, the performance of the NSPL, RSPL, and MSPL objective functions is far better than the others in each figure. That suggests that if all the conditions were similar for both the observation and the hypotheses, the lighting presence with the penalty parameter gives better results than when the lighting is absent.

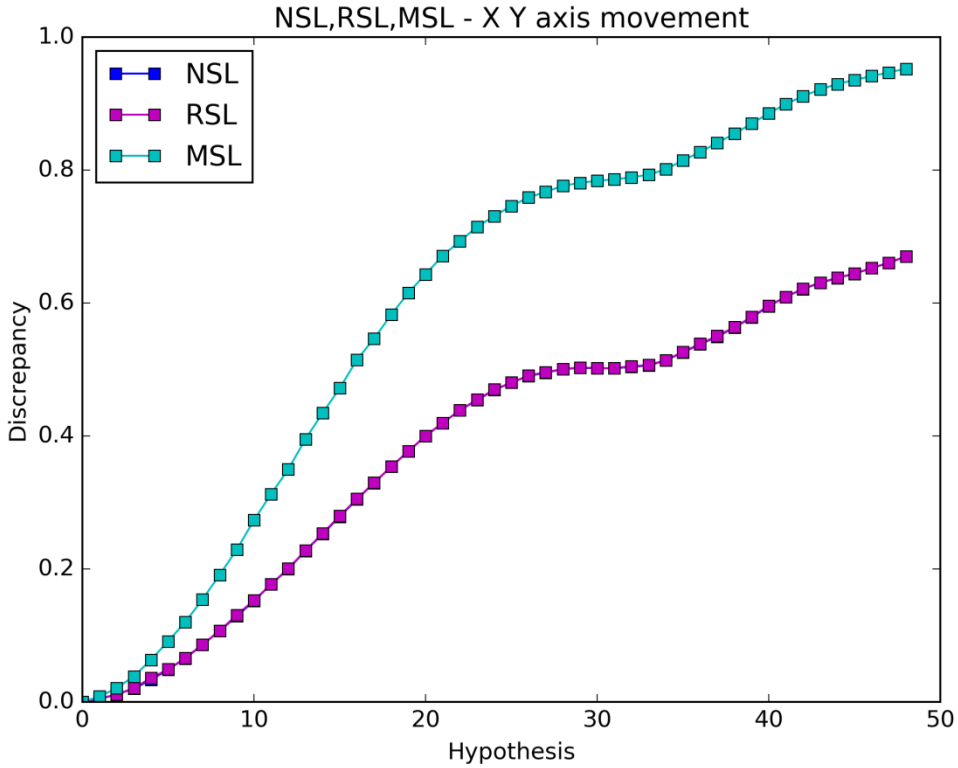

Figure 3.18: NSL, RSL, MSL equations comparison.

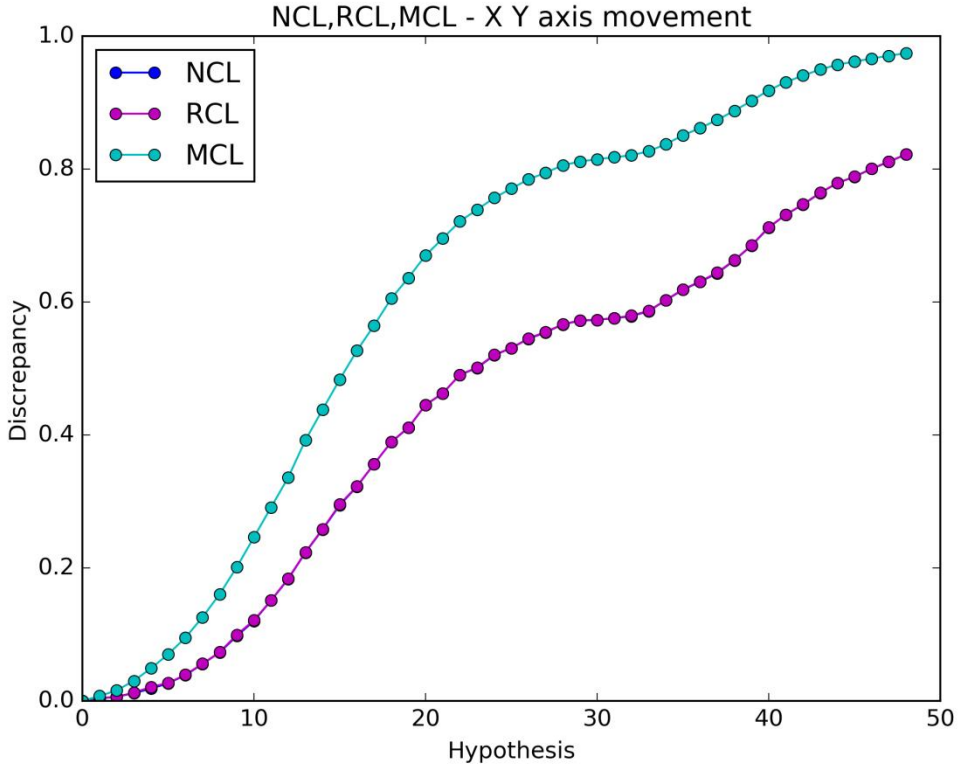

Figure 3.19: NCL, RCL, MCL equations comparison.

The mask parameter (MSL, MCL) is more efficient than the percent parameter (RSL, RCL). In this case, the percent parameter has no influence on the discrepancy value since the responses of RSL and RCL coincide with the responses of NSL and NCL respectively. The reason is that the number of the pixels of the hand model stays the same during the move.

3.1.2. Case 2: Y axis rotation

The RGB image used in this case can be seen in the image below.

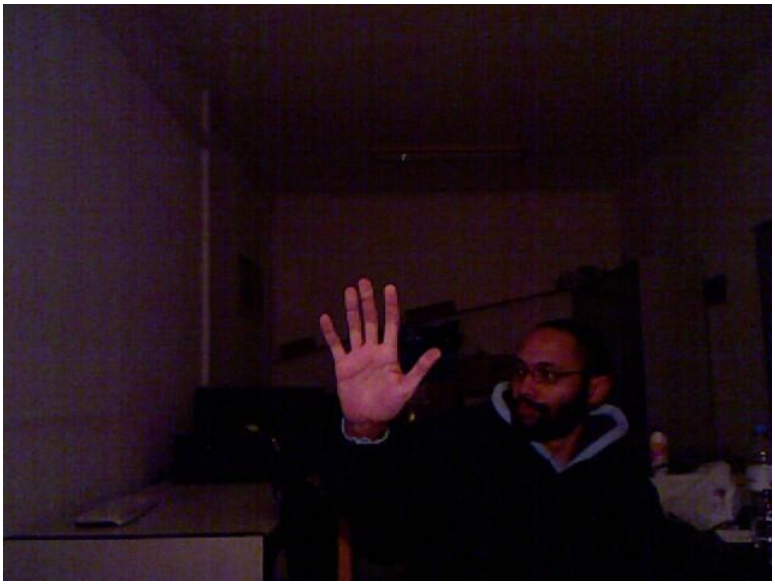

Figure 3.20: RGB image of Case 2.

3.1.2.1. Clockwise Rotation

The result after the foreground extraction, the initial hypothesis, the final hypothesis, and their differences are shown in the figures below.

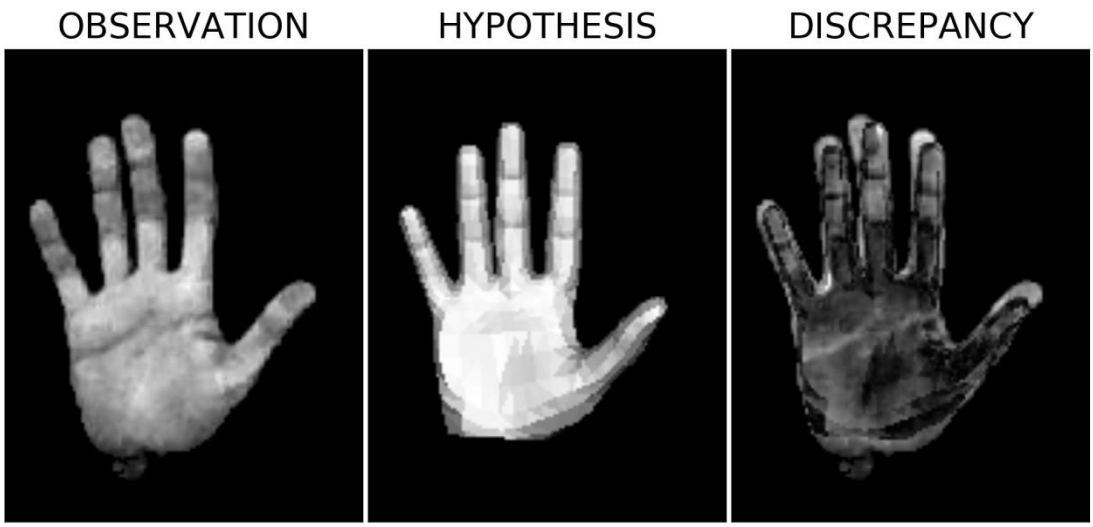

Figure 3.21: Observation (Left), initial hypothesis (Middle), and the differences (Right).

In this case, the initial hypothesis is rotated clockwise in 49 steps.

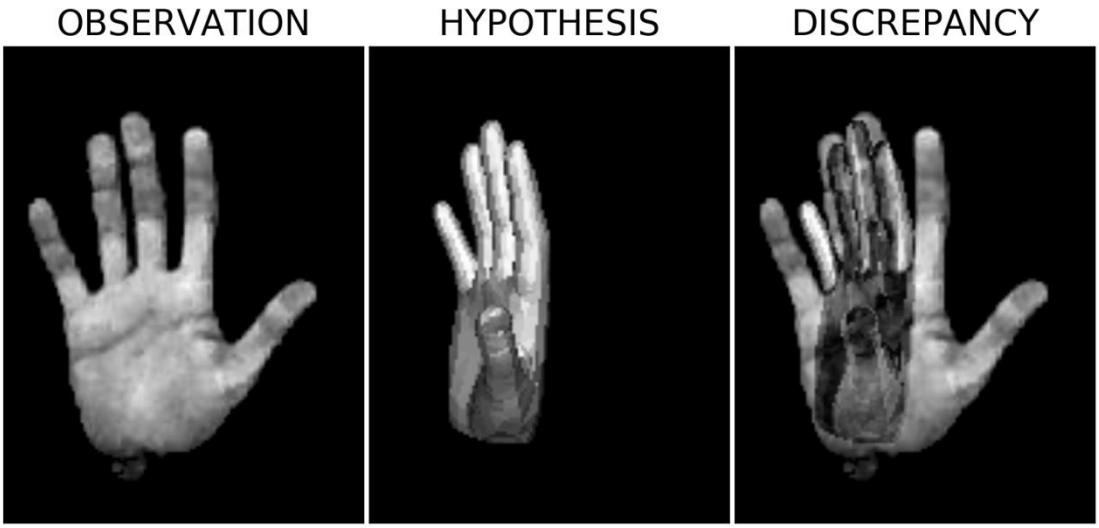

Figure 3.22: Observation (Left), final hypothesis (Middle), and the differences (Right).

The following figures show the cases where a) the lighting is absent, b) the penalty parameter is used and c) only the intersection pixels are used.

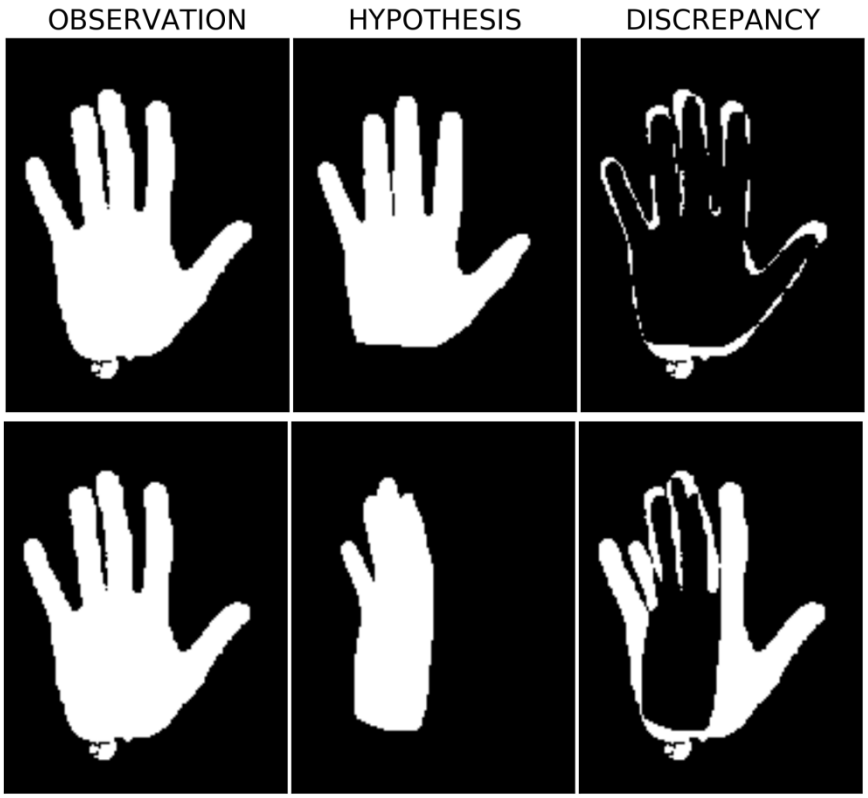

Figure 3.23: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent.

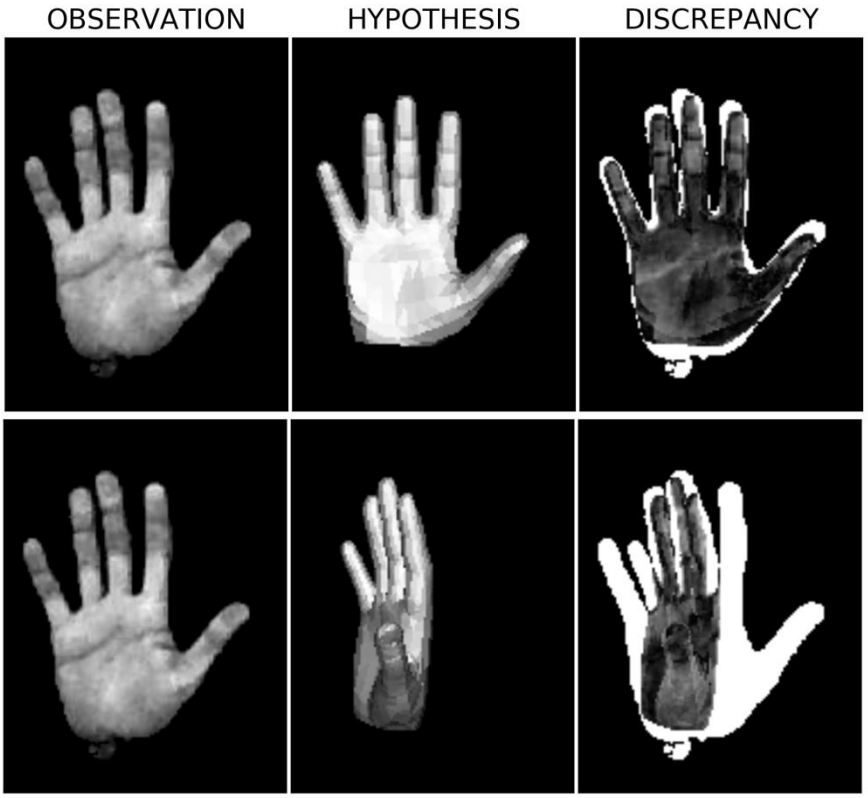

Figure 3.24: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty component is used.

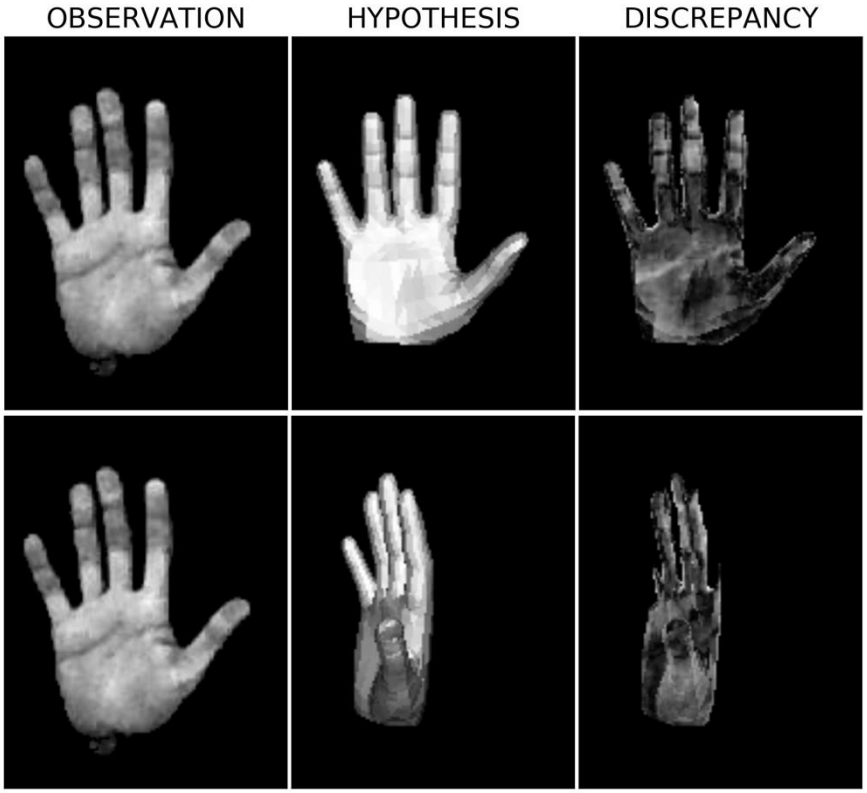

Figure 3.25: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used.

The graph results of the objective functions can be seen in the following figures.

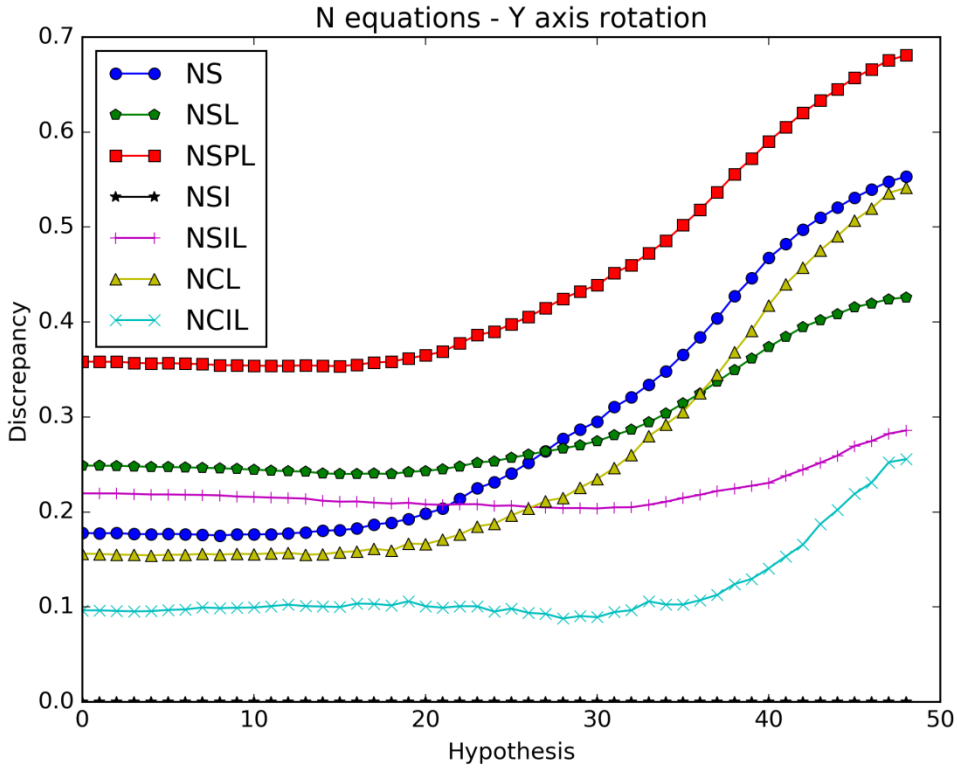

Figure 3.26: N equations.

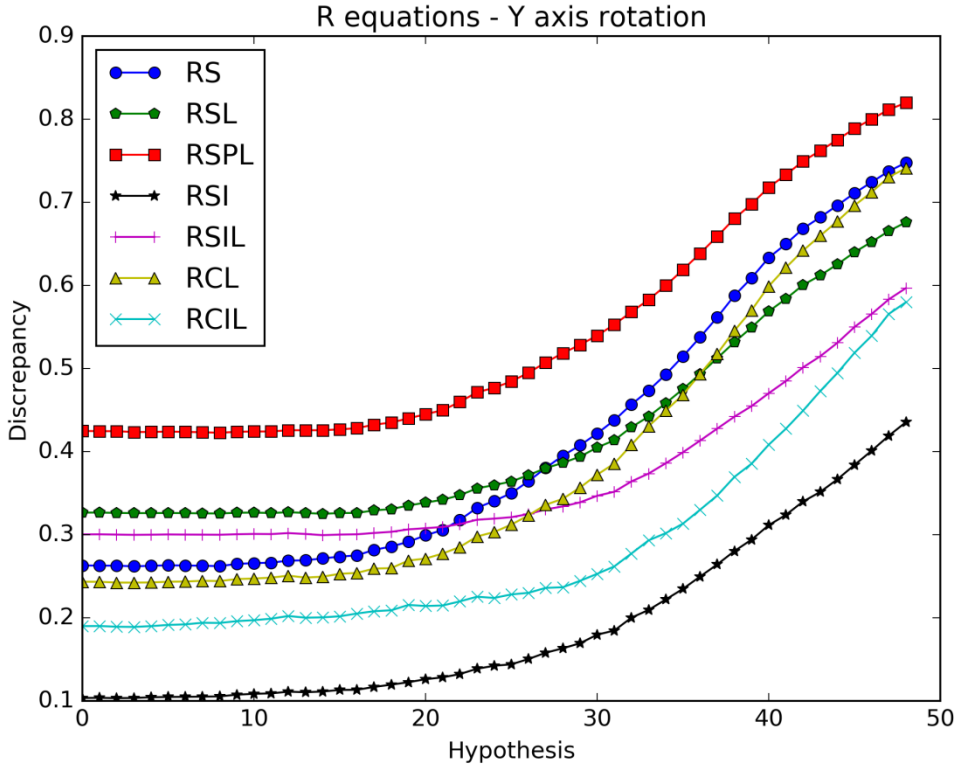

Figure 3.27: R equations.

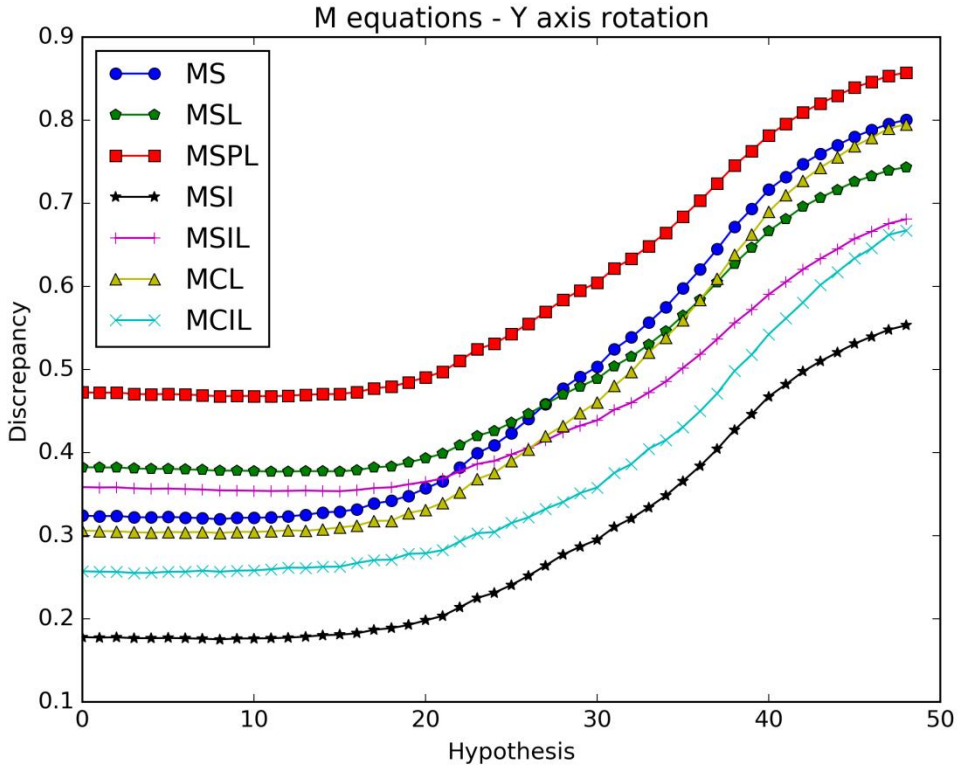

Figure 3.28: M equations.

The responses of the objective functions which use only the intersected pixels rise slower than the others either the lighting is present or absent. So, using only the intersected pixels or the penalty parameter doesn't seem to improve the efficiency of the objective functions. There isn't any lighting objective function to obviously have better response than a lighting absent objective function.

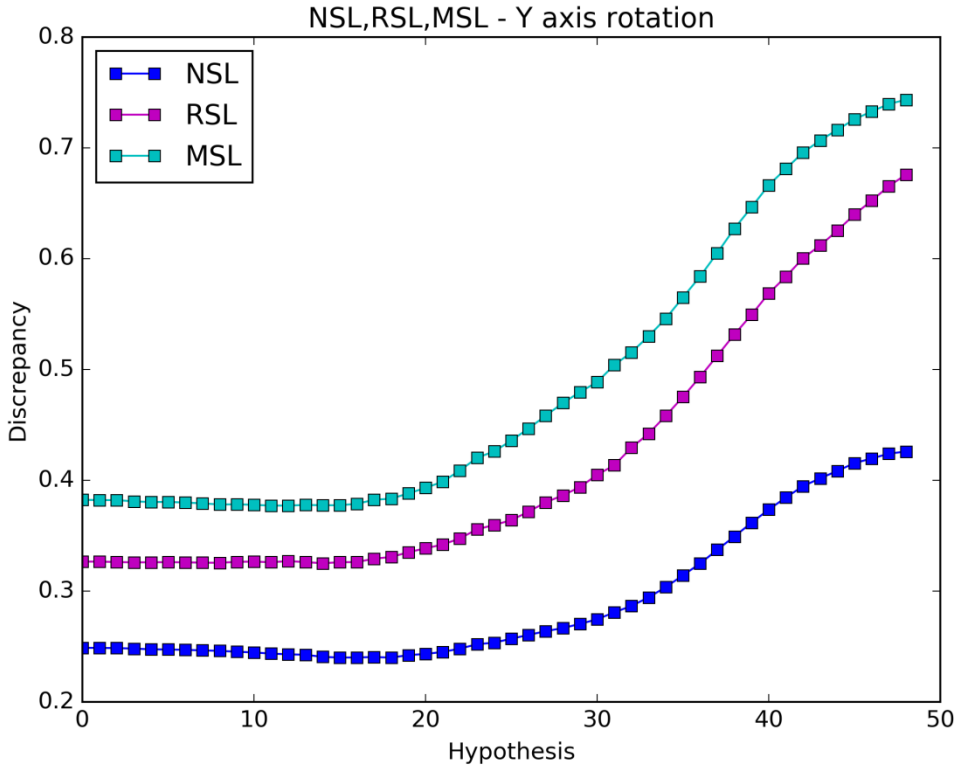

Figure 3.29: NSL, RSL, MSL equations comparison.

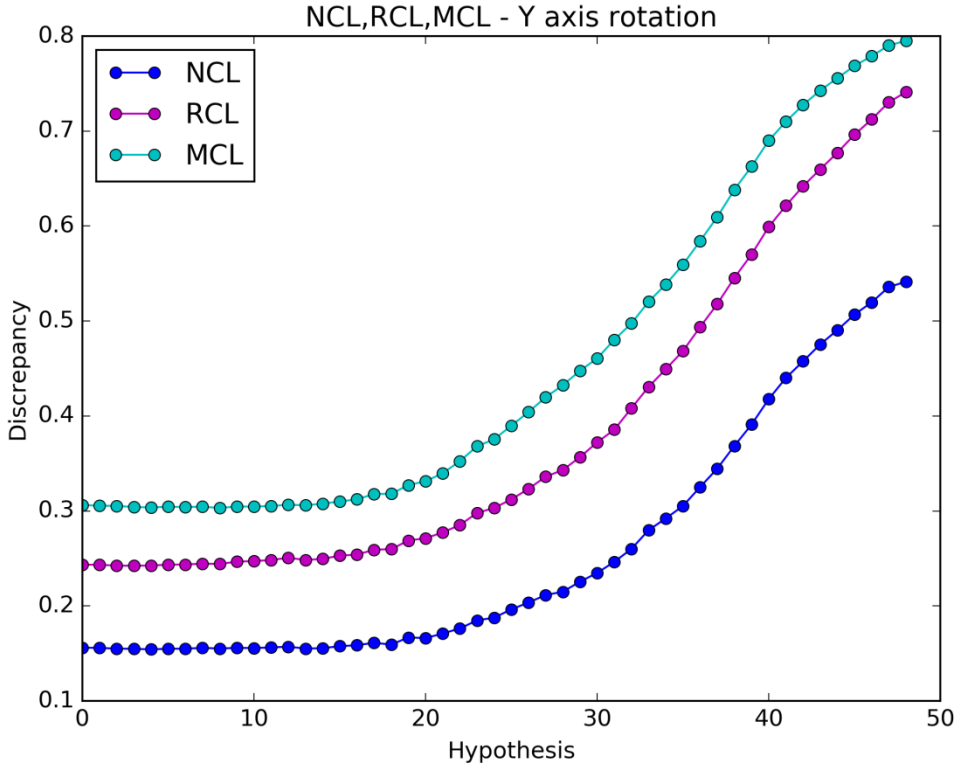

Figure 3.30: NCL, RCL, MCL equations comparison.

In this case, RSL and RCL have similar responses with MSL and MCL respectively. The use of an auxiliary parameter (mask or percent) improves the efficiency of the objective functions.

3.1.2.2. Simulated - Clockwise Rotation

The initial rendered hand pose replaces the observation to see how the objective functions operate when the light conditions and the foregrounds are similar for both the observation and the hypothesis. The initial hypothesis is rotated again in 49 steps.

The result after the foreground extraction, the initial hypothesis, the final hypothesis and their differences are shown in the figures below. Due to the perfect match the initial discrepancy image is black.

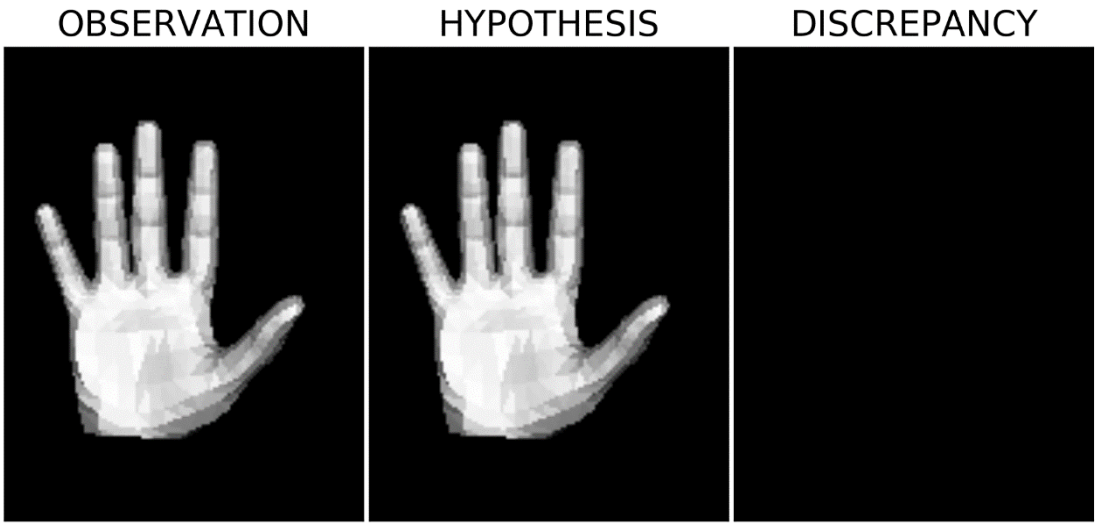

Figure 3.31: Observation (Left), initial hypothesis (Middle), and the differences (Right).

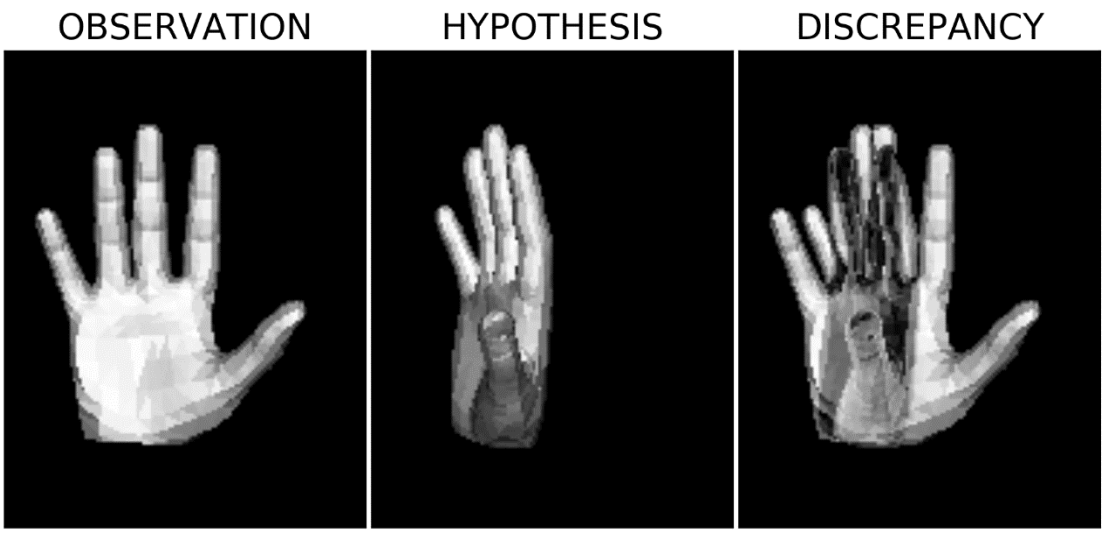

Figure 3.32: Observation (Left), final hypothesis (Middle), and the differences (Right).

The graph results can be seen below.

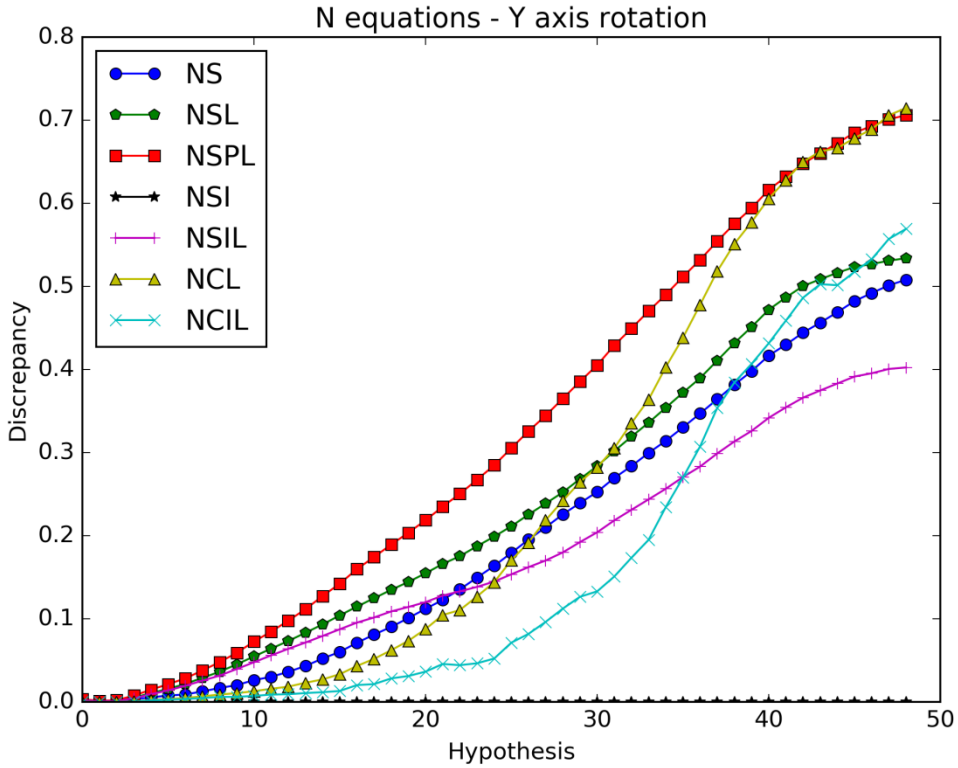

Figure 3.33: N equations.

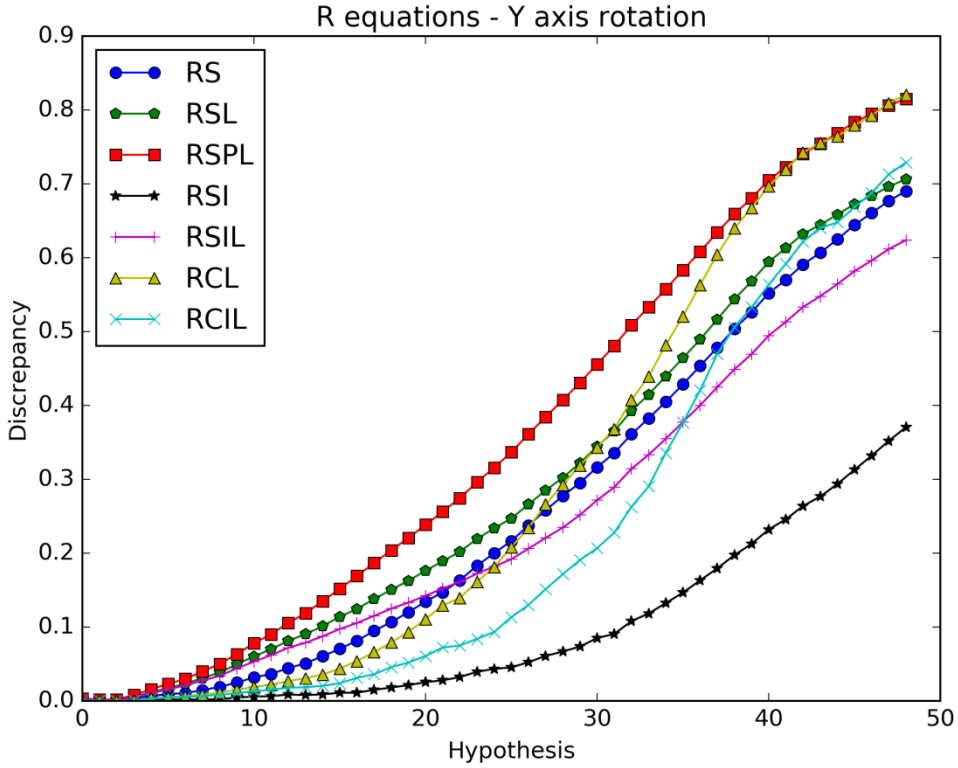

Figure 3.34: R equations.

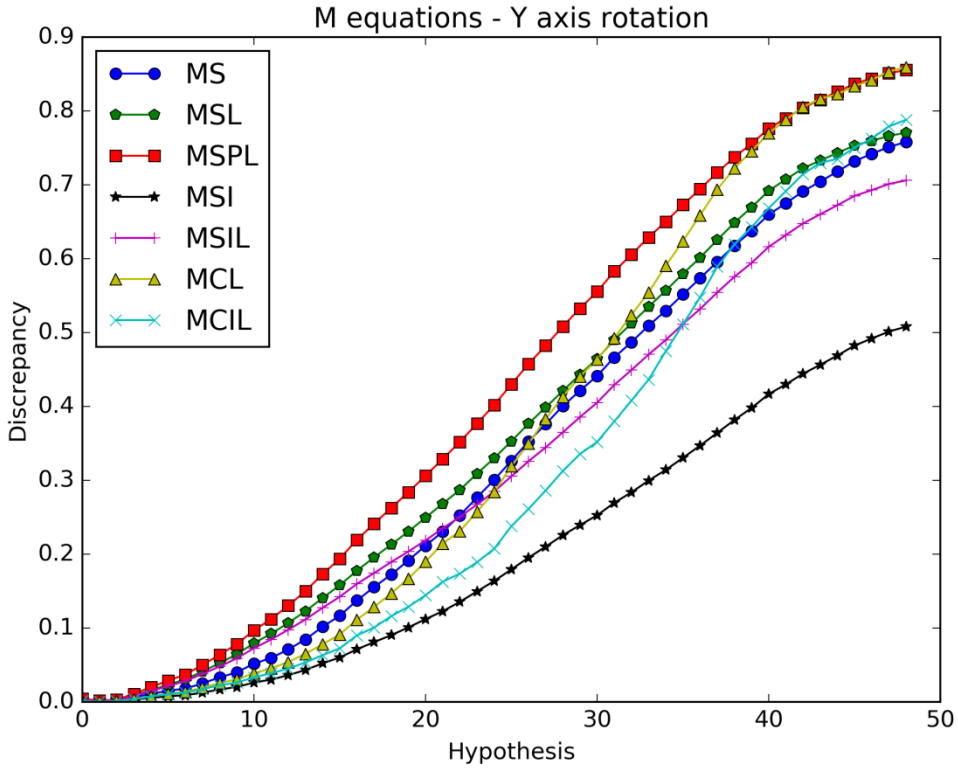

Figure 3.35: M equations.

Using only the intersected pixels doesn't improve the efficiency of the objective functions either the lighting is present or absent. All the responses of the objective functions where the lighting is present increase sharper than the responses of the objective functions where the lighting is absent. That suggests that when all the conditions are similar for both the observation and the hypotheses, the lighting presence gives better results than if the lighting was absent and those results can be further improved if the penalty parameter is included. The NSPL, RSPL, MSPL objective functions are the best in each figure.

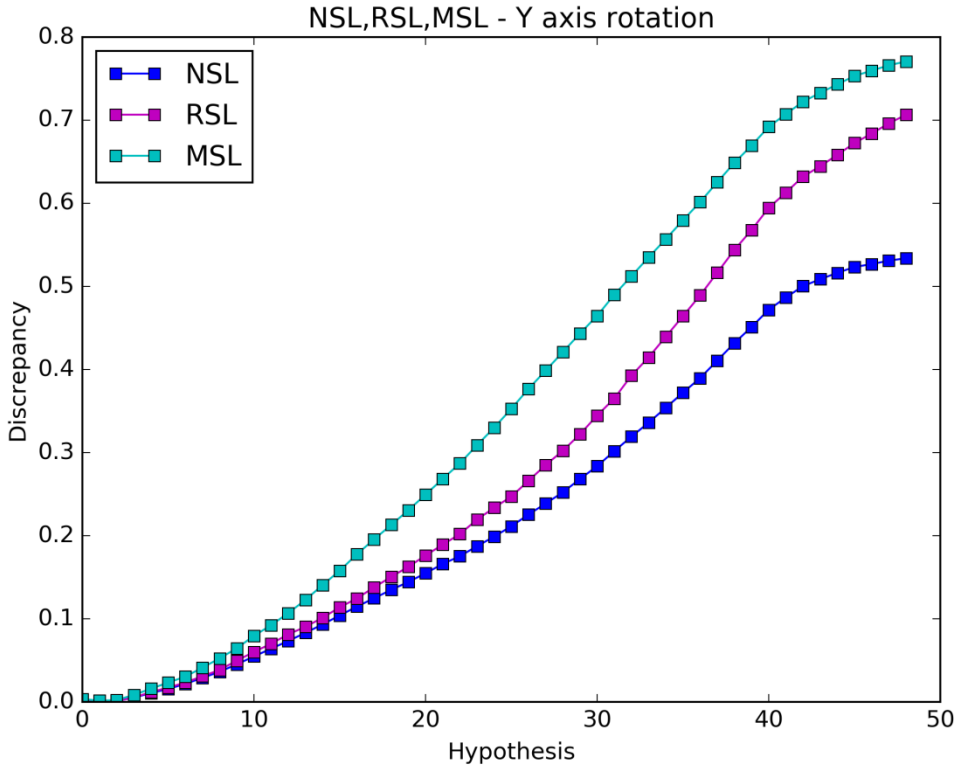

Figure 3.36: NSL, RSL, MSL equations comparison.

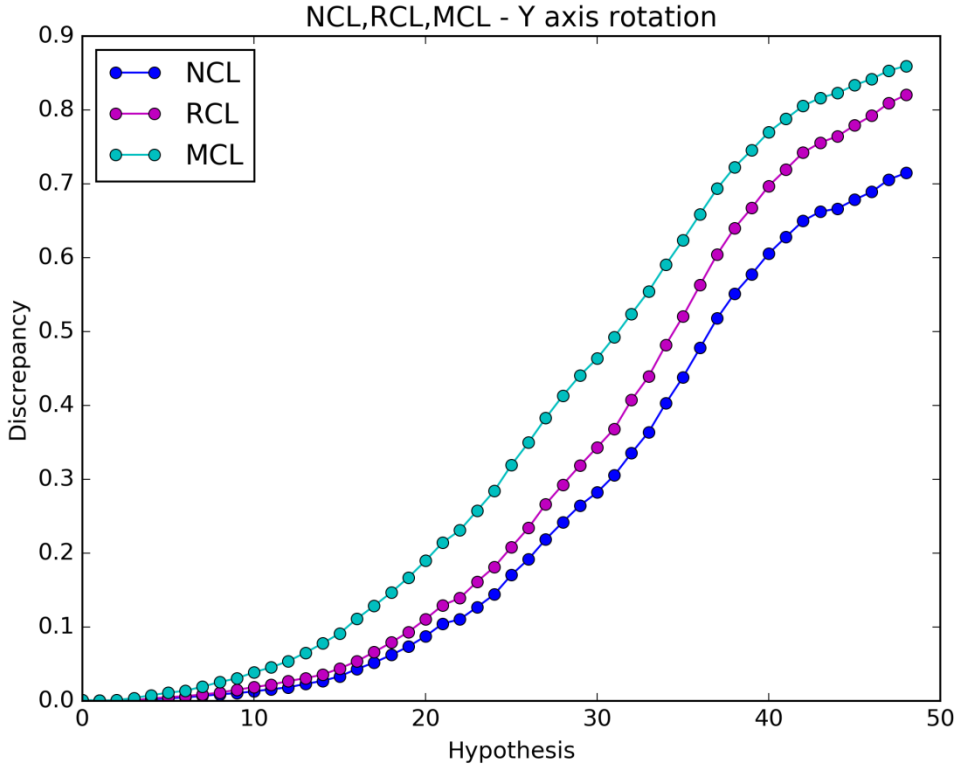

Figure 3.37: NCL, RCL, MCL equations comparison.

From the graphs above we can presume that the mask parameter (MSL, MCL) overweighs both the percent parameter (RSL, RCL) and the case where no auxiliary parameter is used (NSL, NCL). In general, the use of an auxiliary parameter makes the responses of the objective functions increase faster.

3.1.2.3. Counter Clockwise Rotation

The result after the foreground extraction, the initial hypothesis, the final hypothesis and their differences are shown in the figures below.

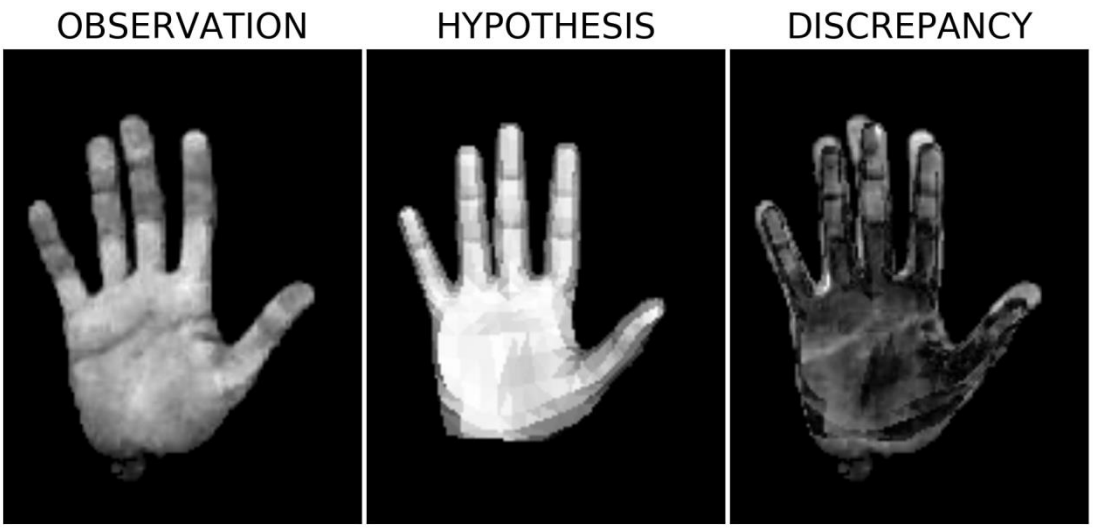

Figure 3.38: Observation (Left), initial hypothesis (Middle), and the differences (Right).

In this case, the initial hypothesis is rotated counter clockwise in 49 steps.

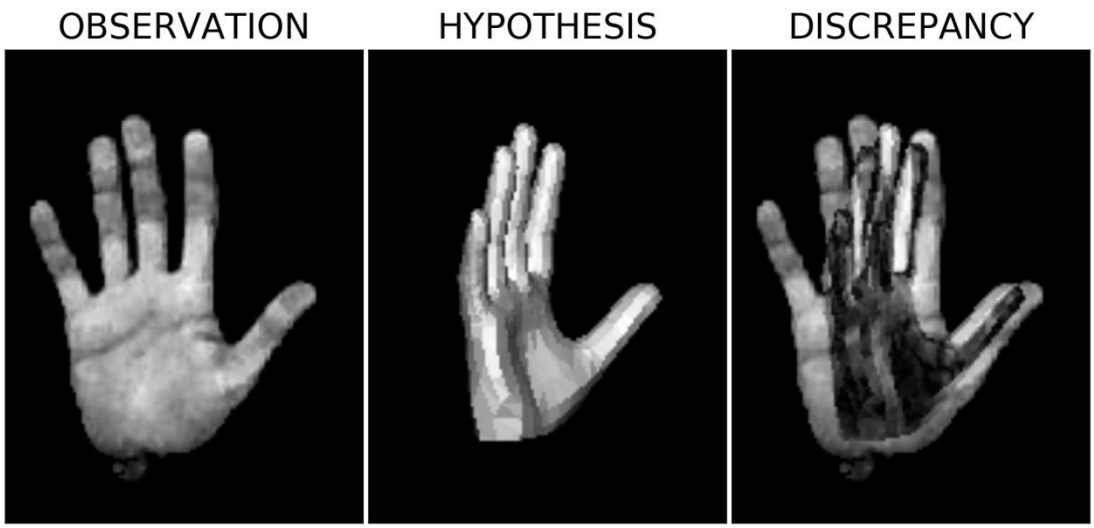

Figure 3.39: Observation (Left), final hypothesis (Middle), and the differences (Right).

The following figures show the cases where a) the lighting is absent, b) the penalty component is used and c) only the intersection pixels are used.

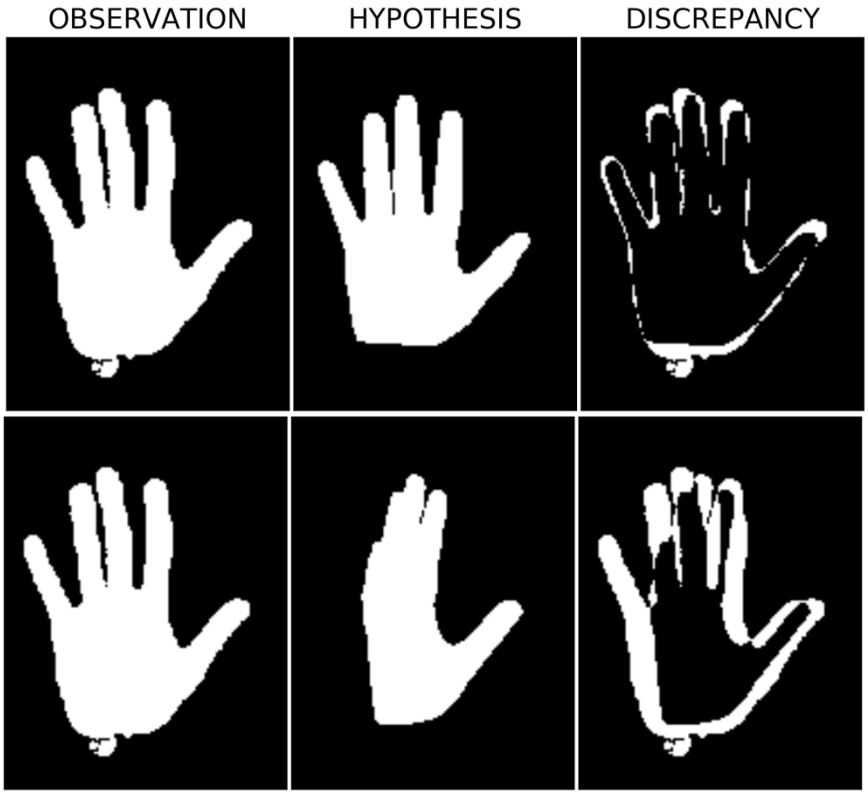

Figure 3.40: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent.

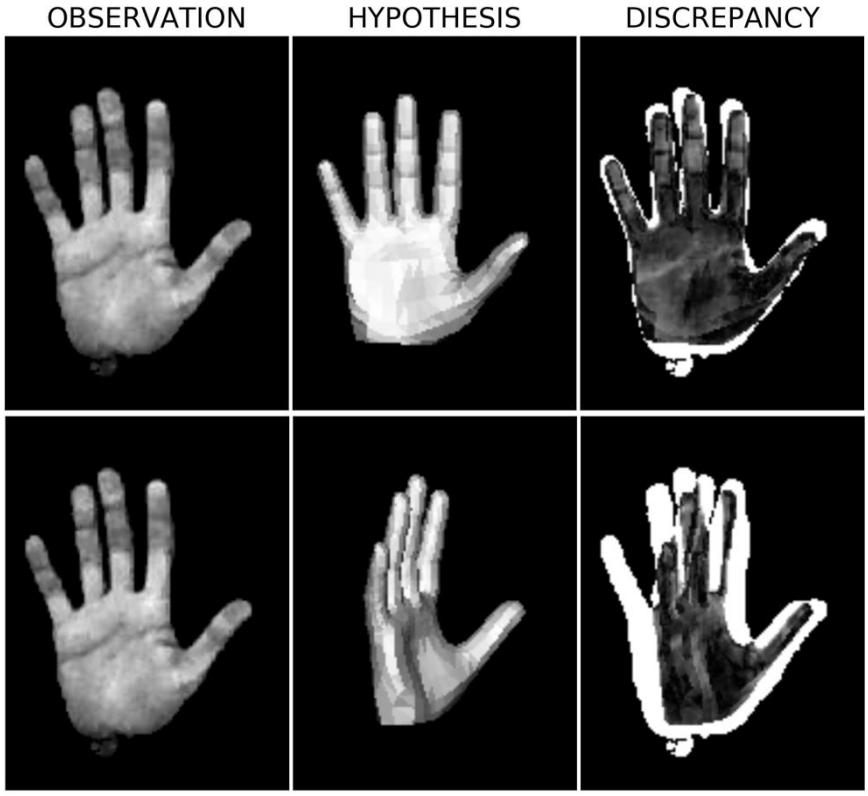

Figure 3.41: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty component is used.

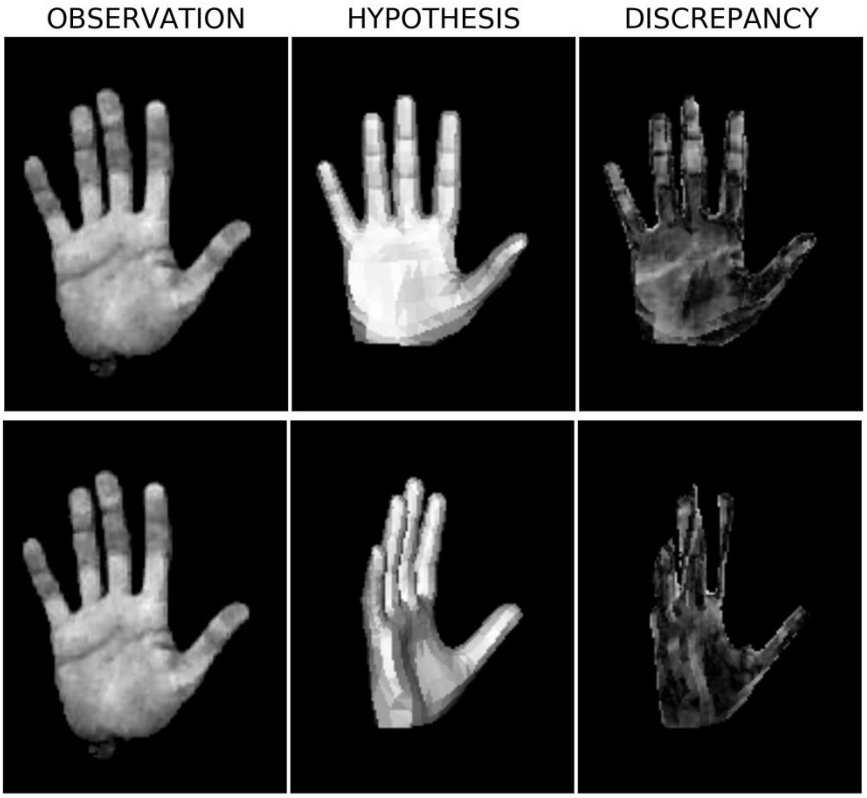

Figure 3.42: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used.

The graph results of the objective functions can be seen in the following figures.

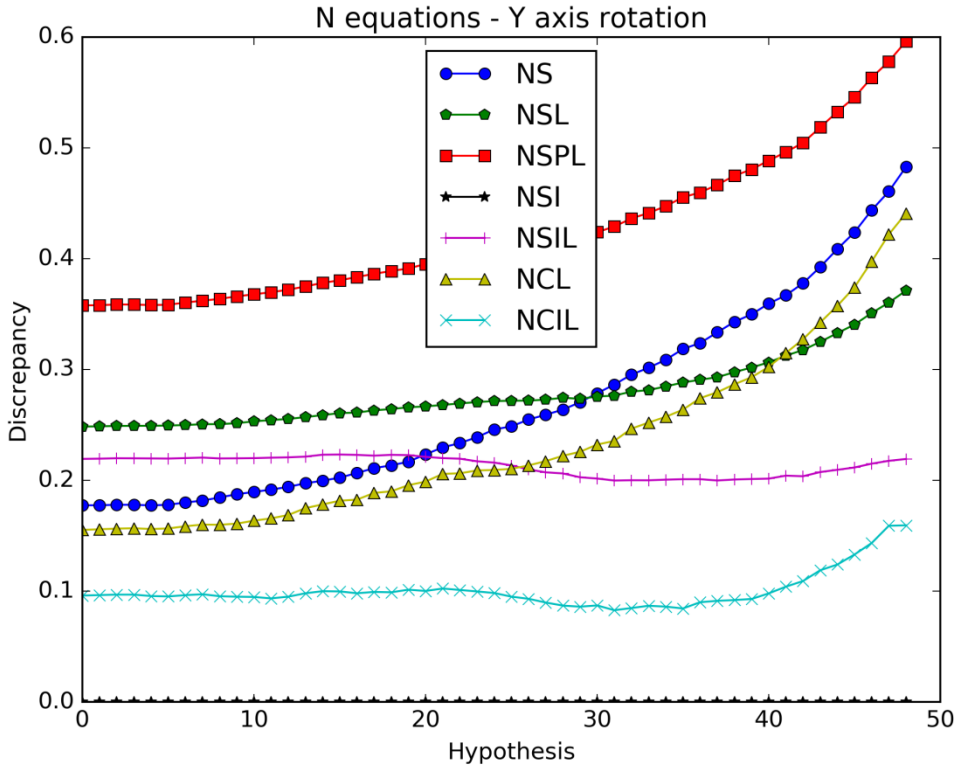

Figure 3.43: N equations.

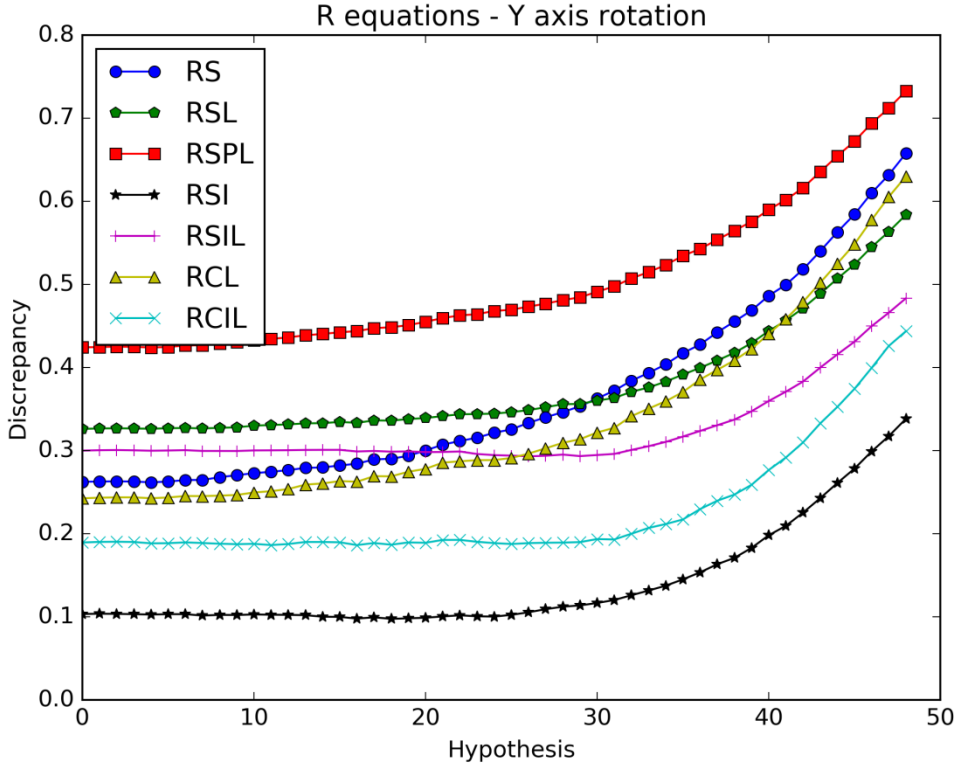

Figure 3.44: R equations.

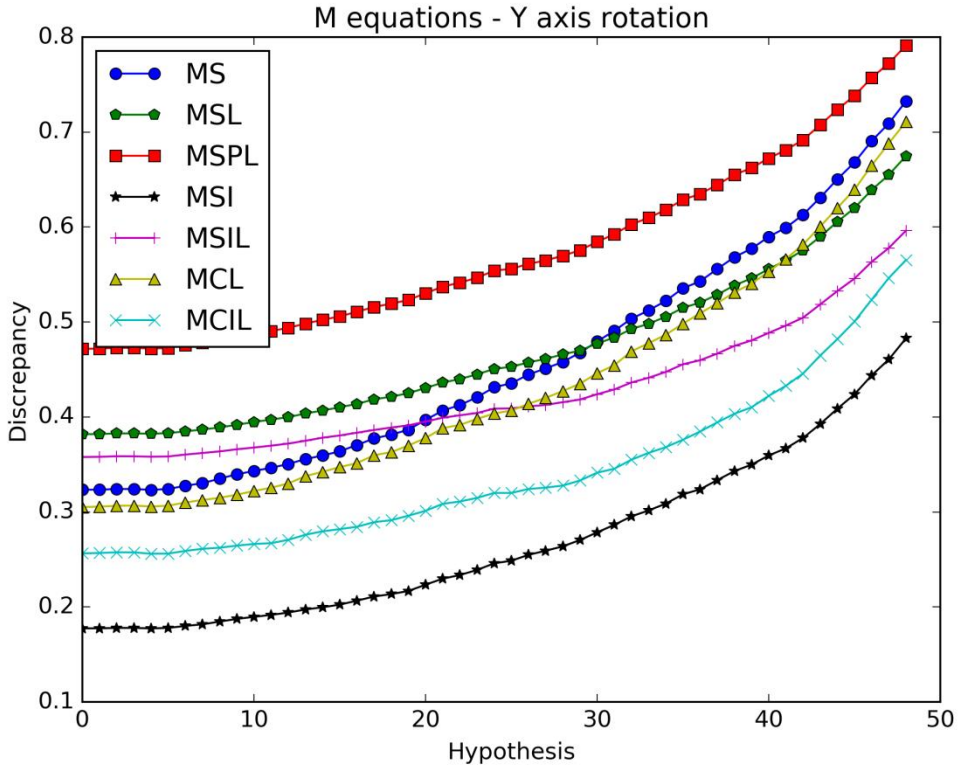

Figure 3.45: M equations.

All the graphs have lower slopes in comparison with the **clockwise** rotation graphs. The position of the light source may be responsible for this. The lamp for this case was placed to the left side of the Kinect sensor so when the hypothesis rotates clockwise then it rotates towards the light source and the differences are bigger than in the case where the hypothesis rotates counter clockwise towards the shadow.

Despite this fact, we conclude to the exact same results as in the clockwise version. Using only the intersected pixels doesn't improve the efficiency of the objective functions. The lighting element in the objective functions seems to have no positive impact to the result.

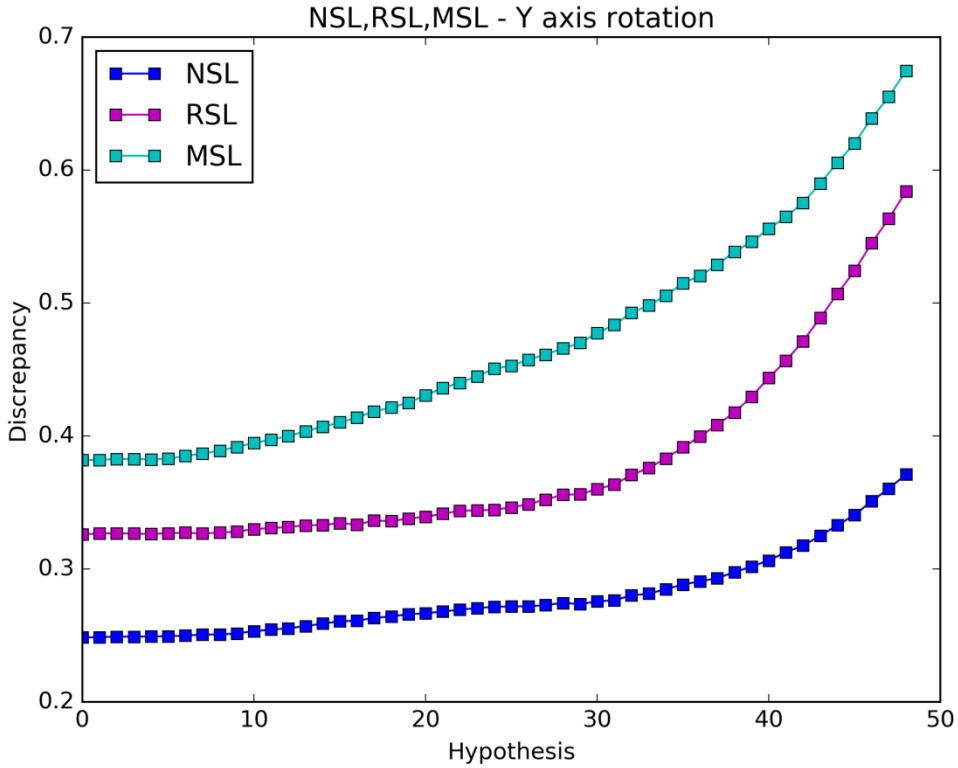

Figure 3.46: NSL, RSL, MSL equations comparison.

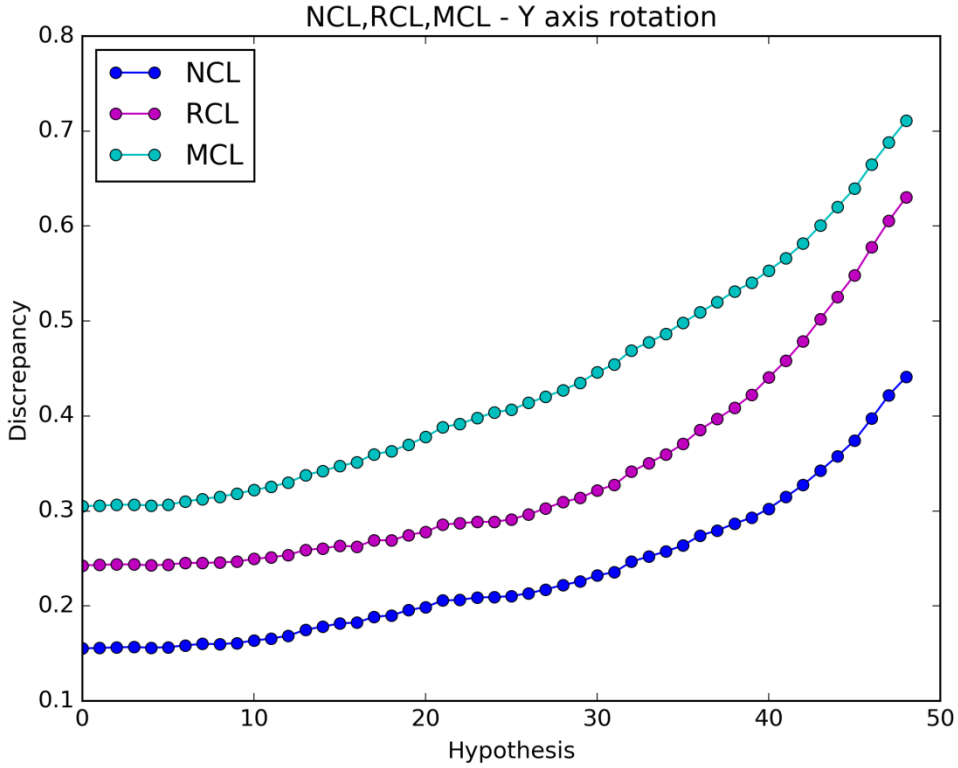

Figure 3.47: NCL, RCL, MCL equations comparison.

The efficiency of the auxiliary parameters isn't obvious from the above graphs.

The positive impact of the lighting presence and the auxiliary parameters is difficult to be noticed. For such a reason we test the objective functions again when the observation and the hypotheses have the same light conditions and similar foregrounds.

3.1.2.4. Simulated – Counter Clockwise Rotation

The initial rendered hand pose replaces the observation to see how the objective functions operate when the light conditions and the foregrounds are similar for both the observation and the hypothesis. The initial hypothesis is rotated again in 49 steps.

The result after the foreground extraction, the initial hypothesis, the final hypothesis, and their differences are shown in the figures below. Due to the perfect match, the initial discrepancy image is black.

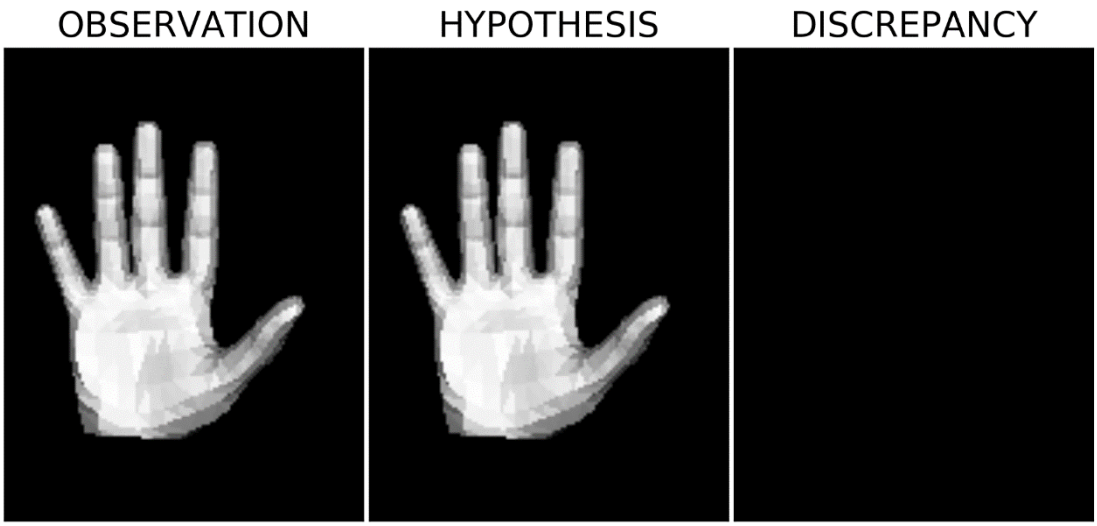

Figure 3.48: Observation (Left), initial hypothesis (Middle), and the differences (Right).

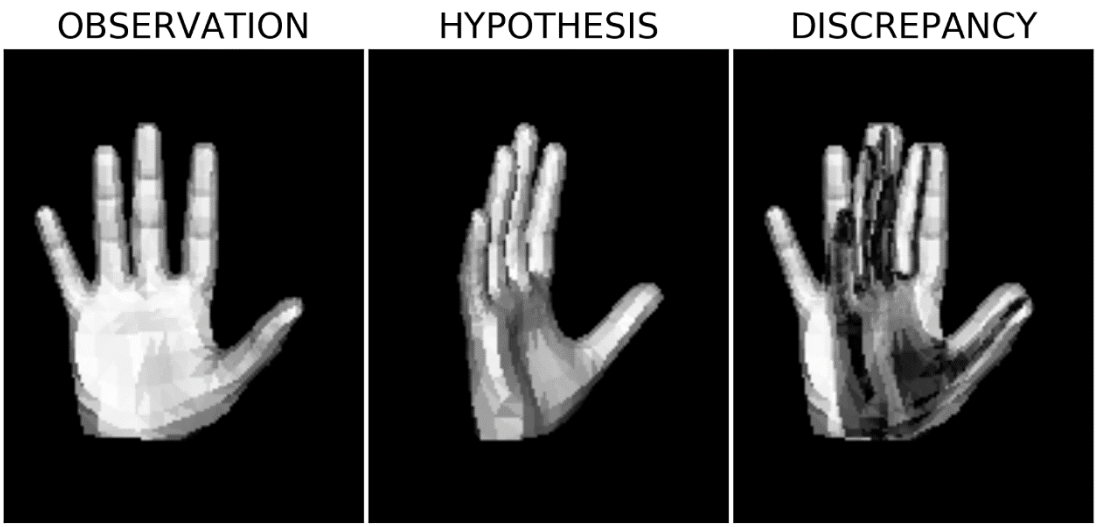

Figure 3.49: Observation (Left), final hypothesis (Middle), and the differences (Right).

The graphs of the objective functions can be seen below.

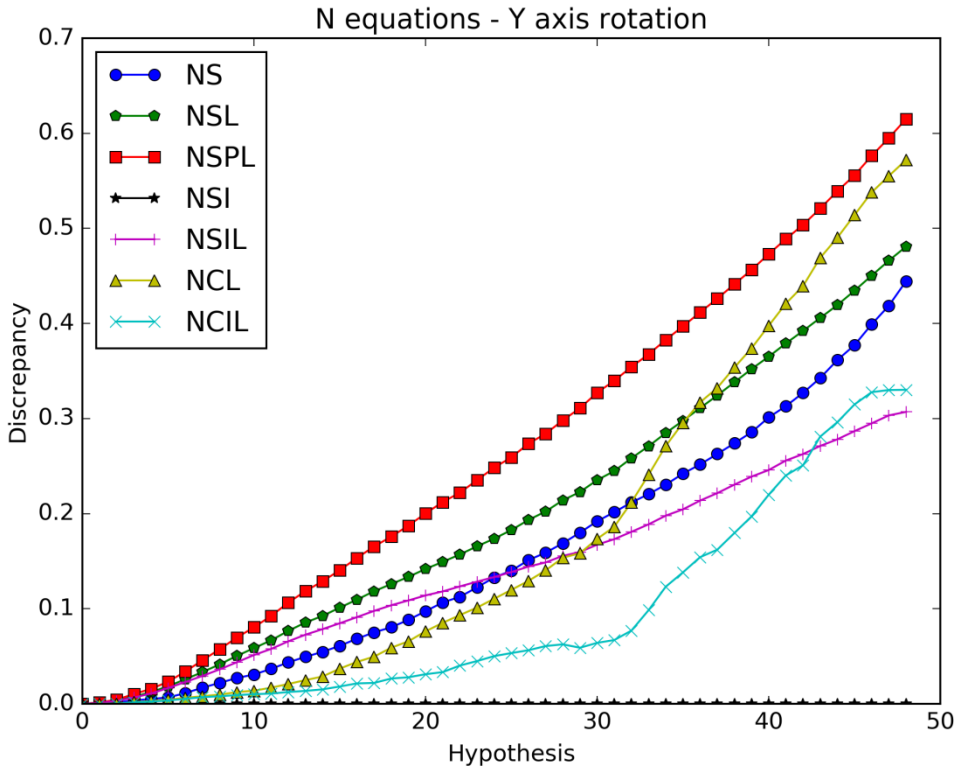

Figure 3.50: N equations.

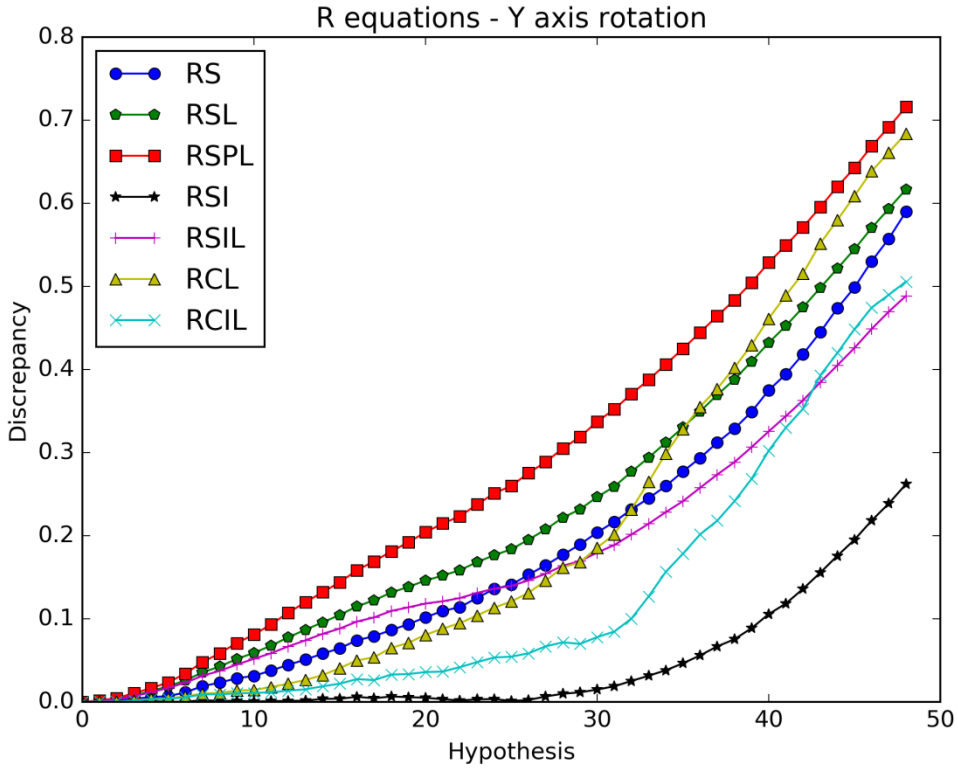

Figure 3.51: R equations.

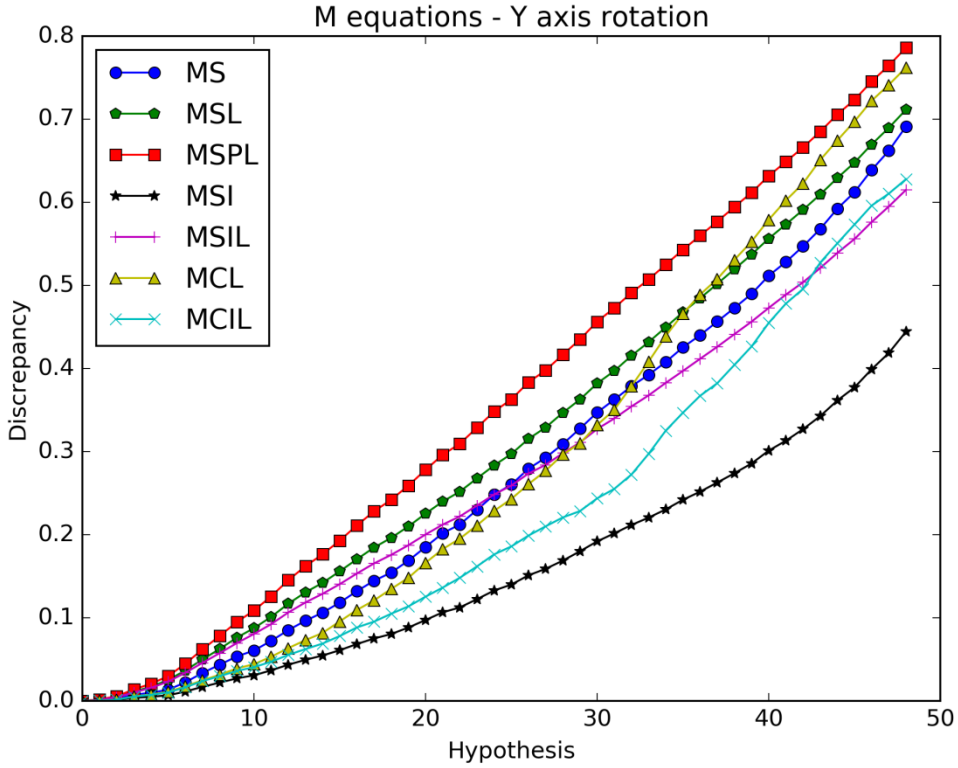

Figure 3.52: M equations.

When all the conditions are similar for both the observation and the hypothesis, the lighting presence gives again better results than if the lighting was absent and those results can be further improved if the penalty parameter is included. Using only the intersected pixels doesn't improve the efficiency of the objective functions either the lighting is present or absent.

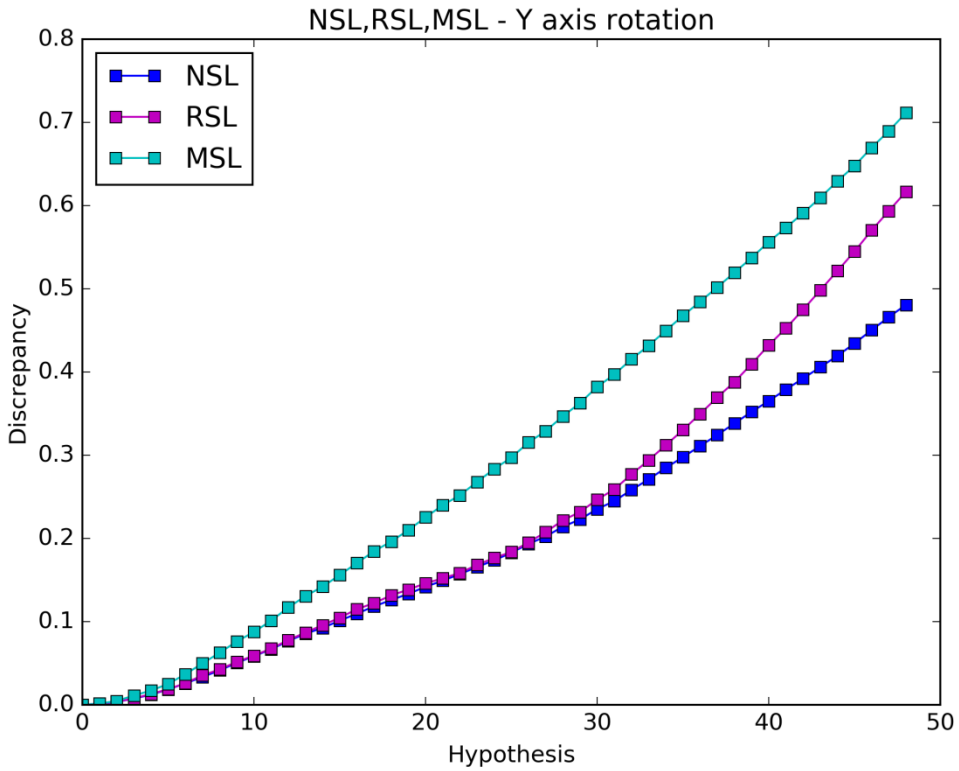

Figure 3.53: NSL, RSL, MSL equations comparison.

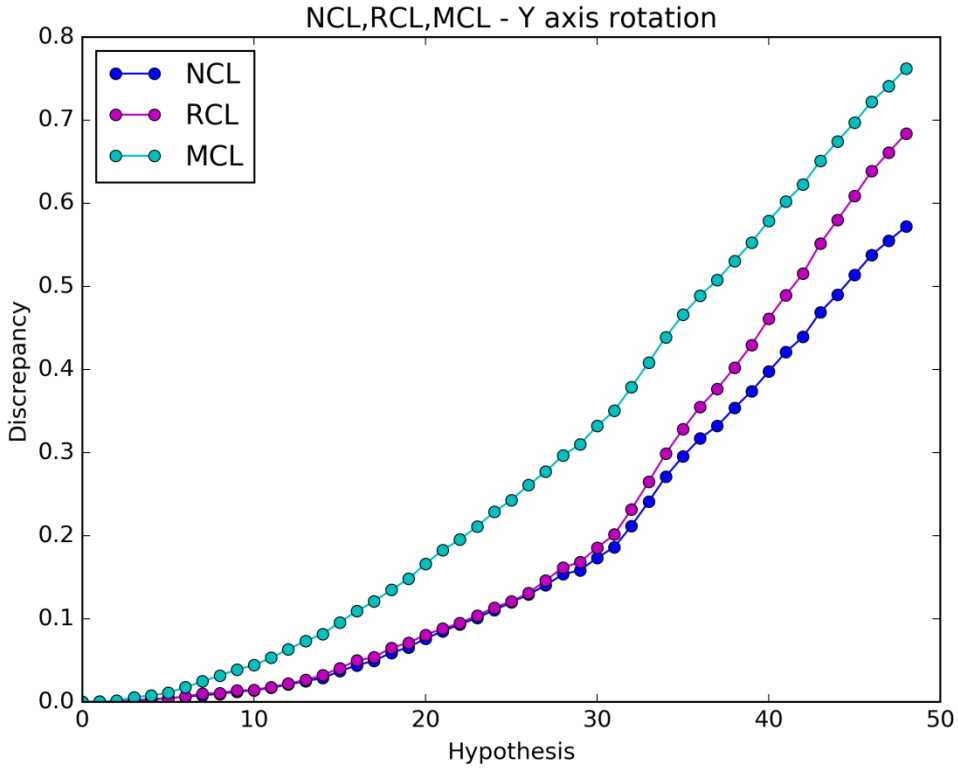

Figure 3.54: NCL, RCL, MCL equations comparison.

The mask parameter is the best auxiliary parameter again.

3.1.3. Case 3: Y axis rotation 2

The RGB image used in this case can be seen in the image below.

Figure 3.19: RGB image of Case 3

The hand in the RGB image is facing left relative to the Kinect camera.

3.1.3.1. Counter Clockwise Rotation

The result after the foreground extraction, the initial hypothesis, the final hypothesis and their differences are shown in the figures below.

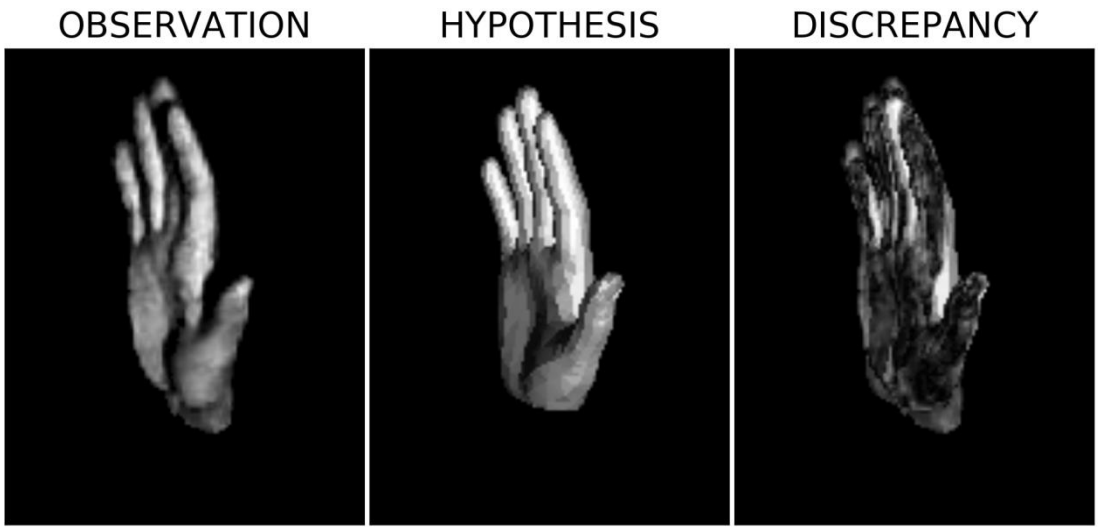

Figure 3.55: Observation (Left), initial hypothesis (Middle), and the differences (Right).

Then, the initial hypothesis is rotated counter clockwise in 49 steps.

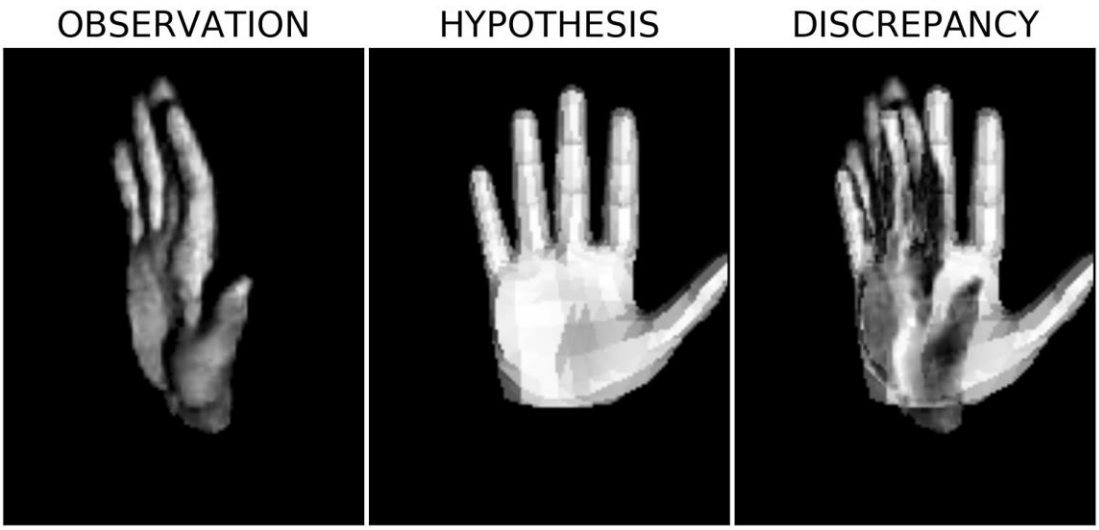

Figure 3.56: Observation (Left), final hypothesis (Middle), and the differences (Right).

The following figures show the cases where a) the lighting is absent, b) the penalty component is used and c) only the intersection pixels are used.

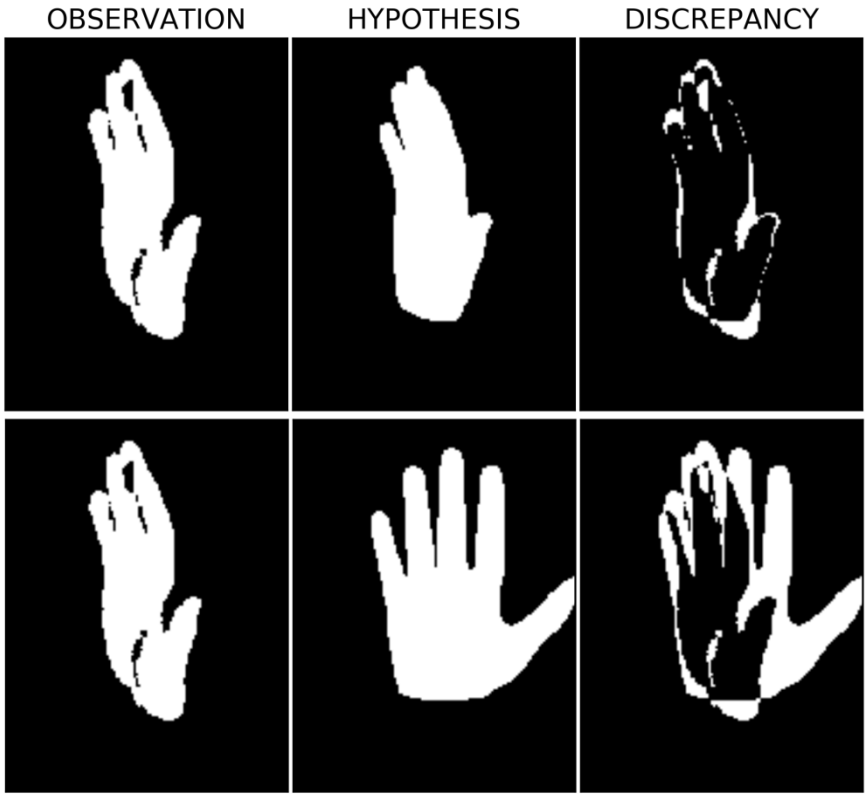

Figure 3.57: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent.

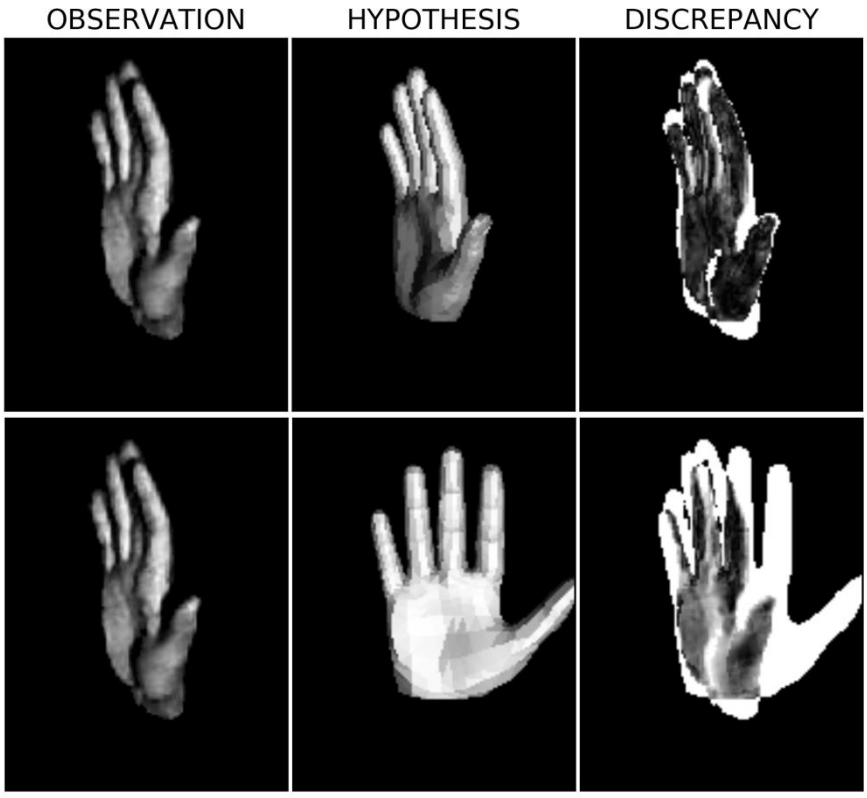

Figure 3.58: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty component is used.

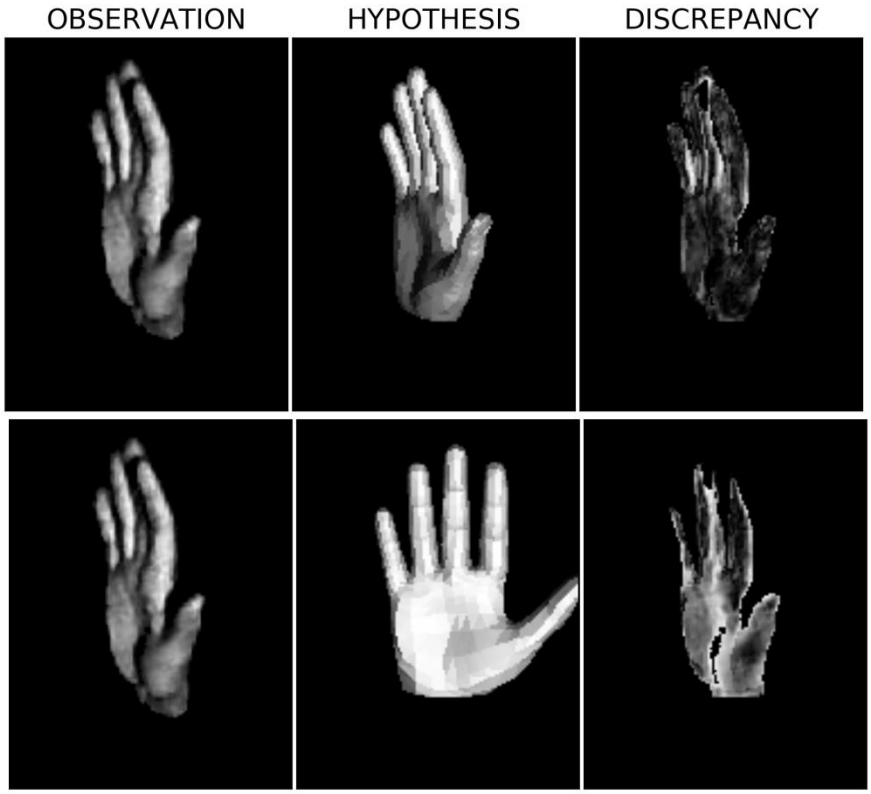

Figure 3.59: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used.

The graph results of the objective functions can be seen in the following figures.

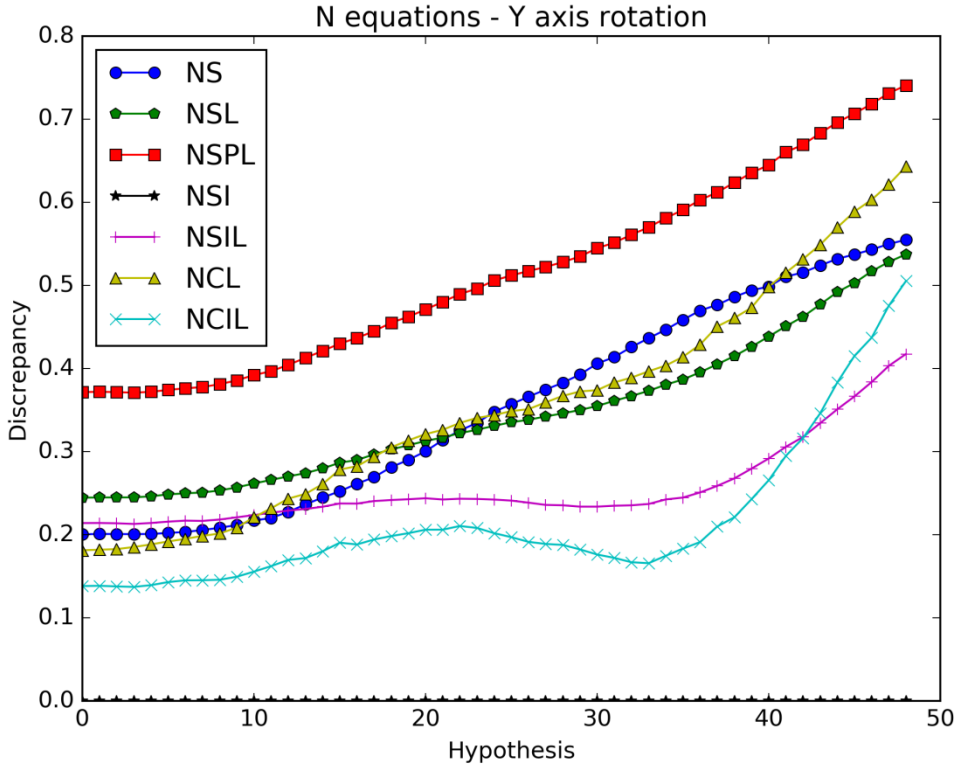

Figure 3.60: N equations.

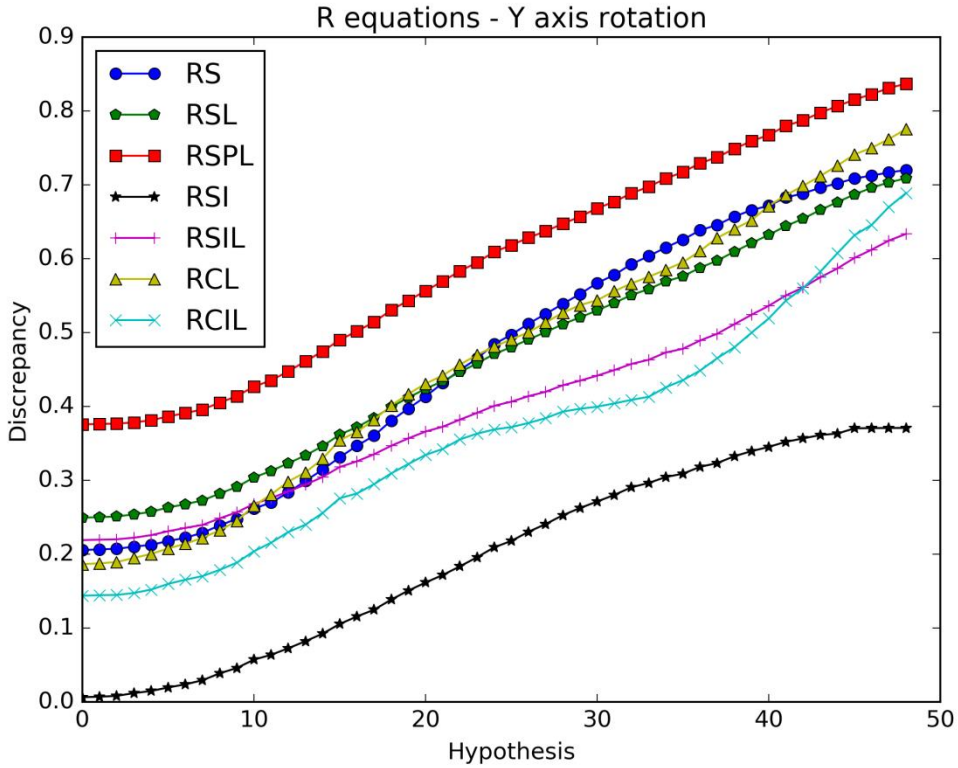

Figure 3.61: R equations.

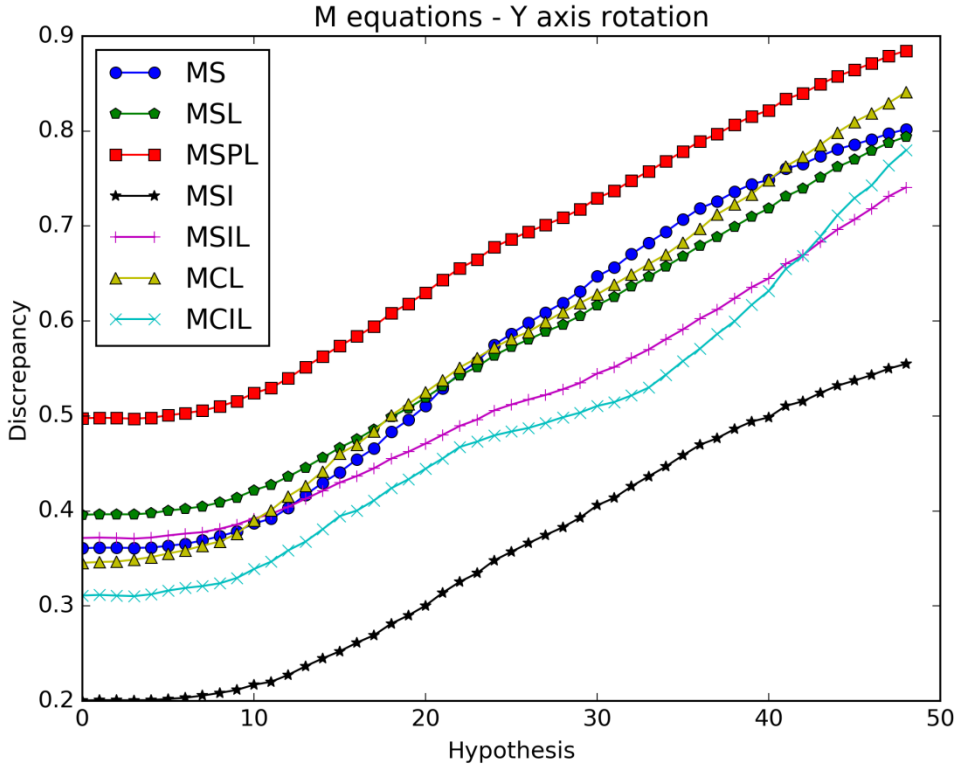

Figure 3.62: M equations.

Using only the intersected pixels doesn't seem to improve the efficiency of the objective functions either the lighting is present or absent. There is no obvious difference between the other graphs in whichever category we look at. All the responses are similar and using the lighting information didn't improve the efficiency of the objective functions.

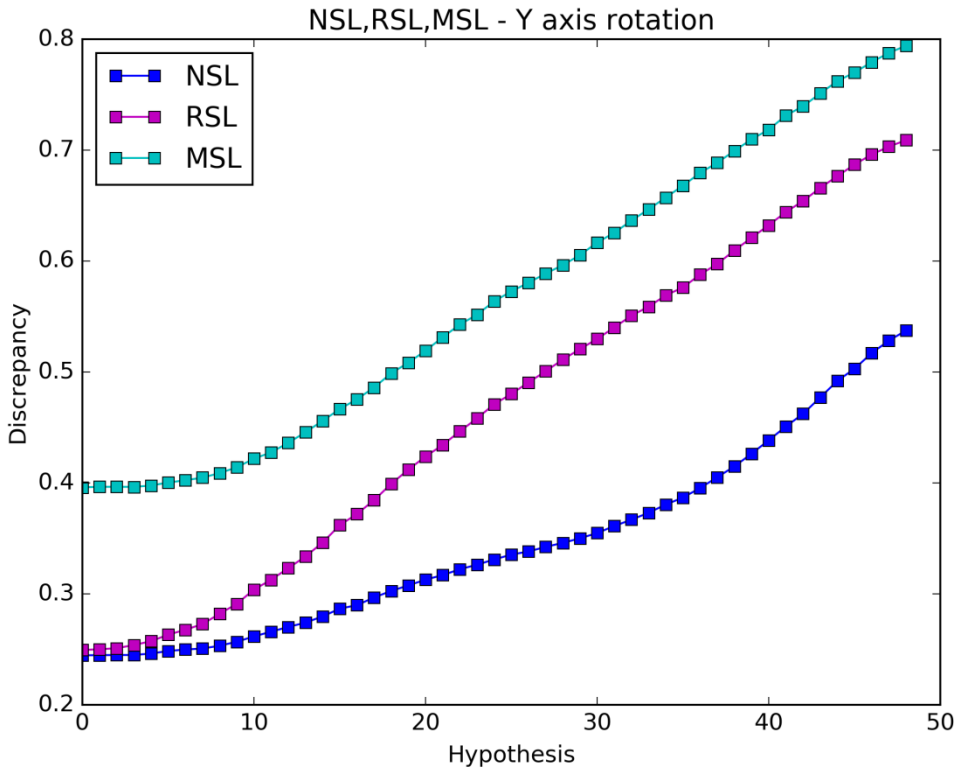

Figure 3.63: NCL, RCL, MCL equations comparison.

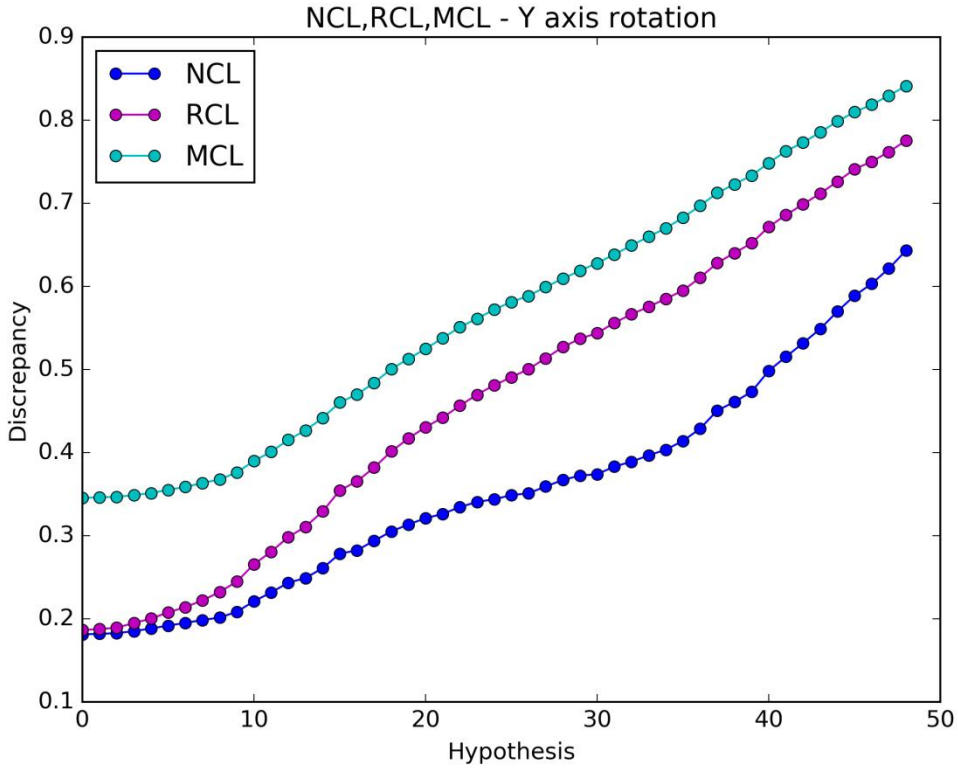

Figure 3.64: NSL, RSL, MSL equations comparison.

In this case, the percent parameter seems to have an advantage over the mask parameter since the RSL and RCL start from a lower point.

3.1.3.2. Simulated – Counter Clockwise Rotation

The initial rendered hand pose replaces the observation to see how the objective functions operate when the light conditions and the foregrounds are similar for both the observation and the hypothesis. The initial hypothesis is rotated again in 49 steps.

The result after the foreground extraction, the initial hypothesis, the final hypothesis and their differences are shown in the figures below. Due to the perfect match the initial discrepancy image is black.

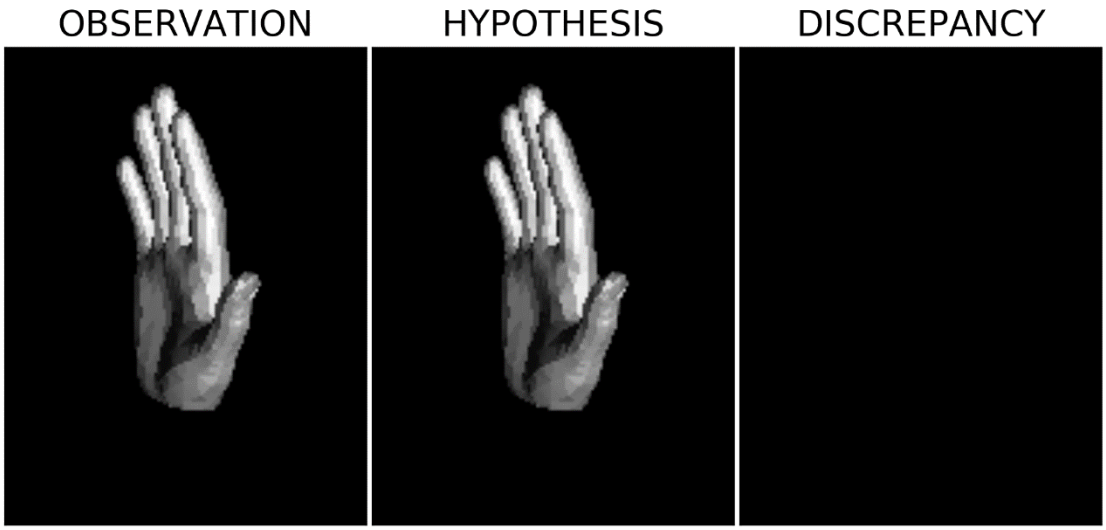

Figure 3.65: Observation (Left), initial hypothesis (Middle), and the differences (Right).

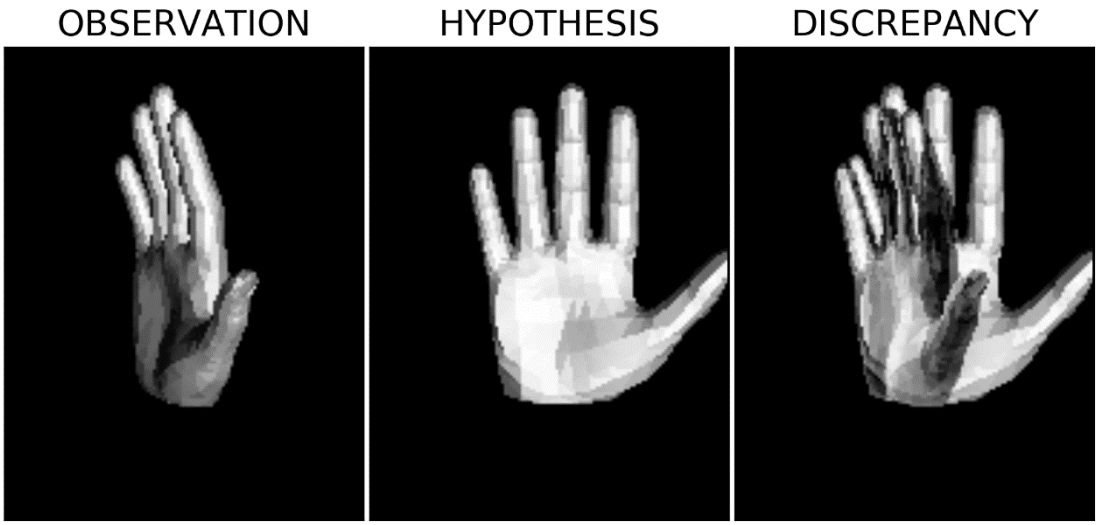

Figure 3.66: Observation (Left), final hypothesis (Middle), and the differences (Right).

The graph results of the objective functions can be seen in the following figures.

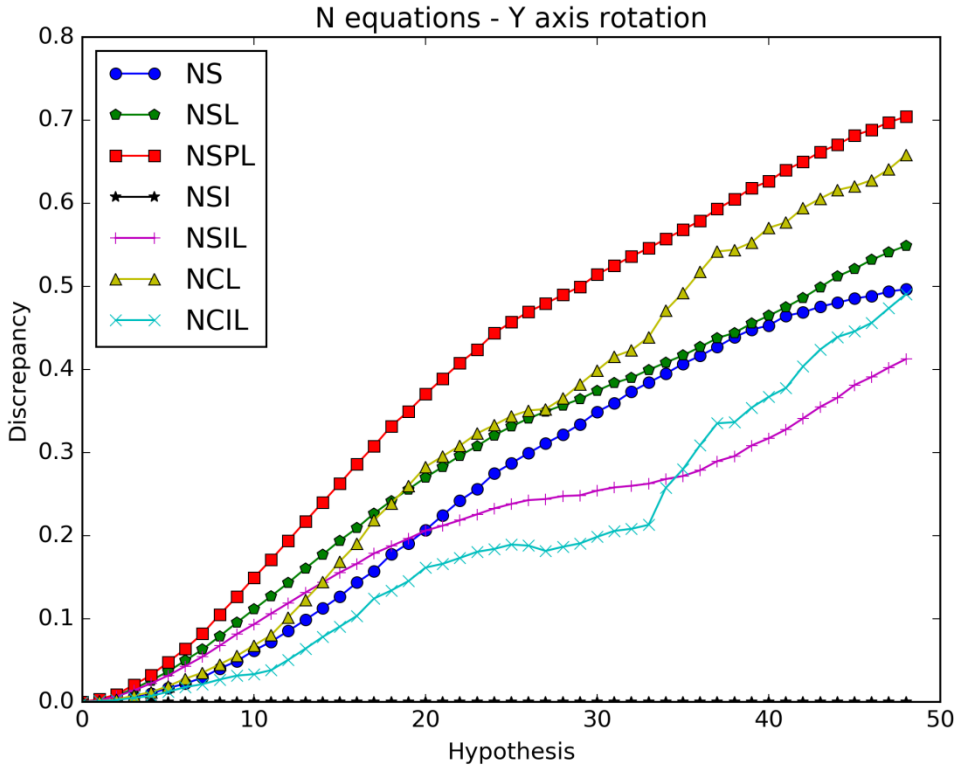

Figure 3.67: N equations.

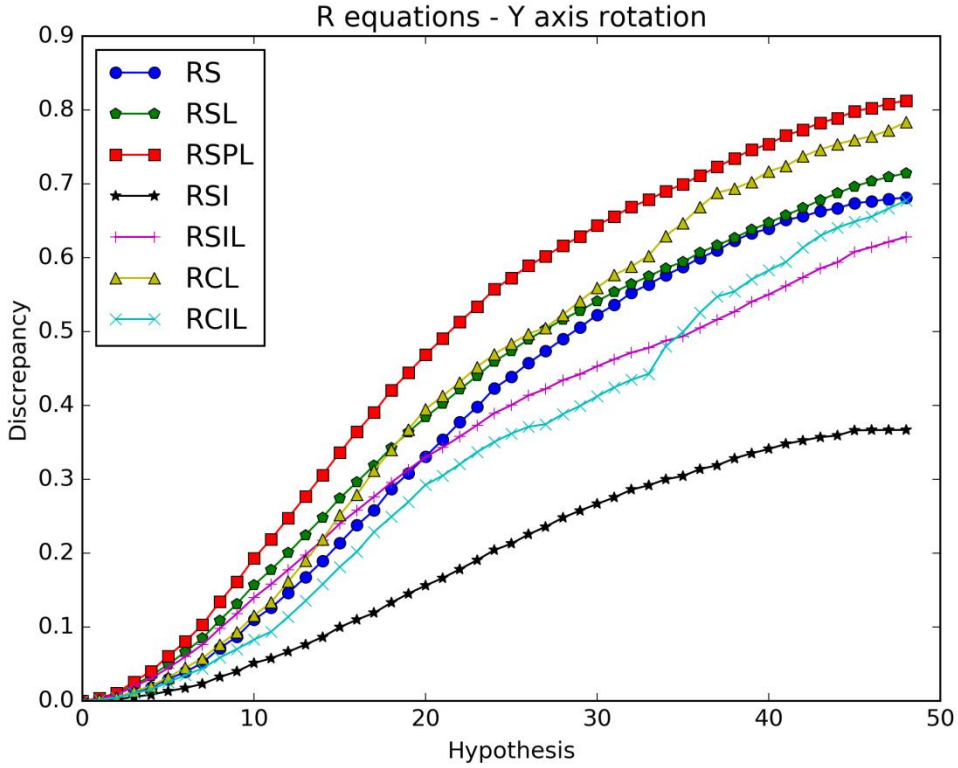

Figure 3.68: R equations.

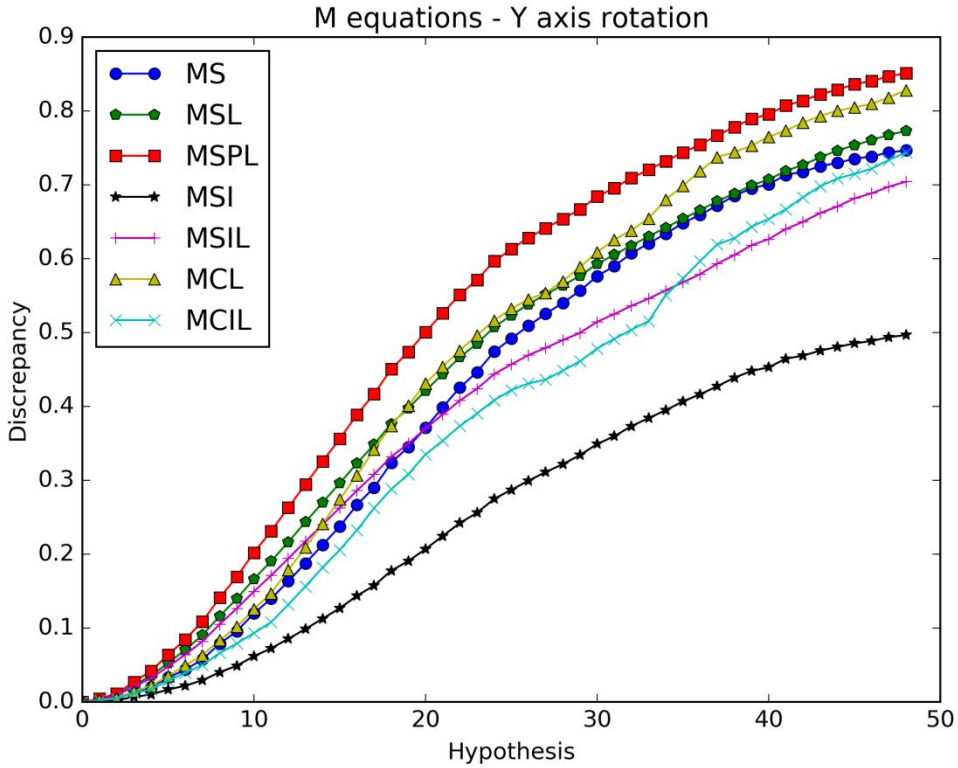

Figure 3.69: M equations.

When all the conditions are similar for both the observation and the hypothesis, the lighting presence gives again better results than if the lighting was absent and those results can be further improved if the penalty parameter is included. Using only the intersected pixels doesn't improve the efficiency of the objective functions either the lighting is present or absent.

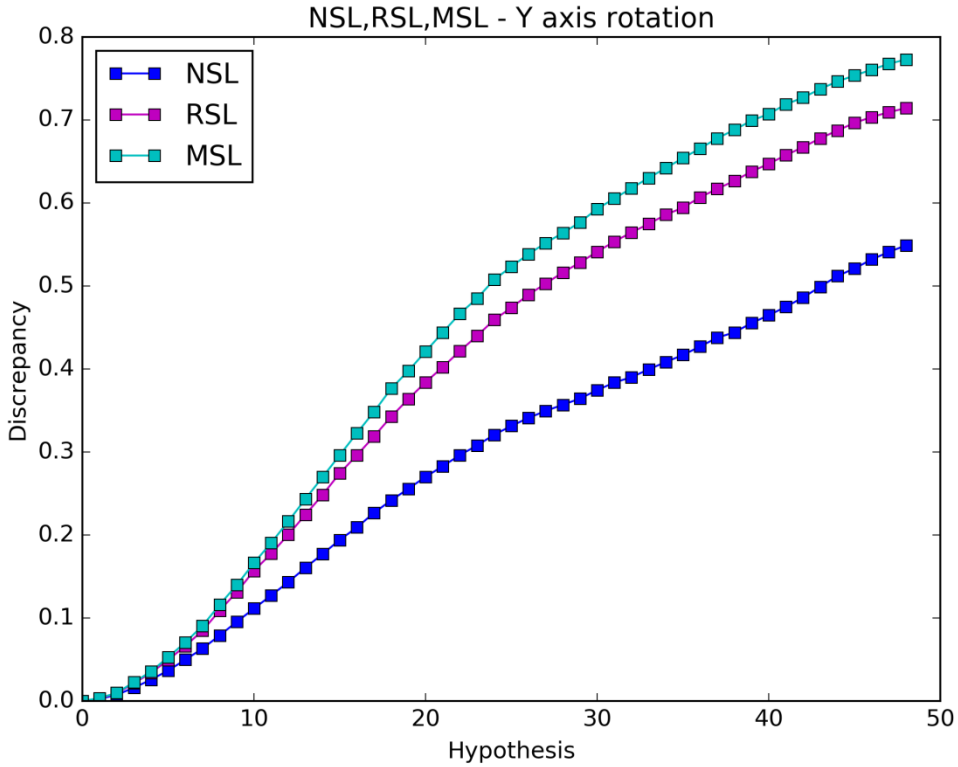

Figure 3.70: NCL, RCL, MCL equations comparison.

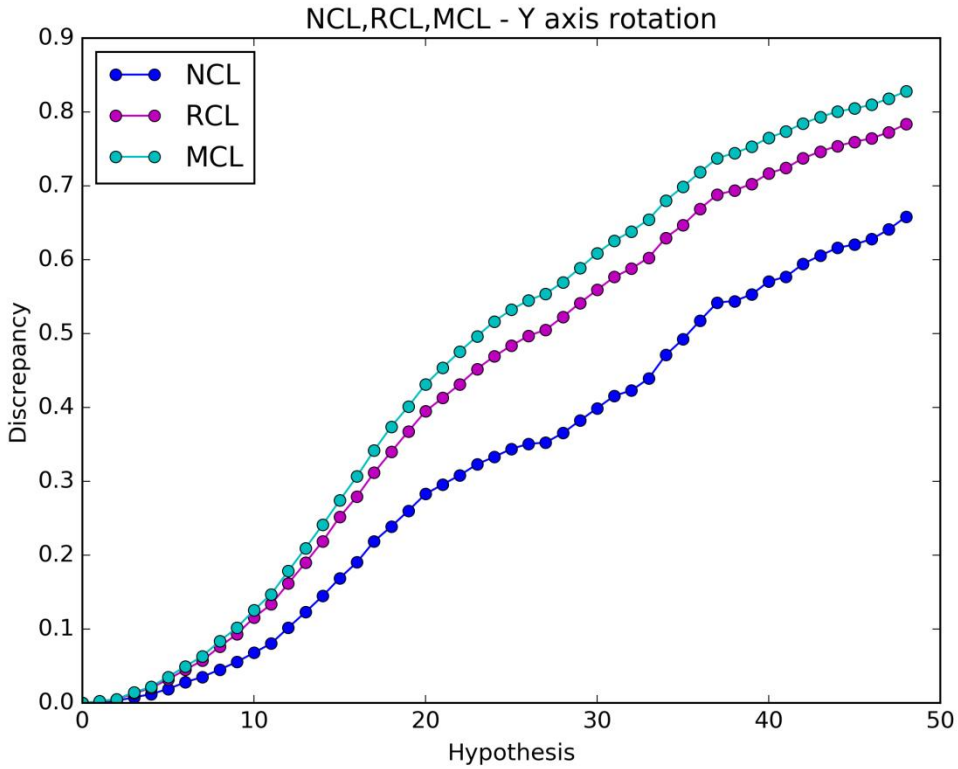

Figure 3.71: NSL, RSL, MSL equations comparison.

The graphs above suggest that the mask parameter enhances the efficiency of the functions more than the percent parameter. In general, the use of an auxiliary parameter gives positive results as their responses increase faster.

3.1.4. Case 4: X axis rotation

The RGB image used in this case can be seen in the image below.

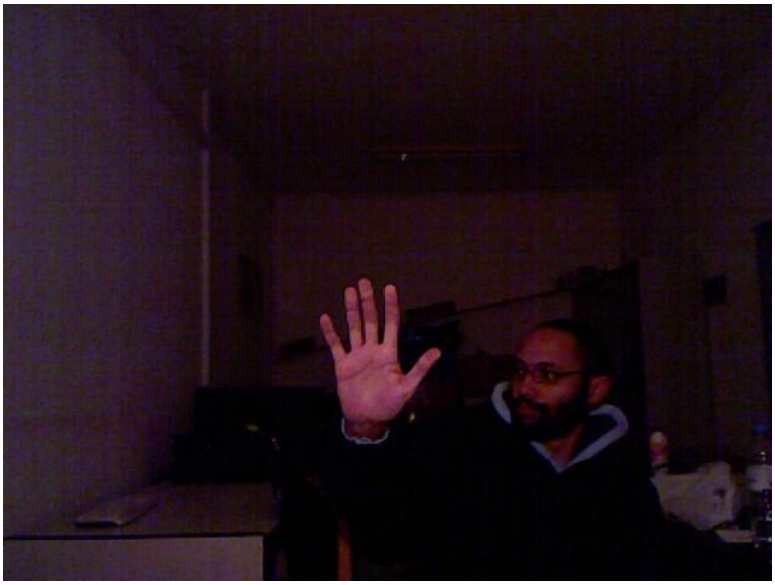

Figure 3.72: RGB image of Case 4.

3.1.4.1. Downward Rotation

The result after the foreground extraction, the initial hypothesis, the final hypothesis, and their differences are shown in the figures below.

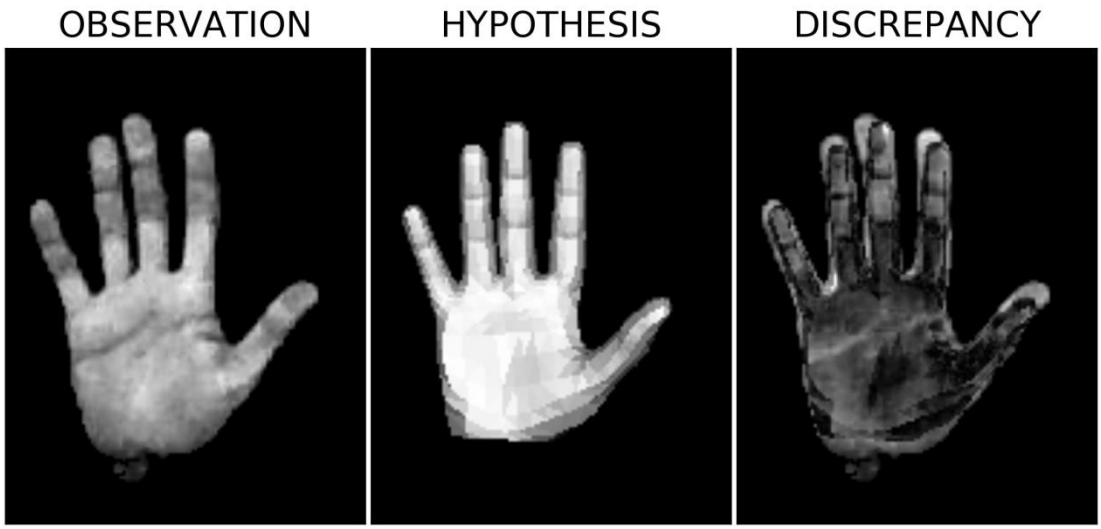

Figure 3.73: Observation (Left), initial hypothesis (Middle), and the differences (Right).

Then, the initial hypothesis is rotated downward in 49 steps.

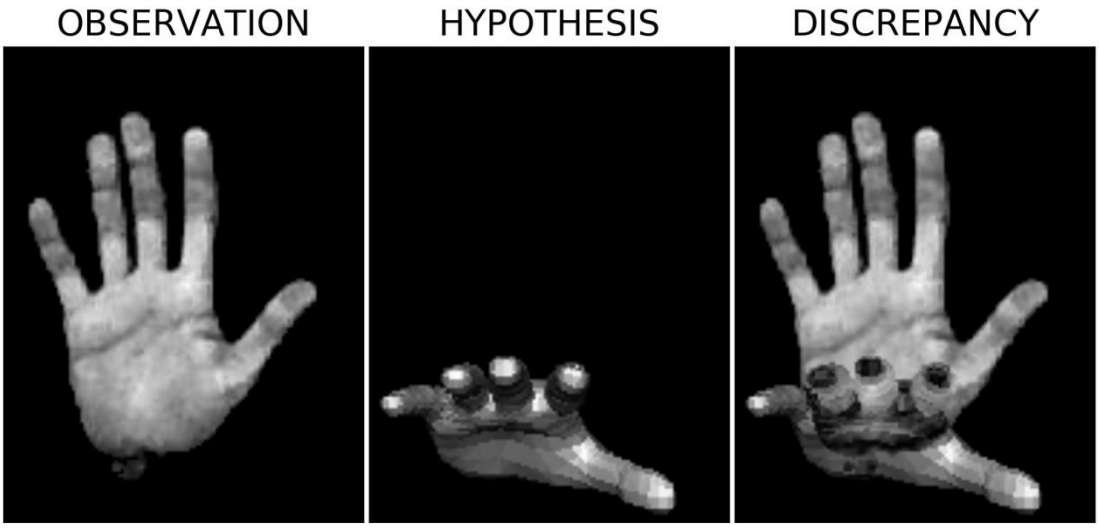

Figure 3.74: Observation (Left), final hypothesis (Middle), and the differences (Right).

The following figures show the cases where a) the lighting is absent, b) the penalty component is used and c) only the intersection pixels are used.

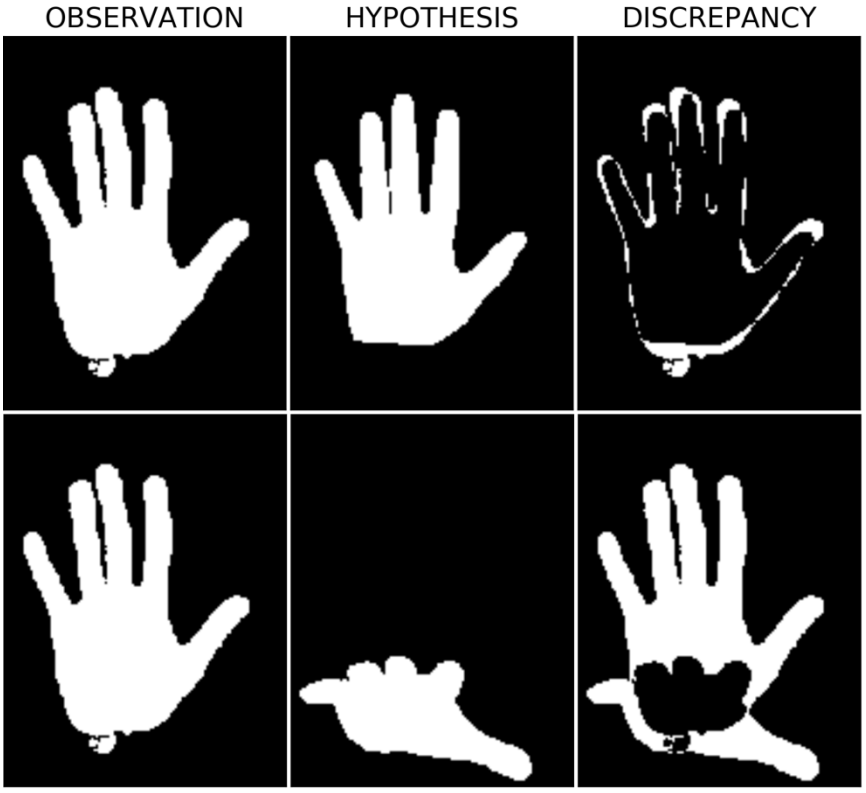

Figure 3.75: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent.

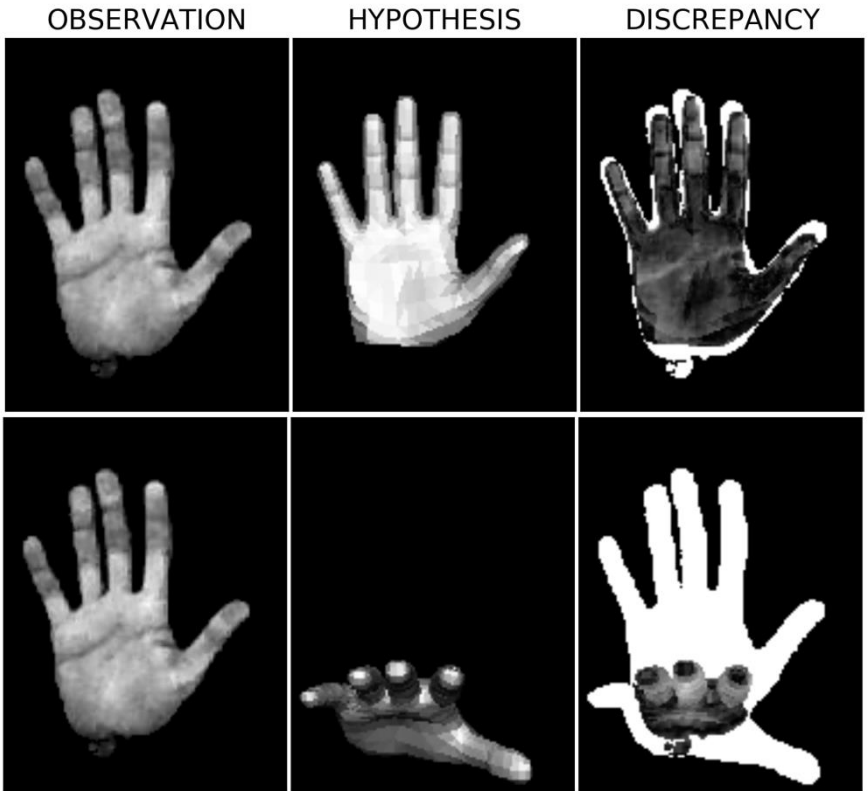

Figure 3.76: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty component is used.

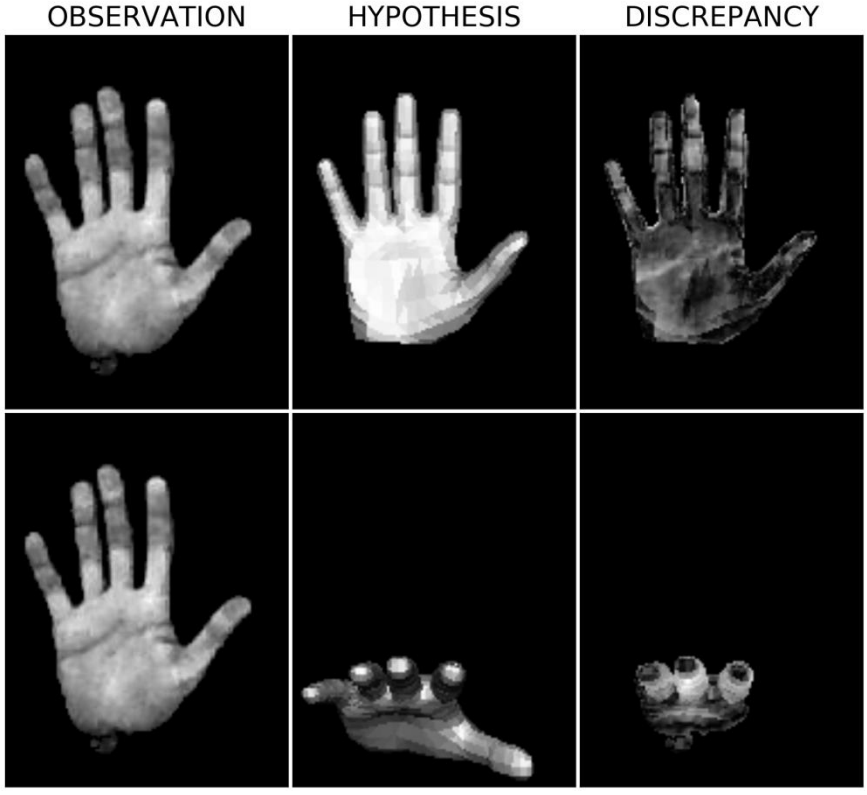

Figure 3.77: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used.

The graph results of the functions can be seen in the following figures.

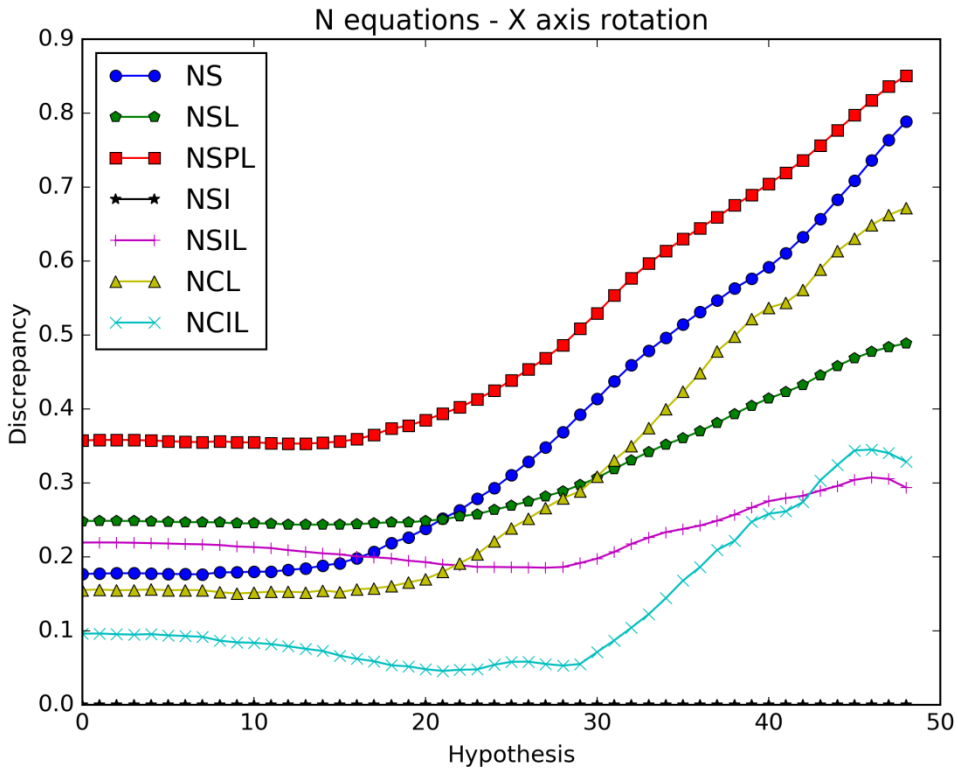

Figure 3.78: N equations.

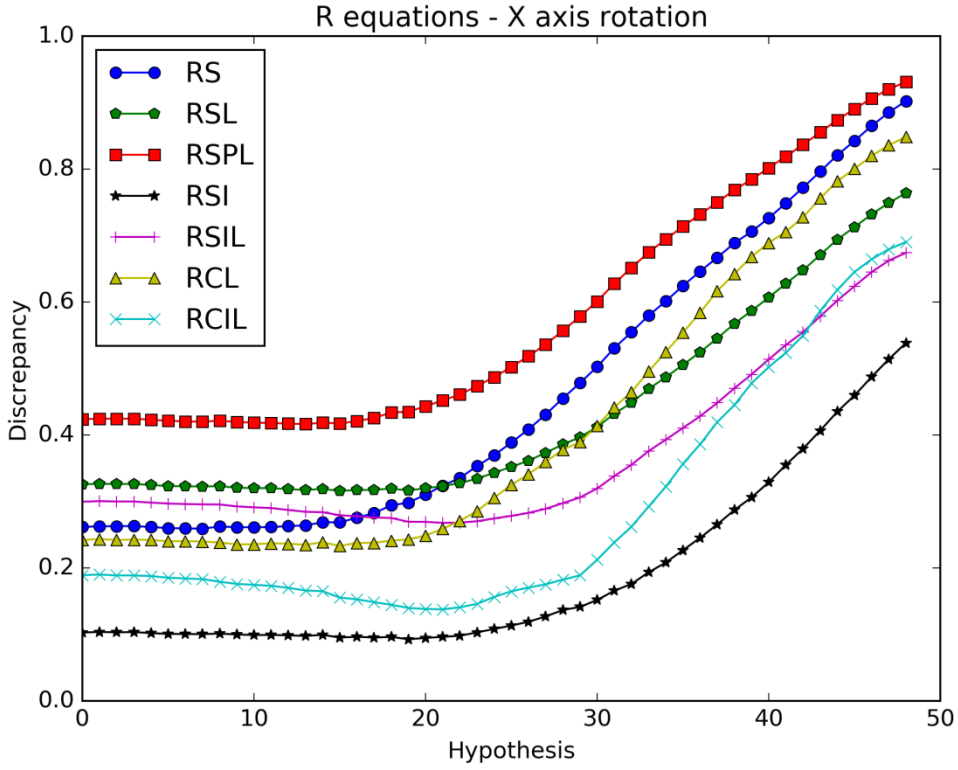

Figure 3.79: R equations.

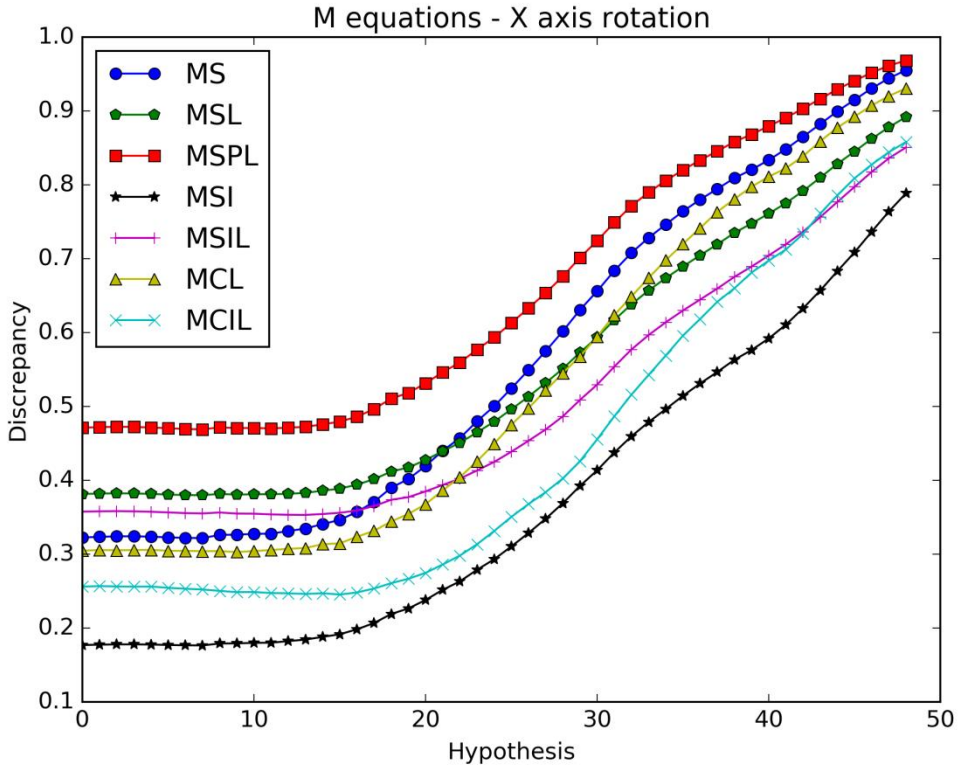

Figure 3.80: M equations.

The responses of the objective functions which use only the intersected pixels are similar or slower than those which use all the pixels to compute the discrepancy value regardless of whether the lighting is present or absent. Thus, we conclude that using only the intersected pixels or the penalty parameter doesn't improve the efficiency of the objective functions. Once more, there is no obvious advantage of an objective function using the lighting information. However, this may change in the experiments where the light conditions are the same for both the observation and the hypotheses.

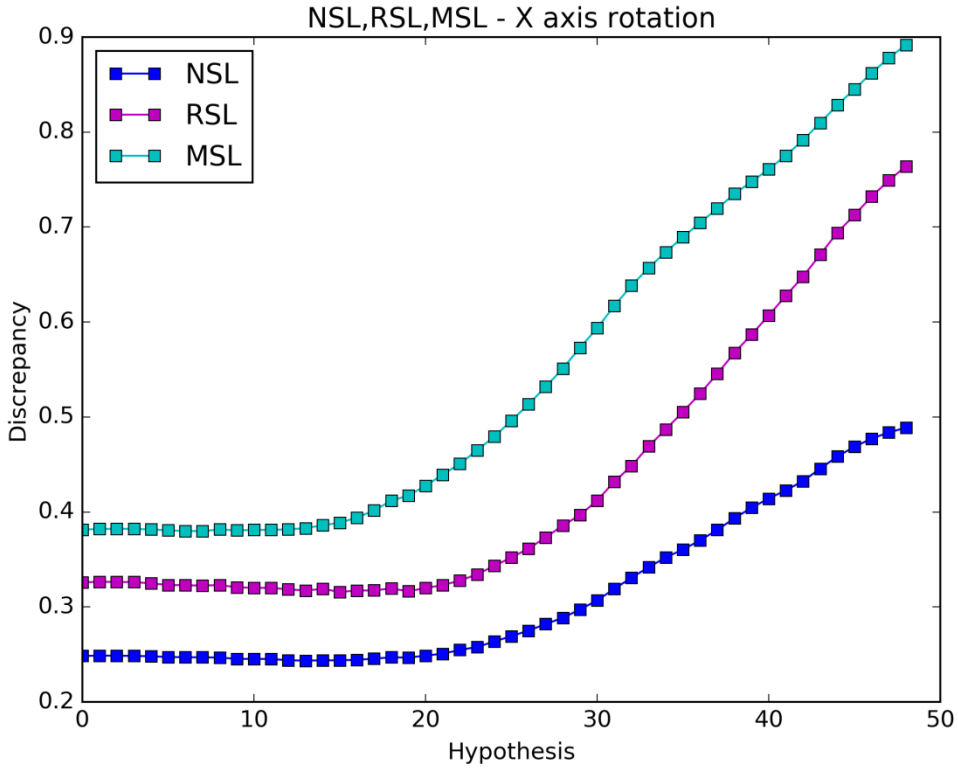

Figure 3.81: NSL, RSL, MSL equations comparison.

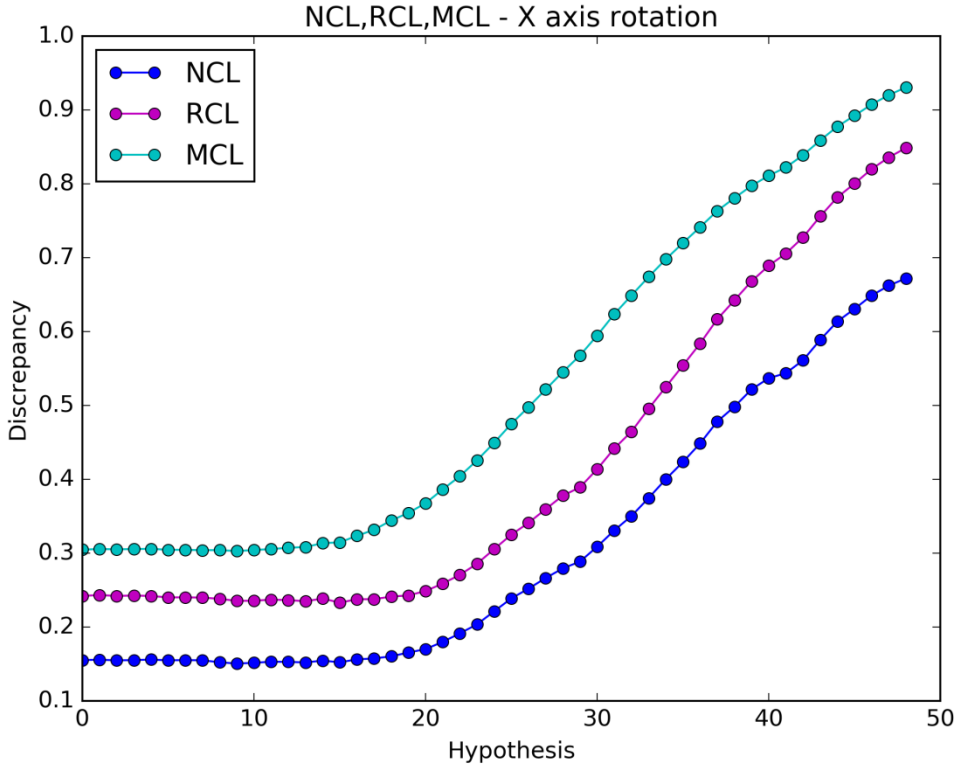

Figure 3.82: NCL, RCL, MCL equations comparison.

The use of the auxiliary parameters doesn't seem to improve the efficiency of the objective functions very much.

3.1.4.2. Simulated – Downward Rotation

The initial rendered hand pose replaces the observation to see how the objective functions operate when the light conditions and the foregrounds are similar for both the observation and the hypothesis. The initial hypothesis is rotated again in 49 steps.

The result after the foreground extraction, the initial hypothesis, the final hypothesis, and their differences are shown in the figures below. Due to the perfect match, the initial discrepancy image is black.

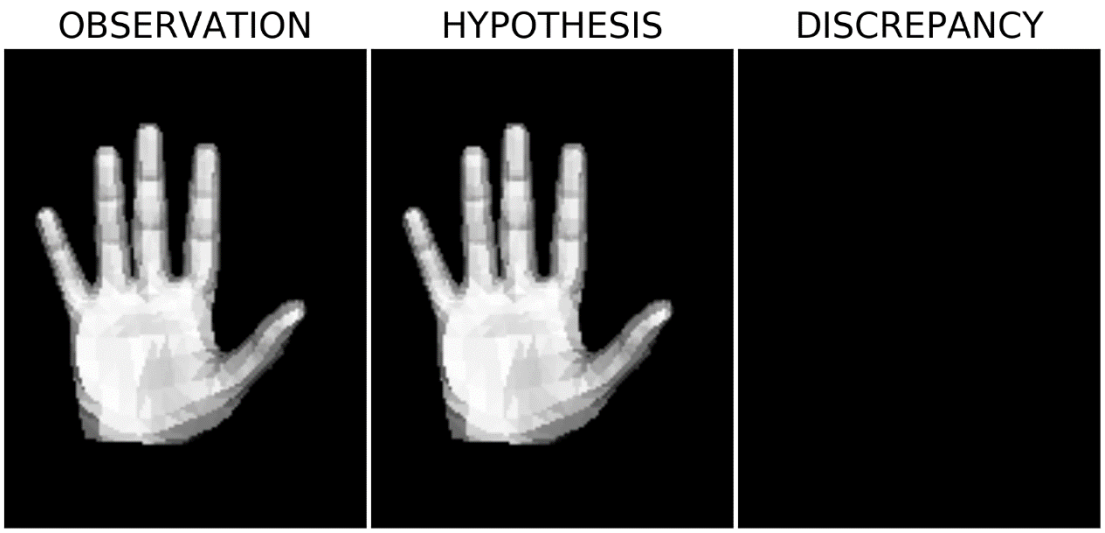

Figure 3.83: Observation (Left), initial hypothesis (Middle), and the differences (Right).

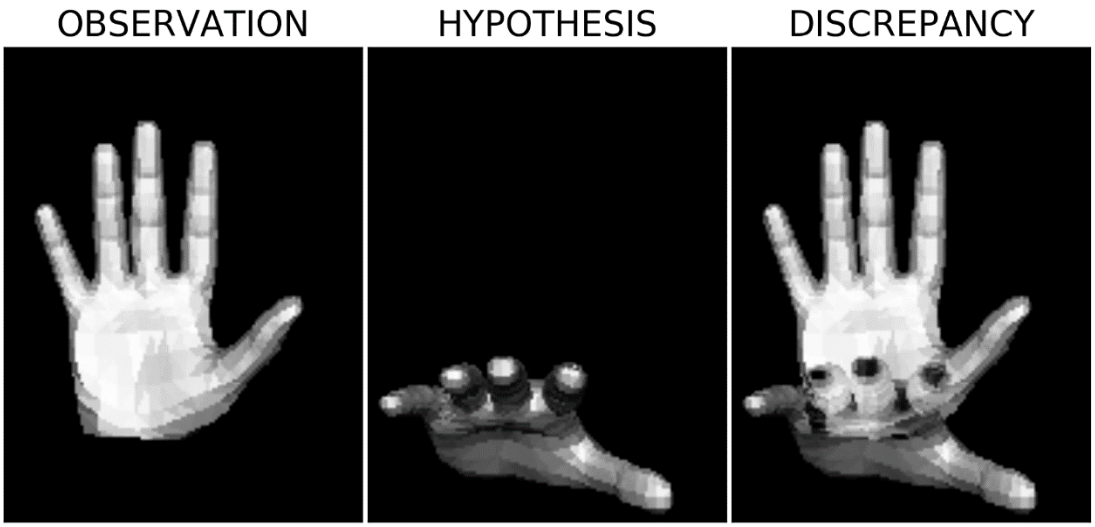

Figure 3.84: Observation (Left), final hypothesis (Middle), and the differences (Right).

The graph results of the functions can be seen in the following figures.

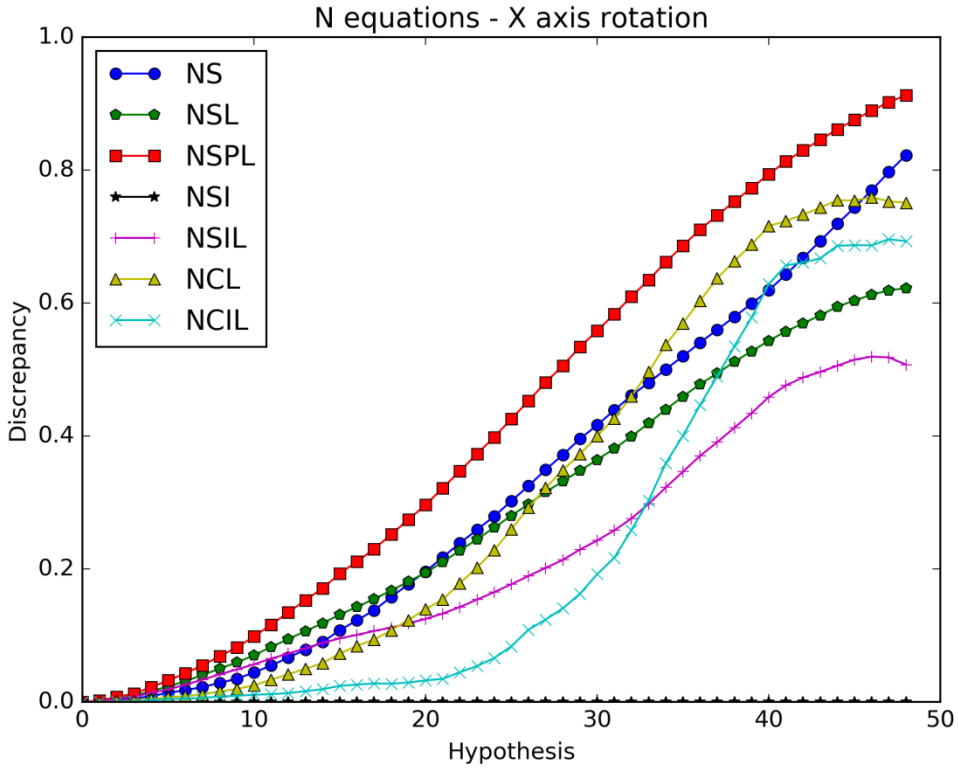

Figure 3.85: N equations.

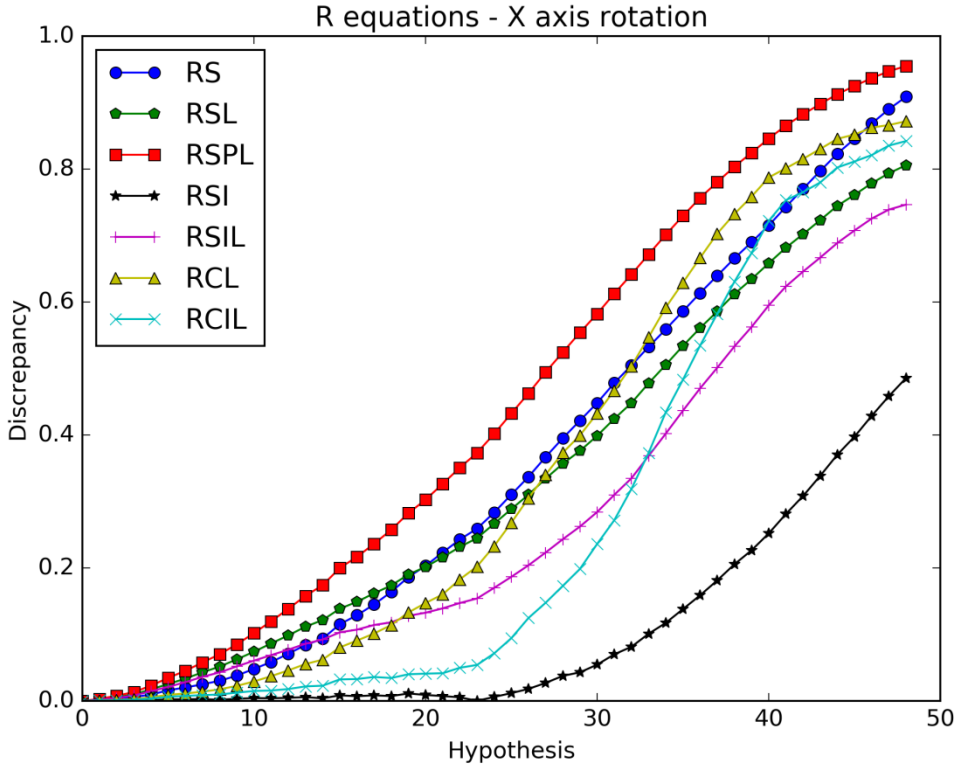

Figure 3.86: R equations.

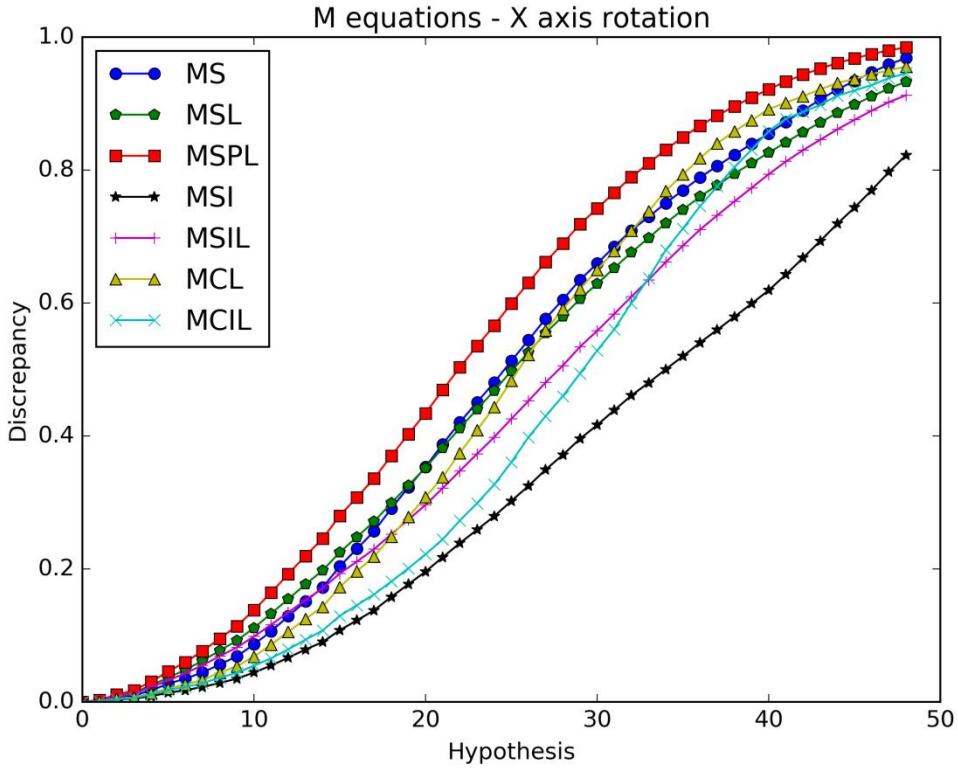

Figure 3.87: M equations.

Using only the intersected pixels doesn't improve the efficiency of the objective functions either the lighting is present or absent. The graphs that stand out are those where the penalty parameter is used. That suggests that if all the conditions are similar for both the observation and the hypothesis, the lighting presence with the penalty parameter gives better results than when the lighting is absent.

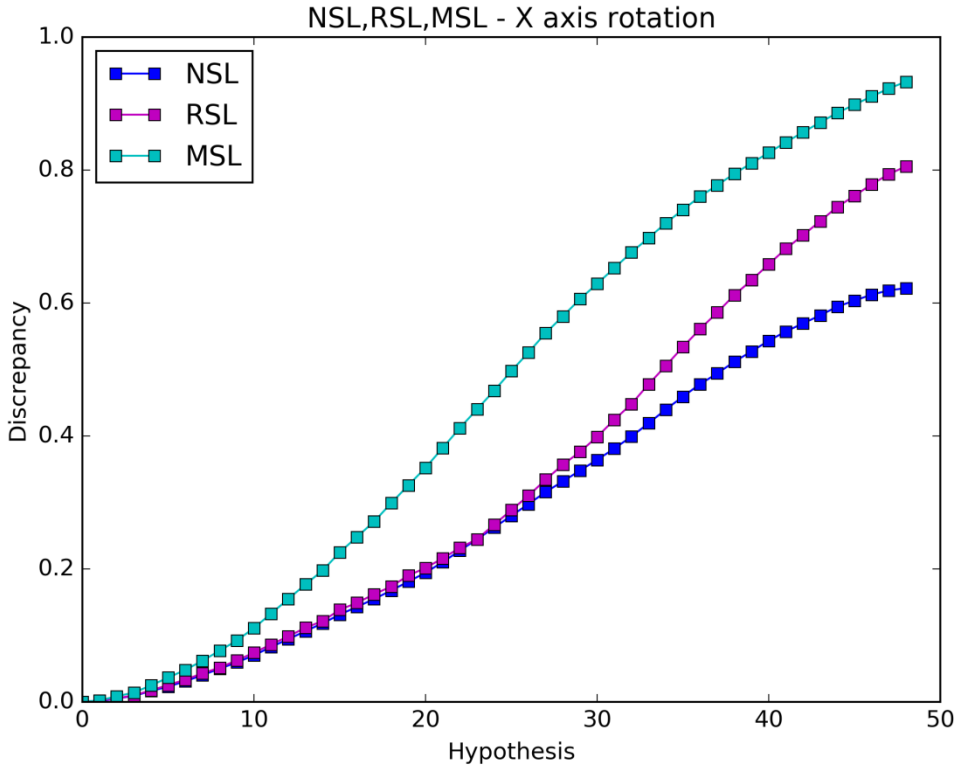

Figure 3.88: NCL, RCL, MCL equations comparison.

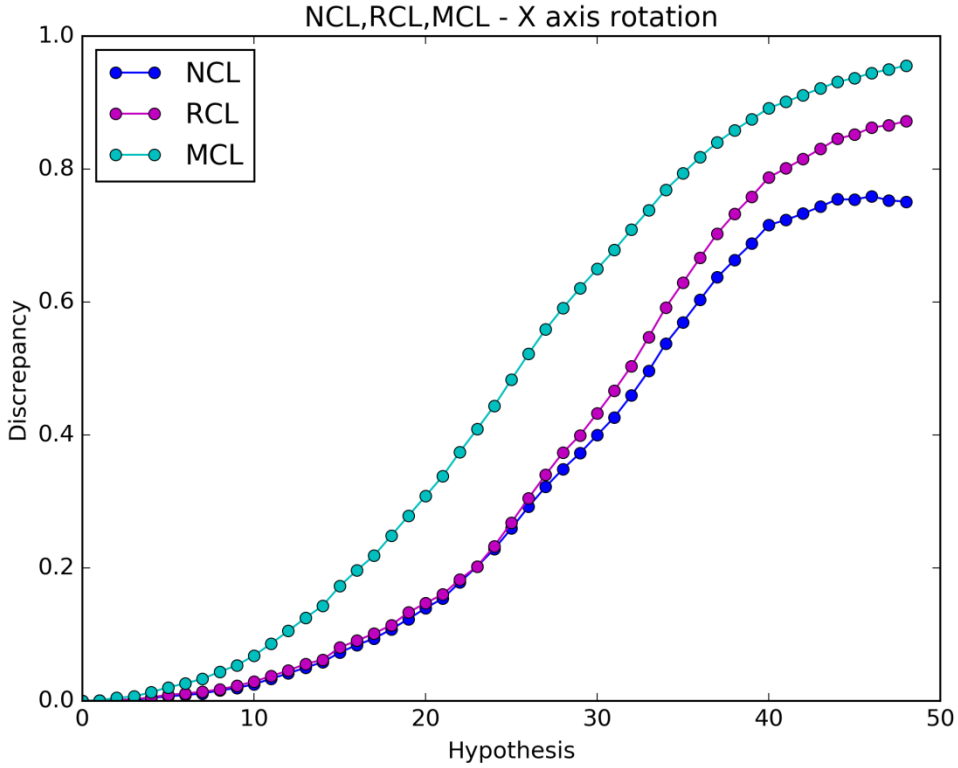

Figure 3.89: NSL, RSL, MSL equations comparison.

The mask parameter is obviously the best auxiliary parameter.

3.1.5. Case 5: Bending

The RGB image used in this case can be seen in the image below.

Figure 3.90: RGB image of Case 5.

3.1.5.1. Bending

The result after the foreground extraction, the initial hypothesis, the final hypothesis, and their differences are shown in the figures below.

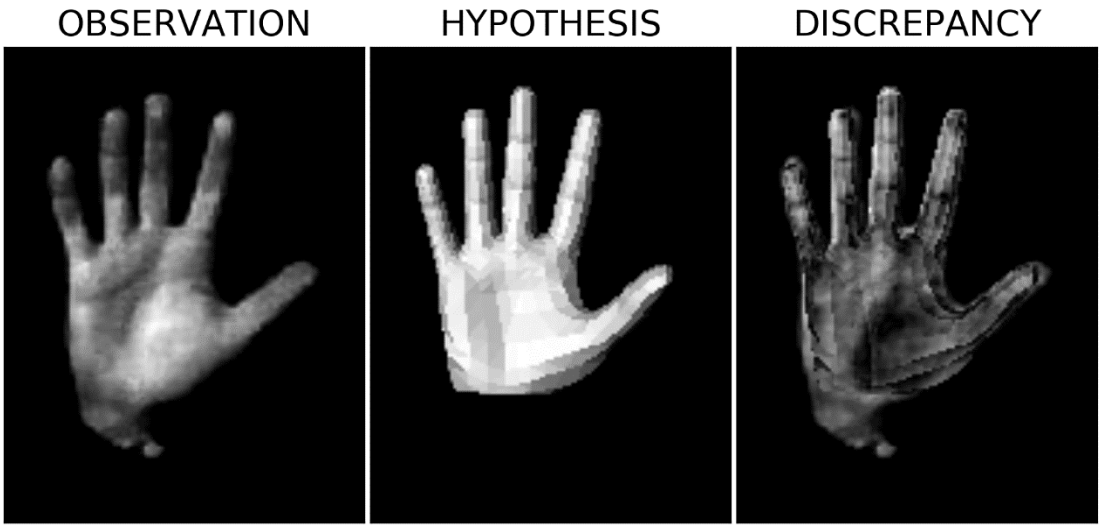

Figure 3.91: Observation (Left), initial hypothesis (Middle), and the differences (Right).

Then, the fingers of the initial hypothesis bend in 49 steps.

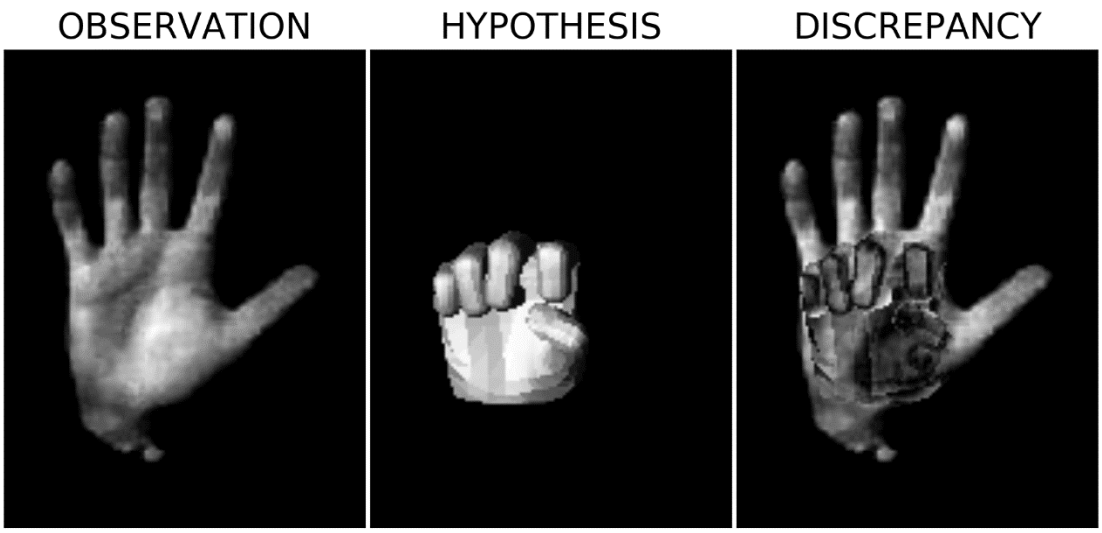

Figure 3.92: Observation (Left), final hypothesis (Middle), and the differences (Right).

The following figures show the cases where a) the lighting is absent, b) the penalty component is used and c) only the intersection pixels are used.

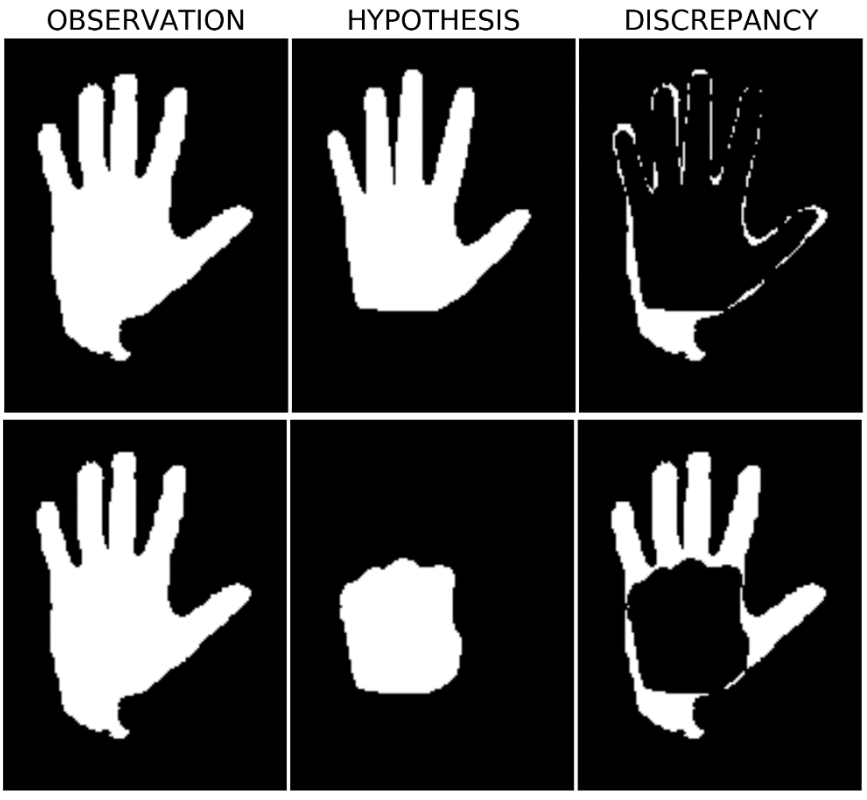

Figure 3.93: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the lighting is absent.

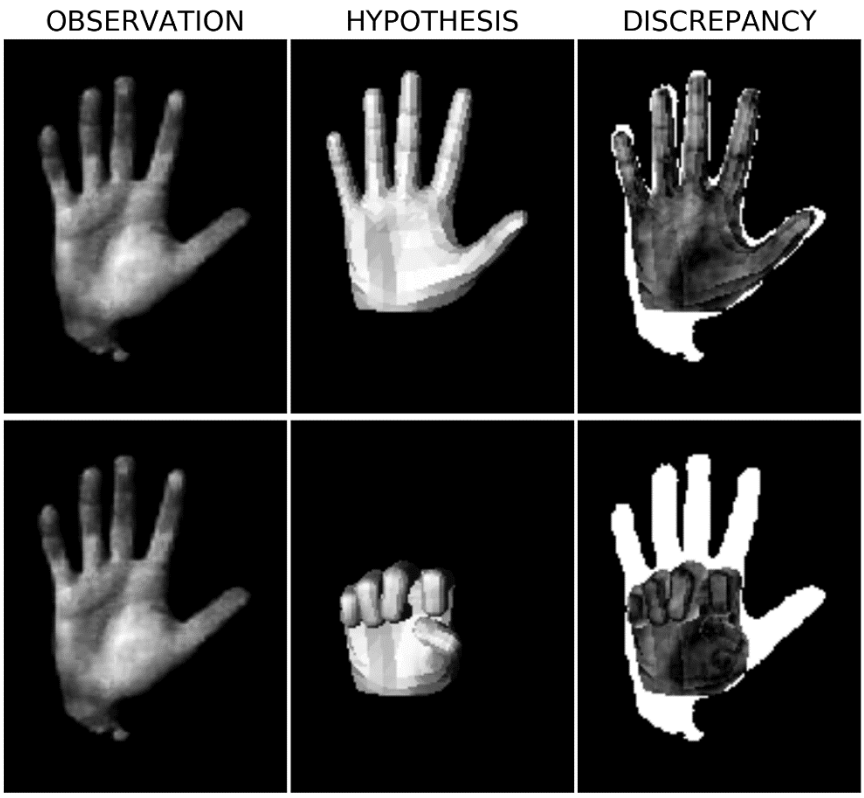

Figure 3.94: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when the penalty component is used.

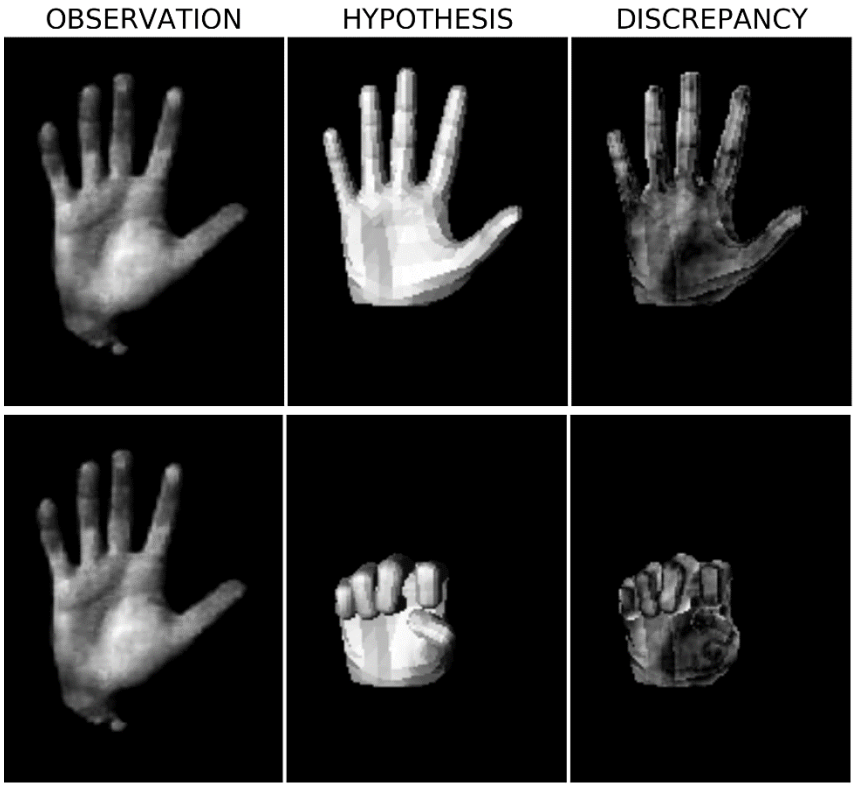

Figure 3.95: Observation (Left), initial hypothesis (Middle-Up), final hypothesis (Middle-Down), and the differences (Right) when only the intersection pixels are used.

The graph results of the objective functions can be seen in the following figures.

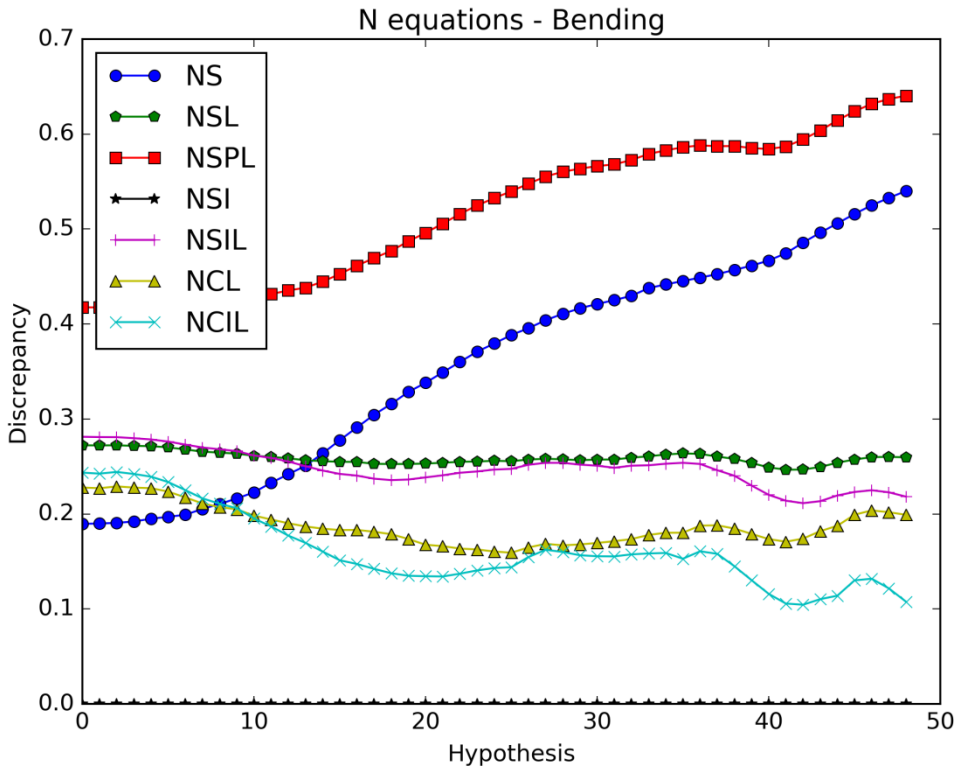

Figure 3.96: N equations.

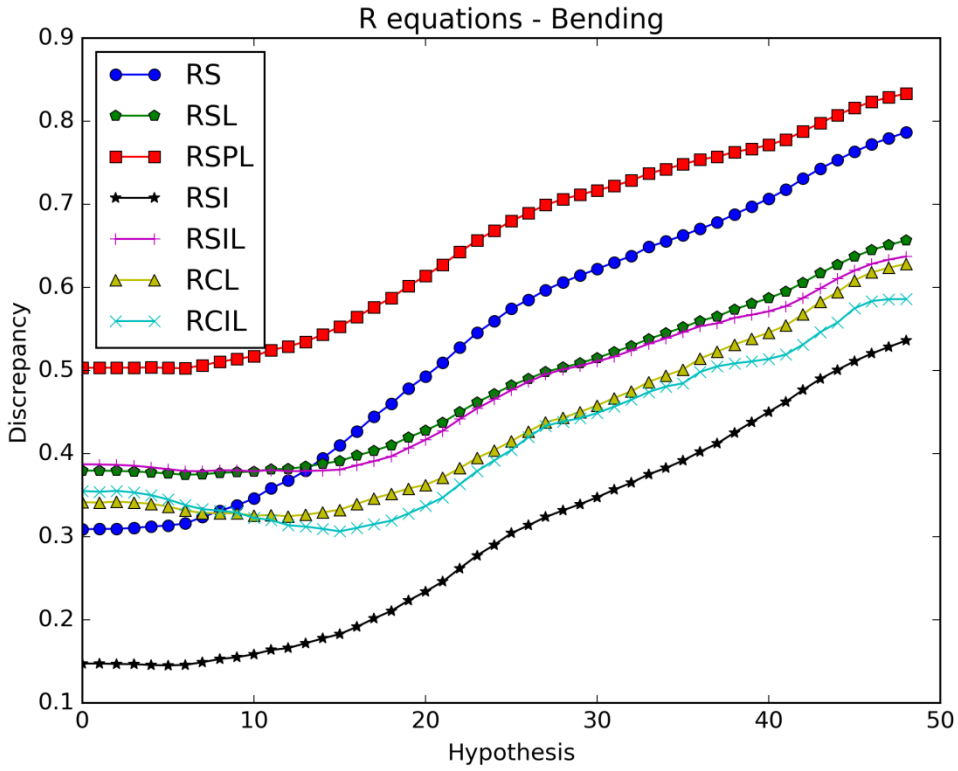

Figure 3.97: R equations.

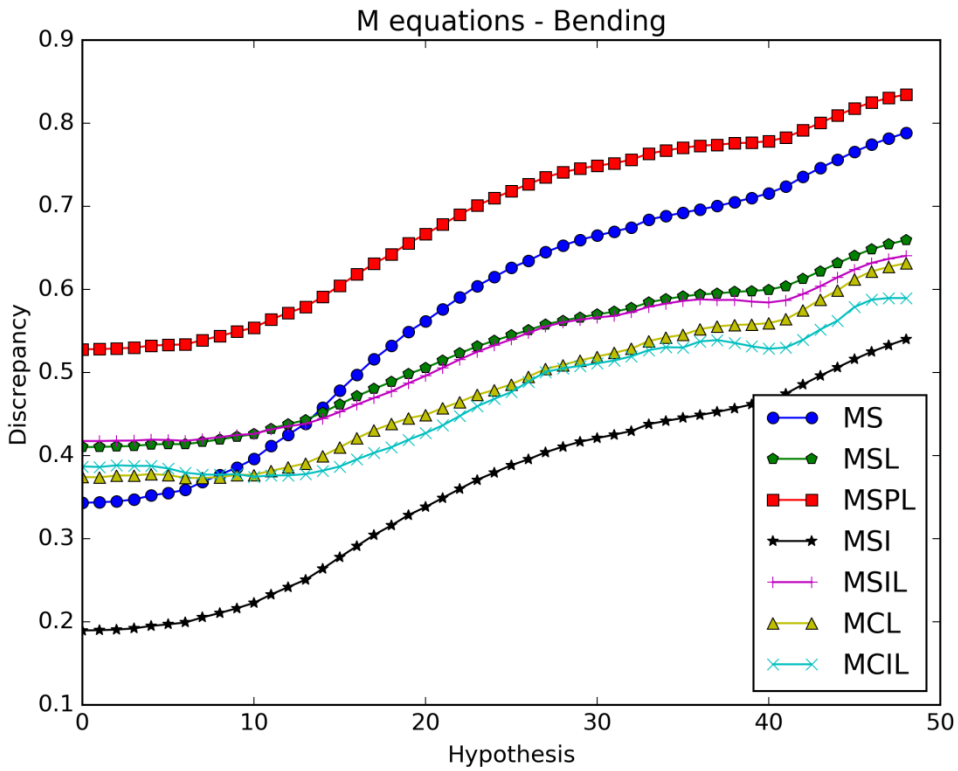

Figure 3.98: M equations.

The objective functions where the lighting is absent seem to have the best responses but there is too much noise in this case and that makes the results untrustworthy. The experiments where the light conditions are similar for both the observation and the hypotheses are more reliable.

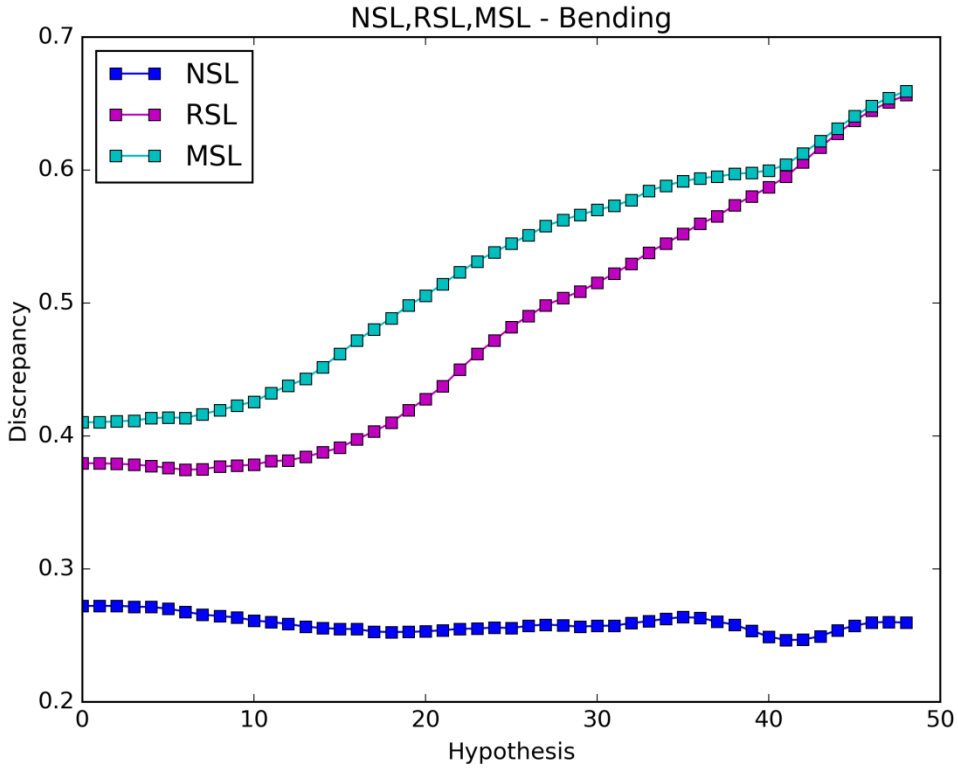

Figure 3.99: NSL, RSL, MSL equations comparison.

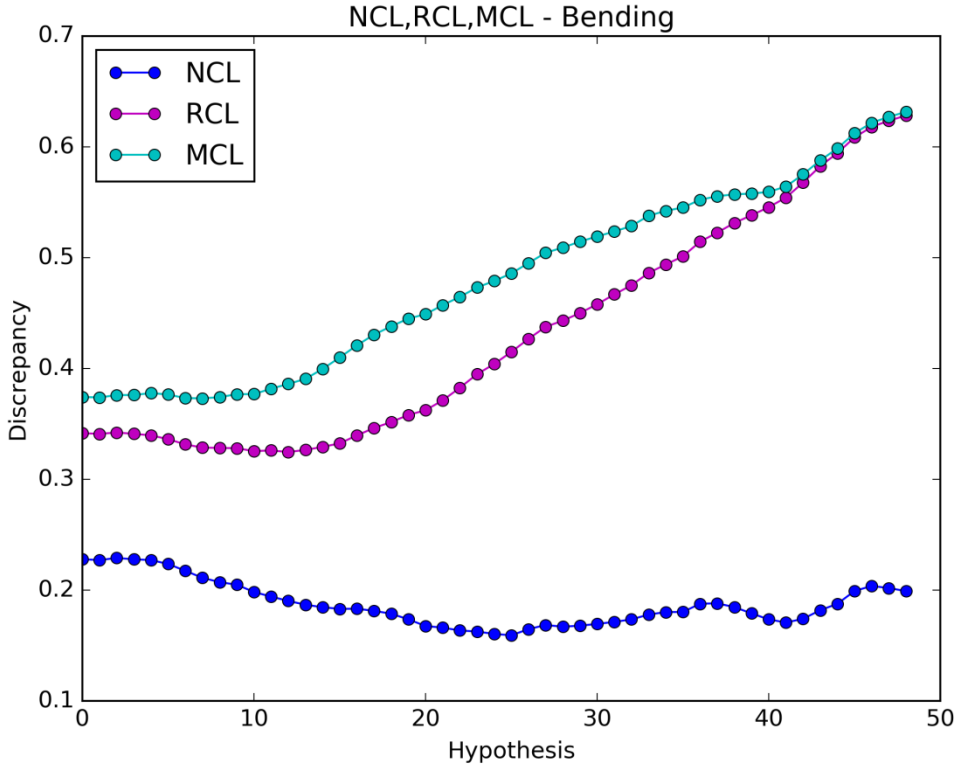

Figure 3.100: NCL, RCL, MCL equations comparison.

Both the mask and the percent parameter improve the efficiency of the objective functions.

3.1.5.2. Simulated – Bending

The initial rendered hand pose replaces the observation to see how the objective functions operate when the light conditions and the foregrounds are similar for both the observation and the hypothesis. The fingers of the initial hypothesis bend again in 49 steps.

The result after the foreground extraction, the initial hypothesis, the final hypothesis, and their differences are shown in the figures below. Due to the perfect match, the initial discrepancy image is black.

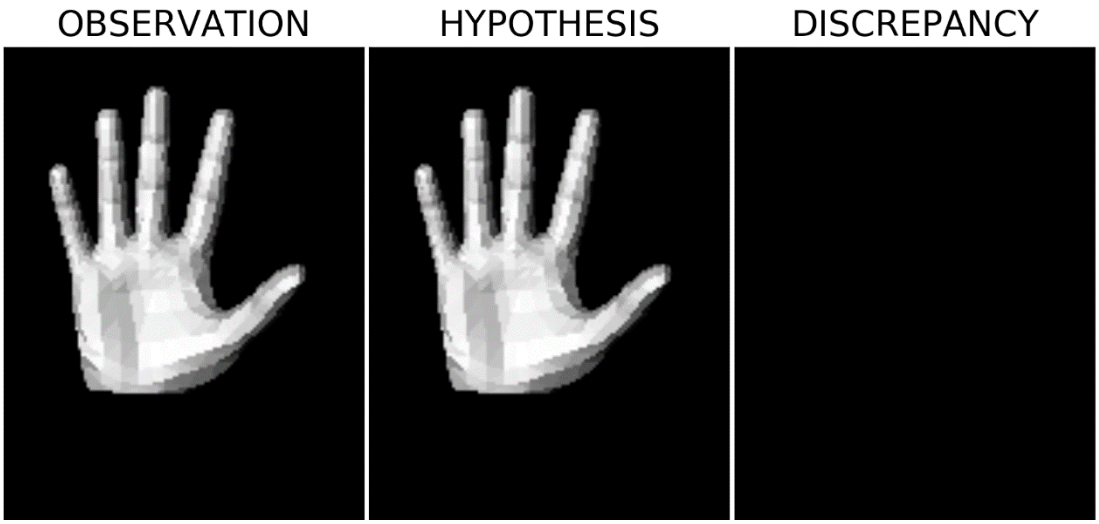

Figure 3.101: Observation (Left), initial hypothesis (Middle), and the differences (Right).

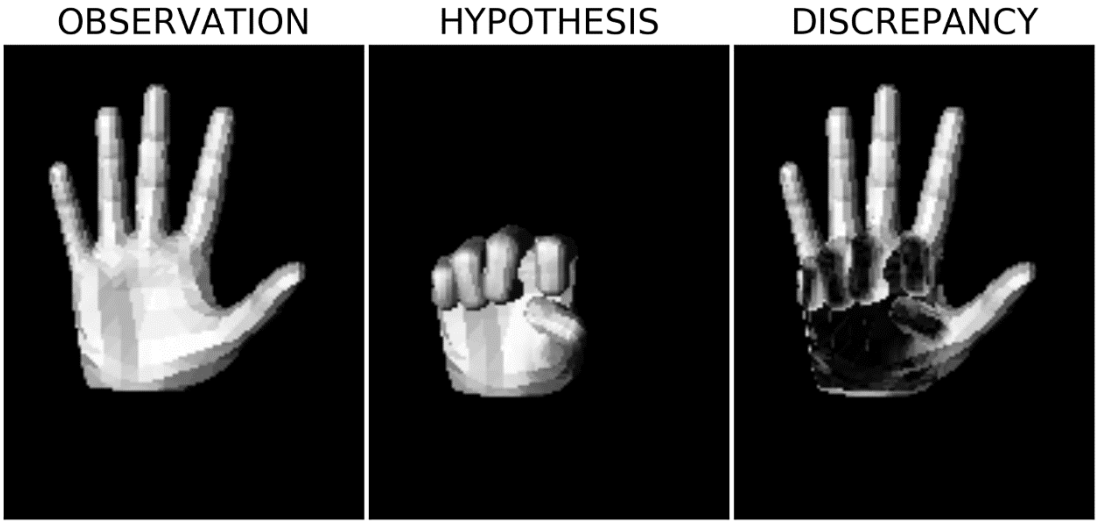

Figure 3.102: Observation (Left), final hypothesis (Middle), and the differences (Right).

The graph results of the functions can be seen in the following figures.

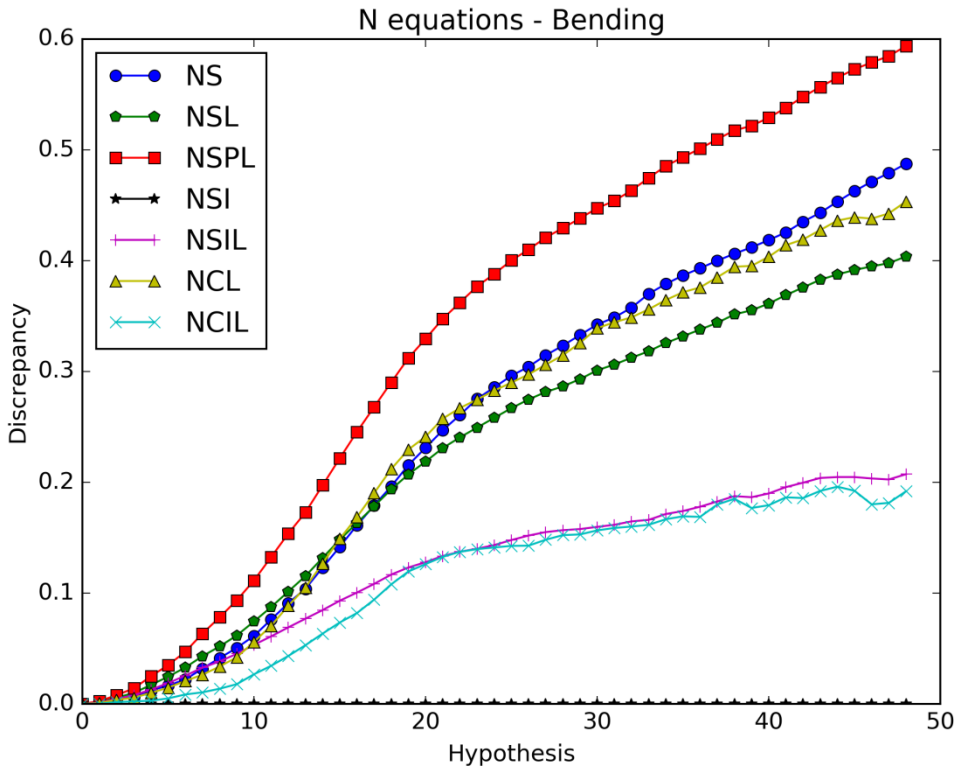

Figure 3.103: N equations.

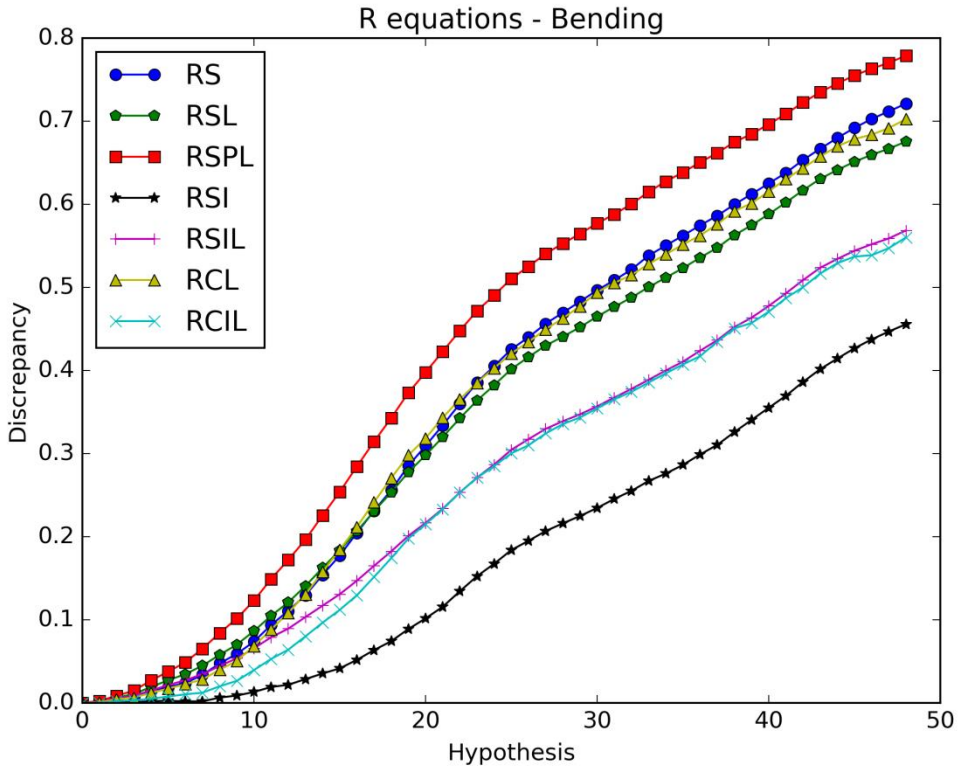

Figure 3.104: R equations.

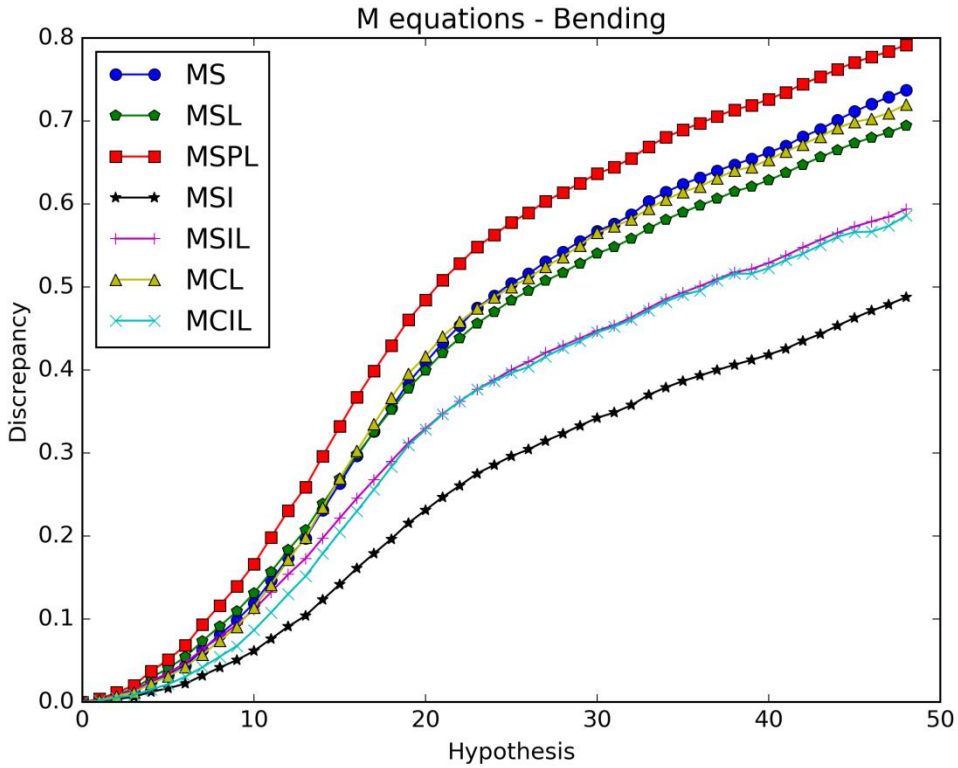

Figure 3.105: M equations.

The results are much better than before. Using only the intersection pixels doesn't improve the efficiency of the objective functions. The graphs that stand out are those where the penalty parameter is used. That suggests that if all the conditions are similar for both the observation and the hypothesis, the lighting presence with the penalty parameter gives better results than when the lighting is absent.

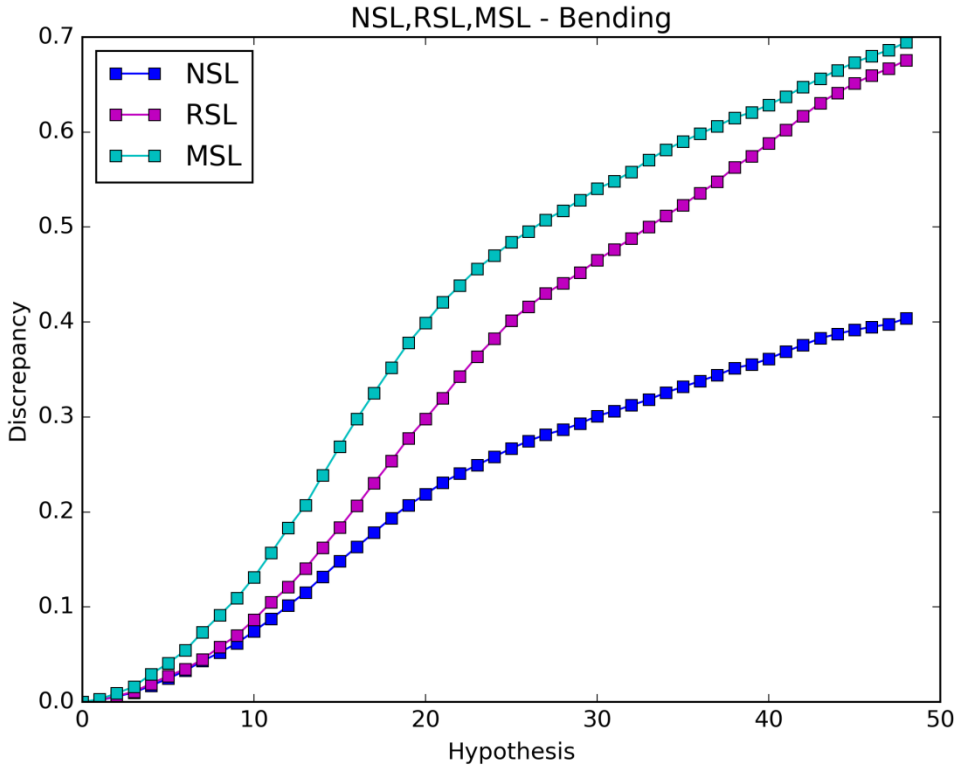

Figure 3.106: NCL, RCL, MCL equations comparison.

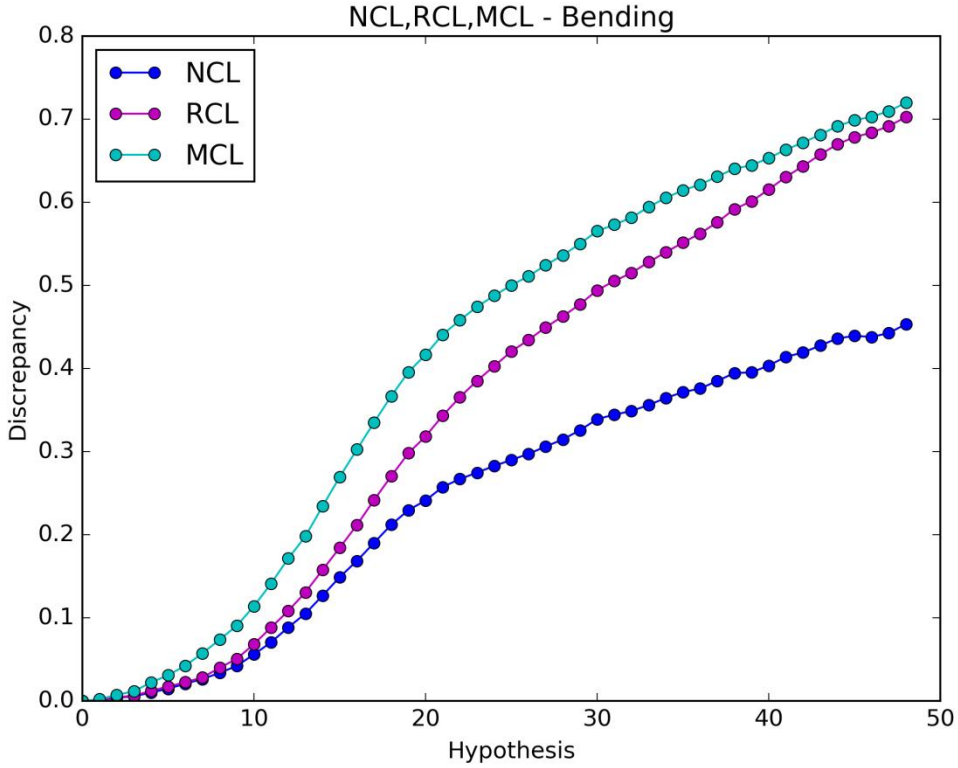

Figure 3.107: NSL, RSL, MSL equations comparison.

The mask parameter is the auxiliary parameter that improves the efficiency of the objective functions the most.

3.1.6. Conclusions of the offline experiments

The offline experiments helped us to form a general view about the efficiency of the objective functions:

- When the light conditions and the foreground between the observation and a hypothesis are not similar enough the results may be a lot different from the results where these two parameters are similar. However, this is an external factor which we cannot avoid, but we needed to test the efficiency of the objective functions when the conditions are similar for both the observation and the hypotheses to know that the lighting information is a useful element to include in our objective functions and that it could improve their efficiency under certain circumstances.
- The mask parameter “m” is an auxiliary parameter that was found to improve the efficiency of the objective functions without an auxiliary parameter (N equations). Also, the experiments showed that it is better than the percent auxiliary parameter “r” which improves the efficiency of the objective functions too.
- Using only the information of the intersected pixels isn’t helpful either the lighting information is present or absent.
- The best objective function using the **sum of absolute differences** was found to be the **MSPL** objective function with the penalty parameter. The best objective function using the **number of big absolute differences** was found to be the **MCL** objective function.
- The Figure below shows that the MSPL objective function (black) is the best objective function since it increases faster than the MCL objective function (blue) in every graph.
- The simulated offline experiments showed that the lighting information may be a useful element and that there is a lot of room for improvement. However, it is a requirement the lighting conditions of the hypothesis to be simulated very well so that to be similar with the lighting conditions of the observation. That means a more complex lighting model is needed, enriched with much more information and characteristics of the scene. Another requirement is the foreground information of the observation and the hypothesis to marginally match as much as possible. A more improved foreground extraction method could do this job.

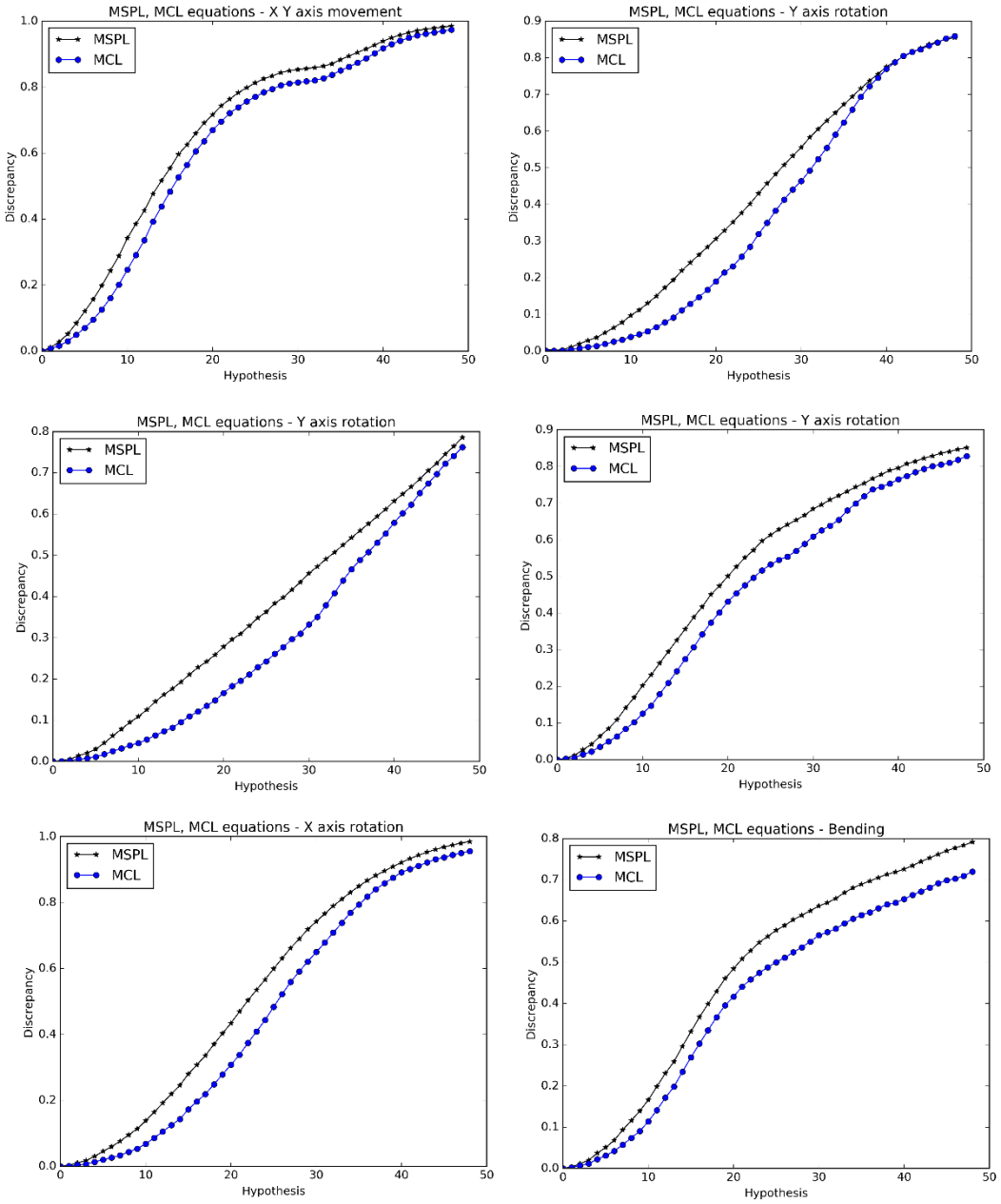

Figure 3.108: MSPL and MCL objective functions comparison.

In the next section, the tracking experiments and their results are going to be presented.

3.2. Tracking experiments

In this section, the efficiency of the best objective functions is tested in real tracking experiments. When the light conditions and the foreground are similar for both the observation and the hypotheses, the offline experiments showed that the best objective functions are: MSPL (when the **sum of absolute differences** is used) and MCL (when the **number of big absolute differences** is used). And when the lighting is absent, the offline experiments showed that the best objective function is: MS (when the **sum of differences** is used). All the objective functions include the **mask parameter** which was found that improves the efficiency the most.

The three objective functions mentioned in the previous paragraph, are tested in **simulated sequence tracking** where the light conditions and the foreground are similar for both the observation and the hypotheses and in **real video tracking**. Their efficiency is also compared with the efficiency of the default objective function which the tracker uses based on the depth information.

3.2.1. Datasets

The table below shows the characteristics of the simulated sequences that were used to test the efficiency of the objective functions. Specifically, for each sequence the table shows the position of the light source in the 3D space, the number of frames, the intensity threshold used in the foreground extraction below which every pixel's intensity is changed to zero (2.3.2.2. Foreground extraction (or background subtraction) and some elements of the move which the hand does in the sequence. The objective functions are tested in six different sequences using 128 particles and they are tested again in the last four sequences using 256 particles. The chance a hypothesis has to match the observation gets higher as the number of the particles (hypotheses) for each frame is increased.

Sequence #	Light Pos. (x, y, z)	Frames	Threshold	Movement
1	(+200, +170, 0)	99	0	Translation
2	(+200, +170, 0)	99	0	Translation, Bending
3	(+600, 0, 0)	99	0	X, Y axis Rotation, Bending
4	(+600, 0, 0)	99	0	Z axis Rotation, Bending
5	(+600, 0, 0)	99	0	X axis Rotation, Bending
6	(0, 0, -100)	99	0	X, Y, Z axis Rotation, Bending

Table 5: Simulated tracking sequences.

In a similar way, the table below shows the characteristics of the real videos that were used to test the efficiency of the objective functions.

Video #	Light Pos. (x, y, z)	Frames	Threshold	Movement
1	(+200, +170, 0)	126	70	Translation, Bending
2	(+200, +170, 0)	214	60	X, Z axis Rotation, Bending
3	(+200, +170, 0)	180	60	Bending
4	(+200, +170, 0)	149	60	Y axis Rotation

Table 6: Real tracking videos.

3.2.2. Simulated sequence tracking

In this section simulated sequences are used to test the efficiency of the objective functions. The light conditions of the scene and the foreground are similar for both the observation and the hypothesis. For every sequence, we display some snapshots (frames) indicatively and two tables in the end show the average error (mm) of each objective function.

3.2.2.1. Simulate sequence tracking – 128 particles

The following tracking experiments were done using 128 hypotheses (particles).

3.2.2.1.1. Sequence 1

The observation is moving as the red lines show in the Figure below. The orientation doesn't change during this movement.

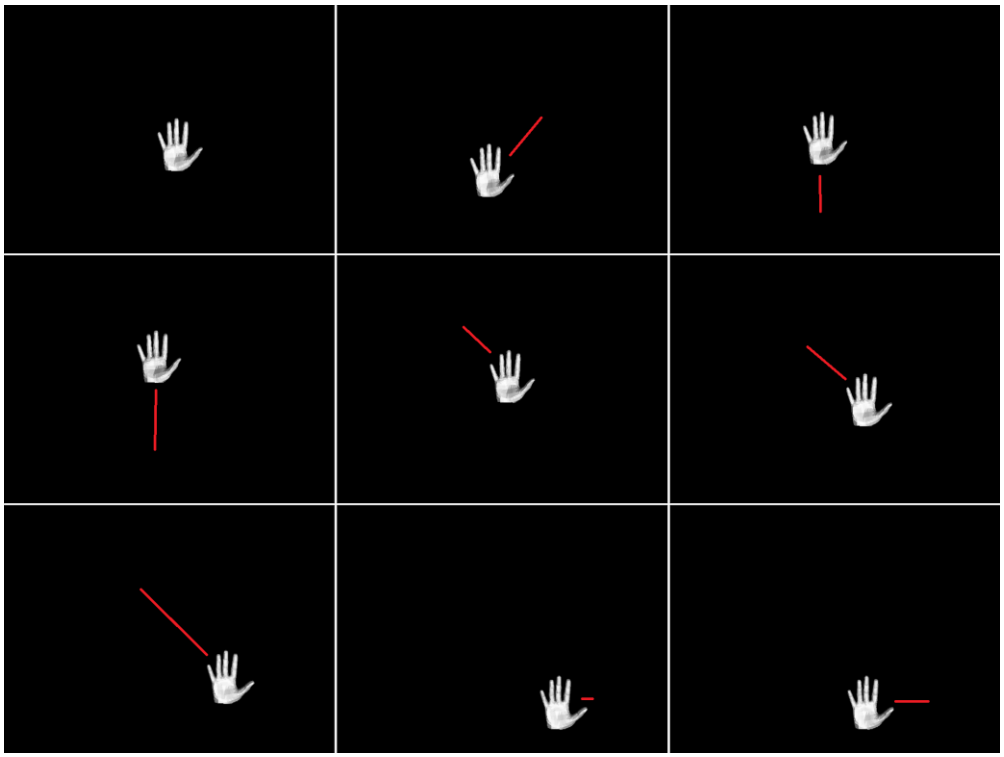

Figure 3.109: Sequence 1 observation images.

3.2.2.1.1.1. Default objective function

The default objective function is used.

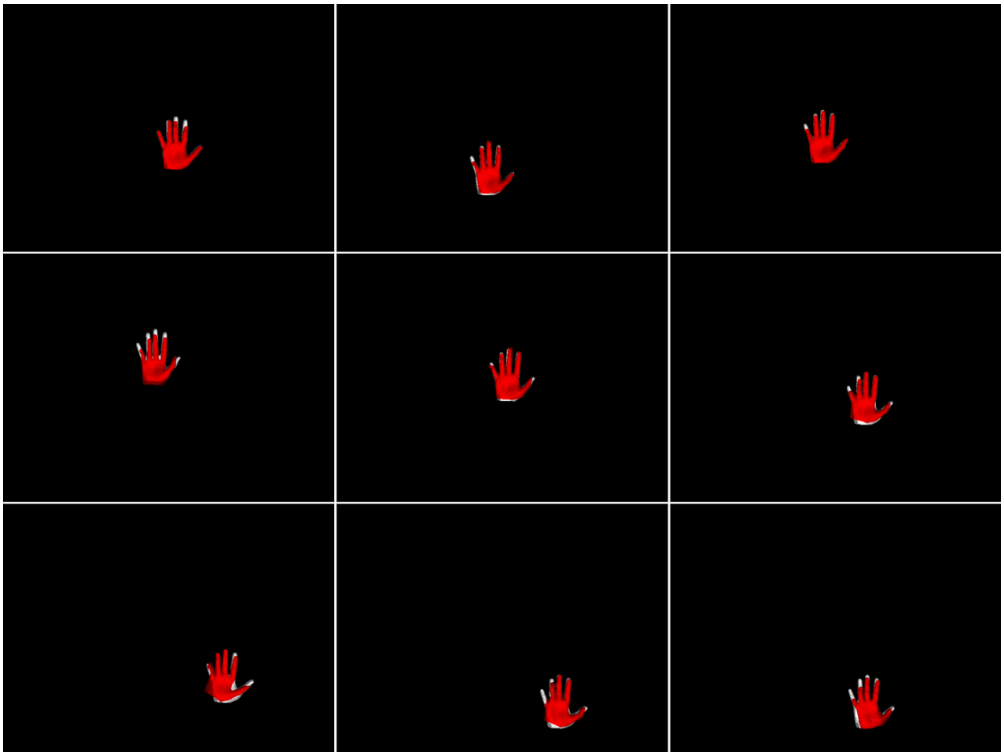

Figure 3.110: Default objective function images.

3.2.2.1.1.2. MS objective function

The MS objective function is used.

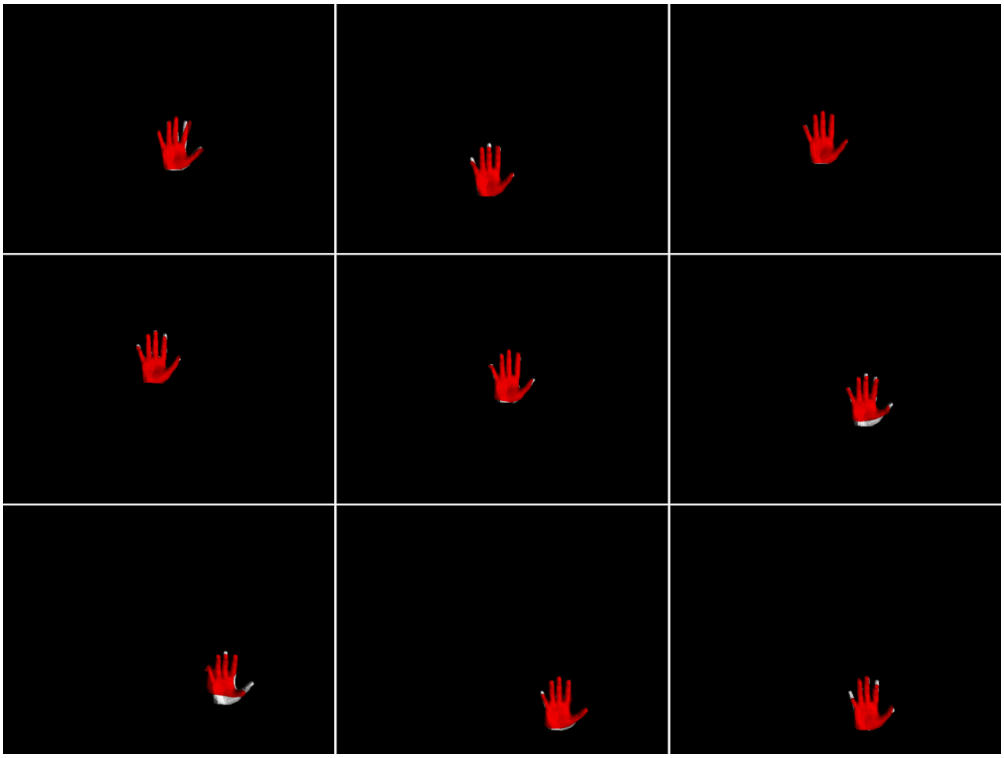

Figure 3.111: MS objective function images.

3.2.2.1.1.3. MSPL objective function

The MSPL objective function is used.

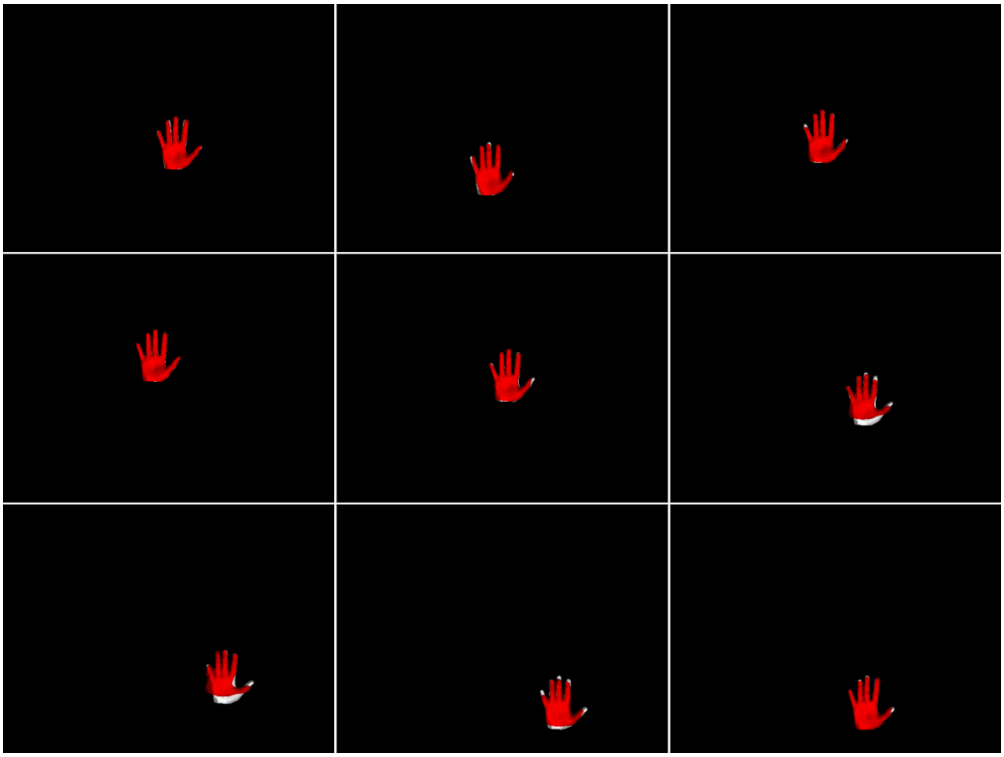

Figure 3.112: MSPL objective function images.

3.2.2.1.1.4. MCL objective function

The MCL objective function is used.

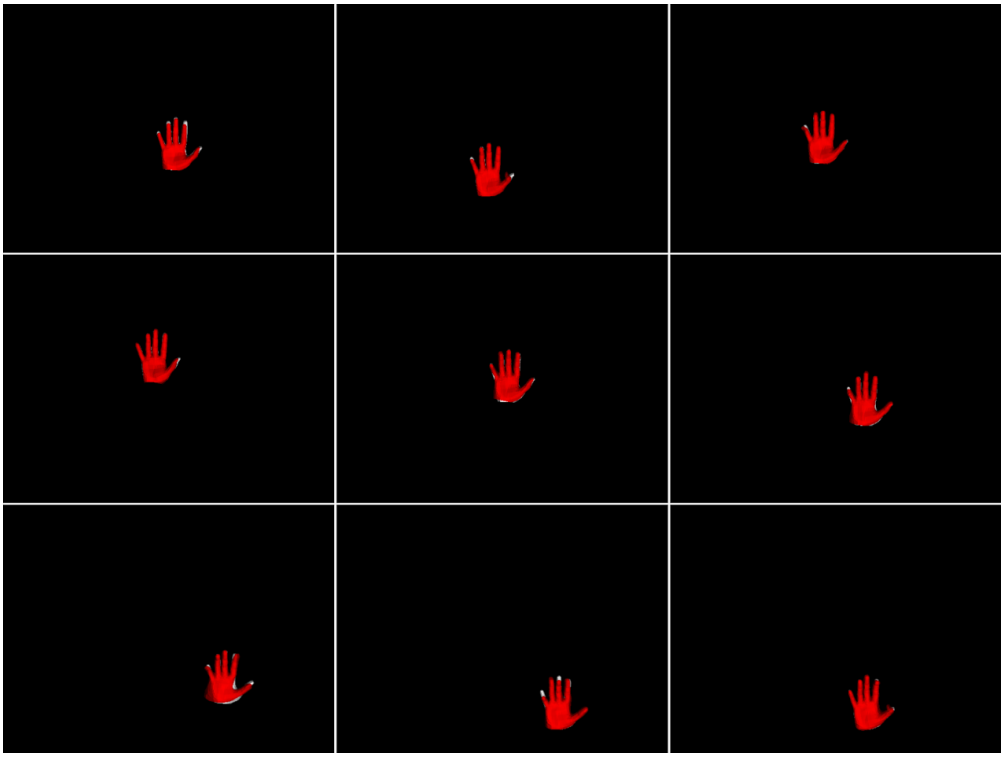

Figure 3.113: MCL objective function images.

3.2.2.1.2. Sequence 2

The fingers of the observation are bending and then they are straightening while the observation is not moving. Then it starts moving to the left and at the same time the thumb is bending.

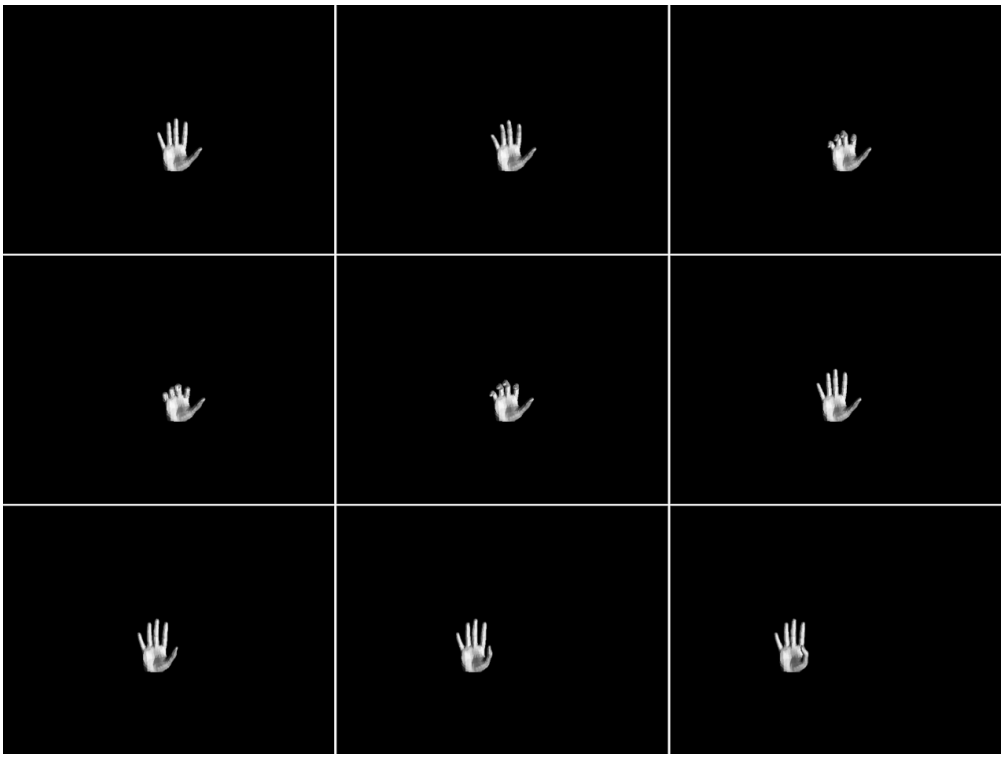

Figure 3.114: Sequence 2 observation images.

3.2.2.1.2.1. Default objective function

The default objective function is used.

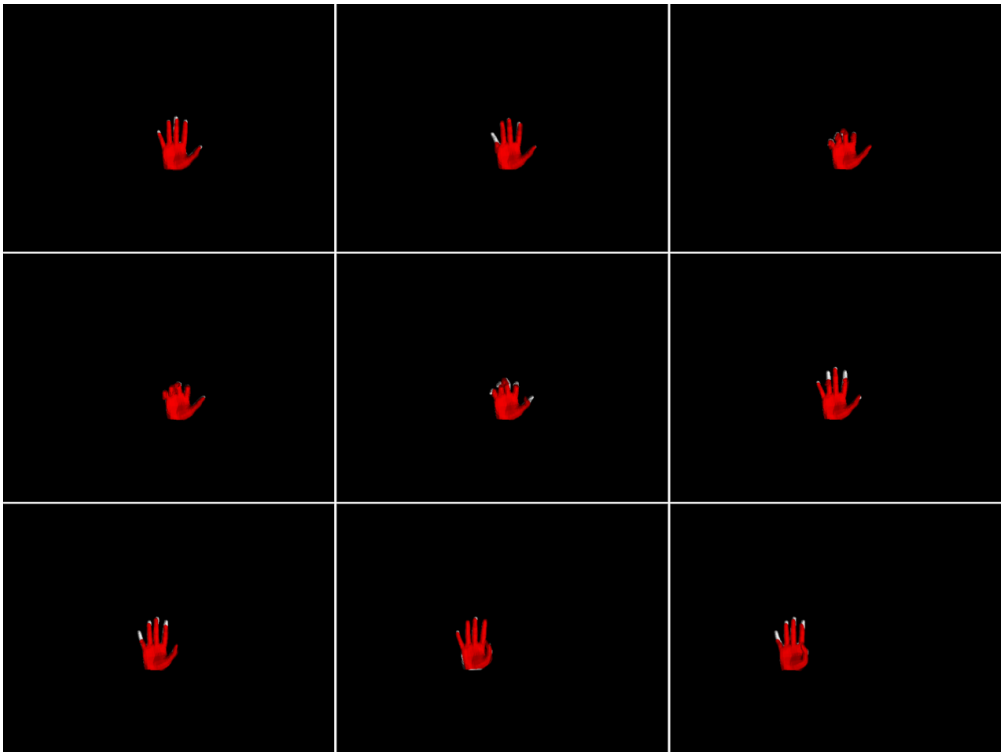

Figure 3.115: Default objective function images.

3.2.2.1.2.2. MS objective function

The MS objective function is used.

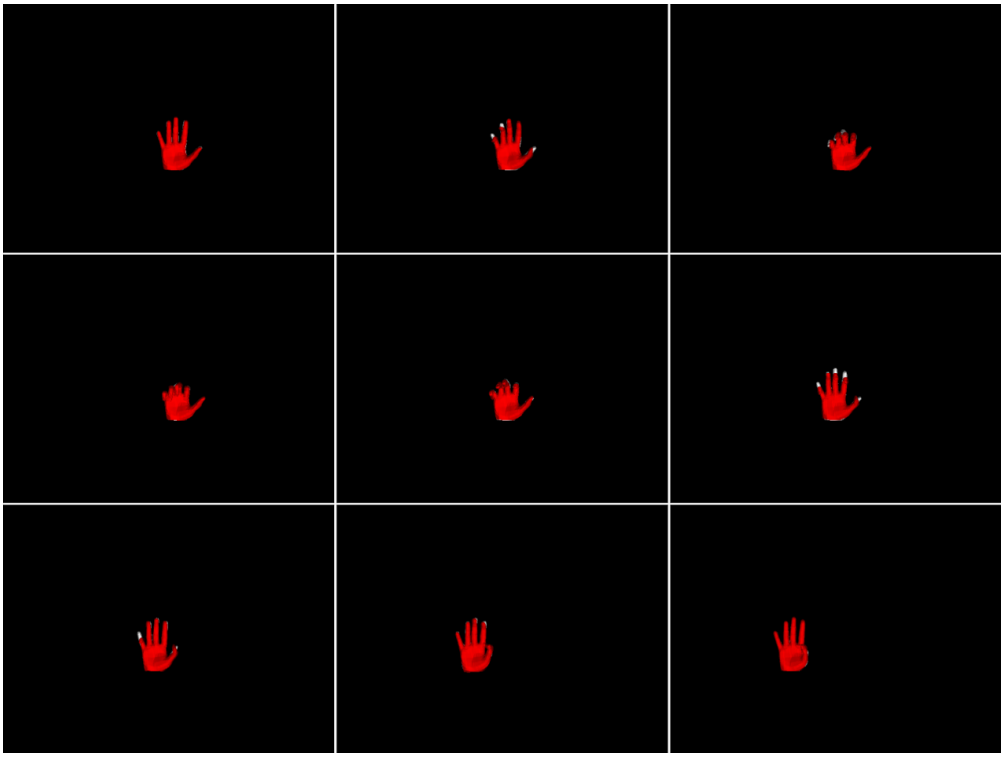

Figure 3.116: MS objective function images.

3.2.2.1.2.3. MSPL objective function

The MSPL objective function is used.

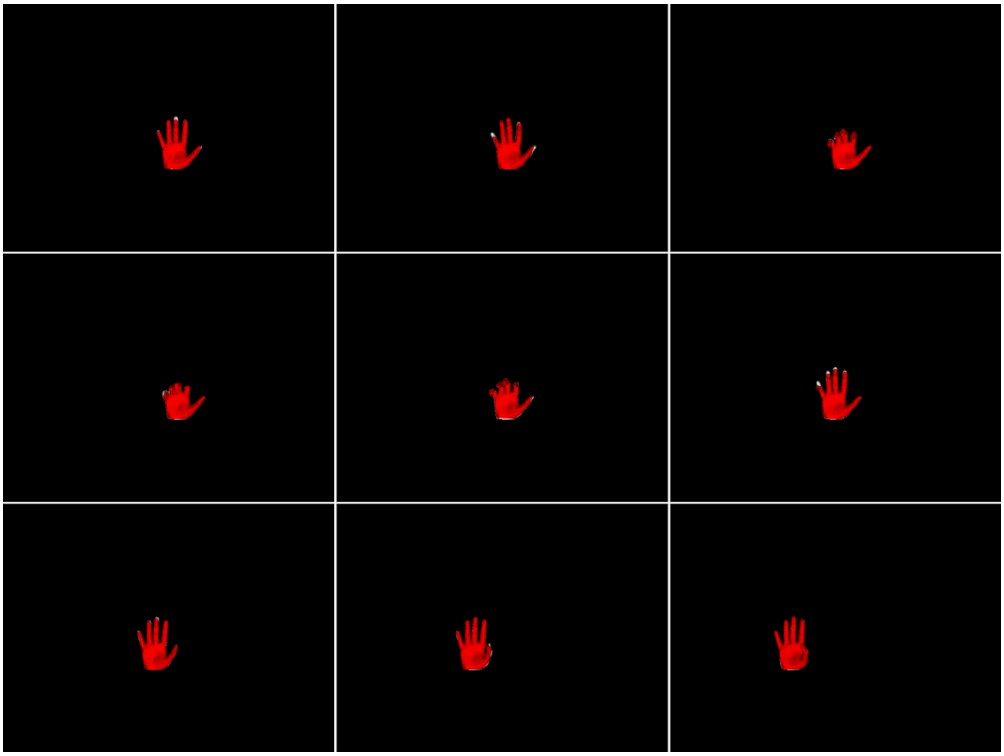

Figure 3.117: MSPL objective function images.

3.2.2.1.2.4. MCL objective function

The MCL objective function is used.

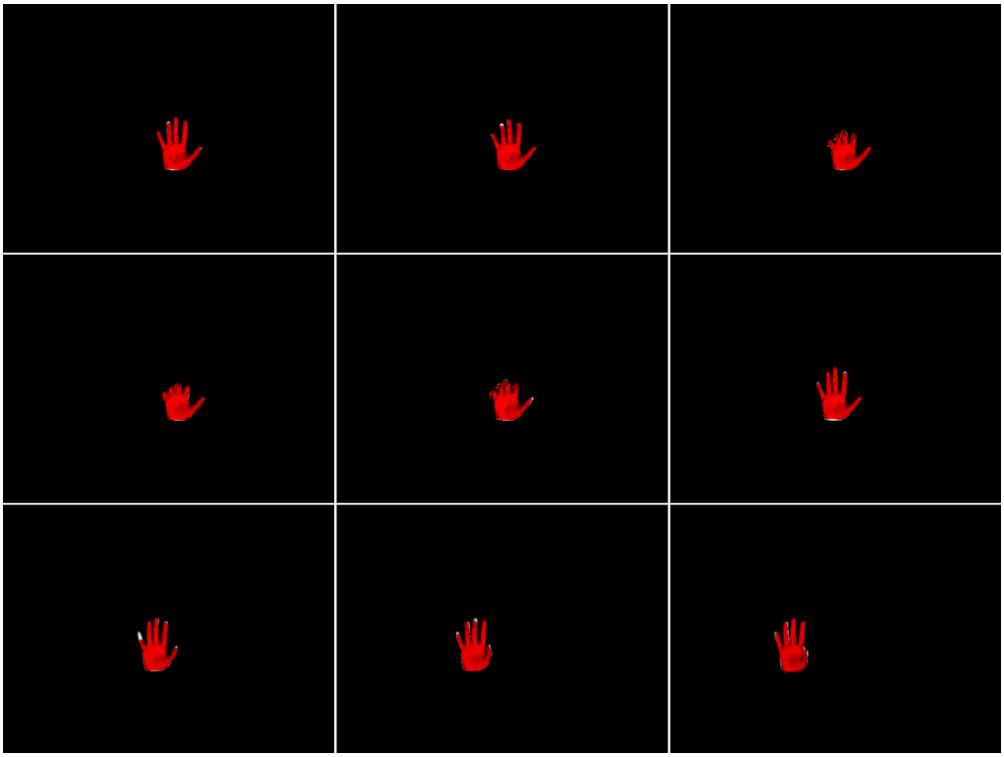

Figure 3.118: MCL objective function images.

3.2.2.1.3. Sequence 3

The observation is rotating forward, towards the camera while the fingers are bending. Then it is rotating to the right while the fingers are straightening and finally is rotating to the left.

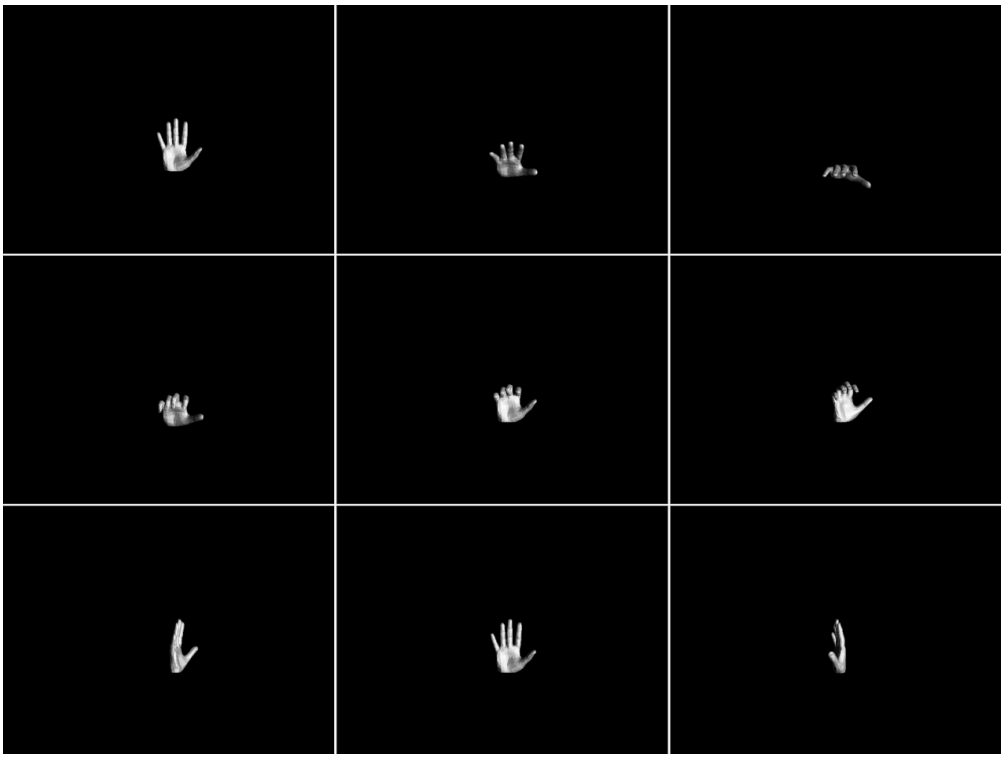

Figure 3.119: Sequence 3 observation images.

3.2.2.1.3.1. Default objective function

The default objective function is used.

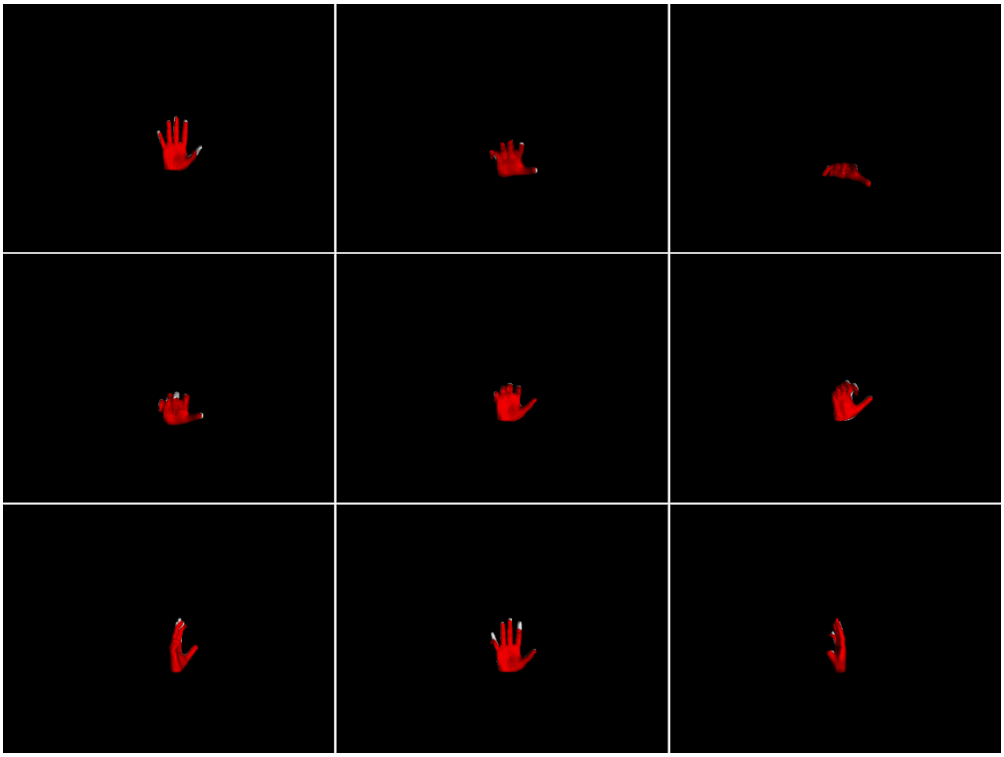

Figure 3.120: Default objective function images.

3.2.2.1.3.2. MS objective function

The MS objective function is used.

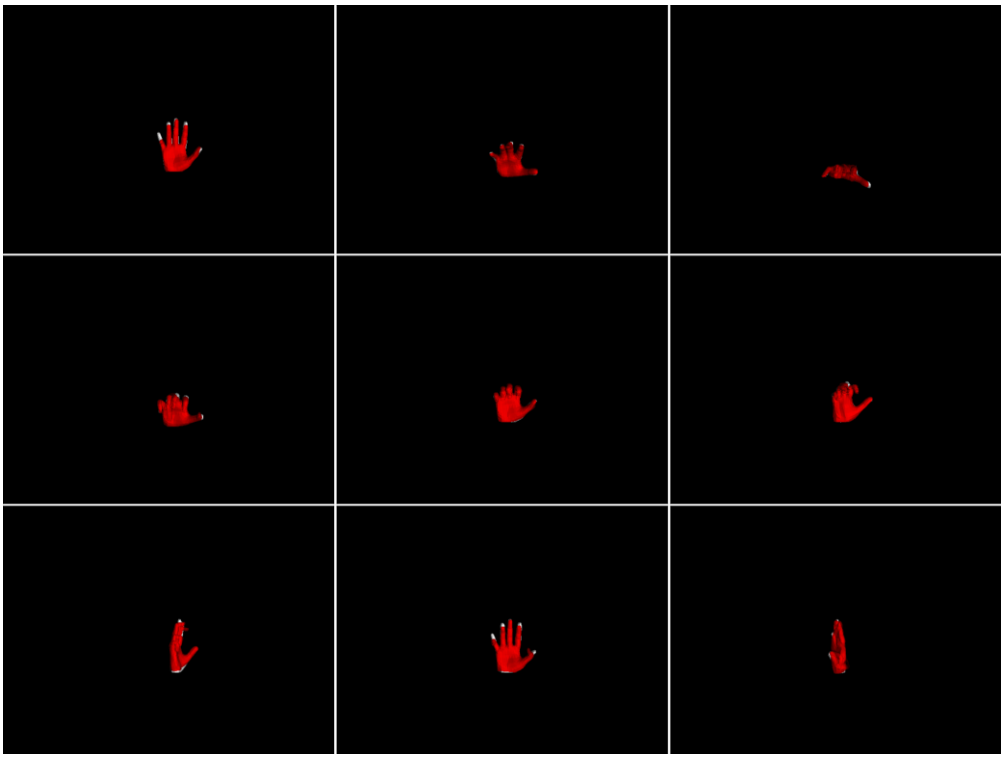

Figure 3.121: MS objective function images.

3.2.2.1.3.3. MSPL objective function

The MSPL objective function is used.

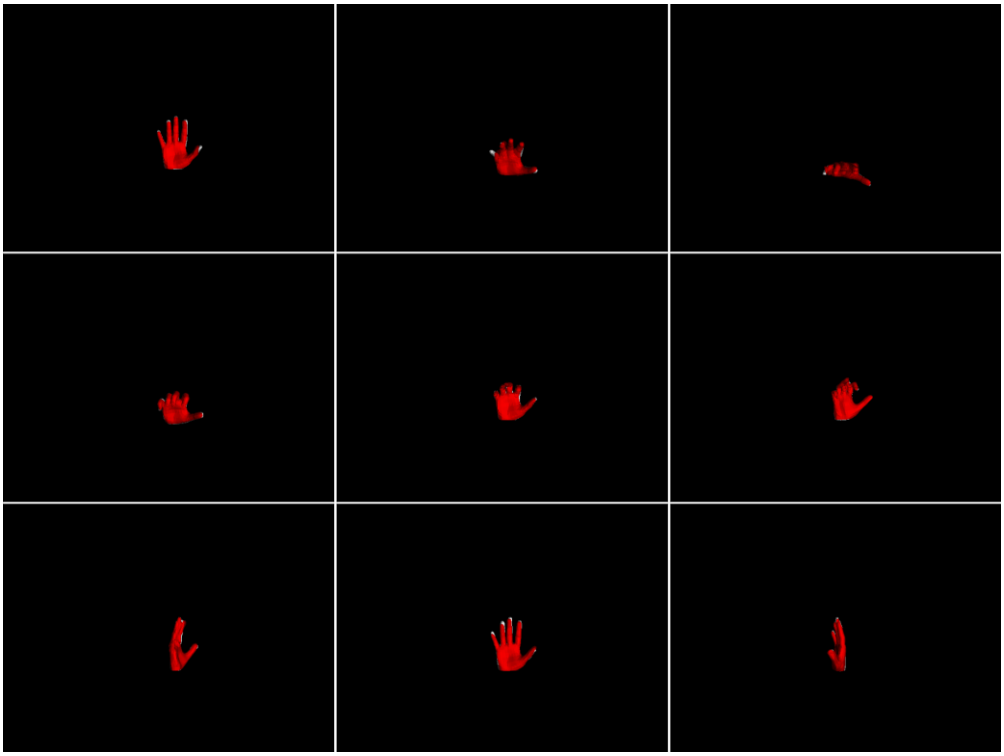

Figure 3.122: MSPL objective function images.

3.2.2.1.3.4. MCL objective function

The MCL objective function is used.

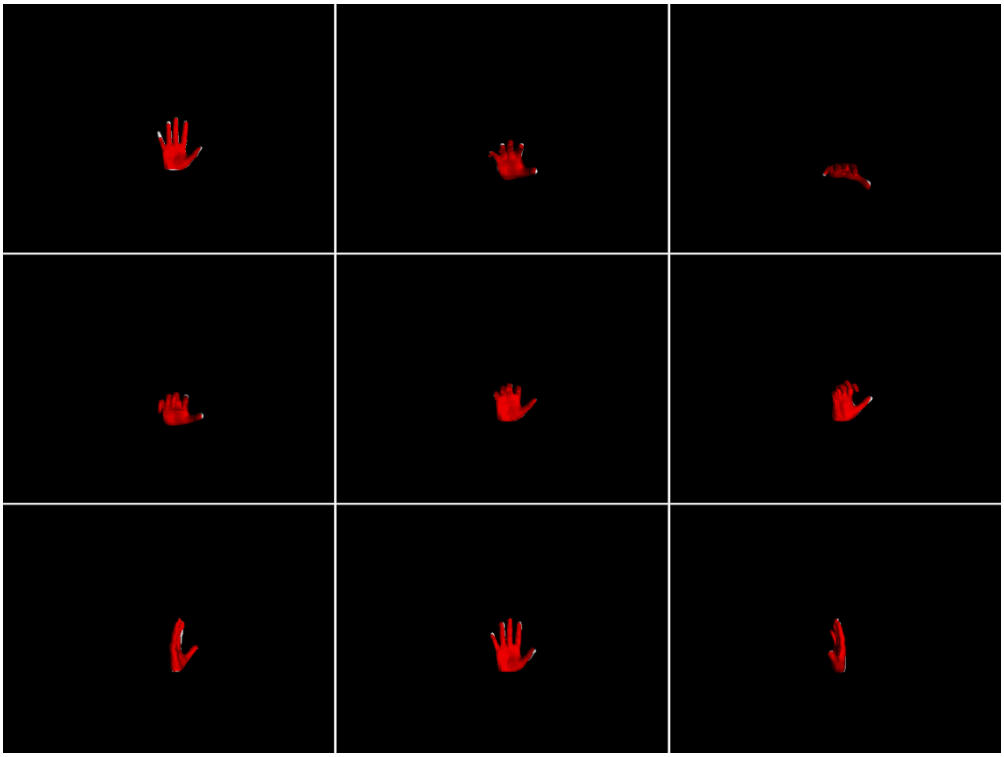

Figure 3.123: MCL objective function images.

3.2.2.1.4. Sequence 4

In this case, the fingers of the observation are bending forming a fist which is rotating to the right and then to the right. At the end, the index and the thumb start to straighten to form a “pistol” gesture that is pointing to the left.

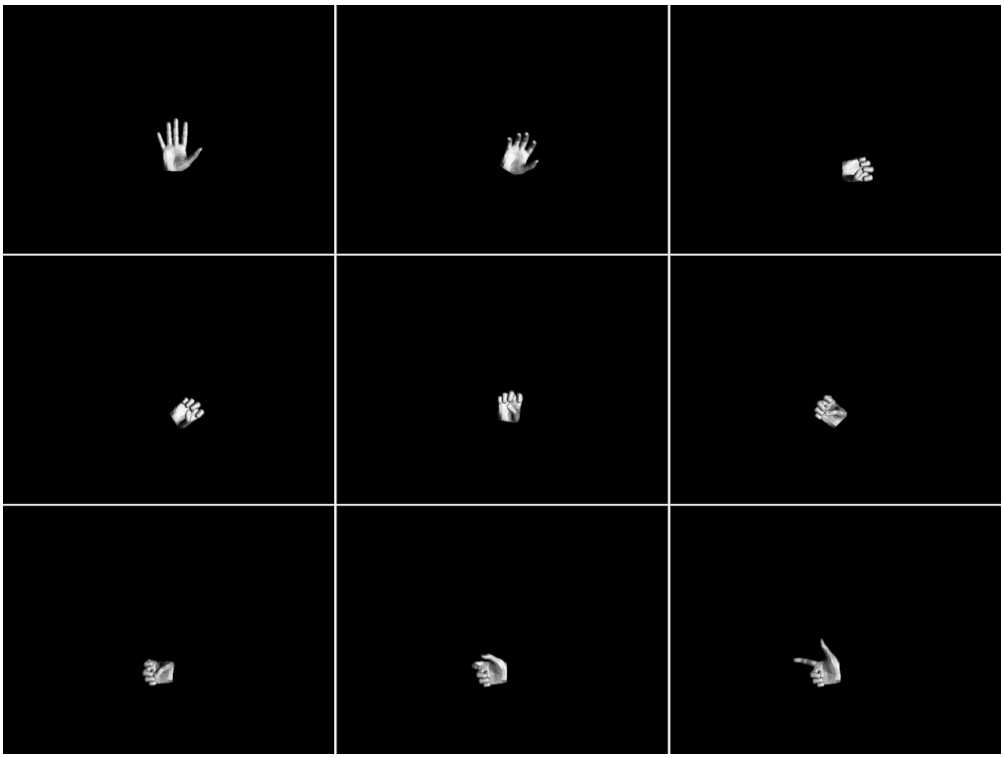

Figure 3.124: Sequence 4 observation images.

3.2.2.1.4.1. Default objective function

The default objective function is used.

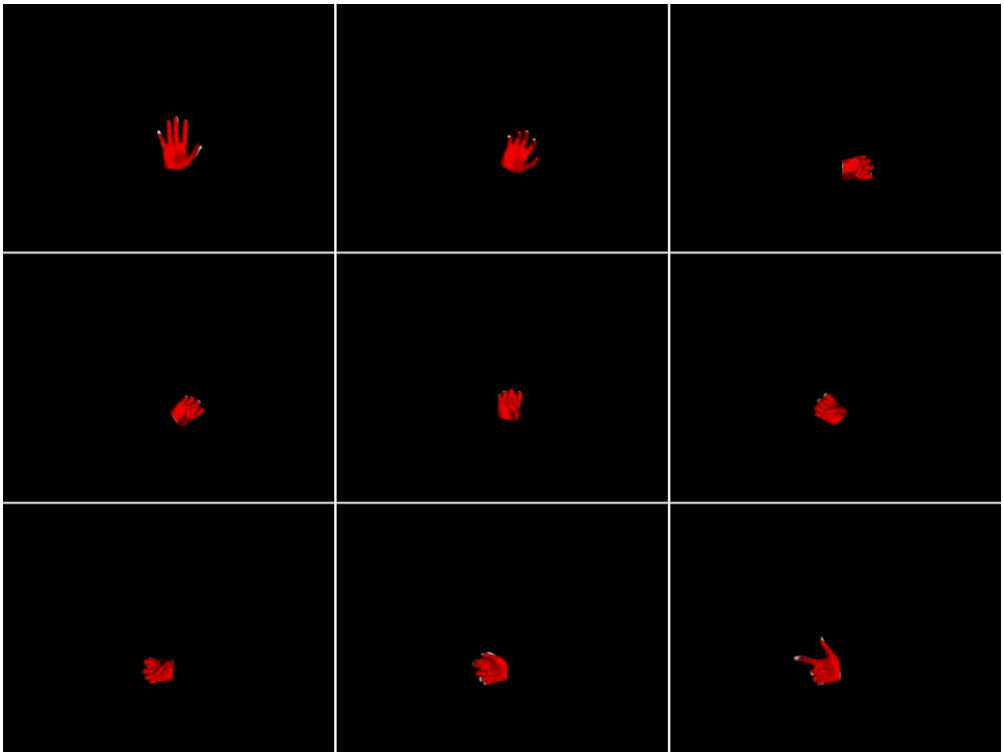

Figure 3.125: MCL objective function images.

3.2.2.1.4.2. MS objective function

The MS objective function is used.

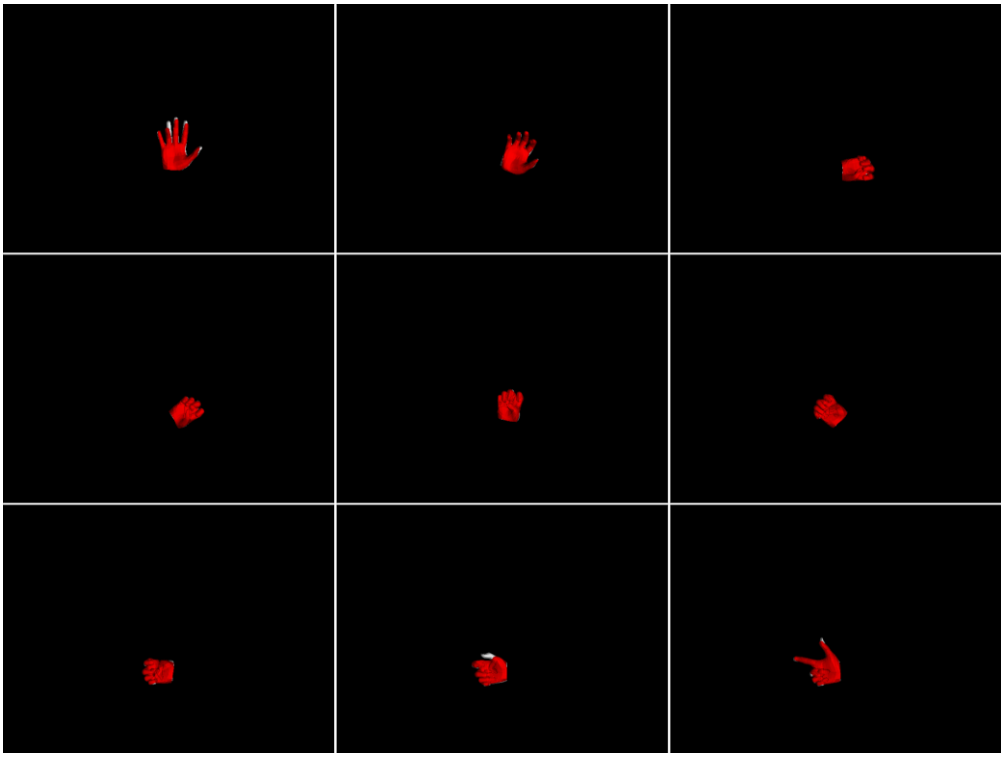

Figure 3.126: MS objective function images.

3.2.2.1.4.3. MSPL objective function

The MSPL objective function is used.

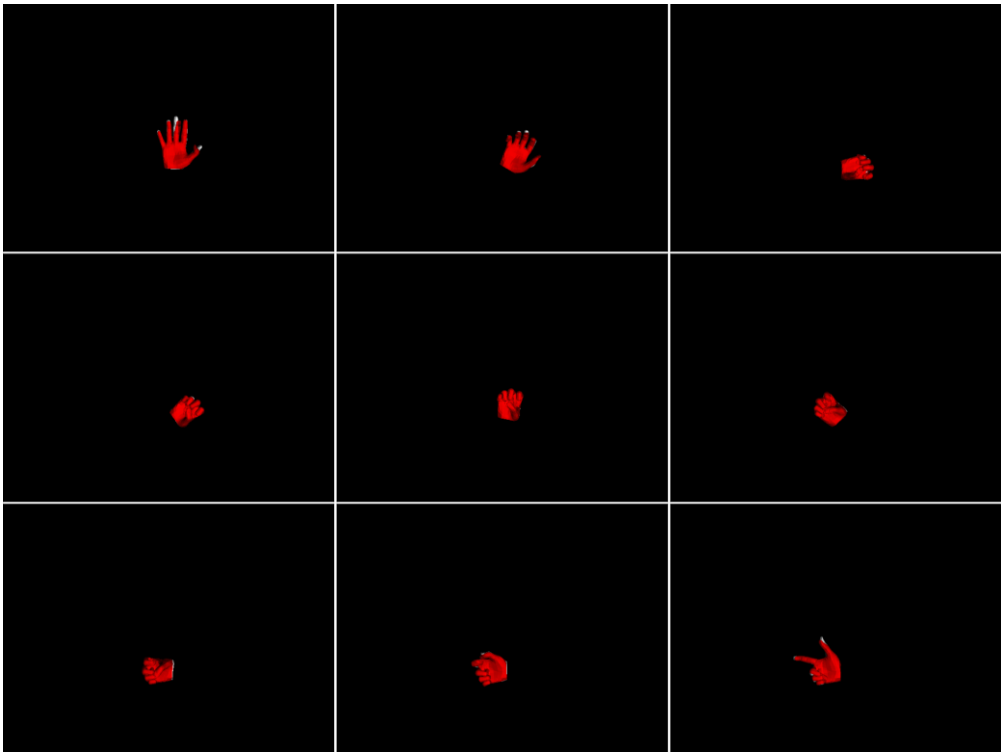

Figure 3.127: MSPL objective function images.

3.2.2.1.4.4. MCL objective function

The MCL objective function is used.

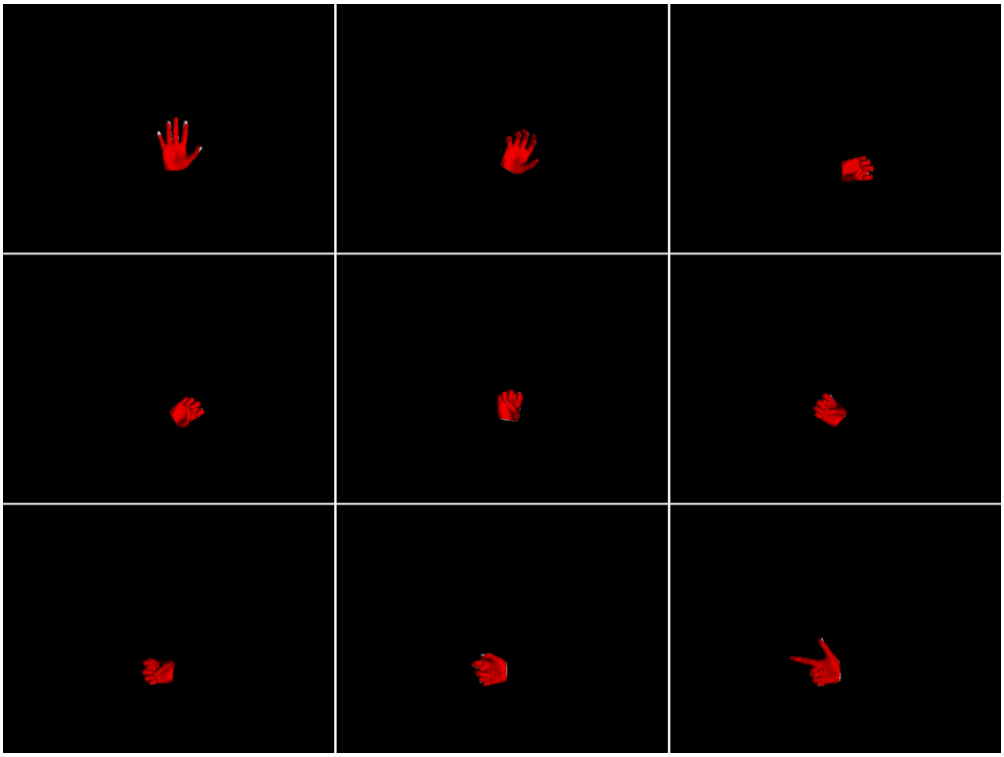

Figure 3.128: MCL objective function images.

3.2.2.1.5. Sequence 5

The observation is rotating forward, towards the camera until a certain point. The fingers start to bend and then to straighten while the observation is rotating back to its initial state.

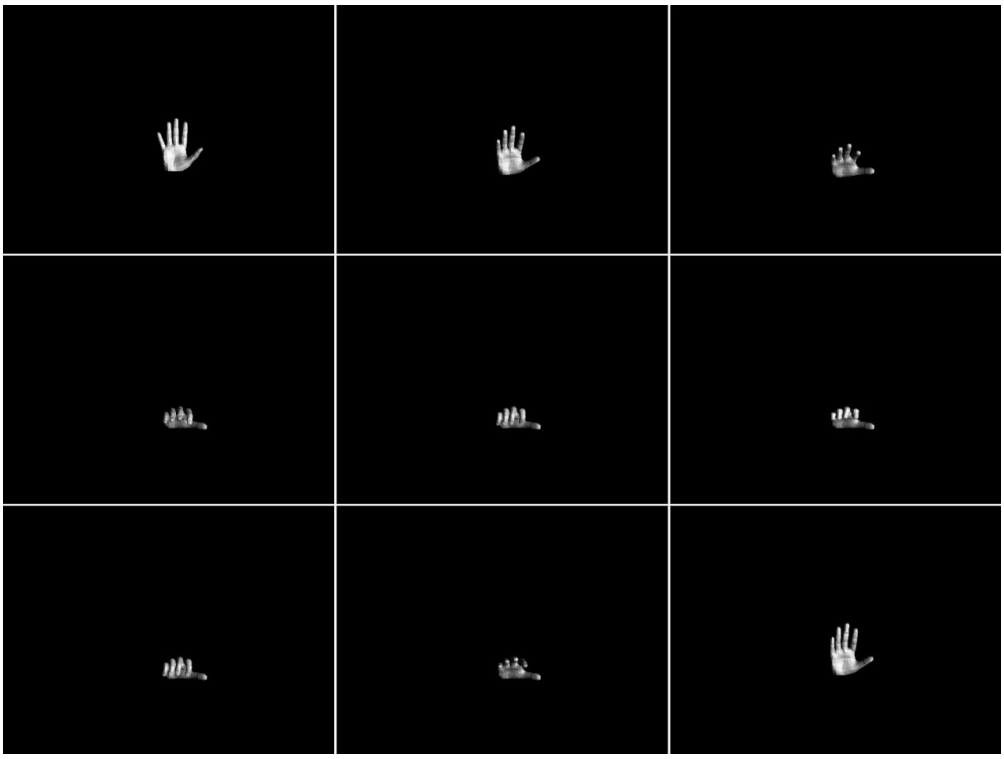

Figure 3.129: Sequence 5 observation images.

3.2.2.1.5.1. Default objective function

The default objective function is used.

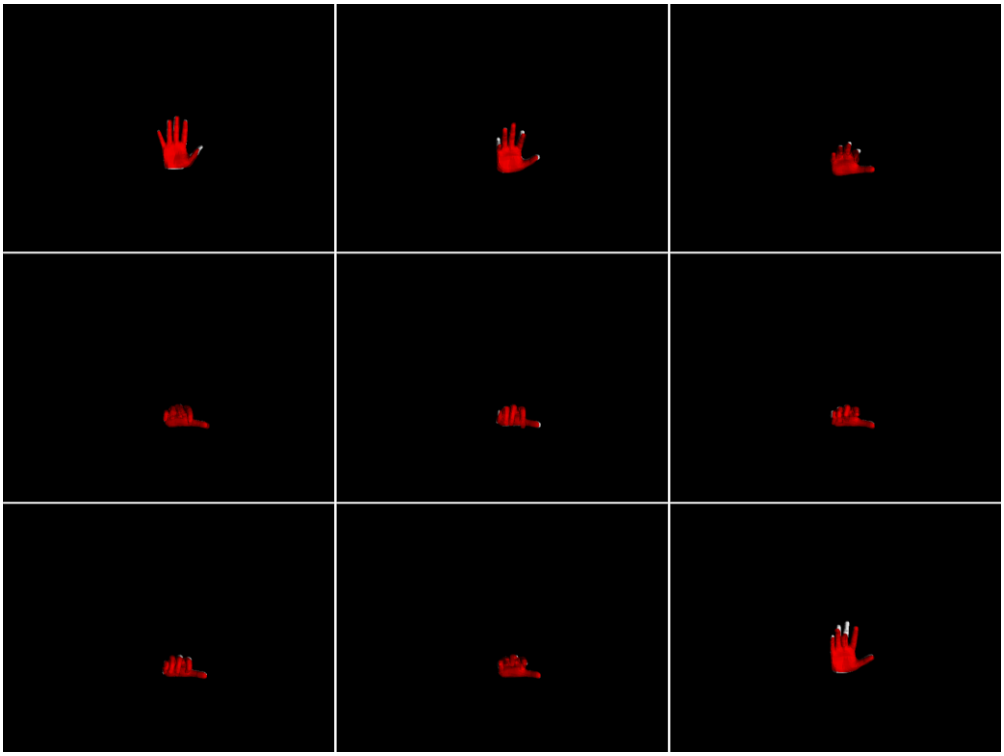

Figure 3.130: Default objective function images.

3.2.2.1.5.2. MS objective function

The MS objective function is used.

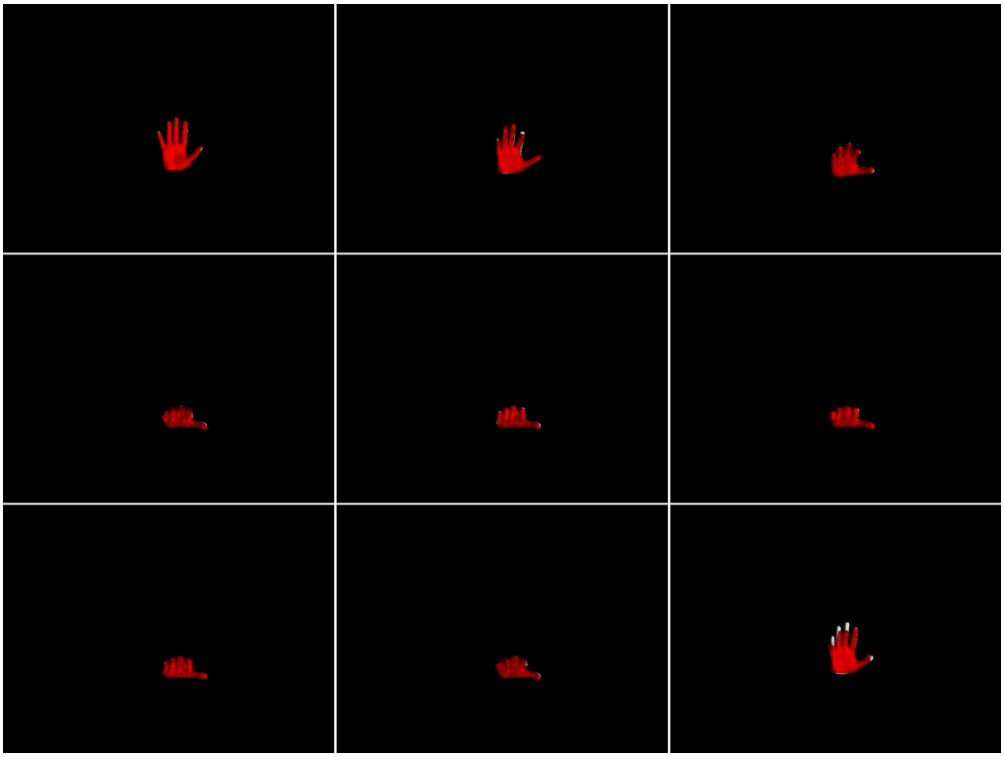

Figure 3.131: MS objective function images.

3.2.2.1.5.3. MSPL objective function

The MSPL objective function is used.

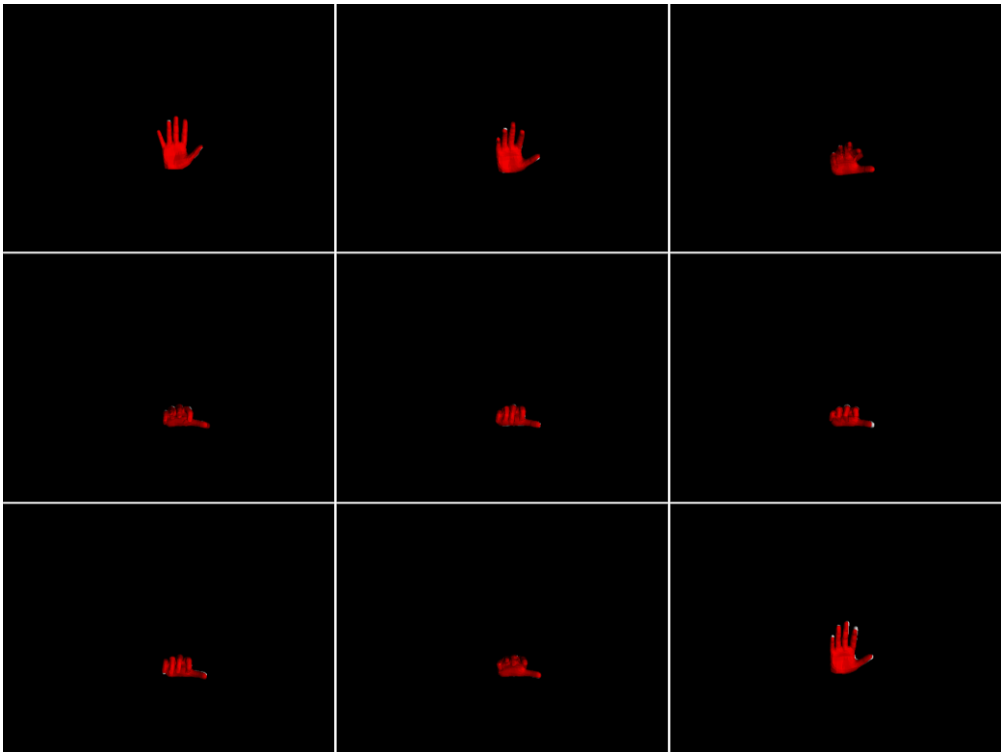

Figure 3.132: MSPL objective function images.

3.2.2.1.5.4. MCL objective function

The MCL objective function is used.

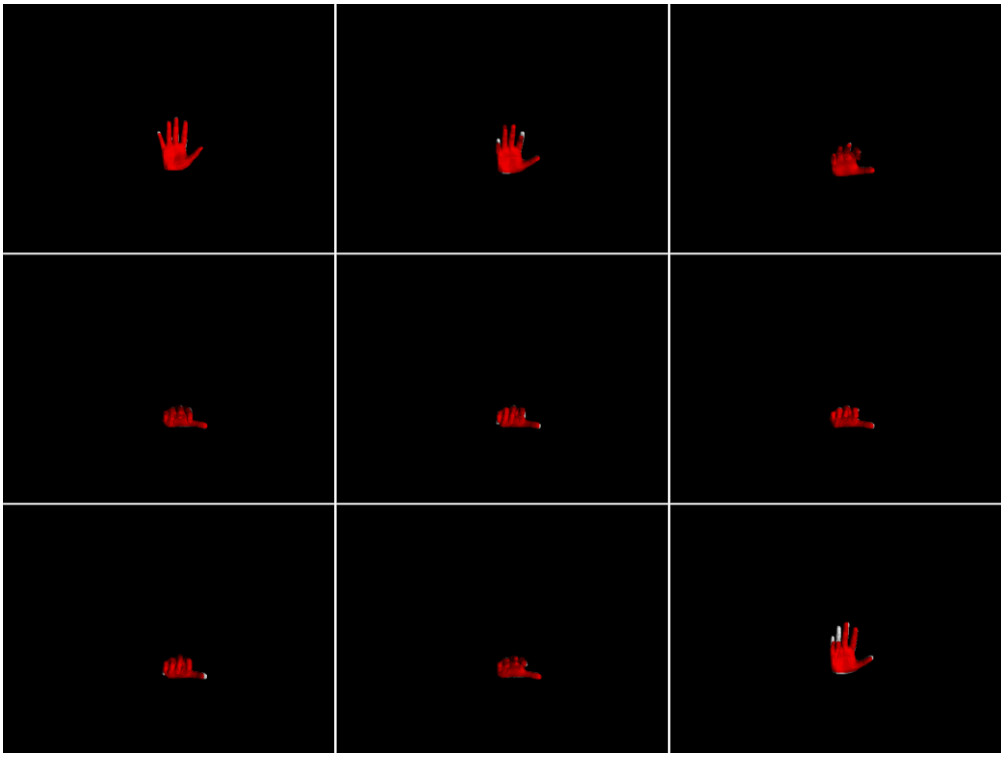

Figure 3.133: MCL objective function images.

3.2.2.1.6. Sequence 6

The observation is moving forward towards to the camera and at the same time is rotating facing to the right. Then it is rotating until the palm is facing to the camera and the ring finger is bent.

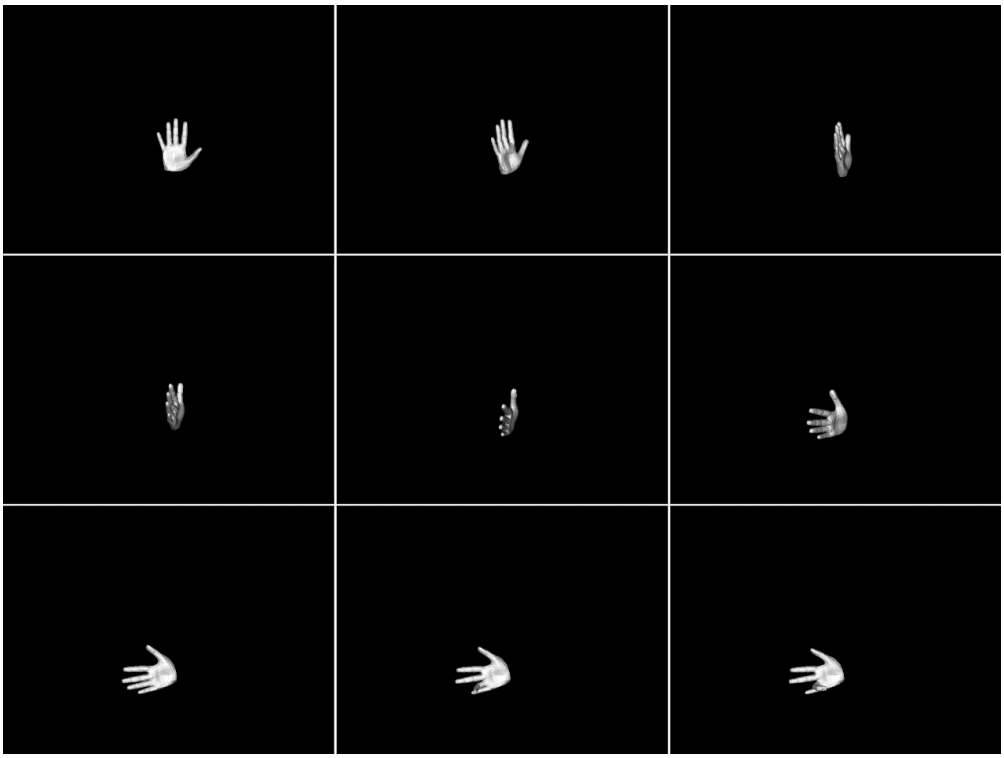

Figure 3.134: Sequence 6 observation images.

3.2.2.1.6.1. Default objective function

The default objective function is used.

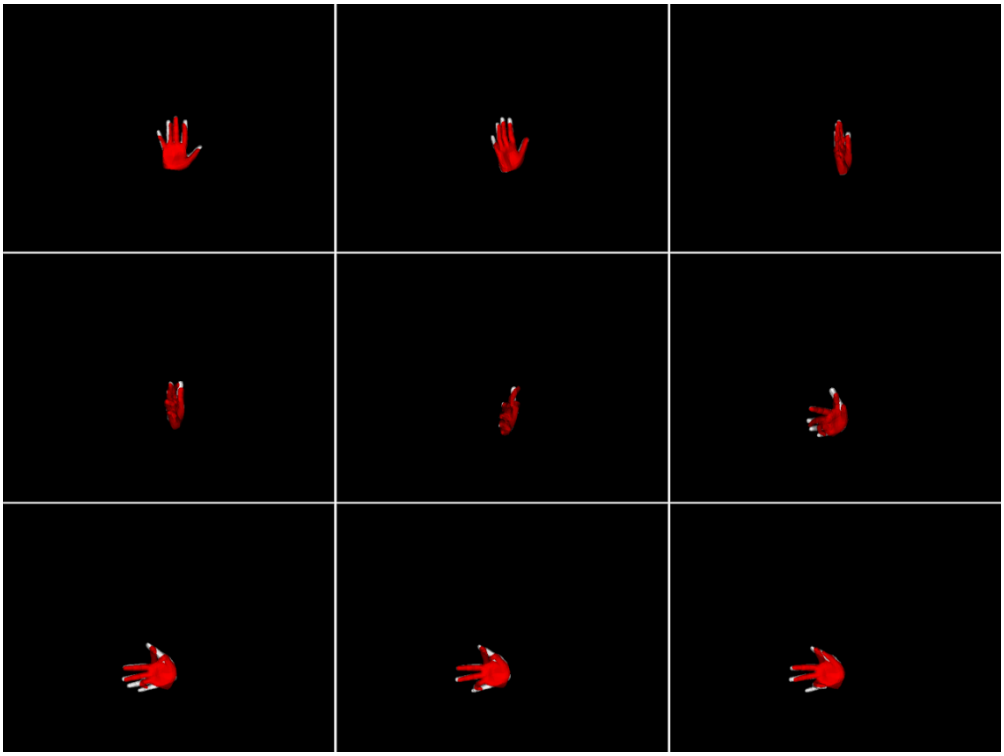

Figure 3.135: Default objective function images.

3.2.2.1.6.2. MS objective function

The MS objective function is used.

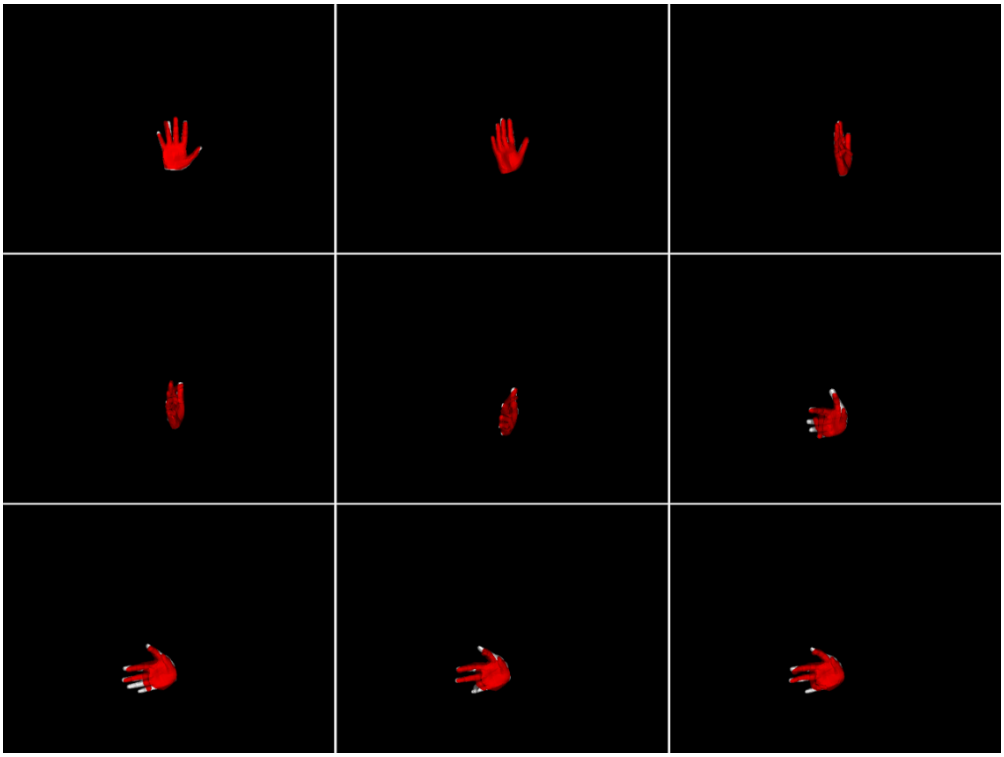

Figure 3.136: MS objective function images.

3.2.2.1.6.3. MSPL objective function

The MSPL objective function is used.

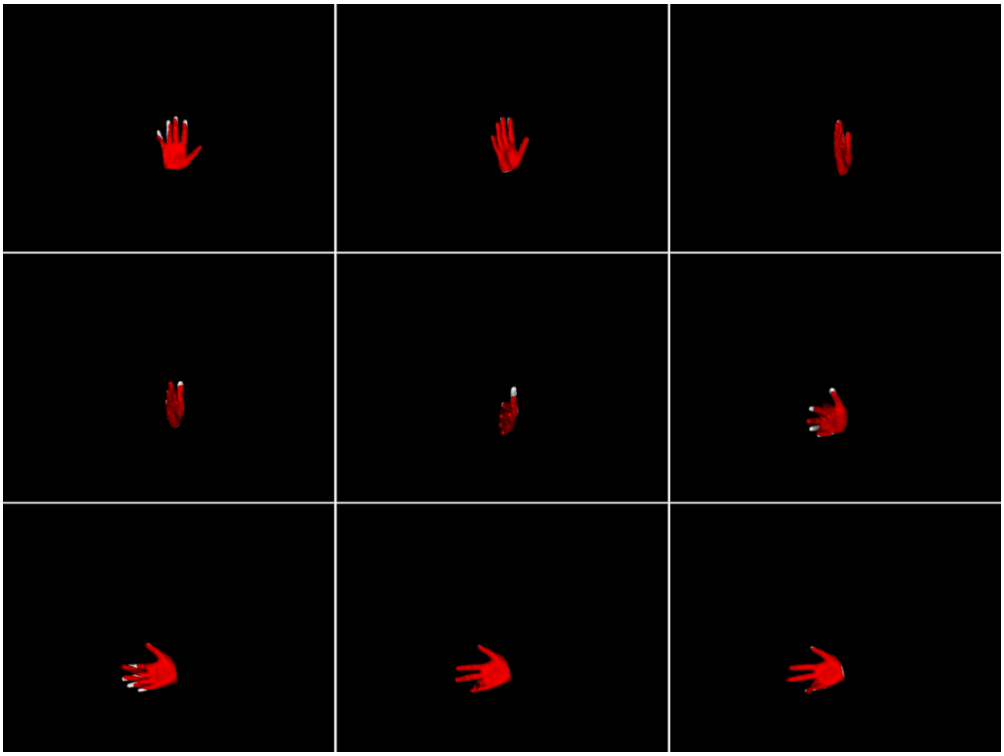

Figure 3.137: MSPL objective function images.

3.2.2.1.6.4. MCL objective function

The MCL objective function is used.

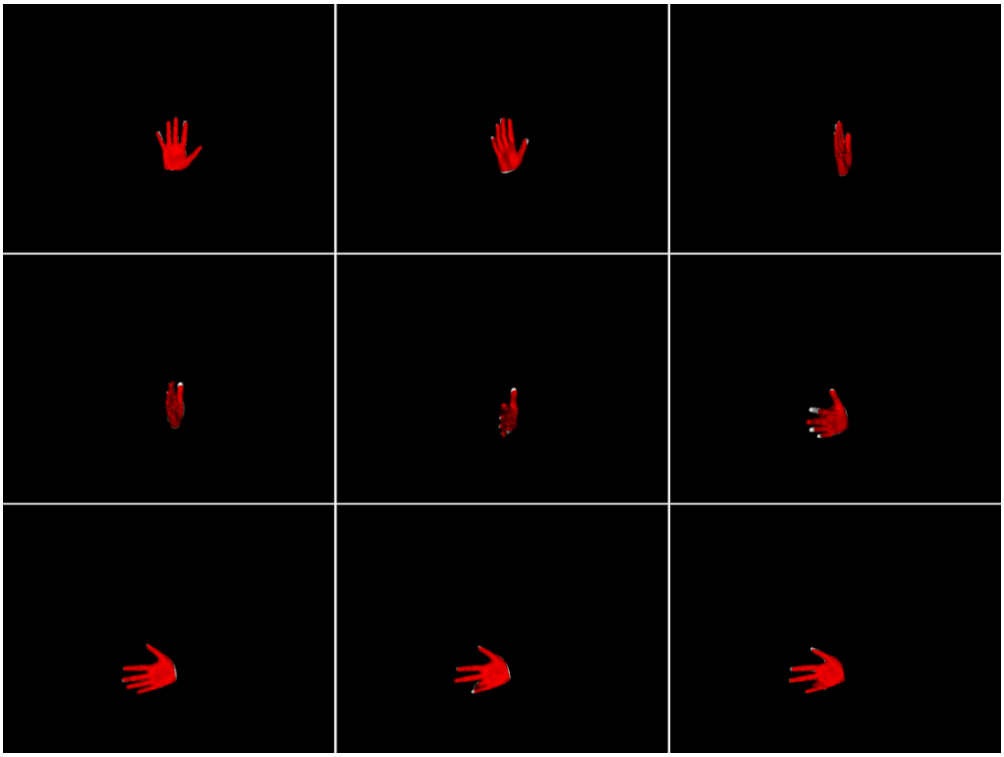

Figure 3.138: MCL objective function images.

3.2.2.2. Simulate sequence tracking – 256 particles

The same sequences that were shown before are going to be tested again using 256 (hypotheses) particles this time. The first two sequences (Sequence 1 and Sequence 2) where there are no complex moves were not tested.

3.2.2.2.3. Sequence 3

The observation is rotating forward, towards the camera while the fingers are bending. Then it is rotating to the right while the fingers are straightening and finally is rotating to the left.

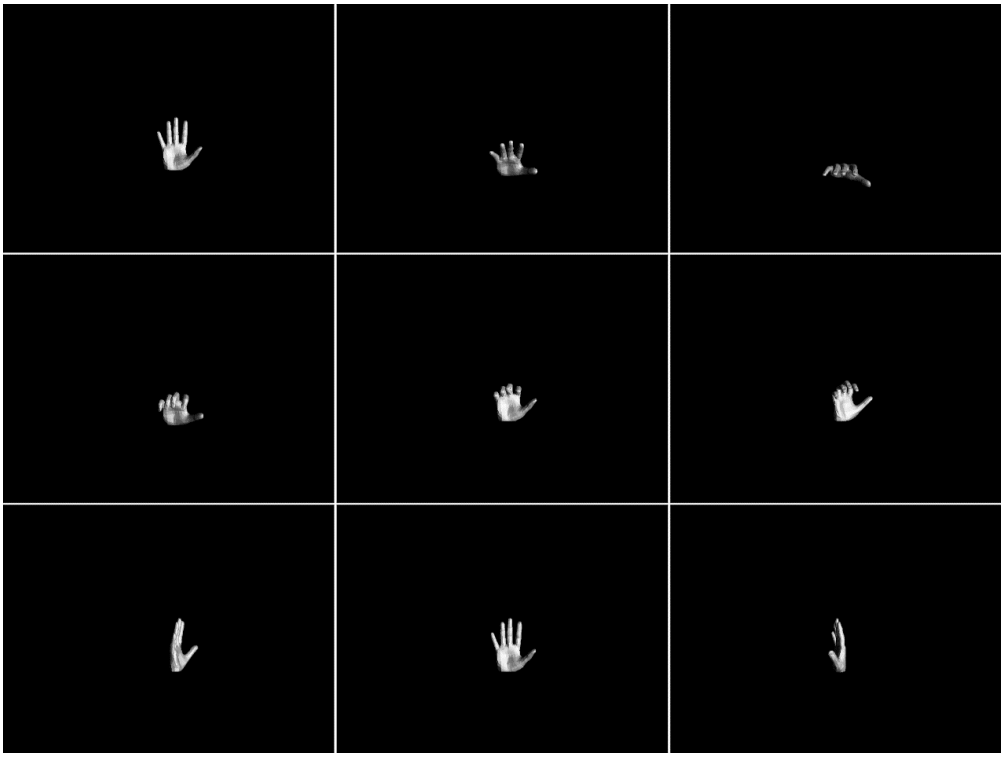

Figure 3.139: Sequence 3 observation images.

3.2.2.2.3.1. Default objective function

The default objective function is used.

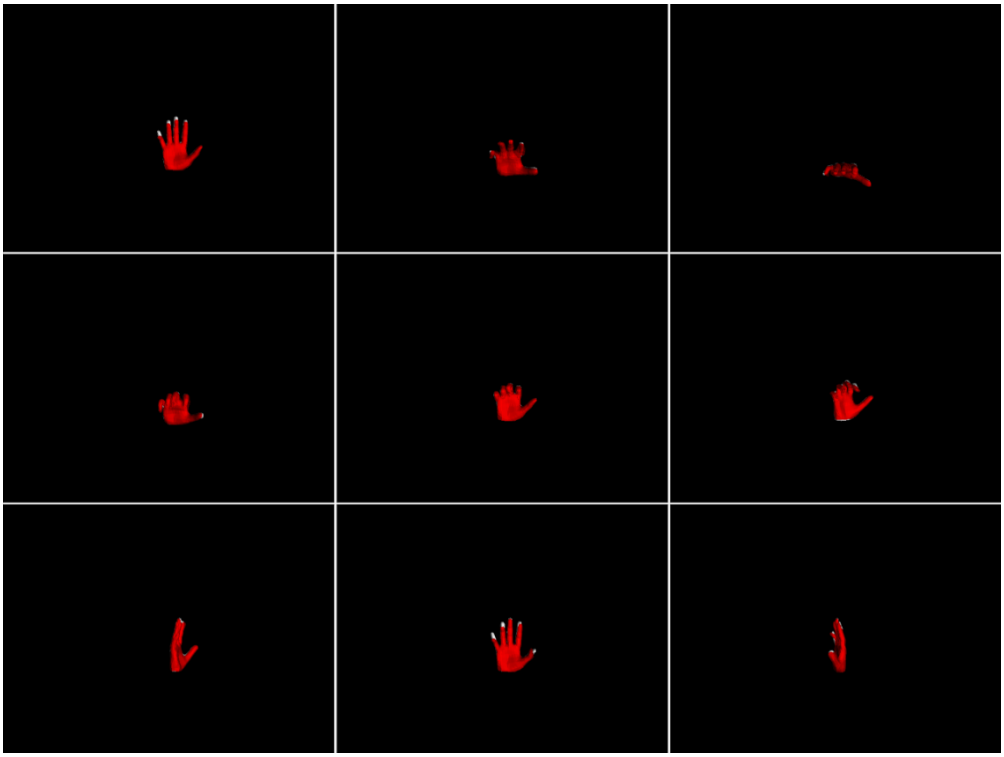

Figure 3.140: Default objective function images.

3.2.2.2.3.2. MS objective function

The MS objective function is used.

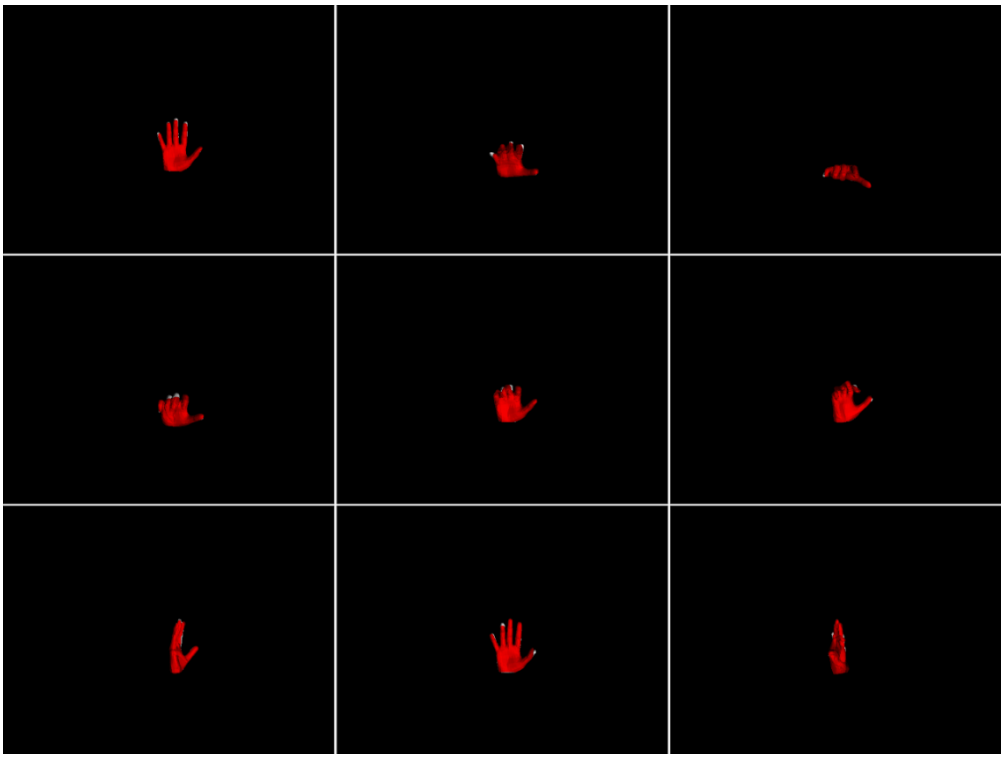

Figure 3.141: MS objective function images.

3.2.2.2.3.3. MSPL objective function

The MSPL objective function is used.

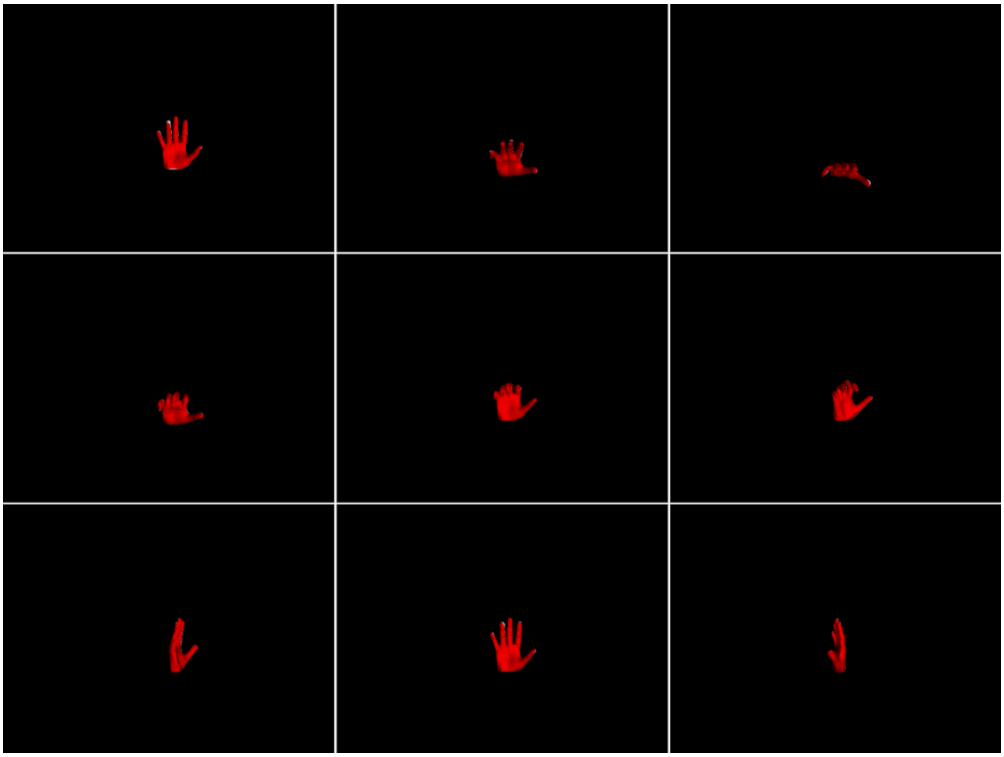

Figure 3.142: MSPL objective function images.

3.2.2.2.3.4. MCL objective function

The MCL objective function is used.

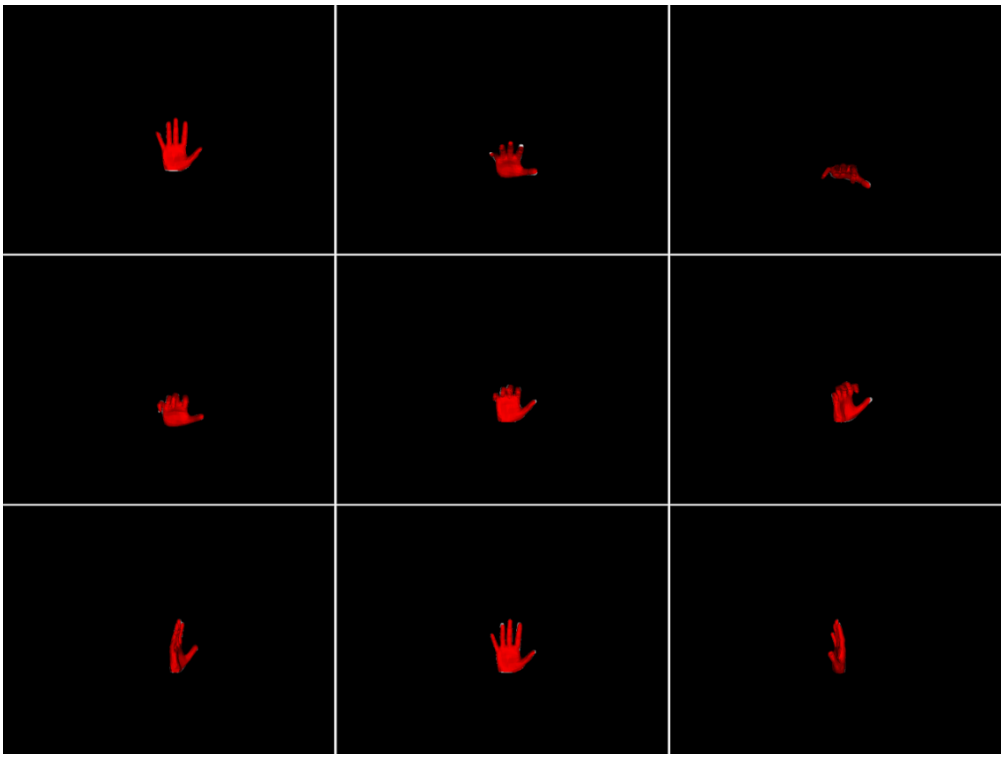

Figure 3.143: MCL objective function images.

3.2.2.2.4. Sequence 4

In this case, the fingers of the observation are bending forming a fist which is rotating to the right and then to the right. At the end, the index and the thumb start to straighten to form a “pistol” gesture that is pointing to the left.

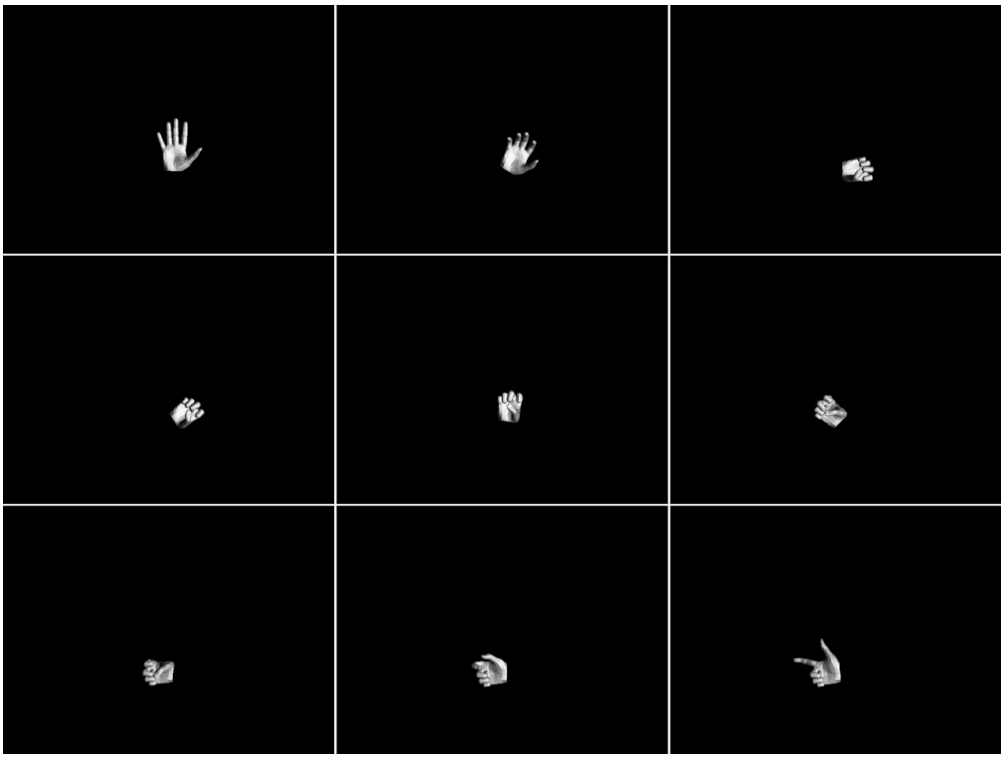

Figure 3.144: Sequence 4 observation images.

3.2.2.2.4.1. Default objective function

The default objective function is used.

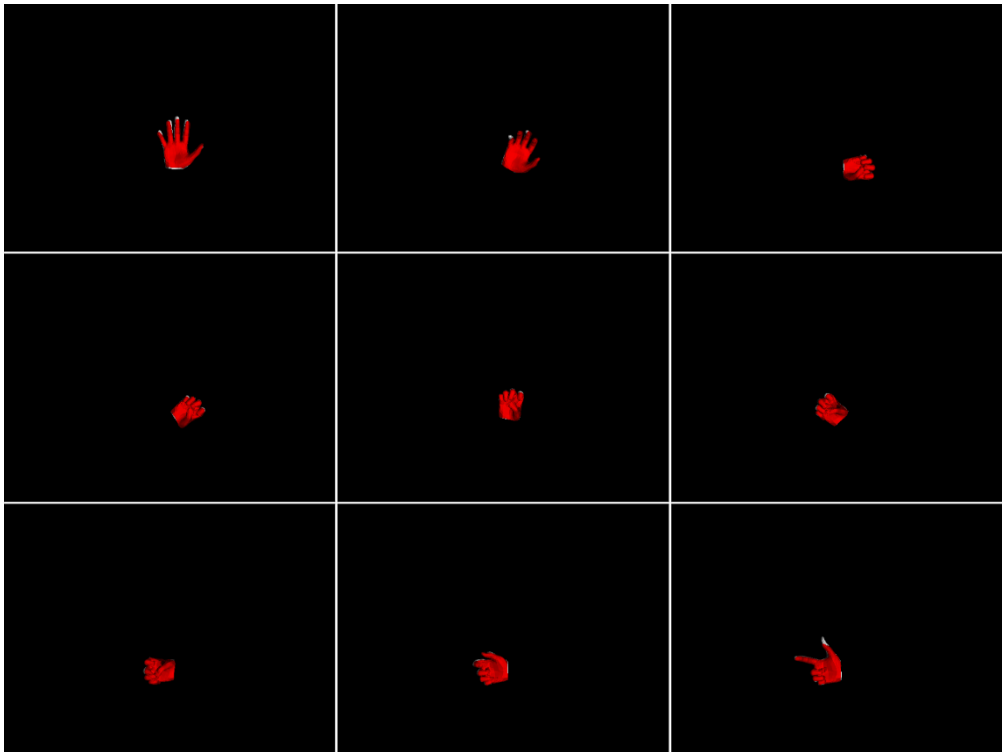

Figure 3.145: Default objective function images.

3.2.2.2.4.2. MS objective function

The MS objective function is used.

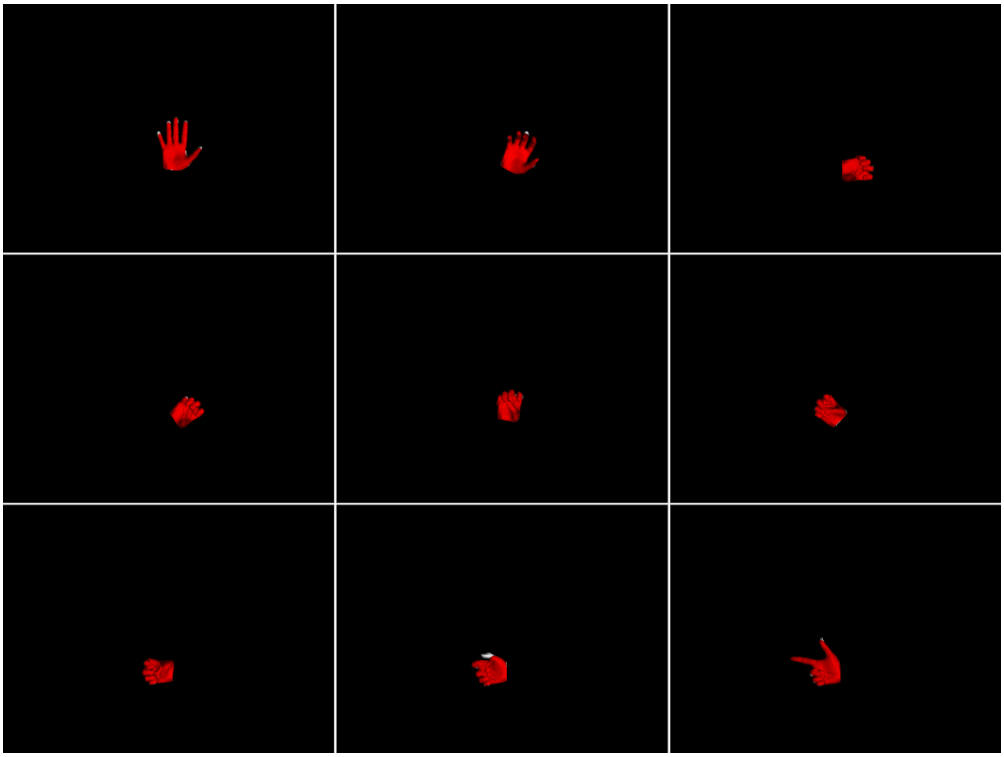

Figure 3.146: MS objective function images.

3.2.2.2.4.3. MSPL objective function

The MSPL objective function is used.

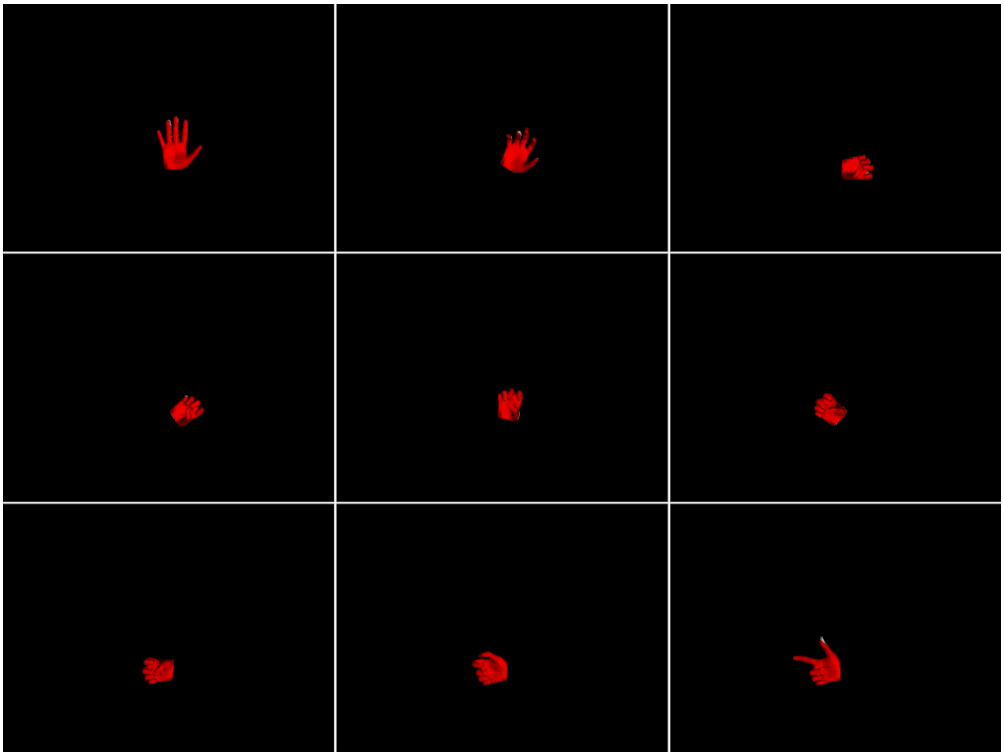

Figure 3.147: MSPL objective function images.

3.2.2.2.4.4. MCL objective function

The MCL objective function is used.

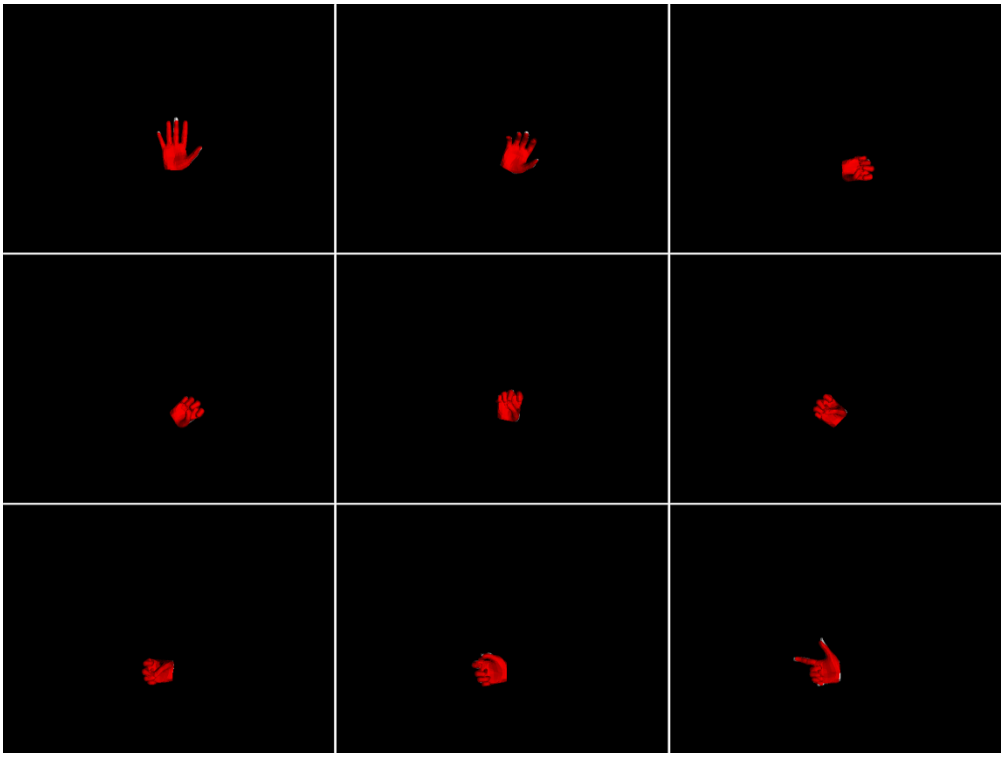

Figure 3.148: MCL objective function images.

3.2.2.2.5. Sequence 5

The observation is rotating forward, towards the camera until a certain point. The fingers start to bend and then to straighten while the observation is rotating back to its initial state.

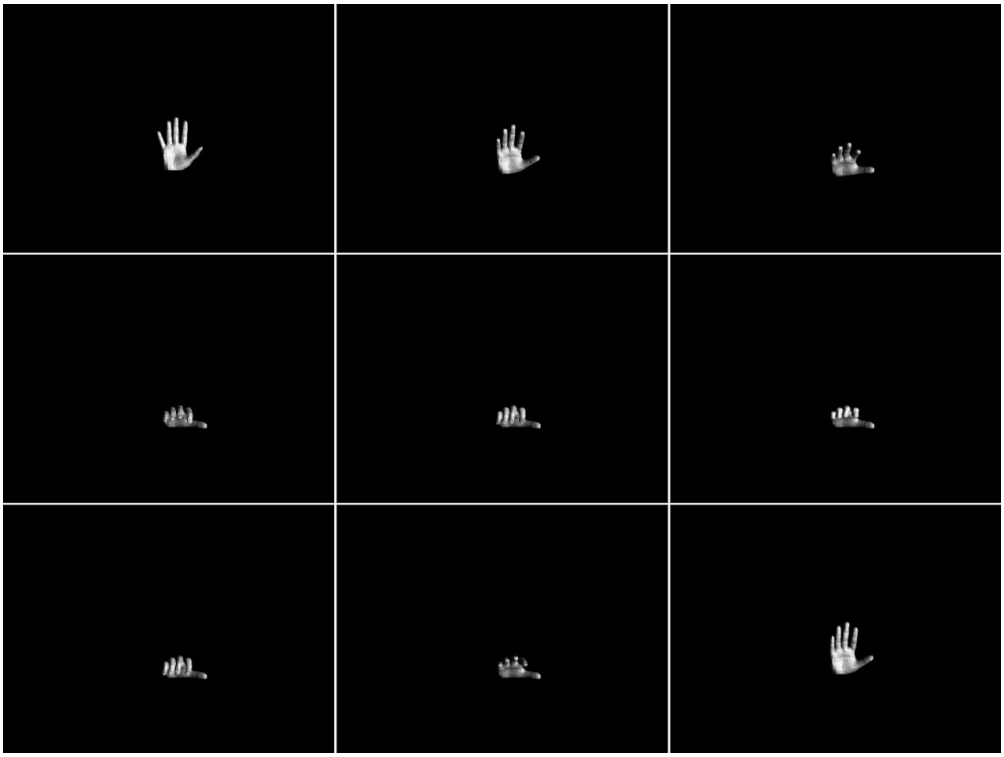

Figure 3.149: Sequence 5 observation images.

3.2.2.2.5.1. Default objective function

The default objective function is used.

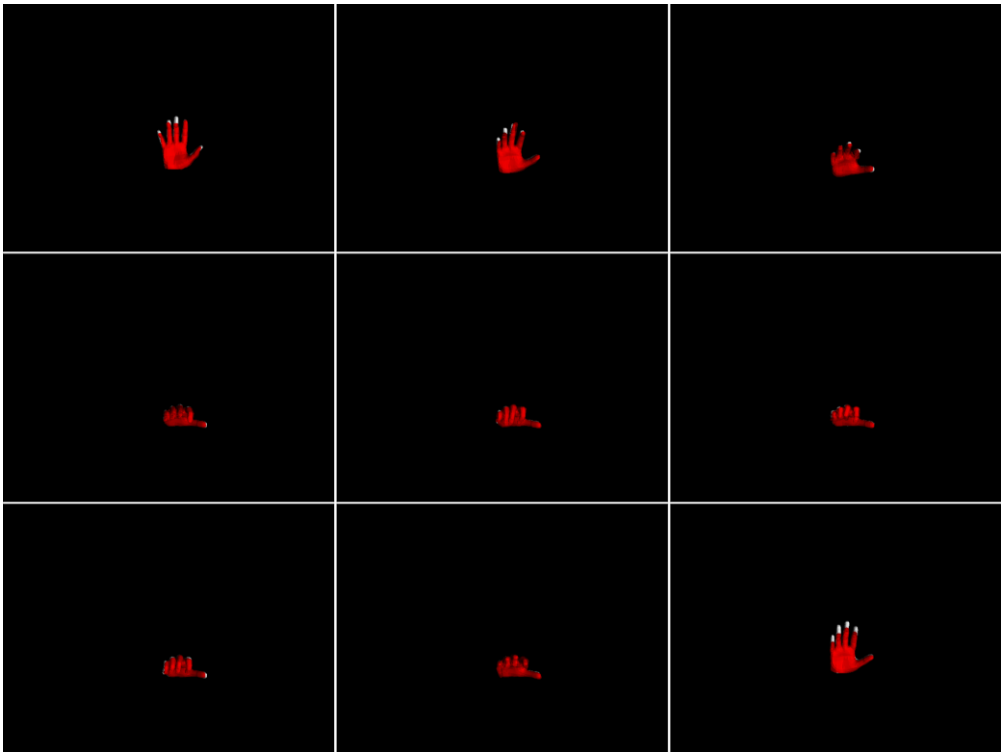

Figure 3.150: Default objective function images.

3.2.2.2.5.2. MS objective function

The MS objective function is used.

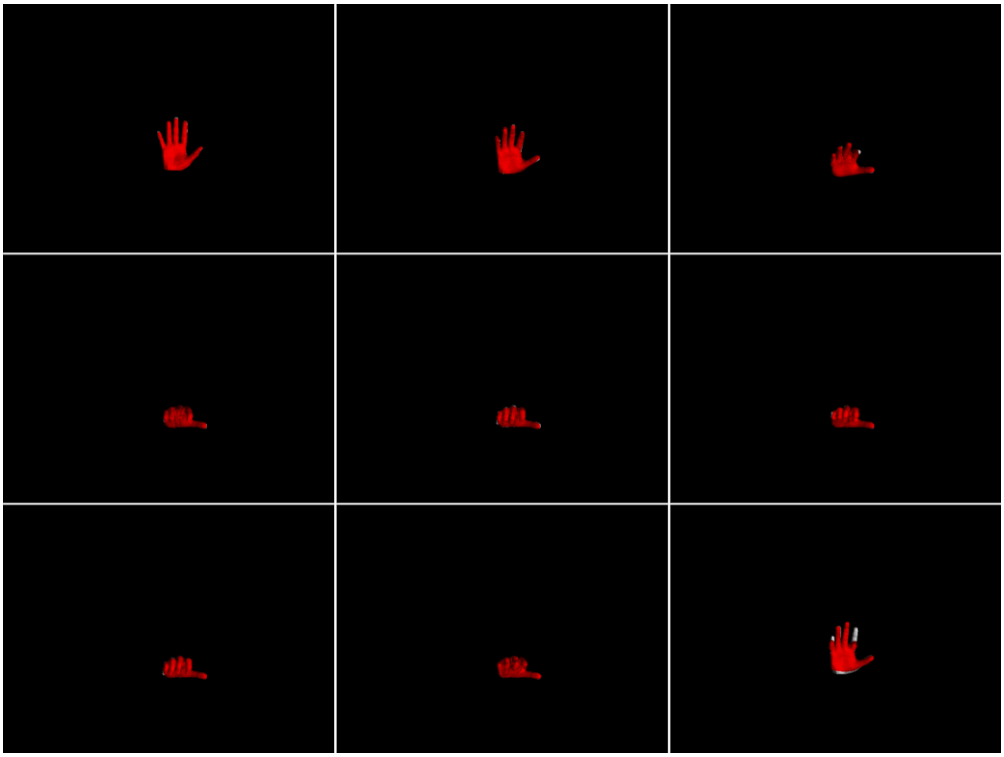

Figure 3.151: MS objective function images.

3.2.2.2.5.3. MSPL objective function

The MSPL objective function is used.

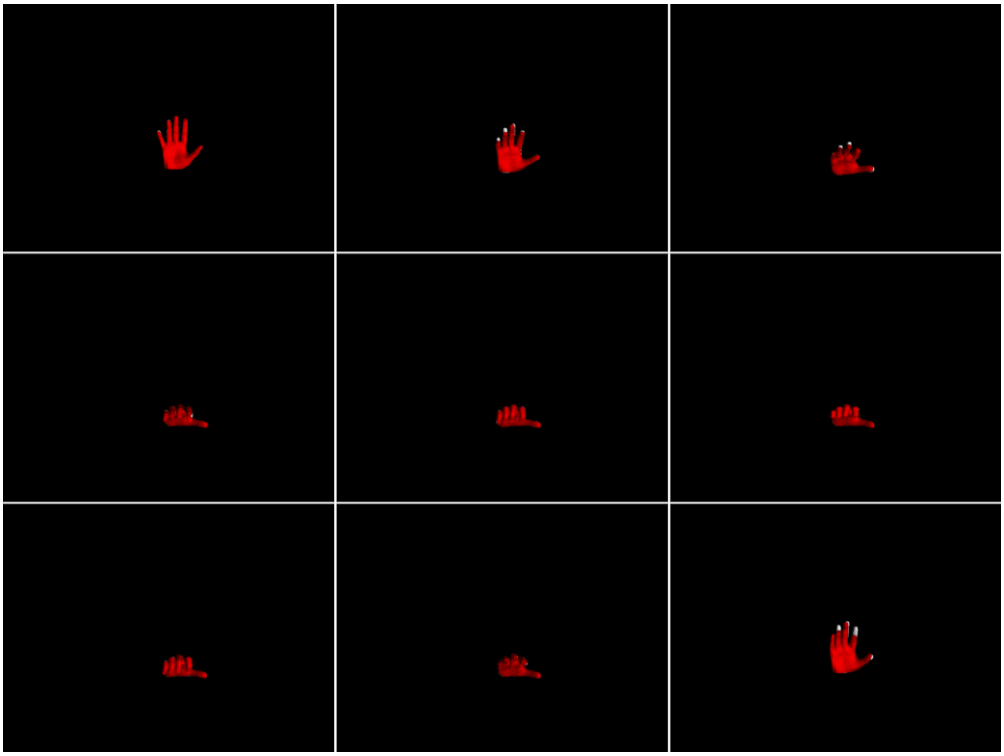

Figure 3.152: MSPL objective function images.

3.2.2.2.5.4. MCL objective function

The MCL objective function is used.

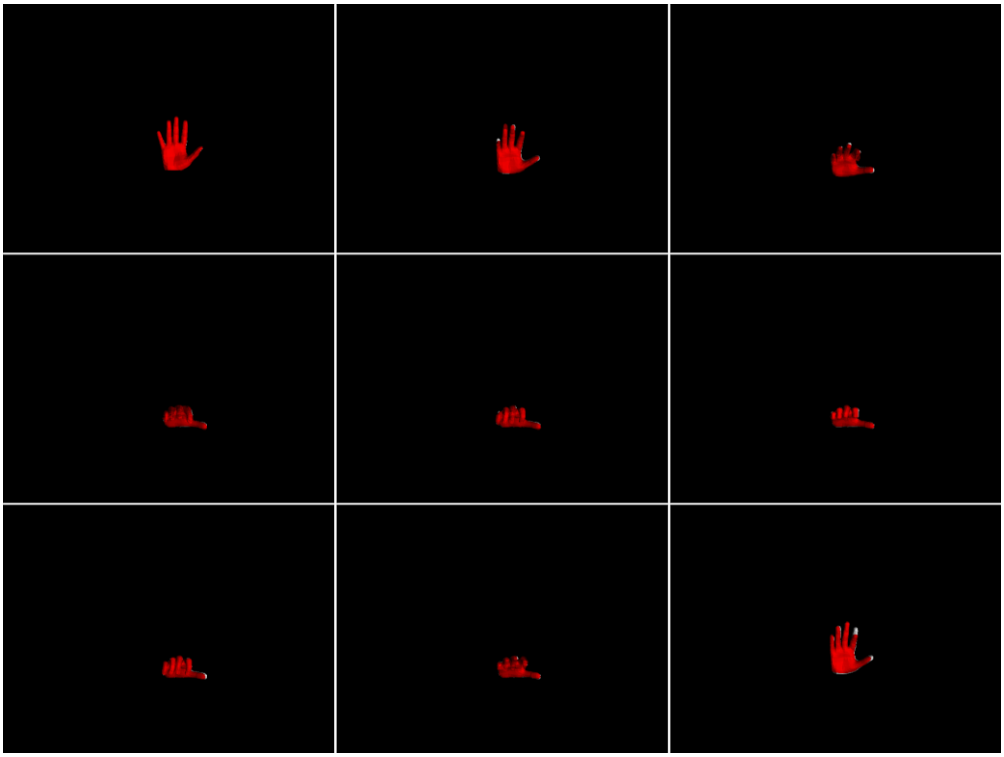

Figure 3.153: MCL objective function images.

3.2.2.2.6. Sequence 6

The observation is moving forward towards to the camera and at the same time is rotating facing to the right. Then it is rotating until the palm is facing to the camera and the ring finger is bent.

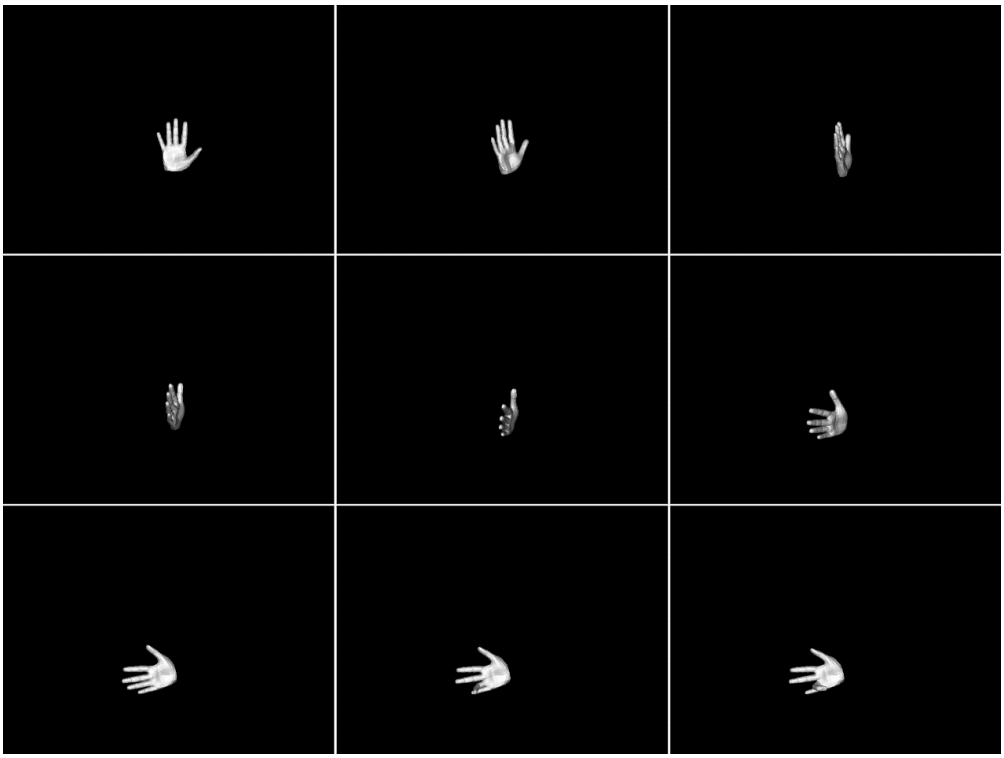

Figure 3.154: Sequence 6 observation images.

3.2.2.2.6.1. Default objective function

The default objective function is used.

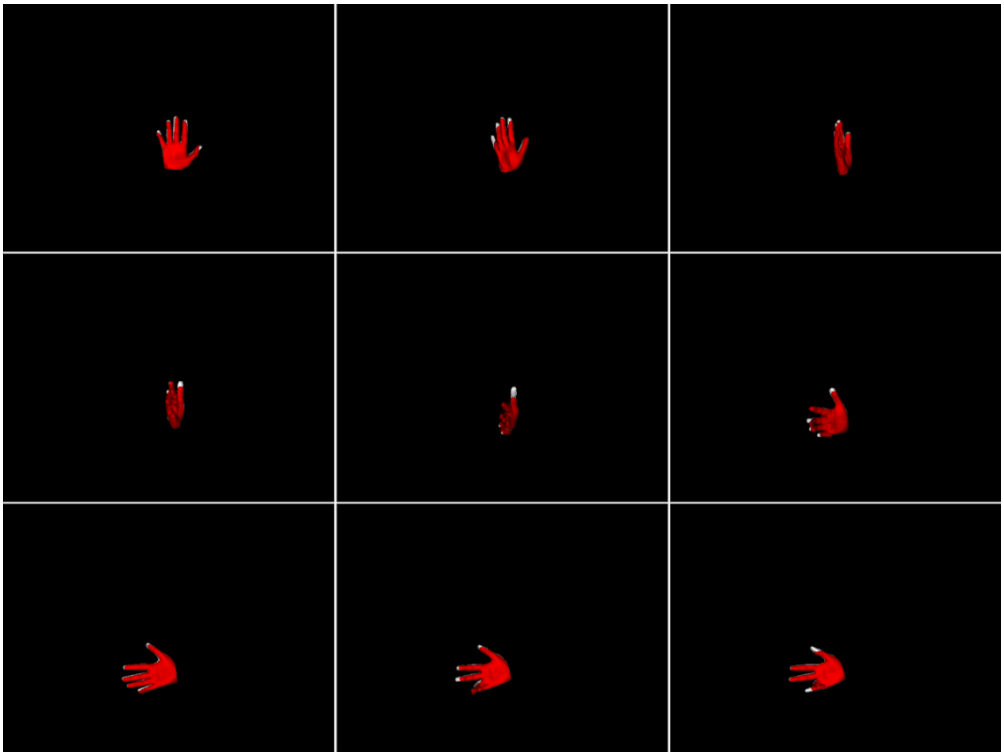

Figure 3.155: Default objective function images.

3.2.2.2.6.2. MS objective function

The MS objective function is used.

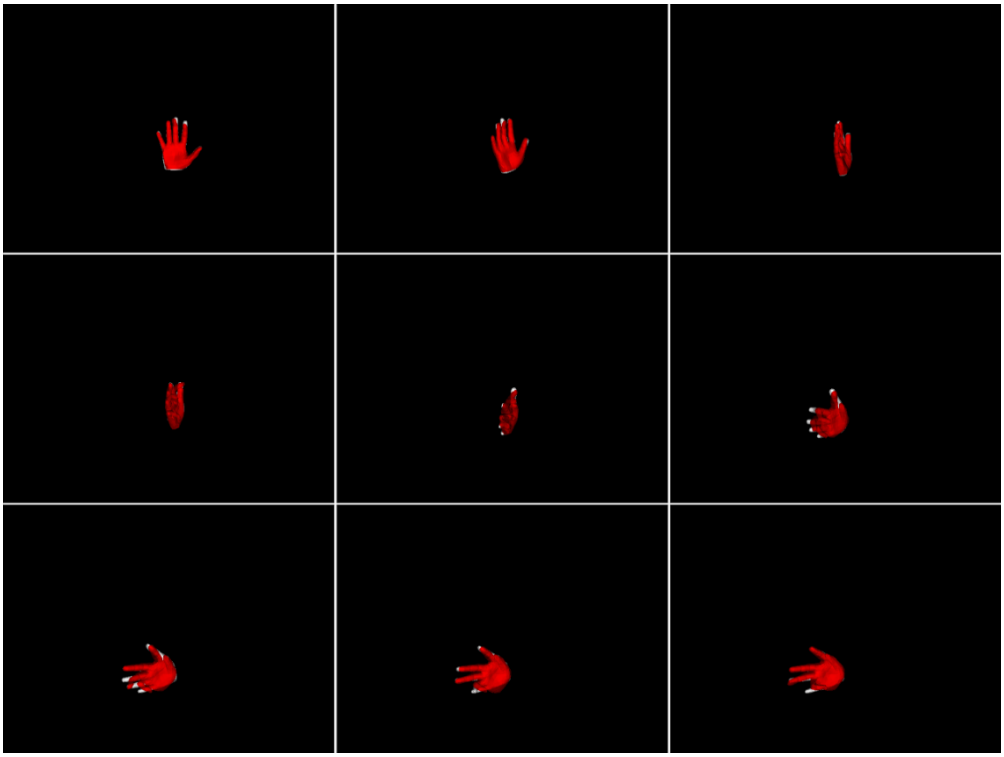

Figure 3.156: MS objective function images.

3.2.2.2.6.3. MSPL objective function

The MSPL objective function is used.

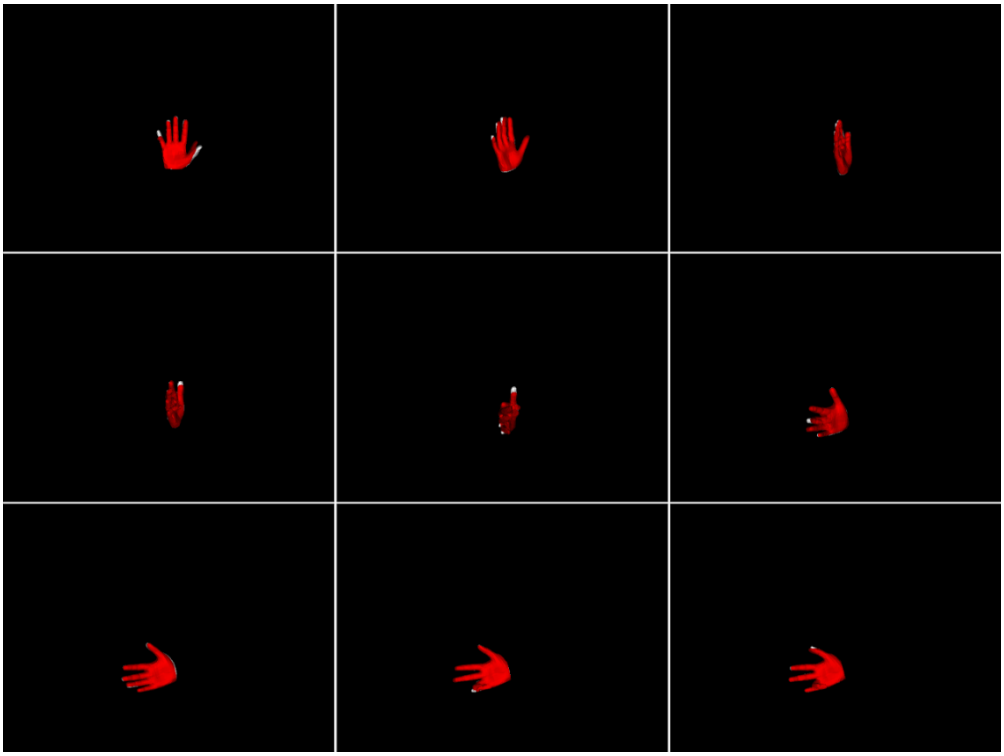

Figure 3.157: MSPL objective function images.

3.2.2.2.6.4. Objective function MCL

The MCL objective function is used.

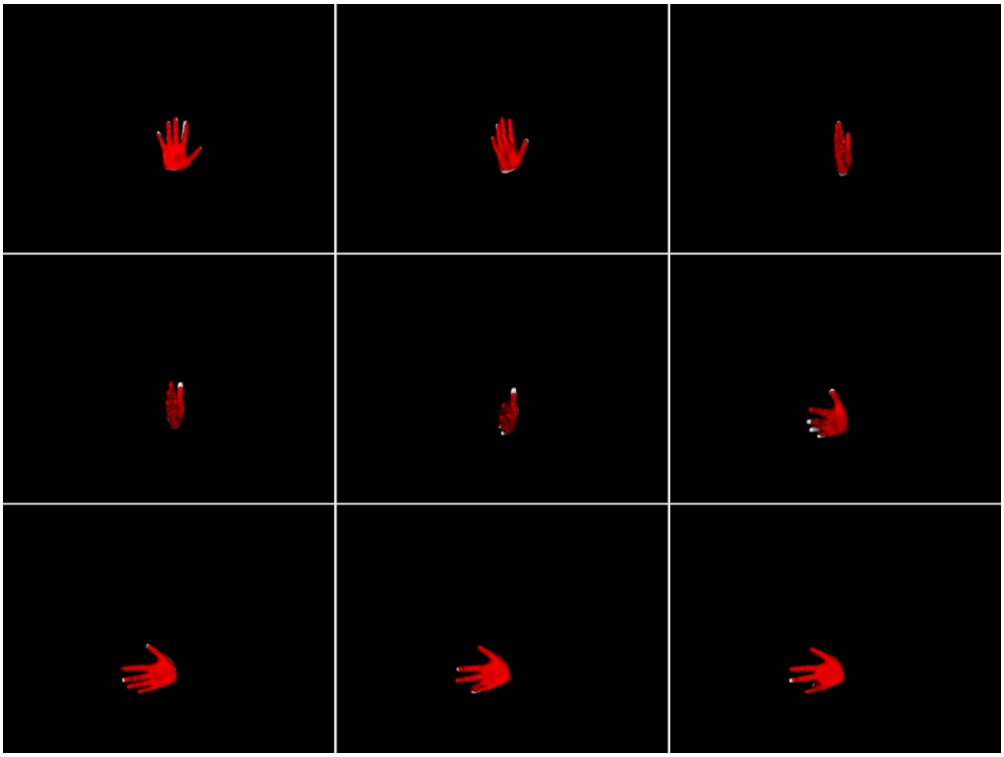

Figure 3.158: MCL objective function images.

3.2.2.3. Quantitative results and conclusions of the simulated sequence tracking experiments

The simulated sequence tracking experiments gave us useful knowledge on the efficiency of the four objective functions. The two tables below show the average error in millimeters of each objective function for 128 and 256 particles.

128 pcls	**Case 1**	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**
Default	37,45	25,17	34,49	32,70	44,71	42,51
MS	52,15	35,58	38,51	54,08	55,98	65,82
MSPL	**40,15**	**23,86**	35,49	44,37	**45,72**	**57,63**
MCL	45,29	27,43	**32,05**	**39,49**	47,63	58,81

Table 7: Simulated sequence tracking average errors [mm] for 128 particles.

256 pcls	**Case 3**	**Case 4**	**Case 5**	**Case 6**
Default	24,88	31,50	38,28	38,57
MS	35,37	53,02	52,94	71,73
MSPL	**25,43**	38,34	**36,28**	59,80
MCL	27,20	**36,85**	43,46	**46,19**

Table 8: Simulated sequence tracking average errors [mm] for 256 particles.

- The results show that the MS objective function where there is no lighting has the highest average errors. That means that the lighting information is a useful element and reduces the average error when its used (MSPL, MCL).
- However, the default objective function which uses the depth information has the lowest average errors. A more realistic lighting model maybe could reduce the average error of the lighting objective functions (MSPL and MCL) even more.
- The use of 256 particles didn't improve the efficiency in every objective function.
- In the tested sequences, the moves included many translations (X-Y axis), rotations (X-Y-Z axis) and bending of the observation's fingers. The experiments showed that the when the lighting information is used the tracking is more accurate and robust.

3.2.3. Real video tracking

The experiments in this section are made using real observations acquired from the Kinect sensor. The three objective functions (MS, MSPL, MCL) are tested in several different instances. Each objective function is tested several times in an instance. The default objective function is tested too. The instances are described below and some tracking images for each objective function are displayed.

3.2.3.1. Video 1

The first video shows a man moving his hand to the left from the camera perspective and at the same time bending his fingers to the inside to form a fist. Then he starts to straighten his fingers while he is moving his hand back to its initial position.

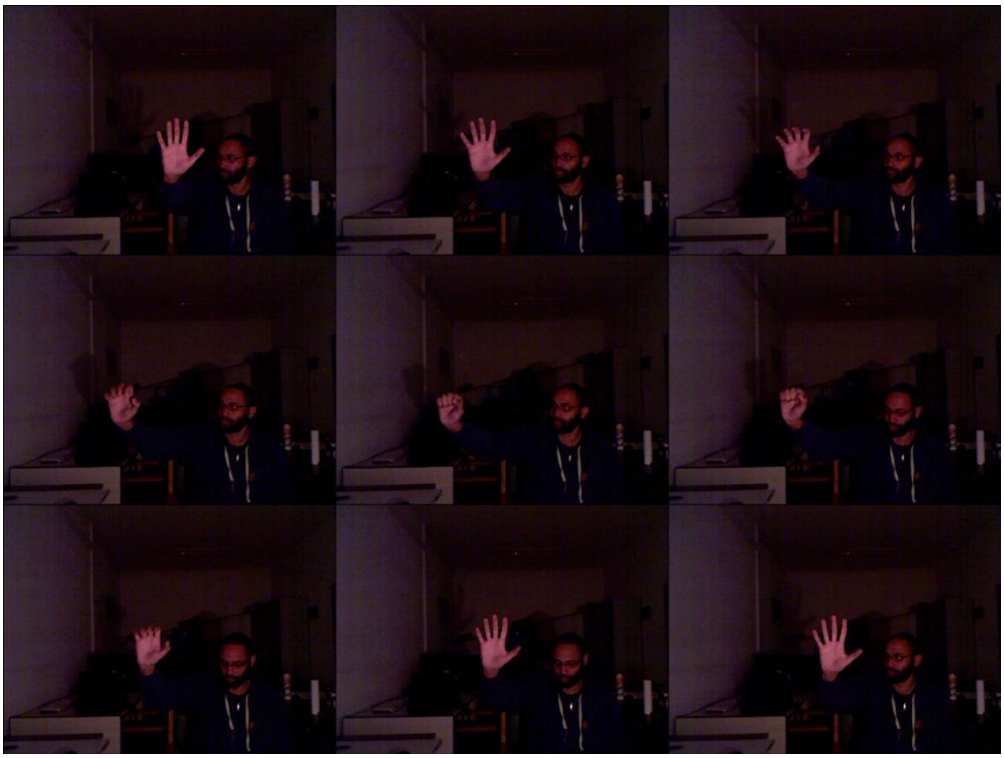

Figure 3.159: Video 1 RGB images.

3.2.3.1.1. Default objective function

The default objective function is used.

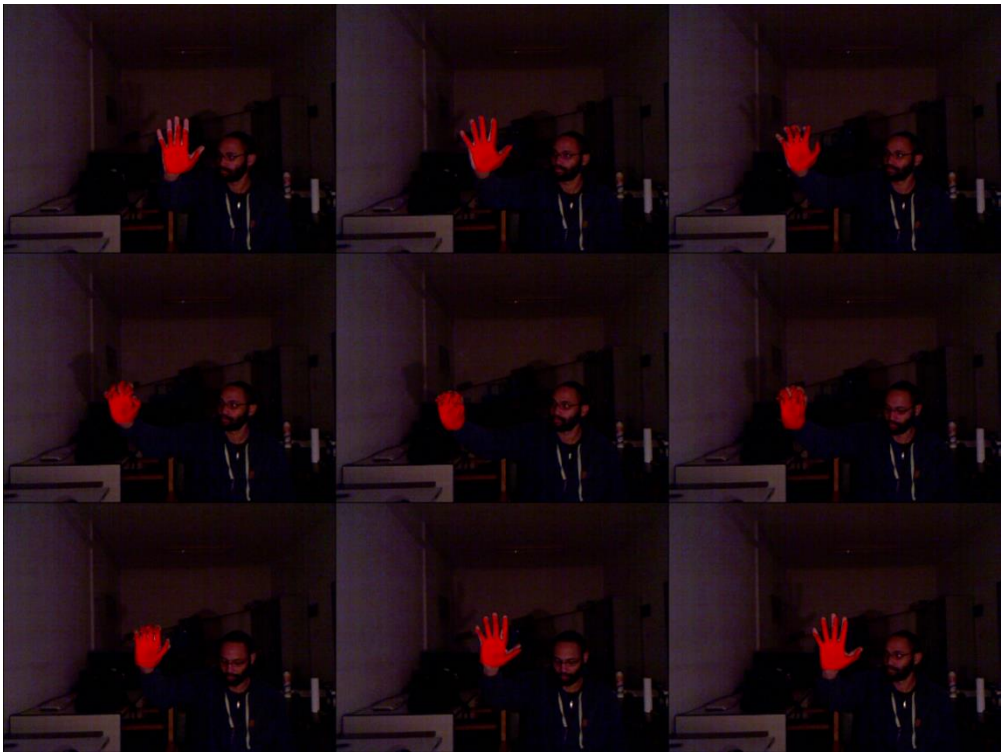

Figure 3.160: Default objective function images.

3.2.3.1.2. MS objective function

The MS objective function is used.

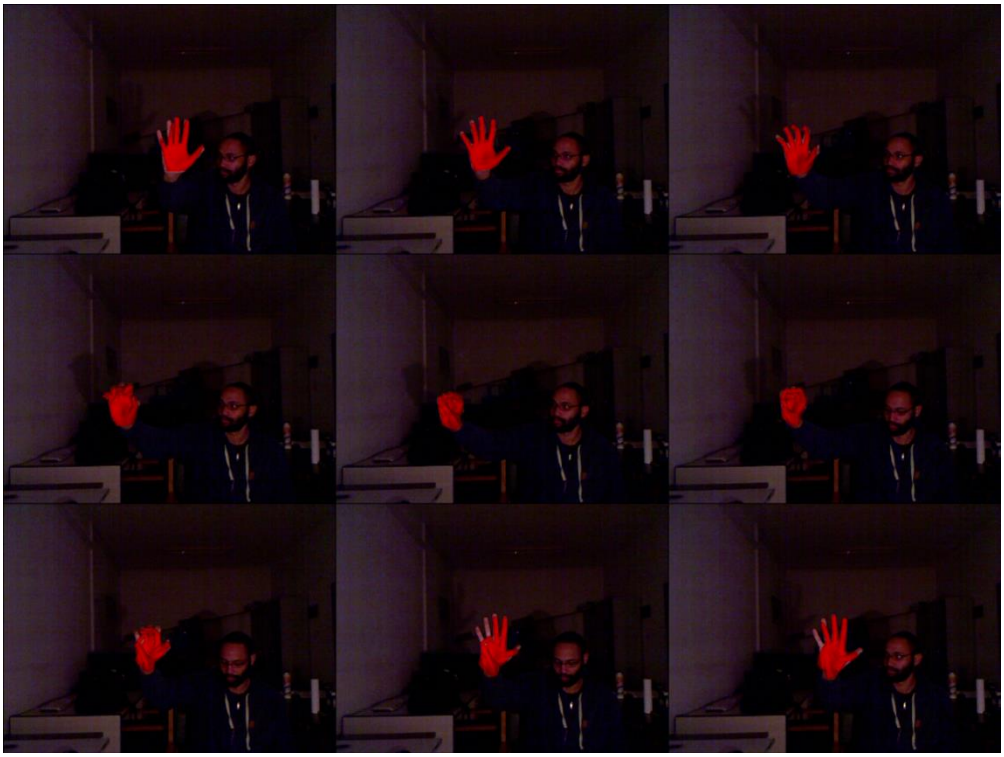

Figure 3.161: MS objective function images.

3.2.3.1.3. MSPL objective function

The MSPL objective function is used.

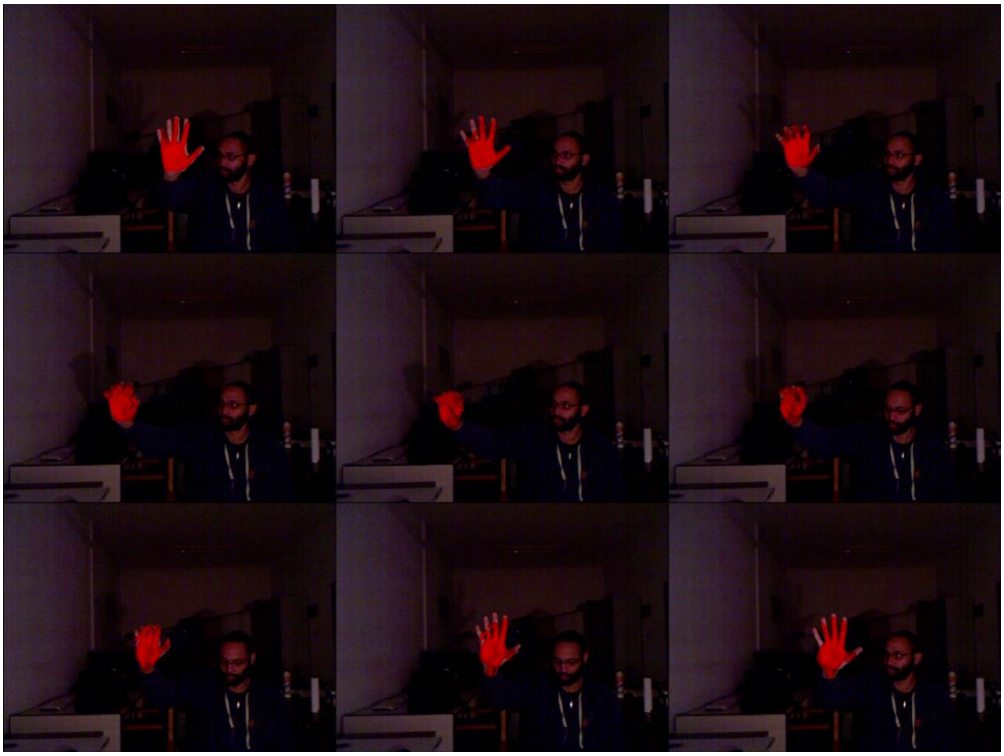

Figure 3.162: MSPL objective function images.

3.2.3.1.4. MCL objective function

The MCL objective is used.

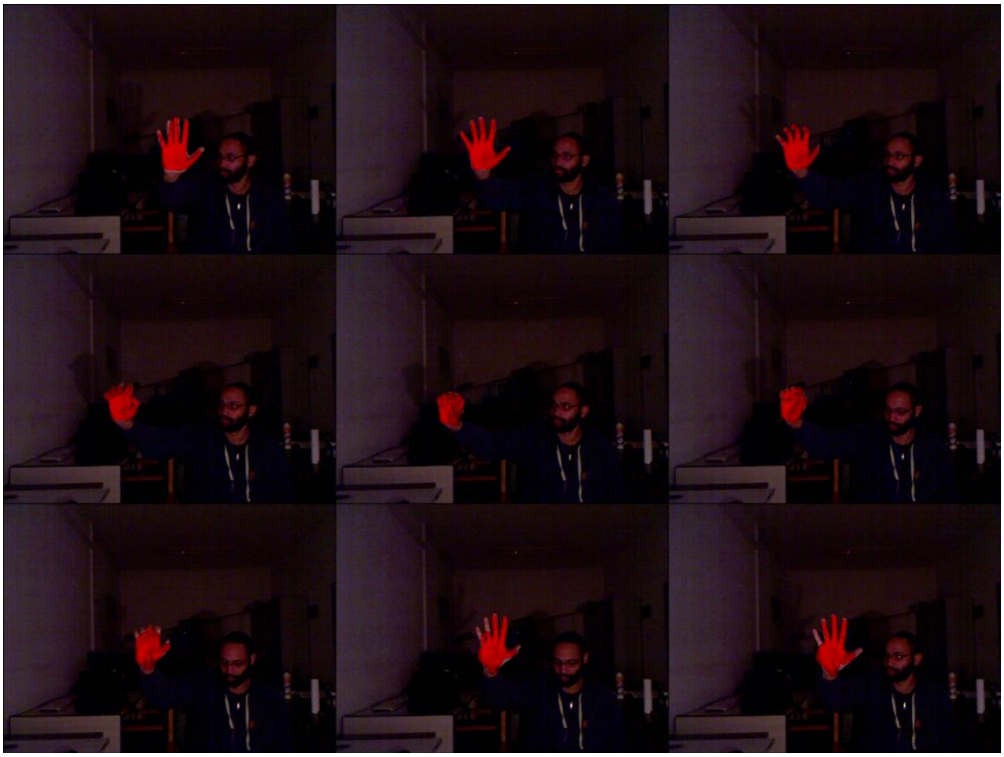

Figure 3.163: MCL objective function images.

3.2.3.2. Video 2

The second instance shows a man rotating his hand toward the camera and down while the fingers are bending forming a “thumb in the middle” gesture. Then the thumb is rotated upwards and “in the middle” again. At the end, he is straightening his fingers and rotates his hand to its initial state.

Figure 3.164: Video 2 RGB images.

3.2.3.2.1. Default objective function

The default objective function is used.

Figure 3.165: Default objective function images.

3.2.3.2.2. MS objective function

The MS objective function is used.

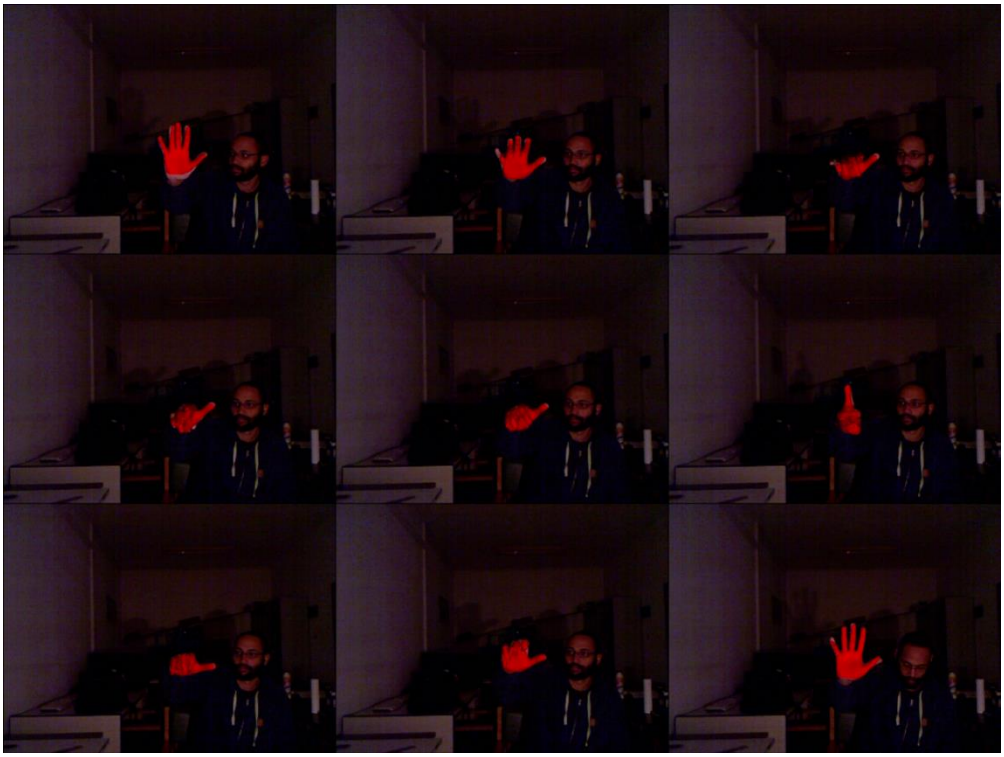

Figure 3.166: MS objective function images.

3.2.3.2.3. MSPL objective function

The MSPL objective function is used.

Figure 3.167: MSPL objective function images.

3.2.3.2.4. MCL objective function

The MCL objective function is used.

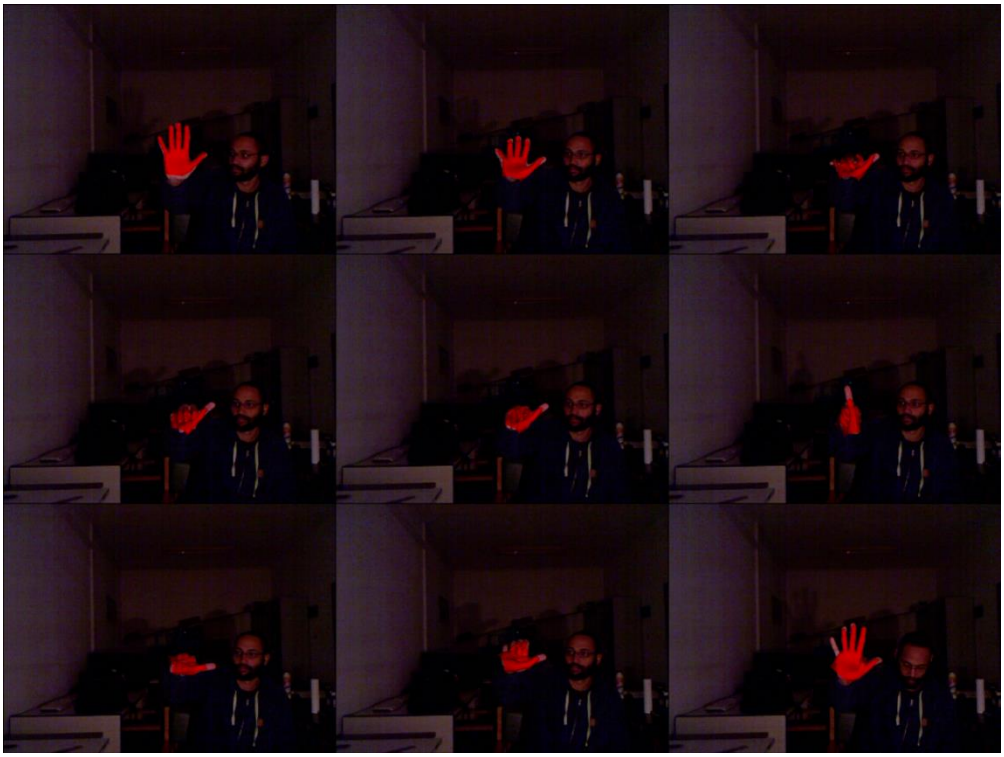

Figure 3.168: MCL objective function images.

3.2.3.3. Video 3

The third instance shows a man bending his fingers to the inside and then gradually straightening them one by one.

Figure 3.169: Video 3 RGB images.

3.2.3.3.1. Default objective function

The default objective function is used.

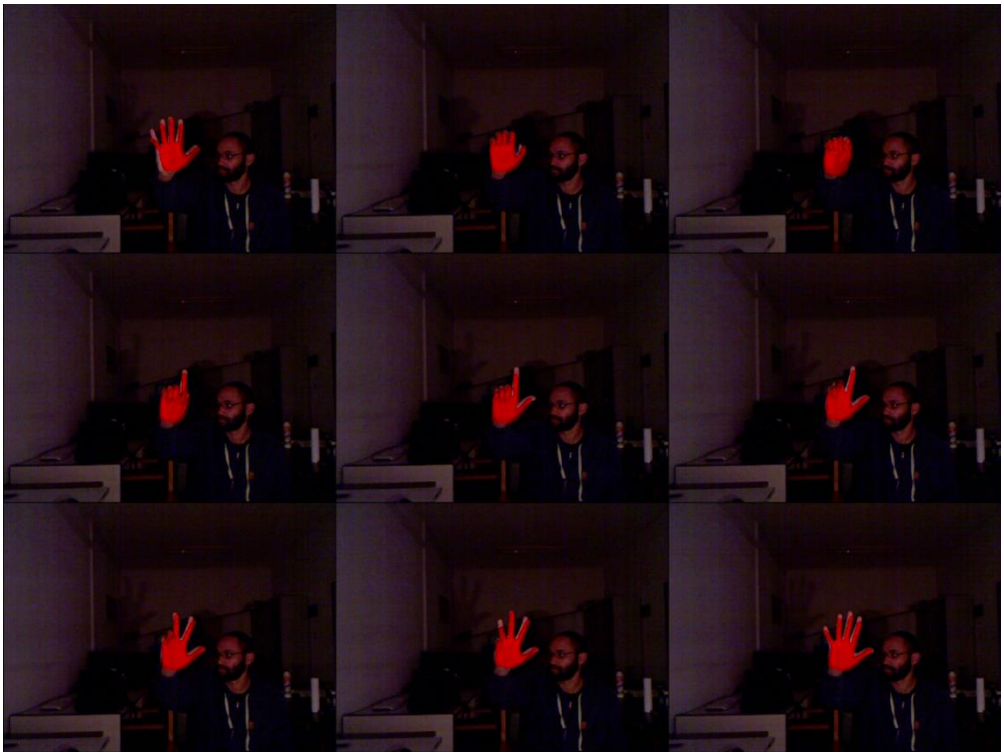

Figure 3.170: Default objective function images.

3.2.3.3.2. MS objective function

The MS objective function is used.

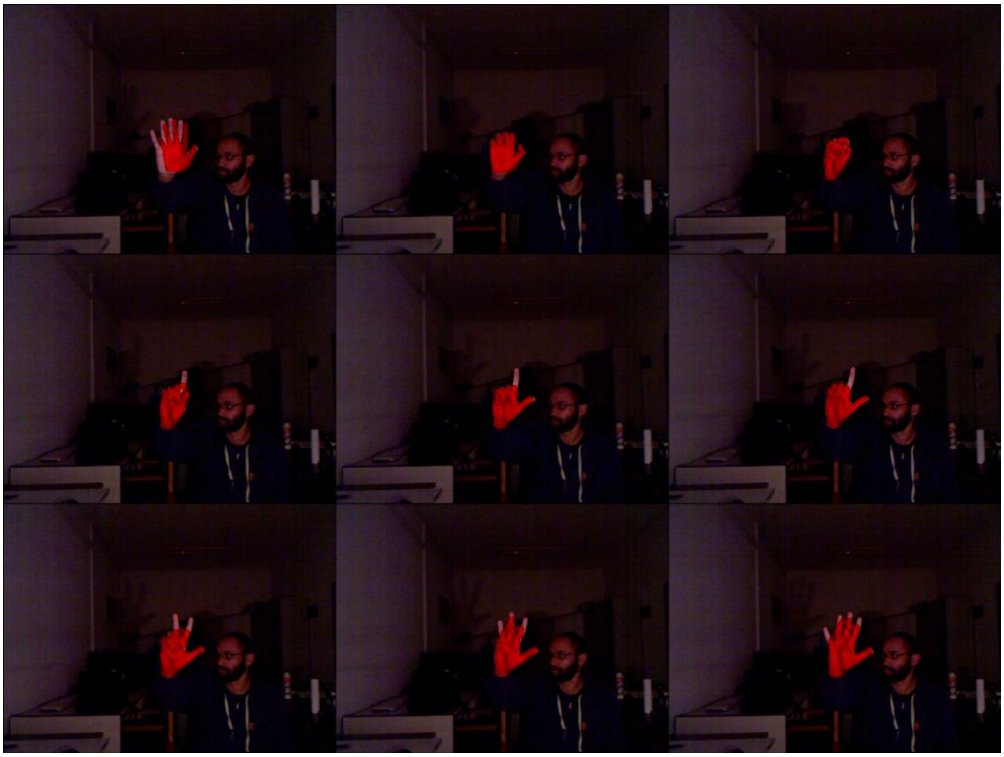

Figure 3.171: MS objective function images.

3.2.3.3.3. MSPL objective function

The MSPL objective function is used.

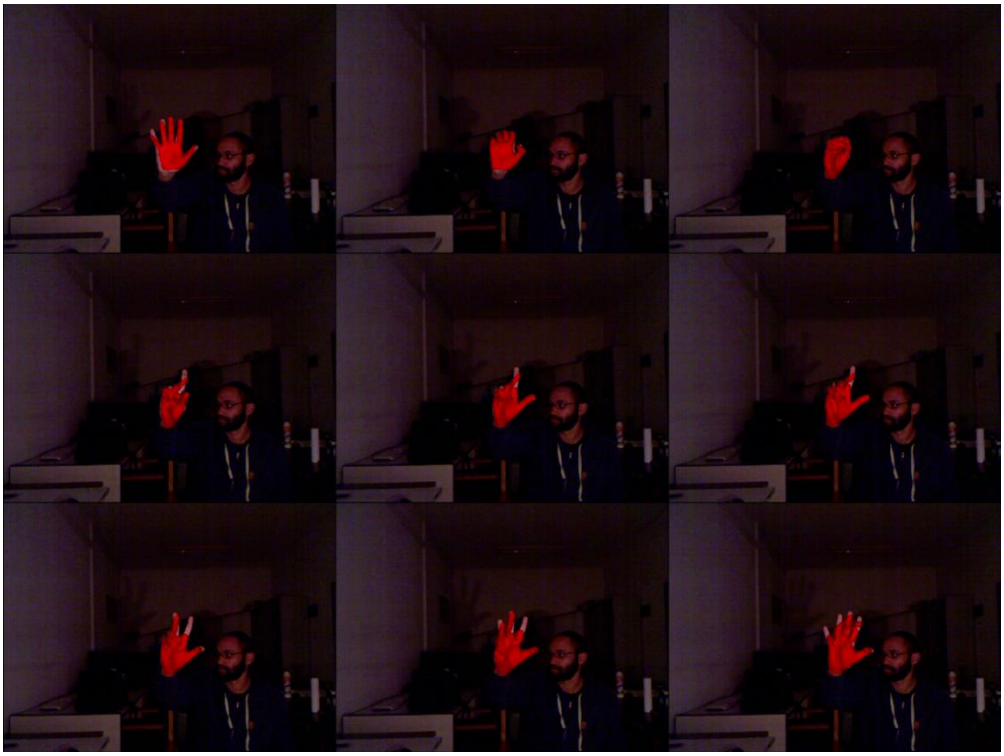

Figure 3.172: MSPL objective function images.

3.2.3.3.4. Objective function MCL

The objective function MCL is used.

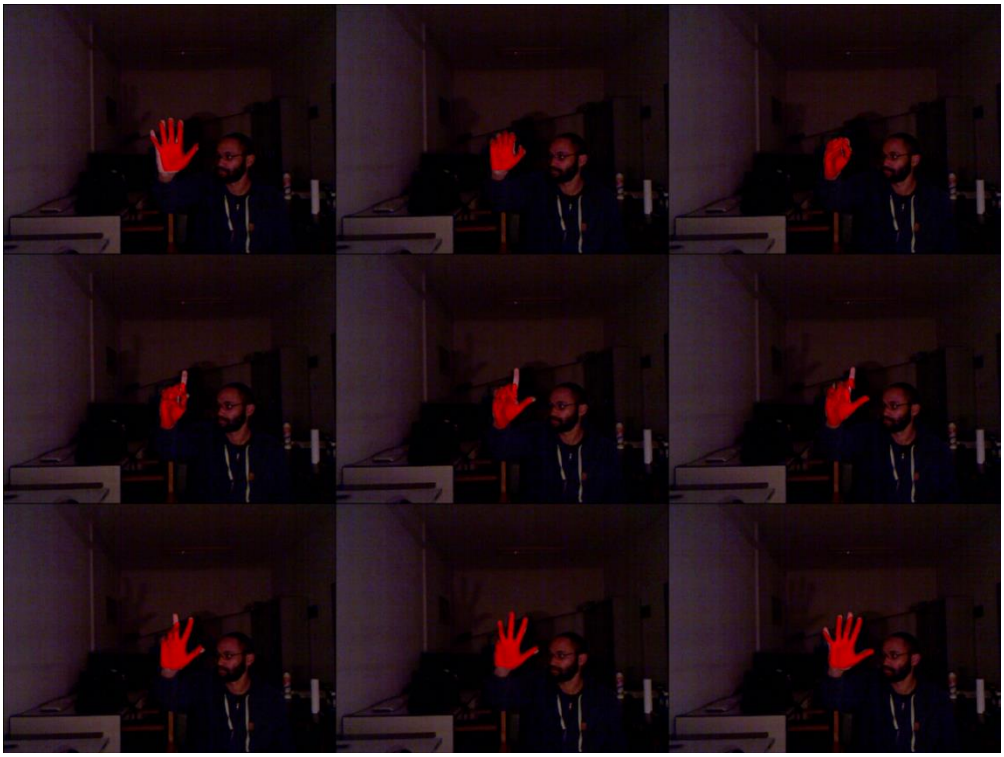

Figure 3.173: MCL objective function images.

3.2.3.4. Video 4

The fourth instance shows a man rotating his hand to the left and then to the right.

Figure 3.174: Video 4 RGB images.

3.2.3.4.1. Default objective function

The default objective function is used.

Figure 3.175: Default objective function images.

3.2.3.4.2. MS objective function

The MS objective function is used.

Figure 3.176: MS objective function images.

3.2.3.4.3. MSPL objective function

The MSPL objective function is used.

Figure 3.177: MSPL objective function images.

3.2.3.4.4. Objective function MCL

The objective function MCL is used.

Figure 3.178: MCL objective function images.

3.2.3.5. Conclusions of the real tracking experiments

The real tracking experiments showed that the default objective function is more robust than our three objective functions but in cases like the Video 2 where the observation's hand rotates toward the camera, due to the loss of depth information the hand model fails to follow. In general, when one of the three objective functions (MS, MSPL, MCL) is used the hand model follows the observation's hand but the matching isn't always very accurate. There is no obvious difference in their efficiency that can be seen in the displayed images to let us understand if using the lighting information in real tracking case is useful. The main reasons are the fact that the lighting conditions as well as the foreground between the observation and the hypotheses are very different.

Chapter 4 – Conclusions and future work

4.1. Conclusions

The experiments on the three objective functions (MS, MSPL, MCL) showed that the lighting element in a scene is useful and an objective function that uses the lighting information to calculate the discrepancy value V(O,H) improves the tracking of the hand compared to an objective function that doesn't include the lighting information.

The default objective function which uses the depth information to extract the foreground and calculate the discrepancy value V(O,H) found to have the best efficiency and its more stable than the other objective functions. However, there are cases where due to depth ambiguities the pose failed to follow the observation. So, in cases like these, where the fingers are bending or the observation rotates forward towards the camera, the tracking can be improved by using the lighting information instead of the depth information.

The above conclusions came from the results of the simulated sequence tracking experiments. The real tracking experiments didn't give any solid results because the illumination of the scene in the RGB images (of the Kinect sensor) is very different and much more complex from the illumination of the scene provided by our lighting model. Our lighting model is based on the proportions of the ambient reflection, the diffuse reflection, and the specular reflection. But the illumination of a scene is much more complex. There are factors like the amount of light that is absorbed or reflected from the walls in the environment, the amount of light that is absorbed by the hand or the rate at which the light intensity attenuates proportional to the distance from the light source. Our lighting model includes many of these variables but their calculation is very difficult so their values are set in a way that doesn't affect the result. Although, one of the most important factors is the shades that may occur while the observation is moving which they cannot be computed and adapted to the hypotheses with our algorithm. Another factor is that the foreground of the observation does always have the same shape with the foreground of the hypotheses and that adds an error to the results.

As an overall conclusion, we can say that the simulated sequence tracking experiments showed that our lighting model is efficient when the lighting conditions between the observation and the hypotheses are similar even if it's not entirely realistic. Its further enrichment could make it more realistic and radically increase the chances for us to see positive results in the real tracking experiments too.

4.2. Future work

The lighting model that was developed and tested in this thesis has much room for improvement since there are many variables in which someone can focus on:

- A variable that may have a significant impact to the efficiency of our lighting model and the lighting objective functions is the shade variable. There are points of the observation that during its movement are shaded. These shades are formed when the fingers are bended or the hand is rotated. In the figure below some these points can be seen. Our lighting model is not able to determine these points. So, during the tracking the corresponding shaded areas of the hypotheses may appear brighter than they should be and this is a significant error. An algorithm capable to identify the shades of the hypotheses (poses) would make our lighting model more realistic, improve its efficiency and we might be able to see positive results not only in the simulated sequence tracking experiments but also in the real tracking experiments.

Figure 4.1: Shaded areas.

- Further algorithm improvements could be related to the foreground extraction method of the observation to eliminate the points of the hand that they are not needed and add some error to the discrepancy value V(O,H). Examples of these points can be seen in the images below. Those points should be minimized without losing any important information from the rest of the hand. This way the observation would marginally match the hypotheses better since their shape would be more alike.

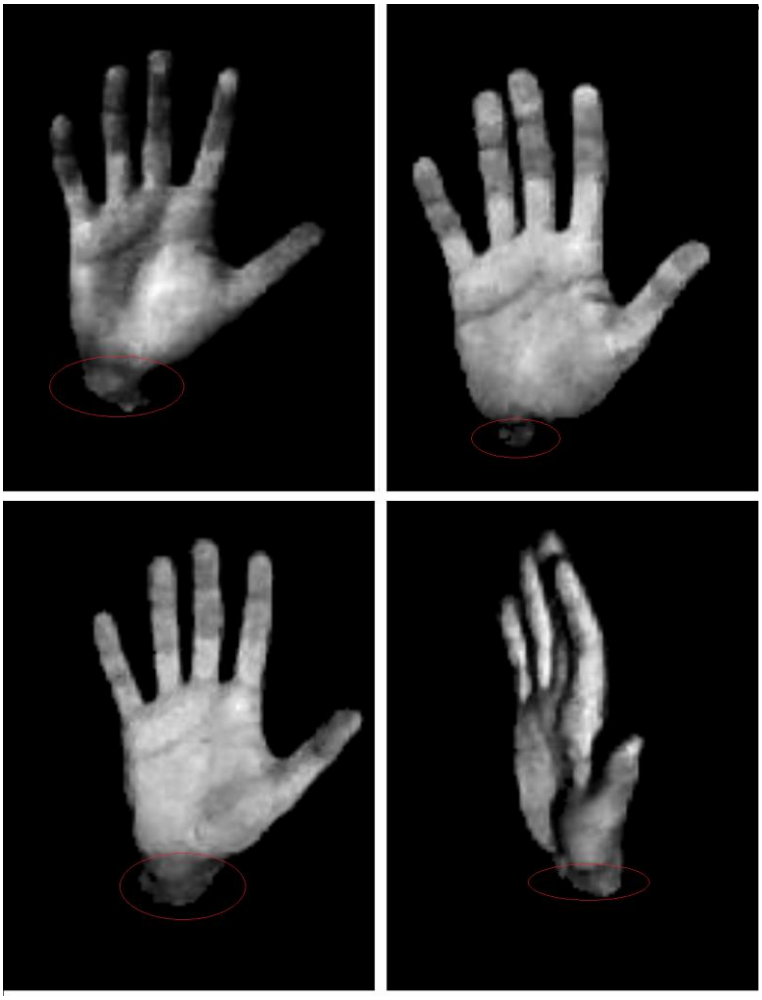

Figure 4.2: Foreground extraction fail points.

- The computation of variables like the amount of light that is absorbed or reflected from the hand depended on the skin color or the rate at which the light intensity attenuates proportional to the distance from the light source could make the lighting model even more realistic and complex at the same time.

References

1. Video tracking. (2017, February 11). In *Wikipedia, The Free Encyclopedia*. Retrieved 15:15, April 21, 2017, from https://en.wikipedia.org/wiki/Video_tracking.
2. Kinect. (2017, April 13). In *Wikipedia, The Free Encyclopedia*. Retrieved 16:31, April 21, 2017, from https://en.wikipedia.org/wiki/Kinect.
3. S. Crawford. How Microsoft Kinect Works. Retrieved from http://electronics.howstuffworks.com/microsoft-kinect.htm.
4. Skeletal Tracking. Retrieved from https://msdn.microsoft.com/en-us/library/hh973074.aspx.
5. FORTH – Model Based Hand Tracker https://github.com/FORTH-ModelBasedTracker/HandTracker.
6. I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. pages 101.1–101.11. British Machine Vision Association, 2011.
7. A. Makris, D. Kosmopoulos, S. Perantonis, and S. Theodoridis. A hierarchical feature fusion framework for adaptive visual tracking. Image and Vision Computing, 29(9): 594 – 606, Aug. 2011.
8. A. Makris, N. Kyriazis, and A. Argyros. Hierarchical Particle Filtering for 3D Hand Tracking. In 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Page(s):8 – 17, 2015.
9. E. Angel, D. Shreiner (2012). Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL (6th Edition).
10. Κ. Μουστάκας, Ι. Παλιόκας, Α. Τσακίρης, Δ. Τζοβάρας (2015). Γραφικά και Εικονική Πραγματικότητα. Retrieved 17:14, May 2, 2017, from https://repository.kallipos.gr/bitstream/11419/4491/1/00_master_document.pdf.
11. M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing, 50(2):174–188, Feb. 2002.
12. Particle filter. (2017, April 13). In *Wikipedia, The Free Encyclopedia*. Retrieved 09:12, May 5, 2017, from https://en.wikipedia.org/wiki/Particle_filter.
13. Bayesian inference. (2017, April 14). In *Wikipedia, The Free Encyclopedia*. Retrieved 09:12, May 5, 2017, from https://en.wikipedia.org/wiki/Bayesian_inference.
14. Recursive Bayesian estimation. (2017, March 20). In *Wikipedia, The Free Encyclopedia*. Retrieved 09:12, May 5, 2017, from https://en.wikipedia.org/wiki/Recursive_Bayesian_estimation.
15. Probability density function. (2017, March 22). In *Wikipedia, The Free Encyclopedia*. Retrieved 09:12, May 5, 2017, from https://en.wikipedia.org/wiki/Probability_density_function.
16. K. Hsiao, H. de Plinval – Salgues, J. Miller. (2005) Particle Filters and Their Applications. Retrieved May 05, 2017 from

https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/a5_hso_plnvl_mlr.pdf.
17. B. Minor (2011) Particle Filters. Retrieved May 05, 2017 from http://www.eecs.wsu.edu/~cook/ml/presentations/bryan.pdf.
18. D. G. Bleser, Prof. Stricker. Computer Vision: Object and People Tracking. Retrieved May 05, 2017 from https://ags.cs.uni-kl.de/fileadmin/inf_ags/opt-ss12/lec08_opt.pdf.
19. Kavan L (2014) Part I: Direct skinning methods and deformation primitives. In: ACM SIGGRAPH 2014 course - Skinning: Real-time shape deformation, pp 1–11.
20. I. Oikonomidis, N. Kyriazis, and A. Argyros. Tracking thearticulated motion of two strongly interacting hands. In 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1862–1869, June 2012.
21. A. Tagliasacchi, M. Schroeder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly. Robust articulated-icp for real-time hand tracking. Technical report, 2015.
22. Sridhar, S., Oulasvirta, A., and Theobalt, C. Interactive markerless articulated hand motion tracking using RGB and depth data. In Proc. ICCV (Dec. 2013).
23. T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton,D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi. Accurate, robust, and flexible real-time hand tracking. CHI, April 2015.
24. Georg Poier, Konstantinos Roditakis, Samuel Schulter, Damien Michel, Horst Bischof, and Antonis A Argyros. Hybrid one-shot 3d hand pose estimation by exploiting uncertainties. 2015.
25. Zhou, X., Wan, Q., Zhang, W., Xue, X., Wei, Y.: Model-based deep hand pose estimation. In: IJCAI (2016)
26. R. Wang, S. Paris, and J. Popovic. 6D hands: markerless hand-tracking for computer aided design. In Proc. ACMUIST, pages 549–558. ACM, 2011
27. Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. Cascaded hand pose regression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 824–832, 2015.
28. C. Keskin, F. Kirac, Y. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pages 1228–1234, Nov. 2011.
29. Martin De La Gorce, Nikos Paragios, and David J. Fleet. Model-based Hand Tracking with Texture, Shading and Self-occlusions. CVPR, (June):1–8, June 2008.
30. A. Balan, M. Black, H. Haussecker, and L. Sigal. Shining a light on human pose: On shadows, shading and the estimation of pose and shape. In ICCV, pages 1–8, 2007.

31. C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 1106–1113. IEEE, 2014.