Stony Brook University

Computer Aided Design of Planar Four-Bar Linkages On iOS

Using a Novel SVD Algorithm

A Master Thesis Presented

by

Xin Ge

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Mechanical Engineering

Stony Brook University

May 2013

Stony Brook University

The Graduate School

Xin Ge

We, the thesis committee for the above candidate for the

Master of Science degree, hereby recommend

acceptance of this thesis.

Qiaode Jeffery Ge – Thesis Advisor
Professor
Mechanical Engineering Department

Anurag Purwar – Thesis Co-Advisor
Research Associate Professor
Mechanical Engineering Department

Yu Zhou – Committee Chairperson
Assistant Professor
Mechanical Engineering Department

This thesis is accepted by the Graduate School

Charles Taber
Interim Dean of the Graduate School

Abstract of the Thesis

Computer Aided Design of Planar Four-Bar Linkages On iOS

Using a Novel SVD Algorithm

by

Xin Ge

Master of Science

in

Mechanical Engineering

Stony Brook University

2013

Motion synthesis has been a practical topic existing for a long time. Various methods have been tried and put into practice. It is well done by the scientists and engineers to solve large equations of matrix. Although it can be done and get the good result, the efficiency is low and labor is very much consuming. Comparatively, our methods are much more easier and efficient, it is more suitable for the software to realize the calculation by itself.

It is mature in PCs to have a motion synthesis software, but since the mobile system is going into everyday life and there is no such an app being able to undertake this function. With the latest theory of Fourier and quaternions, it is able to build this app on iPad to process both the simulation and motion synthesis functionalities with high accuracy and speed.

To

My father Maobin Ge and mother Guifeng Gong.

Thank you for all your support and always love me.

Table of Contents

List of Figures

Accknowledgements

The motion synthesis theories are based on the papers of Qiaode Jeffery Ge, Ping Zhao, Anurag Purwar. The simulation functionality uses Fourier Method which is based on the theory by Xiangyn Li and Qiaode Jeffrey Ge.

Thanks to Shoulin Sun from Dalian University of Technology for the advices to the former works.

Thanks to the financial support from Research Foundation of State University of New York at Stony Brook.

This work was supported by a TALENT grant awarded to Prof. Anurag Purwar (PI) and Prof. Jeff Ge (co-PI) by the Teaching, Learning, and Technology Center at Stony Brook University, NY.

Chapter 1

Introduction and Background

1.1 Background on motion synthesis

Planar linkages have been studied for hundreds of years for it is the most common and basic mechanical system. The earliest research on the motion synthesis was done by Burmester, who developed an approach to deal with at most five position synthesis. He showed that one could get infinite solutions with four positions or less and six, one or no solution with five positions. Ravani and Roth [1, 2] provided the relative approach on the manifold fitting problem and shows the problem can be solved by fitting a pencil of quadrics. Recent works can be found in Ping Zhao, Anurag Purwar, Q.J.Ge [3] who introduce quaternion into motion synthesis calculation to save computational time and simplify calculating steps and Xiangyun, Li, Ping Zhao and Q.J.Ge [4] who use Fourier method to solve path problems. More modern research on the motion synthesis can be referenced in texts of McCarthy [5].

1.2 Background on the mobile system CAD software

Along with the popularity of the mobile devices, people start to use their personal phones or tablets to deal with more daily activities. The mobile devices have

more advantages over the PC in many areas than people usually think. Generally mobile devices are much cheaper than a PC and the performance is not that much lost. It expects the tablet market will grow by 37 percent on average each year between 2012 and 2016, with volumes reaching 389 million units, accounting for 59 percent of total PC shipments [6]. One of the reason it is so popular is the portability of such kinds of small and middle devices.

People download softwares (which is usually called apps on the mobile systems) from the online app stores by separate OS. Normally an app store contains thousands and even millions of different apps online. But among them, most are games and daily useful productivity apps, few are the professional apps facing to the technical persons. We can see some of the engineering apps on the mobile systems. In june 2011, ASME.org recommended 5 mobile apps for the engineers [7]. The five apps are Mechanical Engineer, AutoCAD WS, Cross Section Cal, Mech Ref, Mechanics Basics. Four of them are reference apps and only one is for the real design.

On Fabruary 2013, ASME.org posted another page with the name of "The Engineering Design Apps: A Work in Progress" [8]. It said:" While there are many mobile apps useful to engineers in the areas of calculation tools, formulas, libraries, viewers, and collaboration, there is paucity when it comes to design." The article introduced ForceEffect, which the software developer Gianluca Natalini introduced FingerCAD and Finger3D in 2010, he said they were the only CAD apps on the iTunes store. It allows users to use their fingers to fulfill the functionalities of the CAD softwares almost as the same as it is on the PCs for the first time. Recently the

ForceEffect joined with ForceEffect Motion, the latter of which can also do the motion simulation of a multi-linkage mechanism drawn by the users.

It is obvious that there is still a large market and blank waiting to be filled on the mobile apps for the engineers. With such a need in design, we have developed the technical app to realize not only the functionality of simulation but also synthesis.

1.3 iOS introduction

iOS (previously iPhone OS) is the mobile operating system developed by Apple Inc. It is used as the operating system in iPhone, iPod touch, iPad and Apple TV.

According to data published by IDC and cited by AppleInsider [9], Apple's global share of the smartphone market grew from 18.8 percent to 25.1 percent during 2012. iPad had 43.6% of global tablet market share. This makes it the biggest market in worldwide. According to TechCrunch, iOS aquired almost 80% of the enterprise market share compared with Android in 2011 and 2012. From the statistics, iOS gains a large amount of users among the world.

iOS is developed, managed and distributed only by Apple Inc. and the apps can only be downloaded through app store, which make it the most secure operating system in the world. The most distinguished feature of iOS is the multi-touch functionality with high smoothness, sensitivity, accuracy and speed.

Objective-C is the only core programming language for developing iOS apps, although many other languages, C++ and C mostly, are also accepted within the code.

According to the TIOBE Programming Community Index for April 2013 [10], Objective-C is the fourth mostly used programming language worldwide which is 19.598%, next only to C (17.862%), Java (17.681%) and C++ (9.714%). It is the language ascends fastest for the past three years.

Objective-C was created in the 1980s and is inherited from the C language. It adds to the traditional C some objective oriented structures. Objective-C syntax is much easier to read than other programming language. It is arranged in a way that you can understand it when you look at it, just like look at any human language.

For example, the following line from Objective-C compares whether the contents of a variable called someName is equal to David:

```
[someName isEqualToString:@"David"]
```

It doesn’t take much energy to understand what is to be achieved in the code snippet. In traditional C, this could be written as follows:

```
strcmp (someName,"David")
```

We can see that even though the C statement is a bit shorter, it doesn't declare much what the code is actually doing.

There are many advantages using Objective-C language, except for understandability and efficiency. More can be found in "Sams Teach Yourself iOS 6

Application Development in 24 Hours(4th Edition)" by John Ray [11].

1.4 The objective of this app

This app was originally faced to college students as an educational software. They could use it in class or off class, both for presentation and textbook supplement. It can also extended into research assisting tool while users may use it to test their ideas expediently, verify certain theories or just try to get some inspirations from it. Based on the applicability, the app may have a helpful role in the industry fields. It will help engineers visually understand and correct some design work benefited from our latest motion synthesis theories. Even for the common users, they can still gain much knowledge from our well-classified four bar linkages user interface.

Chapter 2

Theoretical Foundation

2.1 Introduction on the simulation

Four-bar linkage is the easiest and most widely used planar linkage mechanism. Other multi-bar linkages can all be treated as constructed by several four-bar linkages. A four-bar linkage of which all the kinematic joints are revolute joints is called a revolute four-bar mechanism, about which our app only concerns.

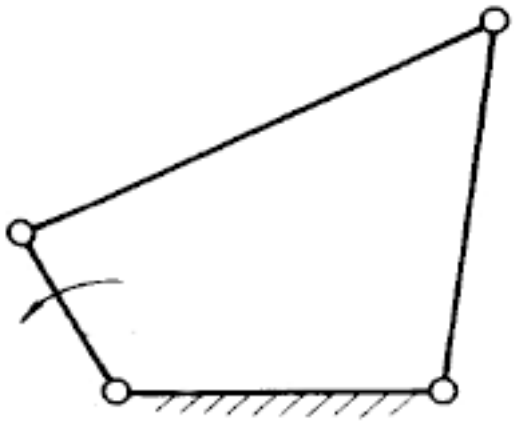


Figure 1.1: A revolute four-bar mechanism

A four-bar linkage met the condition:

$$l_{min} + l_{max} \leq l' + l''$$

is called Grashof linkage. The sum of the lengths of the shortest and longest link must

be less (at most equal) than the sum of the lengths of the other two links. Another is non-Grashof linkage.

Followed by the classification from SoftIntegration Inc [12], the two categories can be classified as such:

Grashof linkage:

For linkages of this type continuous relative motion between the shortest link and its adjacent links is possible.

Crank-Rocker:

This type of Grashof linkage is obtained when the shortest link is the input link (r2). The input link has full motion. The output link has a limited range of motion that is defined as follows:

lower limit: $\theta_4 = 180° + \theta_1 - \theta_4^{''}$

upper limit: $\theta_4 = 180° + \theta_1 - \theta_4^{'}$

where $\theta_4^{'} = a\cos[\frac{r_1^2 + r_4^2 - (r_3 - r_2)^2}{2r_1 r_4}]$

and

$\theta_4^{''} = a\cos[\frac{r_1^2 + r_4^2 - (r_3 + r_2)^2}{2r_1 r_4}]$

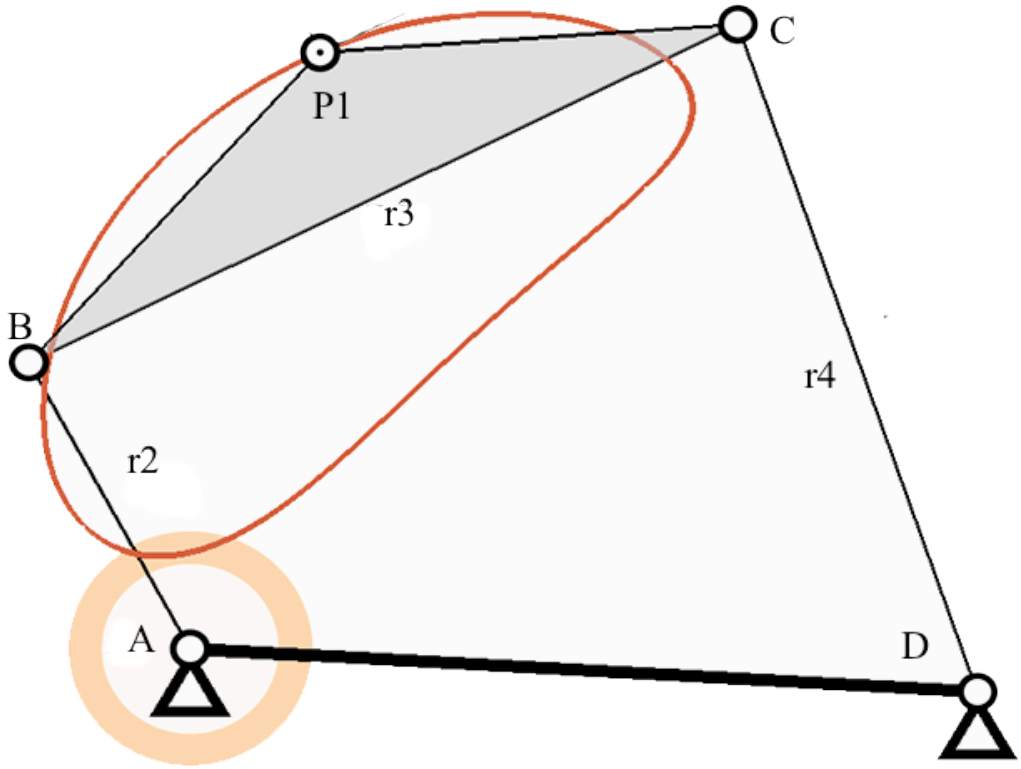


Figure 2.2: Crank-rocker linkage

Double-Rocker:

This type of Grashof linkage is obtained when the shortest link is the floating link (r3). Note that the complete relative motion between the shortest link and its adjacent links are still possible. Both the input and output links have limited ranges of motion that are defined as follows:

Input Link

lower limit: $\theta_2 = \theta_1 + \theta_2^{'}$

upper limit: $\theta_2 = \theta_1 + \theta_2^{''}$

where $\theta_2^{'} = a\cos[\frac{r_1^2 + r_2^2 - (r_3 - r_4)^2}{2r_1 r_2}]$

and

$$\theta_2^{''} = a\cos[\frac{r_1^2 + r_2^2 - (r_3 + r_4)^2}{2r_1r_2}]$$

Output Link

lower limit: $\theta_4 = 180° + \theta_1 - \theta_4^{''}$

upper limit: $\theta_4 = 180° + \theta_1 - \theta_4^{''}$

where $\theta_4^{''} = a\cos[\frac{r_1^2 + r_4^2 - (r_3 - r_2)^2}{2r_1r_4}]$

and

$$\theta_4^{''} = a\cos[\frac{r_1^2 + r_4^2 - (r_3 + r_2)^2}{2r_1r_4}]$$

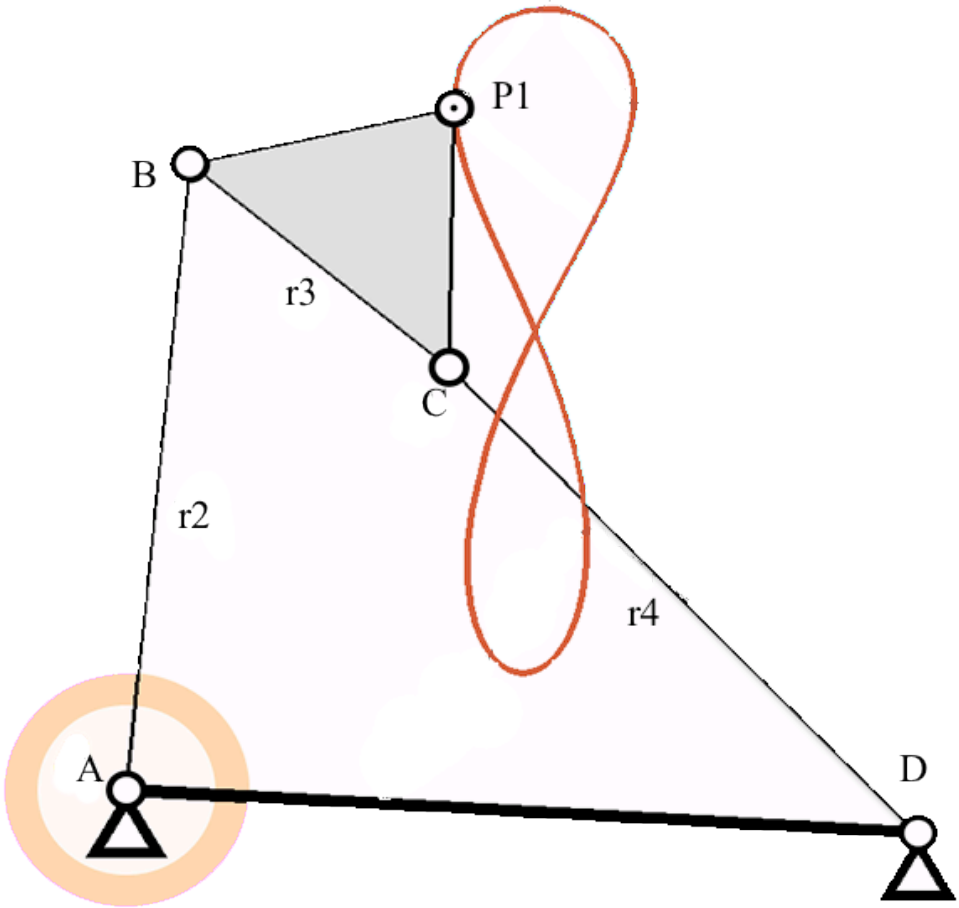


Figure 2.3: Double-rocker linkage

Rocker-Crank:

This type of Grashof linkage is obtained when the shortest link is the output link (r4). The output link has full motion. The input link has a limited range of motion that is defined as follows:

lower limit: $\theta_2 = \theta_1 + \theta_2^{''}$

upper limit: $\theta_2 = \theta_1 + \theta_2^{''}$

where $\theta_2^{'} = a\cos[\frac{r_1^2 + r_2^2 - (r_3 - r_4)^2}{2r_1r_2}]$

and

$\theta_2^{'} = a\cos[\frac{r_1^2 + r_2^2 - (r_3 + r_4)^2}{2r_1r_2}]$

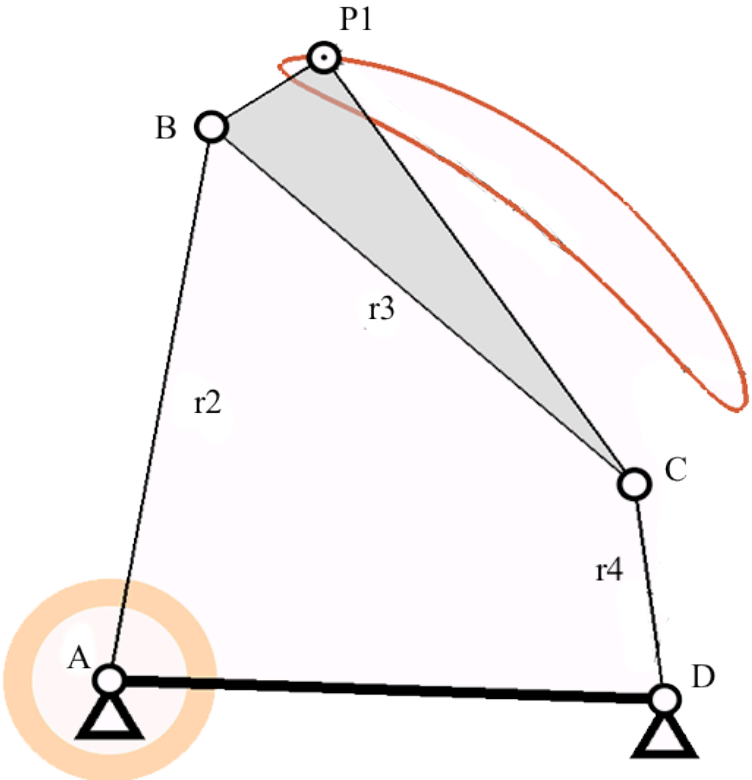


Figure 2.4: Rocker-crank linkage

Double-Crank:

This type of Grashof linkage is obtained when the shortest link is the ground link (r1). Both the input and output links have full motion.

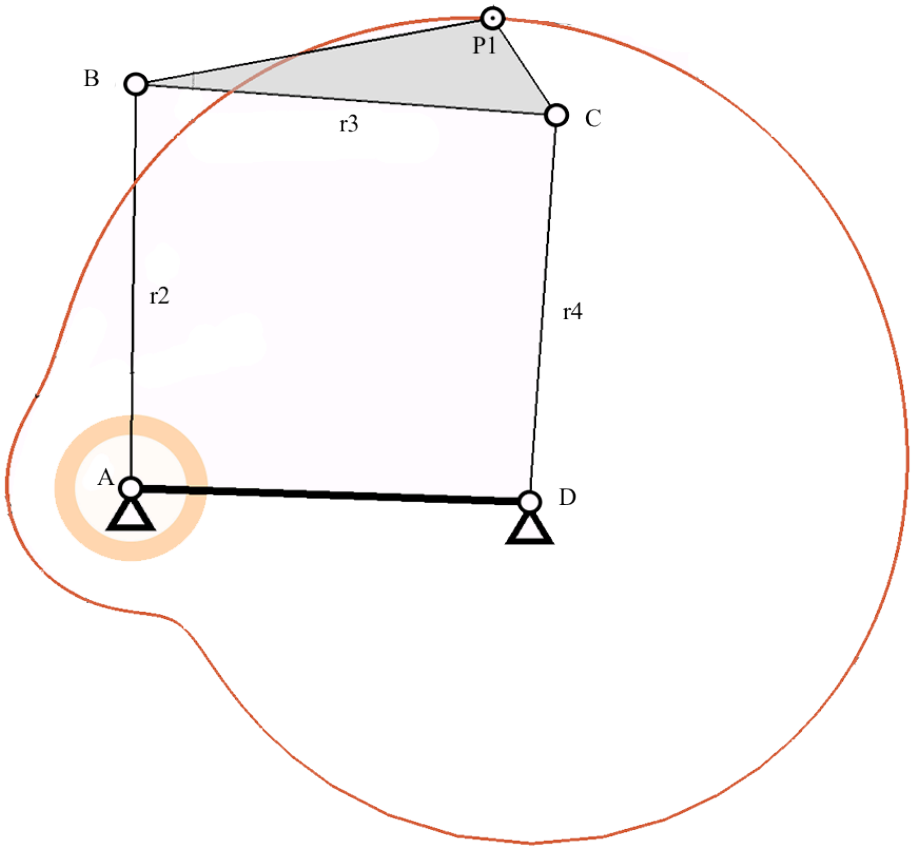


Figure 2.5: Double-crank linkage

Non-Grashof linkage:

For linkages of this type continuous relative motion between any two of its links is not possible.

Inward/Inward limited triple rocker:

When r1 + r2 < r3 + r4 the input link (r2) of the linkage is said to be inward limited. When r1 + r4 < r2 + r3 the output link (r4) of the linkage is said to be inward limited.

When a link is inward limited there are limits on the possible values of the angle of the link that the actual angle must remain outside (the link is limited when moving in the inward direction).

The limits are defined as follows:

Input Link

lower limit: $\theta_2 = \theta_1 + \theta_2^{''}$

upper limit: $\theta_2 = \theta_1 + 360° - \theta_2^{'}$

where $\theta_2^{'} = a\cos[\frac{r_1^2 + r_2^2 - (r_4 - r_3)^2}{2r_1 r_2}]$

Output Link

lower limit: $\theta_4 = \theta_1 - 180° + \theta_4^{'}$

upper limit: $\theta_4 = \theta_1 + 180° - \theta_4^{'}$

where $\theta_4^{'} = a\cos[\frac{r_1^2 + r_4^2 - (r_2 - r_3)^2}{2r_1 r_4}]$

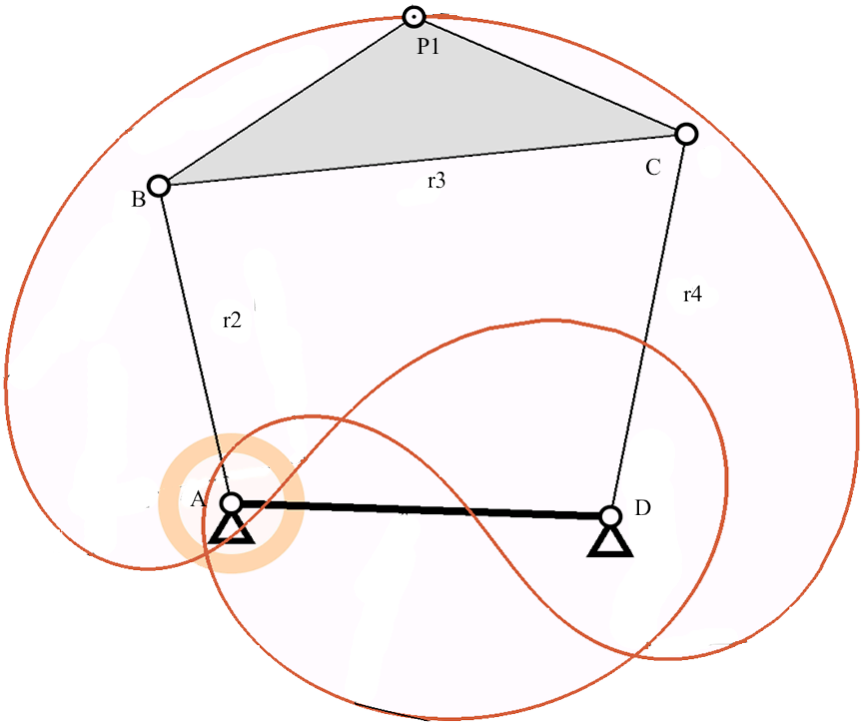


Figure 2.6: Inward/Inward limited triple rocker

Outward/Inward Limited Triple Rocker:

When r1 + r2 >= r3 + r4 the input link (r2) of the linkage is said to be outward limited. When r1 + r4 < r2 + r3 the output link (r4) of the linkage is said to be inward limited.

When a link is outward limited there are limits on the possible values of the angle of the link that the actual angle must remain inside (the link is limited when moving in the outward direction).

When a link is inward limited there are limits on the possible values of the angle of the link that the actual angle must remain outside (the link is limited when moving in the inward direction).

These limits are defined as follows:

Input Link

lower limit: $\theta_2 = \theta_1 - \theta_2^{''}$

upper limit: $\theta_2 = \theta_1 + \theta_2^{''}$

where $\theta_2^{''} = a\cos[\frac{r_1^2 + r_2^2 - (r_4 + r_3)^2}{2r_1 r_2}]$

Output Link

lower limit: $\theta_4 = \theta_1 - 180° + \theta_4^{'}$

upper limit: $\theta_4 = \theta_1 + 180° - \theta_4^{'}$

where $\theta_4^{'} = a\cos[\frac{r_1^2 + r_4^2 - (r_2 - r_3)^2}{2r_1 r_4}]$

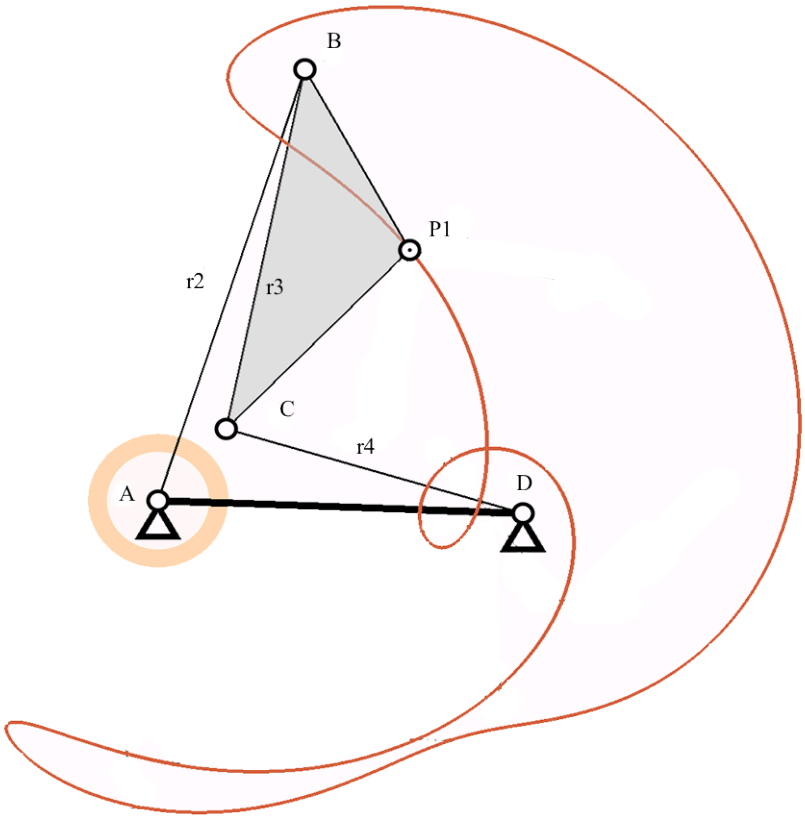


Figure 2.7: Outward/Inward Limited Triple Rocker

Inward/Outward Limited Triple Rocker:

When r1 + r2 < r3 + r4 the input link (r2) of the linkage is said to be inward limited. When r1 + r4 >= r2 + r3 the output link (r4) of the linkage is said to be outward limited.

When a link is inward limited there are limits on the possible values of the angle of the link that the actual angle must remain outside (the link is limited when moving in the inward direction).

When a link is outward limited there are limits on the possible values of the angle of the link that the actual angle must remain inside (the link is limited when moving in the outward direction).

These limits are defined as follows:

Input Link

lower limit: $\theta_2 = \theta_1 + \theta_2^{'}$

upper limit: $\theta_2 = \theta_1 + 360° - \theta_2^{'}$

where $\theta_2^{'} = a\cos[\frac{r_1^2 + r_2^2 - (r_4 - r_3)^2}{2r_1 r_2}]$

Output Link

lower limit: $\theta_4 = \theta_1 + 180° - \theta_4^{''}$

upper limit: $\theta_4 = \theta_1 + 180° + \theta_4^{''}$

where $\theta_4^{''} = a\cos[\frac{r_1^2 + r_4^2 - (r_2 + r_3)^2}{2r_1 r_4}]$

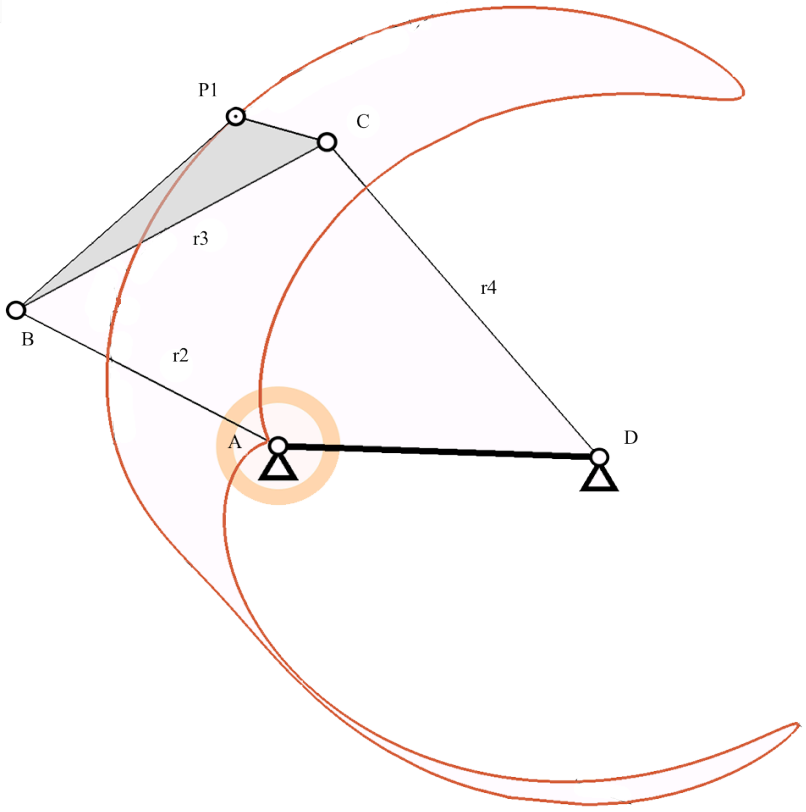


Figure 2.8: Inward/Outward Limited Triple Rocker

Outward/Outward Limited Triple Rocker:

When r1 + r2 >= r3 + r4 the input link (r2) of the linkage is said to be outward limited. When r1 + r4 >= r2 + r3 the output link (r4) of the linkage is said to be outward limited.

When a link is outward limited there are limits on the possible values of the angle of the link that the actual angle must remain inside (the link is limited when moving in the outward direction).

When a link is outward limited there are limits on the possible values of the angle of the link that the actual angle must remain inside (the link is limited when

moving in the outward direction).

These limits are defined as follows:

Input Link

lower limit: $\theta_2 = \theta_1 + \theta_2^{''}$

upper limit: $\theta_2 = \theta_1 + \theta_2^{''}$

where $\theta_2^{''} = a\cos[\frac{r_1^2 + r_2^2 - (r_4 + r_3)^2}{2r_1 r_2}]$

Output Link

lower limit: $\theta_4 = \theta_1 + 180° - \theta_4^{''}$

upper limit: $\theta_4 = \theta_1 + 180° + \theta_4^{''}$

where $\theta_4^{''} = a\cos[\frac{r_1^2 + r_4^2 - (r_2 + r_3)^2}{2r_1 r_4}]$

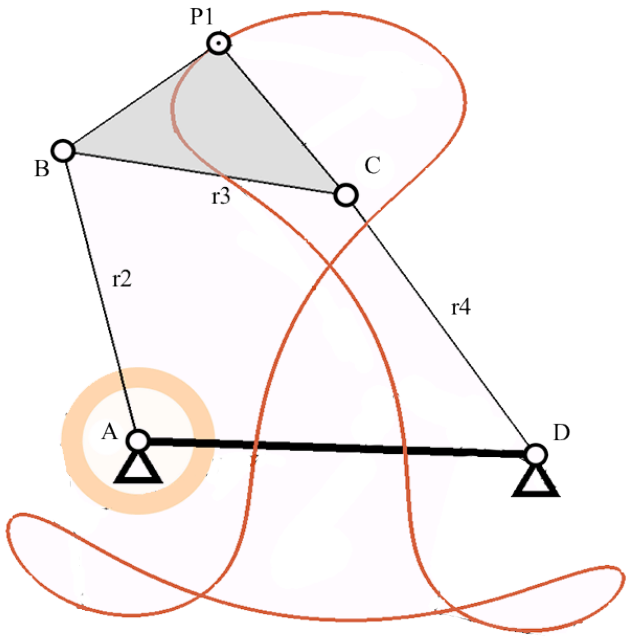


Figure 2.9: Outward/Outward Limited Triple Rocker

2.2. The relative theories on simulation

2.2.1 The Rotation of Coupler Link

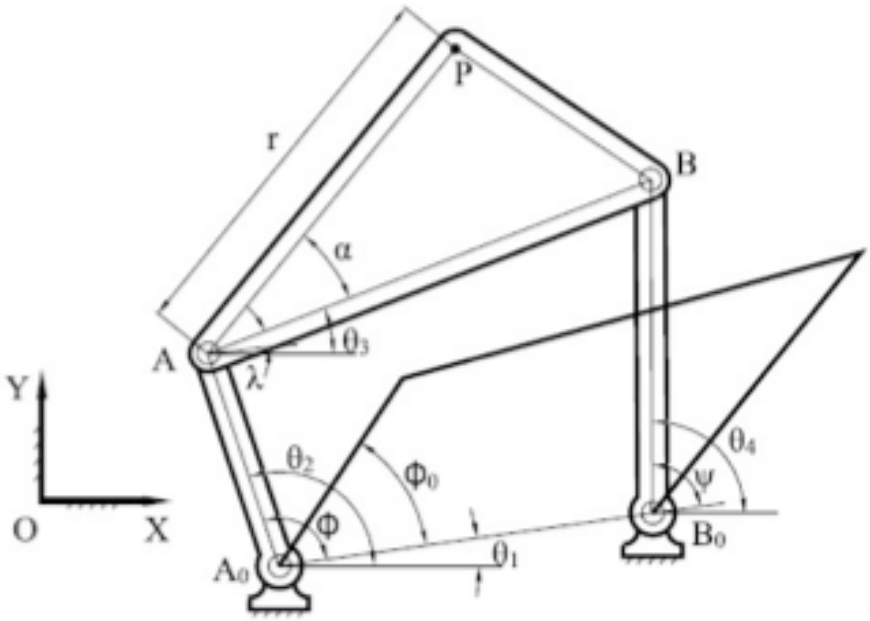


Figure 2.10: Four-bar mechanism example

Figure 2.10 shows a four-bar mechanism in the fixed coordinate XOY. A_0 and B_0 are the two fixed pivots of the ground links. Let AA_0 be the input link, AB be the coupler and BB_0 be the driven link. The length of A_0B_0, AA_0, AB and BB_0 are l_1, l_2, l_3 and l_4 respectively.

With the constant angular velocity, we have the equation

$$\Phi = \omega t + \varphi 0 \qquad (1)$$

where $\varphi 0$ is the initial input angle and ω is the angular velocity.

The point of the simulation problem if to find the coupler angle λ which depends

on the input angle and the link ratios as following

$$l21 = l2/l1,\ l31 = l3/l1,\ l41 = l4/l1 \tag{2}$$

With the loop closure equations, we can get the coupler angle λ

$$e^{i\lambda} = \frac{-B(\phi) \pm \sqrt{\Delta_1(\phi)\Delta_2(\phi)}}{2A(\phi)} \tag{3}$$

where

$$\begin{aligned} A(\varphi) &= l31\ (\ l21e^{-j\varphi} - 1)\ , \\ B(\varphi) &= 1 + l_{21} + l_3{}^2{}_1 - l_4{}^2{}_1 - 2l21\cos\varphi, \\ \Delta1(\varphi) &= 1 + l_{21} - (l31 + l41)^2 - 2l21\cos\varphi, \\ \Delta2(\varphi) &= 1 + l_{21} - (l31 - l41)^2 - 2l21\cos\varphi. \end{aligned} \tag{4}$$

The sign ± represents the two configurations of the same four-bar linkages.

2.2.2 The Path of Coupler Point

To get the path of the coupler point, we can use Fourier representation of the coupler point path of a four-bar mechanism.

Let $A0 = x0 + iy0$ be the complex number representing the fixed pivot $A0$ and let

$z = re^{i\alpha}$ be the position of point P on the moving frame with respect to the coupler link AB. The position of P in the global frame XOY can be given by

$$P=A0+l2e^{i\theta 2}+ze^{i\theta 3}=A0+l2e^{i\theta 1}e^{i\varphi}+ze^{i\theta 1}e^{i\lambda} \quad (5)$$

2.3 Introduction on the motion synthesis

The motion synthesis is very important in the mechanism design when the aim is to find the best solution by the given positions as the output poses. With the right approach, one can get the idealized multi-bar (four-bar in our app) linkages passing through all the poses exactly (five positions or less) or most closely (six positions or more). A better approach will minimize the cost and maximize the efficiency. It can also provide the simplest mechanism to the designer.

For the finite position synthesis, let's introduce the most famous Burmester theory [13]. Burmester theory can be used to seek the circling point in a moving body by the given movements of the same body.

The following is the two-position synthesis example

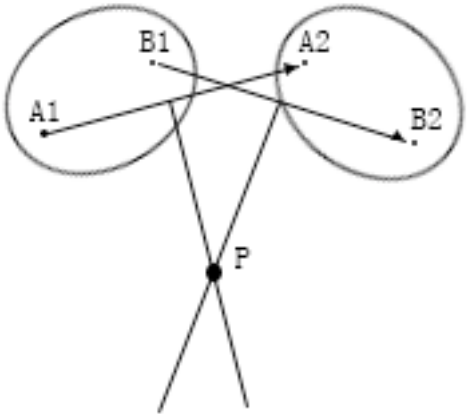


Figure 2.11: Example of two position synthesis

There are two points on the same moving body A and B, which moves from A_1 to A_2 and B_1 to B_2. The circling point of the moving body in this example can be achieved by finding the intersection of two lines perpendicular to the A_1A_2 and B_1B_2, respectively. Based on the theory, positions of four or less will give infinite solutions.

As to the five positions synthesis, Burmester obtained the circling points curve generated by the four positions of the five. Burmester shows that these curves can intersect as many as four points, which means it can have six solutions, one solution or no solution.

2.4 The relative theories on the motion synthesis

2.4.1 Brief Introduction on Quaternion

The theory implements quaternion into the calculation as an input format. Before

going into the details of the theory, a brief introduction on quaternion should be completed.

The quaternions are a way of formatting number system with extension to complex numbers. They were first presented by Irish mathematician William Rowan Hamilton in 1843.

Quaternions combine both translation and rotation in one format of numbers. They form a four-dimensional space while complex numbers form a three-dimensional space.

Quaternion is a number with the form of ai+bj+ck+d (a, b, c, d are real numbers, $i^2 = j^2 = k^2 = -1$, $ij = k, ji = -k, jk = i, kj = -i, ki = j, ik = -j$). $a^2 + b^2 + c^2 + d^2$ is called the modulus of a quaternion. The calculation rules can be easily found in any of the texts mentioning quaternions.

Quaternions can be used in computer graphics and graphical analysis to present the position and rotation of a 3D object. It is also used in control theory, signal processing, positioning control, physics and mechanics. Quaternions are more stable and easier than most of the matrix transformation.

2.4.2 Kinematic Mapping of Planar Kinematics

The general solution of the displacement of a rigid body in a plan can be introduced as the translation part and the rotation part. Let M represent a coordinate frame of the moving body and F be a fixed global frame. Then the planar

displacement can be represented as a transformation of point or line coordinates from M to F. The point coordinate transformation associated with a planar displacement is given by

$$[H]=\begin{bmatrix}\cos\phi & -\sin\phi & d_1\\ \sin\phi & \cos\phi & d_2\\ 0 & 0 & 1\end{bmatrix} \tag{6}$$

The line coordinate transformation for the same displacement is given by the transpose of the inverse of [H]

$$[\overline{H}]=([H]^{-1})^T=\begin{bmatrix}\cos\phi & -\sin\phi & 0\\ \sin\phi & \cos\phi & 0\\ -d_1\cos\phi-d_2\sin\phi & d_1\sin\phi-d_2\cos\phi & 1\end{bmatrix} \tag{7}$$

If one introduces

$$\begin{aligned} Z_1 &= \frac{1}{2}(d_1\sin\frac{\phi}{2}-d_2\cos\frac{\phi}{2}),\\ Z_2 &= \frac{1}{2}(d_1\cos\frac{\phi}{2}+d_2\sin\frac{\phi}{2}),\\ Z_3 &= \sin\frac{\phi}{2},\\ Z_4 &= \cos\frac{\phi}{2}. \end{aligned} \tag{8}$$

The former can be transferred into

$$[H]=\begin{bmatrix} Z_4^2-Z_3^2 & -2Z_3Z_4 & 2(Z_1Z_3+Z_2Z_4) \\ 2Z_3Z_4 & Z_4^2-Z_3^2 & 2(Z_2Z_3-Z_1Z_4) \\ 0 & 0 & Z_3^2+Z_4^2 \end{bmatrix} \tag{9}$$

and

$$[\overline{H}]=\begin{bmatrix} Z_4^2-Z_3^2 & -2Z_3Z_4 & 0 \\ 2Z_3Z_4 & Z_4^2-Z_3^2 & 0 \\ 2(Z_1Z_3-Z_2Z_4) & 2(Z_2Z_3+Z_1Z_4) & Z_3^2+Z_4^2 \end{bmatrix} \tag{10}$$

of which $Z_3^2+Z_4^2=1$.

The four-dimensional vector $Z=(Z_1;Z_2;Z_3;Z_4)$ is said to define a point in projective three-space called the Image Space of planar displacement, denoted as S. In this way, a planar displacement is represented by a point in S; a 1- DOF motion is represented by a curve and a 2-DOF motion is represented by a surface.

Let (x; y) denote the coordinates of a point in the moving frame M and (X; Y) their corresponding coordinates with respect to the fixed frame F. It follows from (9) that

$$\begin{aligned} X &= \frac{(Z_4^2-Z_3^2)x-2Z_3Z_4y+2(Z_1Z_3+Z_2Z_4)}{Z_3^2+Z_4^2}, \\ Y &= \frac{2Z_3Z_4x+(Z_4^2-Z_3^2)y+2(Z_2Z_3-Z_1Z_4)}{Z_3^2+Z_4^2}. \end{aligned} \tag{11}$$

Let (l_1; l_2; l_3) denote the coordinates of a line in M where $l_1^2+l_2^2=1$ and the absolute value of l_3 is the distance from the origin of M to the line. Let (L_1; L_2; L_3) be

the coordinates of the same line with respect to F. It follows from (10) that

$$
\begin{aligned}
L_1 &= \frac{(Z_4^2 - Z_3^2)l_1 - 2Z_3Z_4l_2}{Z_3^2 + Z_4^2}, \\
L_2 &= \frac{2Z_3Z_4l_1 + (Z_4^2 - Z_3^2)l}{Z_3^2 + Z_4^2}, \\
L_3 &= \frac{2(Z_1Z_3 - Z_2Z_4)l_1 + 2(Z_2Z_3 + Z_1Z_4)l_2}{Z_3^2 + Z_4^2} + l_3.
\end{aligned}
\tag{12}
$$

2.4.3 RR and PR Dyads: the Original Formulation

A planar RR dyad defines a 2-DOF motion that one point stays on a circle. In general, a circle can be given as a set of four homogeneous coordinates $(a_0; a_1; a_2; a_3)$ (where $a_0 \neq 0$) as

$$
a_0(X^2 + Y^2) + 2a_1X + 2a_2Y + a_3 = 0, \tag{13}
$$

where the center (X_c; Y_c) of the circle and the radius R are given by

$$
X_c = -\frac{a_1}{a_0}, Y_c = -\frac{a_2}{a_0}, R = \frac{\sqrt{a_1^2 + a_2^2 - a_0a_3}}{a_0}. \tag{14}
$$

and when $a_0 = 0$, what we get is $2a_1X + 2a_2Y + a_3 = 0$ which is the expression of a straight line. Substituting (11) into (13), we obtain an equation of fourth order in terms Z_i (i = 1; 2; 3; 4), which factors into two quadratic terms. One is $Z_3^2 + Z_4^2$ and

the other is as follows

$$\begin{aligned}
&a_0(Z_1^2+Z_2^2)+(-a_0x+a_1)Z_1Z_3+(-a_0y+a_2)Z_2Z_3\\
&+(-a_0y-a_2)Z_1Z_4+(a_0x+a_1)Z_2Z_4+(-a_1y+a_2x)Z_3Z_4\\
&+(a_0(x^2+y^2)-2a_1x-2a_2y+a_3)Z_3^2/4\\
&+(a_0(x^2+y^2)+2a_1x+2a_2y+a_3)Z_4^2/4=0.
\end{aligned} \tag{15}$$

This quadric surface is called the constraint manifold of the RR dyad.

When $a_0 = 0$, Eq.(13) reduces to a linear equation representing a line and Eq.(15) reduces to

$$\begin{aligned}
&a_1(Z_1Z_3+Z_2Z_4)+a_2(Z_2Z_3-Z_1Z_4)+(-a_1y+a_2x)Z_3Z_4\\
&+((a_1x+a_2y)/2)(Z_4^2-Z_3^2)+(a_3/4)(Z_3^2+Z_4^2)=0.
\end{aligned} \tag{16}$$

This is the constraint manifold of a 2-DOF motion for which one point stays on a line, i.e., the motion of a PR dyad.

2.4.4 RR and PR Dyads: the Unified Formulation

We note that all five terms involving Z_i (i = 1; 2; 3; 4) in Eq. (16) are exactly elements of the matrix [H] as given by (9). Furthermore, if we let

$$a_4 = (-a_1y+a_2x), a_5 = (a_1x+a_2y)/2, \tag{17}$$

it follows that the constraint manifold is defined by five unconstrained homogeneous parameters $a_i(i = 1;2;\cdots;5)$. This gives us a clue to rewrite Eq.(15) into:

$$\begin{aligned} &p_1(Z_1^2 + Z_2^2) + p_2(Z_1Z_3 - Z_2Z_4) + p_3(Z_2Z_3 + Z_1Z_4) \\ &+ p_4(Z_1Z_3 + Z_2Z_4) + p_5(Z_2Z_3 - Z_1Z_4) + p_6Z_3Z_4 \\ &+ p_7(Z_3^2 - Z_4^2) + p_8(Z_3^2 + Z_4^2) = 0, \end{aligned} \tag{18}$$

where

$$\begin{aligned} &p_1 = a_0, p_2 = -a_0x, p_3 = -a_0y, \\ &p_4 = a_1, p_5 = a_2, \\ &p_6 = -a_1y + a_2x, p_7 = (a_1x + a_2y)/2, \\ &p_8 = (a_0(x^2 + y^2) + a_3)/4. \end{aligned} \tag{19}$$

Note that Eq.(18) contains all elements of the matrices [H] and [H] as given by (9) and (10), respectively. The only new term in Eq.(18) is $Z_1^2 + Z_2^2$. The set of eight homogeneous parameters $p_i(i = 1;2;\cdots;8)$ are not independent, however. It follows from Eq.(19) that they must satisfy the following two relations:

$$\begin{aligned} &p_1p_6 + p_2p_5 - p_3p_4 = 0, \\ &2p_1p_7 - p_2p_4 - p_3p_5 = 0. \end{aligned} \tag{20}$$

It is clear from (19) that when $p_1 = a_1 = 0$, one has $p_2 = p_3 = 0$ and that both relations in (20) are automatically satisfied.

2.4.5 RP Dyad: the Unified Formulation

The motion of a RP dyad is constrained such that a line L = (L_1; L_2; L_3) of M stays tangent to a given circle C. Let the circle be defined in the same way as before, i.e., by (13), which can be expressed in matrix form:

$$\begin{bmatrix} X & Y & 1 \end{bmatrix} \begin{bmatrix} a_0 & 0 & a_1 \\ 0 & a_0 & a_2 \\ a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = 0. \tag{21}$$

The adjoint of the coefficient matrix above is given by

$$\begin{bmatrix} C_{adj} \end{bmatrix} = \begin{bmatrix} a_0 a_3 - a_2^2 & a_1 a_2 & -a_0 a_1 \\ a_1 a_2 & a_0 a_3 - a_1^2 & -a_0 a_2 \\ -a_0 a_1 & -a_0 a_2 & a_0^2 \end{bmatrix} \tag{22}$$

It is well known in projective geometry of conics that a line L = (L_1; L_2; L_3) stays tangent to the circle C when

$$L^T \begin{bmatrix} C_{adj} \end{bmatrix} L = 0 \tag{23}$$

which is known as the dual conic.

Instead of substituting (12) directly into (23) to obtain the constraint manifold of RP dyad, we first seek to simplify [C_{adj}]. In view of Eq.(14), we have $a_0 a_3 = a_1^2 + a_2^2 - a_0^2 R^2$, where R is the radius of the circular constraint. Substituting

this relationship into (22), we can decompose [C_{adj}] into

$$\left[C_{adj}\right]=\begin{bmatrix} a_1^2 & a_1a_2 & -a_0a_1 \\ a_1a_2 & a_2^2 & -a_0a_2 \\ -a_0a_1 & -a_0a_2 & a_0^2 \end{bmatrix}-\begin{bmatrix} a_0^2R^2 & 0 & 0 \\ 0 & a_0^2R^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{24}$$

Substituting (24) into (23), we obtain

$$(-a_1L_1-a_2L_2+a_0L_3)^2-a_0^2R^2(L_1^2+L_2^2)=0. \tag{25}$$

The solution to Eq.(25) yields

$$\frac{-a_1L_1-a_2L_2+a_0L_3}{a_0\sqrt{L_1^2+L_2^2}}=\pm R, \tag{26}$$

which indicates that the distance from the center of the circle C with homogeneous coordinates $(-a_1,-a_2,a_0)$ to the tangent line L is equal to the radius of C.

Furthermore, in view of (12), we have

$$\sqrt{L_1^2+L_2^2}=(Z_3^2+Z_4^2)\sqrt{l_1^2+l_2^2}=Z_3^2+Z_4^2 \tag{27}$$

where $l_1^2+l_2^2=1$. Substituting (12) and (27) into (26), we can put the resulting constraint manifold in the same form as (18) where

$$
\begin{aligned}
&p_1 = 0, p_2 = 2a_0 l_1, p_3 = 2a_0 l_2, p_4 = 0, p_5 = 0, \\
&p_6 = 2a_1 l_2 + 2a_2 l_1, p_7 = 2a_1 l_1 + 2a_2 l_2, p_8 = a_0 (l_3 \pm R).
\end{aligned} \tag{28}
$$

Again, the above coefficients satisfy the relations (20). Thus we have the following Theorem regarding the representation of kinematic dyads, RR, PR, and RP in the image space:

Theorem: All three dyads can be represented by quadric surfaces of the form (18) with eight homogeneous coefficients $(p_1, p_2, \cdots, p_8)$. In addition, only a subset of these quadrics such that the first seven coefficients satisfy (20) is constraint manifolds for the dyads.

2.4.6 Algebraic Fitting of a Pencil of Quadrics for Finite-Position Synthesis

Now consider the problem of fitting a pencil of quadrics to a set of N points, arranged such that they define an image curve rather than a surface. This problem can be formulated an over-constrained linear problem [A]p = 0 where the coefficient matrix becomes the matrix [A] is given by:

$$
[A] = \begin{bmatrix}
A_{11} & A_{12} & A_{13} & A_{14} & A_{15} & A_{16} & A_{17} & A_{18} \\
\vdots & & & & & & & \vdots \\
\vdots & & & \ddots & & & & \vdots \\
\vdots & & & & & & & \vdots \\
A_{N1} & A_{N2} & A_{N3} & A_{N4} & A_{N5} & A_{N6} & A_{N7} & A_{N8}
\end{bmatrix} \tag{29}
$$

where

$$
\begin{aligned}
&A_{i1} = Z_{i1}^2 + Z_{i2}^2, A_{i2} = Z_{i1}Z_{i3} - Z_{i2}Z_{i4}, \\
&A_{i3} = Z_{i2}Z_{i3} + Z_{i1}Z_{i4}, A_{i4} = Z_{i1}Z_{i3} + Z_{i2}Z_{i4}, \\
&A_{i5} = Z_{i2}Z_{i3} - Z_{i1}Z_{i4}, A_{i6} = Z_{i3}Z_{i4}, \\
&A_{i7} = Z_{i3}^2 - Z_{i4}^2, A_{i8} = Z_{i3}^2 + Z_{i4}^2.
\end{aligned} \tag{30}
$$

are the terms of the ith position represented in quaternion form.

The over-constrained system of linear equations, [A]p = 0, which is equivalent as $[A]^T[A]p = 0$, can be viewed as to solve for the eigenvectors v of $[A]^T[A]$, with their corresponding eigenvalues to be zero, or as close to zero as possible. This idea comes from the definition of a matrix's eigenvector: $(\lambda[I] - [A]^T [A])v = 0$, when λ equals to zero, its corresponding eigenvector v is obviously the solution of $[A]^T[A]p = 0$, Likewise, if l is very close to zero, v becomes an approximated solution.

In [23], we have found that for a set (more than 5) of displacements that define an image curve, there exists a pencil of quadrics that best fit the given data points. The coefficients $p = (p_1; p_2; \ldots; p_8)$ for the pencil of quadrics are defined by the eigenvectors $v_\phi = \lfloor v_{\phi 1} \quad v_{\phi 2} \cdots \quad v_{\phi 8} \rfloor$ and $v_\beta = \lfloor v_{\beta 1} \quad v_{\beta 2} \cdots \quad v_{\beta 8} \rfloor$, corresponding to two smallest singular values, i.e.,

$$
p = \phi v_\phi + \beta v_\beta \tag{31}
$$

where ϕ, β are real scalars.

For five-position synthesis, since the rank of matrix [A] is in general five, there will be three zero eigenvalues for $[A]^T[A]$ that lead to three eigenvectors: v_α, v_β and

vγ.

Thus the coefficients for the pencil of quadrics now becomes:

$$p = \alpha v_\alpha + \beta v_\beta + \gamma v_\gamma . \tag{32}$$

Only those that satisfy (20) correspond to constraint manifolds of planar dyads.

Substituting Eq. 32 into (20), we obtain the following two quadratic equations,

$$\begin{aligned} K_{10} r_{\alpha\gamma}^2 + K_{11} r_{\beta\gamma}^2 + K_{12} r_{\alpha\gamma} r_{\beta\gamma} + K_{13} r_{\alpha\gamma} + K_{14} r_{\beta\gamma} + K_{15} = 0, \\ K_{20} r_{\alpha\gamma}^2 + K_{21} r_{\beta\gamma}^2 + K_{22} r_{\alpha\gamma} r_{\beta\gamma} + K_{23} r_{\alpha\gamma} + K_{24} r_{\beta\gamma} + K_{25} = 0, \end{aligned} \tag{33}$$

where $r_{\alpha\gamma} = \alpha/\gamma$ and $r_{\beta\gamma} = \beta/\gamma$. These two quadratic equations can be converted to one quartic equation and hereby can be analytically solved. Note that a quartic equation has up to four real roots, which means we will find up to four constraint manifolds that fit the five given position precisely. Since the intersection of any two of the four constraint manifolds defines a four-bar linkage, there are up to six four bar linkages that interpolates through the given five positions. This is consistently with the classical Burmester theory. Moreover, this new formulation allows handling of four bar linkages formed not only by revolute joints but prismatic joints as well.

After the coefficient vector p, which satisfies (20) is obtained, the type of dyads can be easily identified:

1. If $p_1 \neq 0$, p represents an RR dyad.

2. If $p_1 = 0$ as well as $p_2 = p_3 = 0$, p represents a PR dyad.

3. If $p_1 = 0$ as well as $p_4 = p_5 = 0$, p represents an RP Dyad.

Chapter 3

Inside the Program

3.1 Introduction on the overall scheme

"To design is much more than simply to assemble, to order, or even to edit; it is to add value and meaning, to illuminate, to simplify, to clarify, to modify, to dignify, to dramatize, to persuade, and perhaps even to amuse."

- Paul Rand

3.1.1 Basic elements of the software UI design

A blog named《Principles of User Interface Design》of famous user interface designer Joshua Porter [15] shows 19 basic principles of UI design:

1. Clarity

2. Interfaces allowing interaction

3. Try everything to catch attention

4. Guide the users

5. Direct manipulation

6. One main action per screen

7. Keep secondary actions secondary

8. Every step is natural

9. Appearance follows behavior

10. Consistency

11. Strong visual hierarchies

12. Have a good organization

13. Highlight

14. Progressive disclosure

15.Help people inline

16. A crucial moment: the zero state

17. Great design is invisible

18. Keep in mind with other design disciplines

19. Interfaces should be useful

3.1.2 Basic elements of the software program design

In one of the blog by Christopher Diggins [16] named《The Principles of Good Programming》, he gives 17 basic principles of program design:

1. Don't repeat

2. Abstract

3. Keep it as simple as possible

4. Avoid Creating a YAGNI

5. Do the simplest thing

6. Don’t make the users think

7. Open/Closed Principle

8. Write Code good for maintaining

9. Principle of least astonishment

10. Single Responsibility Principle

11. Minimize Coupling

12. Maximize Cohesion

13. Hide Implementation Details

14. Law of Demeter

15. Avoid Premature Optimization

16. Reuse code

17. Separate concerns

3.2 Simulation

3.2.1 Overall construction

The aim of the simulation is to allow users to draw any four-bar linkages by fingers on the screen and with enough information (ground links and power) the system shows the animation of the movement of the given four-bar.

There are three steps users should provide to the system: draw a four-bar linkage, give two ground links, choose a power link. If one wants to see the motion simulation of any point on the coupler link, they can give any single point by touch on the screen.

The animation will show the moving path of that particular point with a fixed coordinate on the coupler. It also shows the classification of the four-bar linkages in real time.

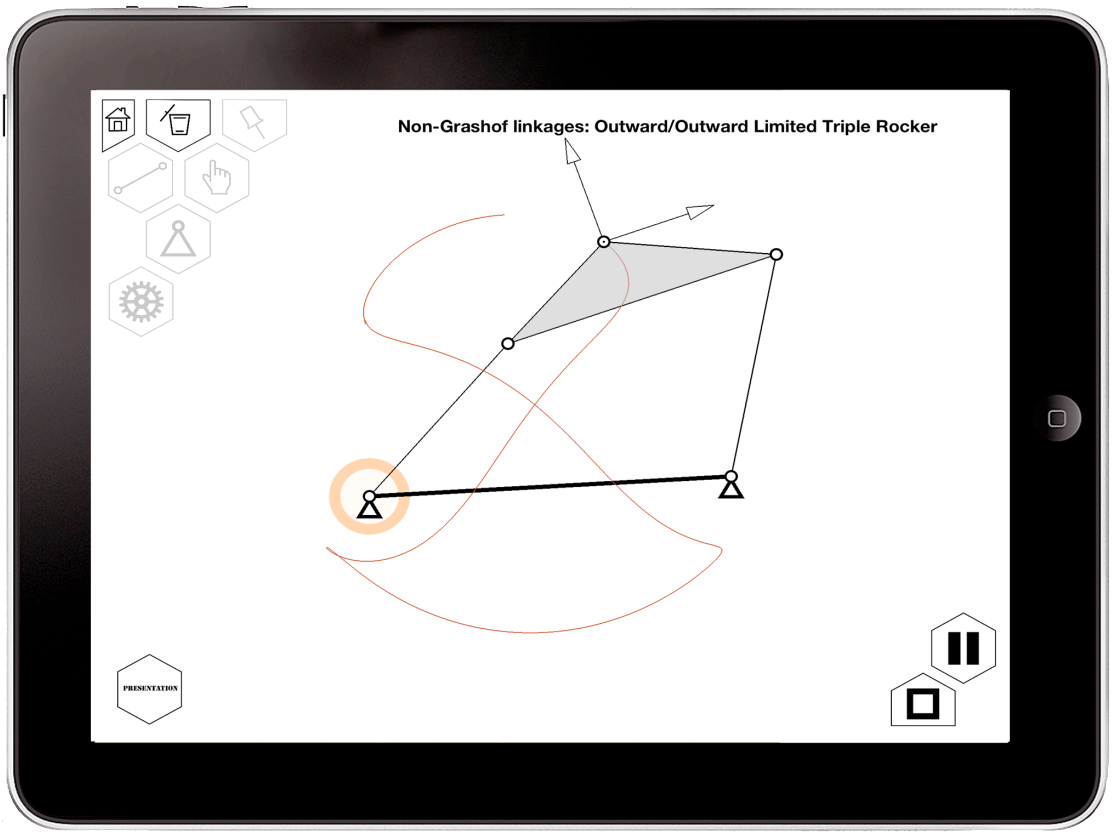


Figure 3.1: Screenshot of the simulation interface on iPad

3.2.2 UI design

Figure 3.1 shows a real user interface on iPad. One may see that all the buttons are in the shape of hexagon. One of the reasons is to save the space on the screen and maximize the input areas. Another reason is to make an impression of "growing" by showing buttons based on the step user is going on.

The buttons on the upper left corner of the screen are the main operating buttons for input. Users can use two fingers to move the mechanism (translation movement).

The functionality of each button is as following:

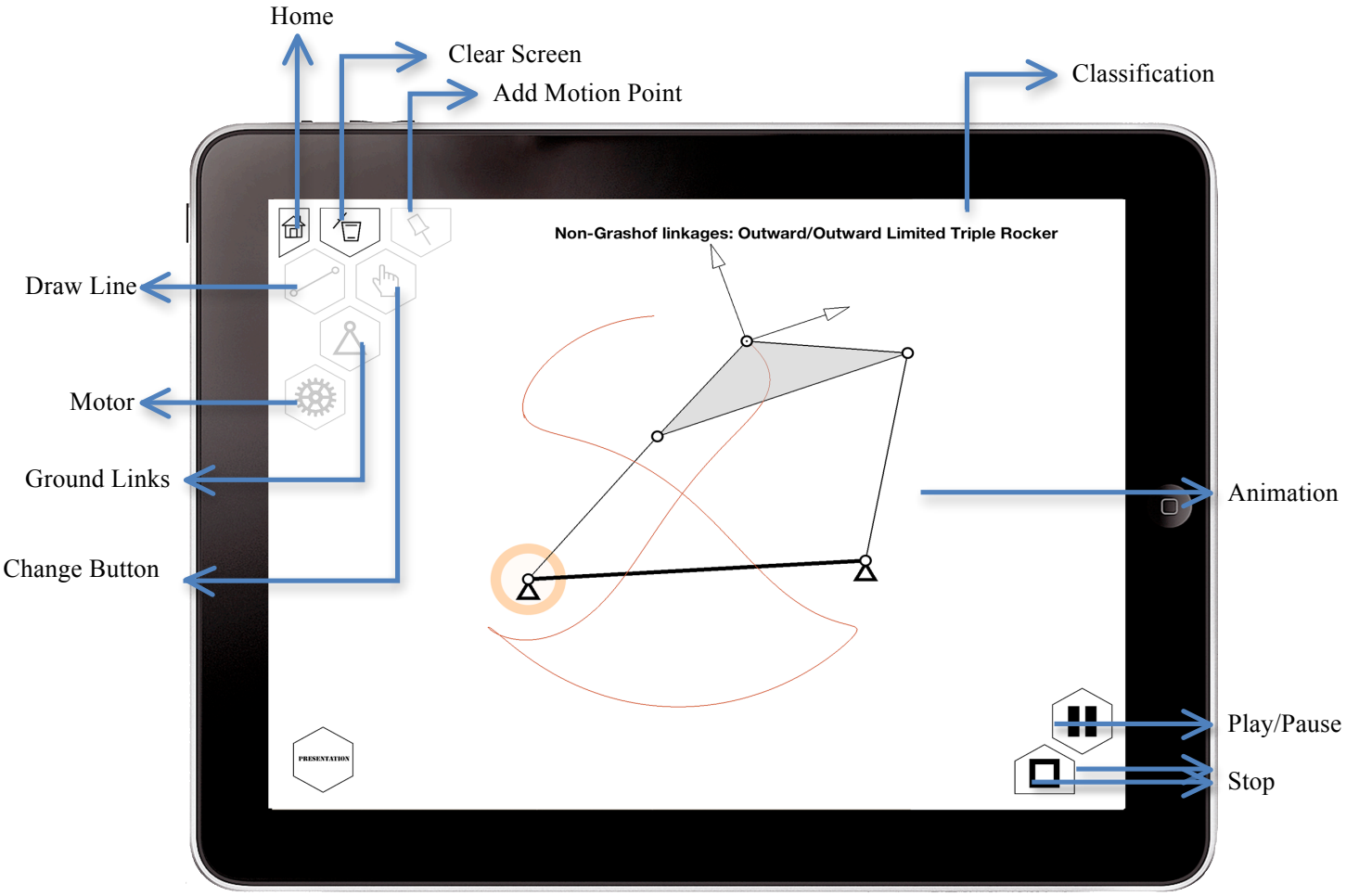


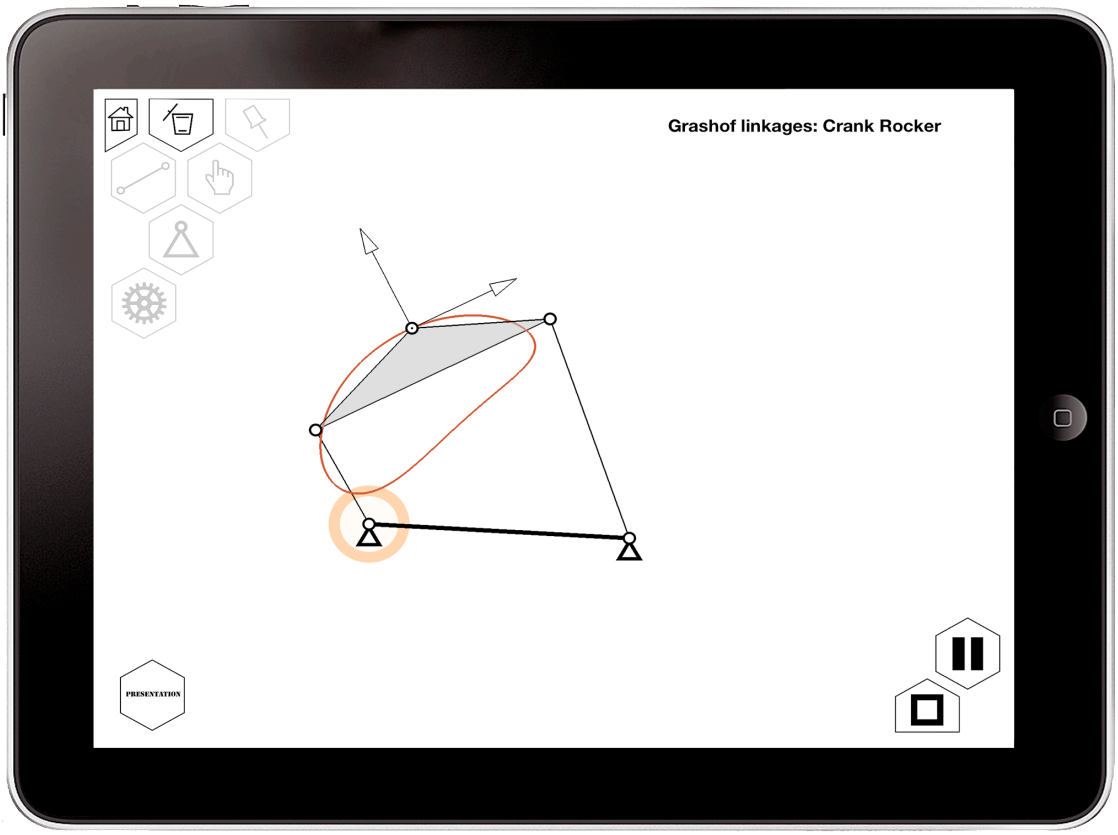
Figure 3.2: Buttons functionalities (simulation)

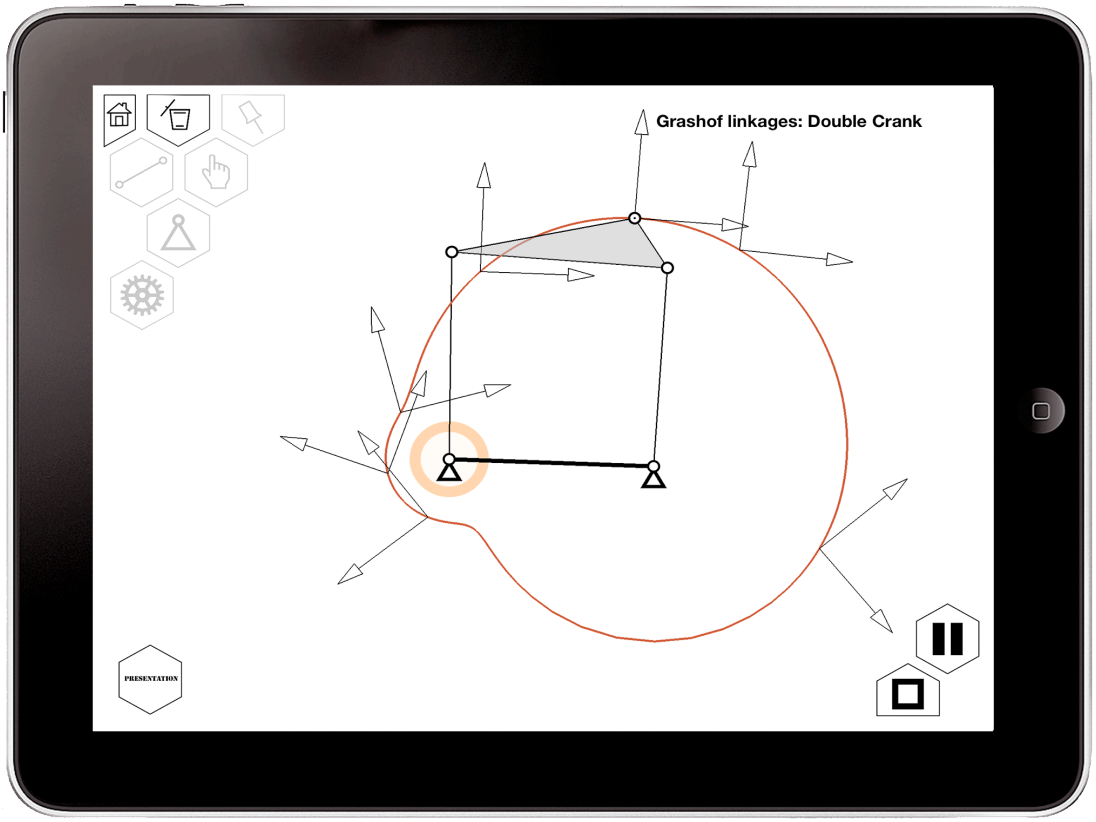
The special feature of the operating button on the upper left corner will appear one by one based on which step the users are utilizing. When moving to the next step, the previous step will automatically fade and be disabled for avoiding errors. This also helps to act as an instructive guide for the users since users will know easily when and where to press the next button.

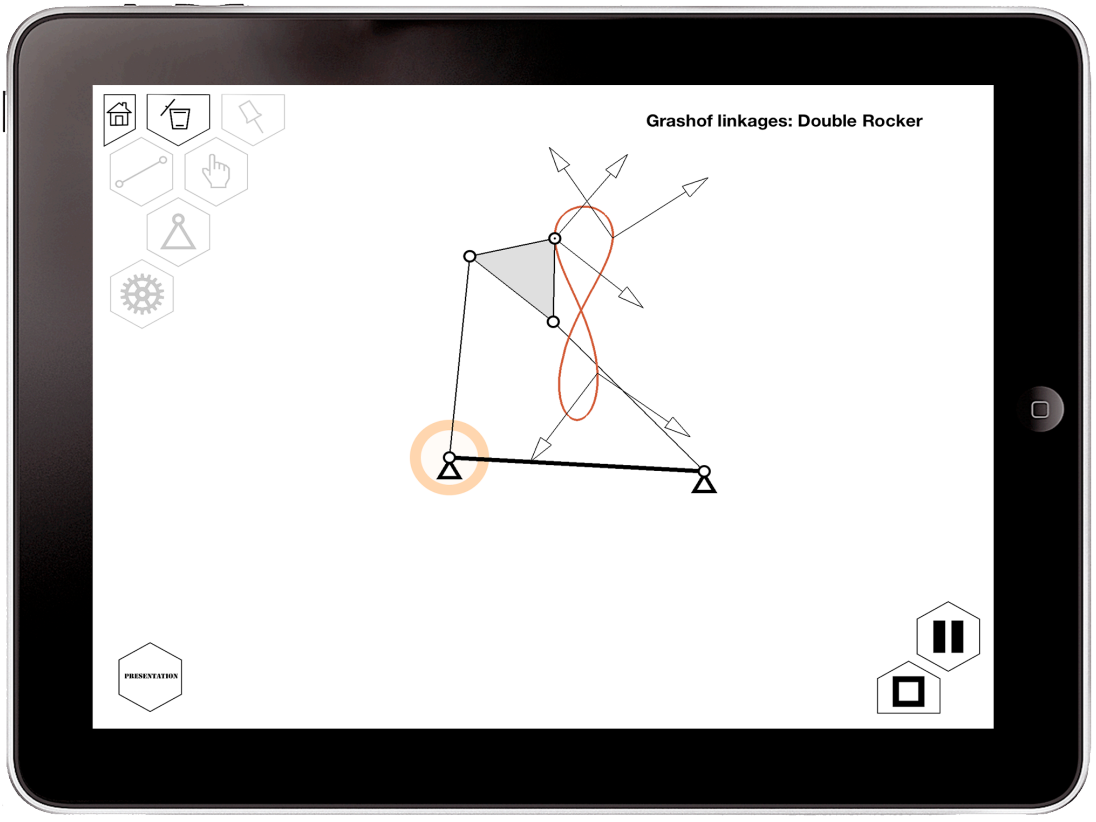
The classification label on the upper right corner is based on the same theories in chapter 2. The hints will change in real time along with the four-bar linkage users give.

Here are some mechanisms in each category:

Grashof:




Grashof linkages: Double Crank
PRESENTATION


Grashof linkages: Double Rocker
PRESENTATION

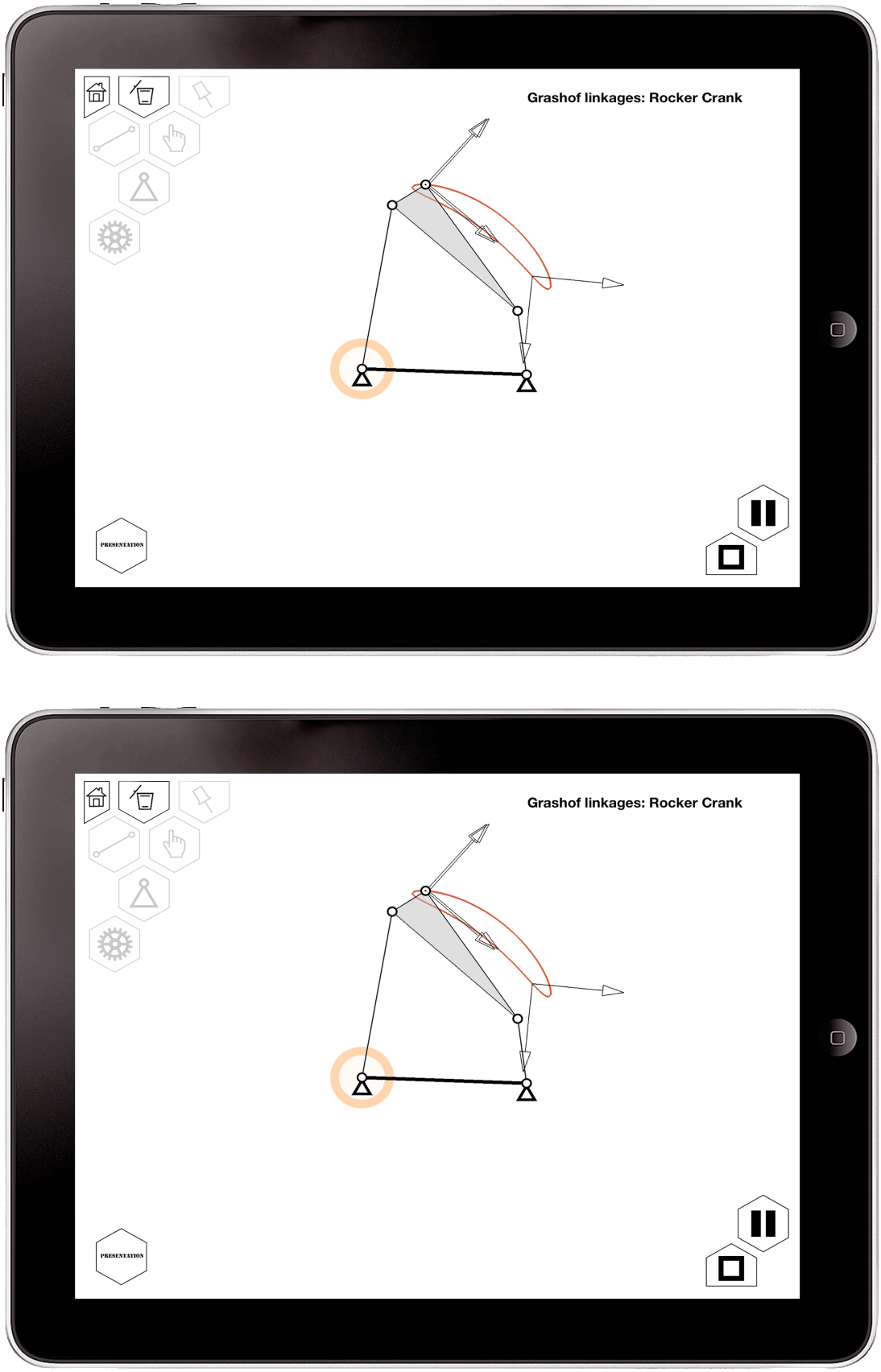
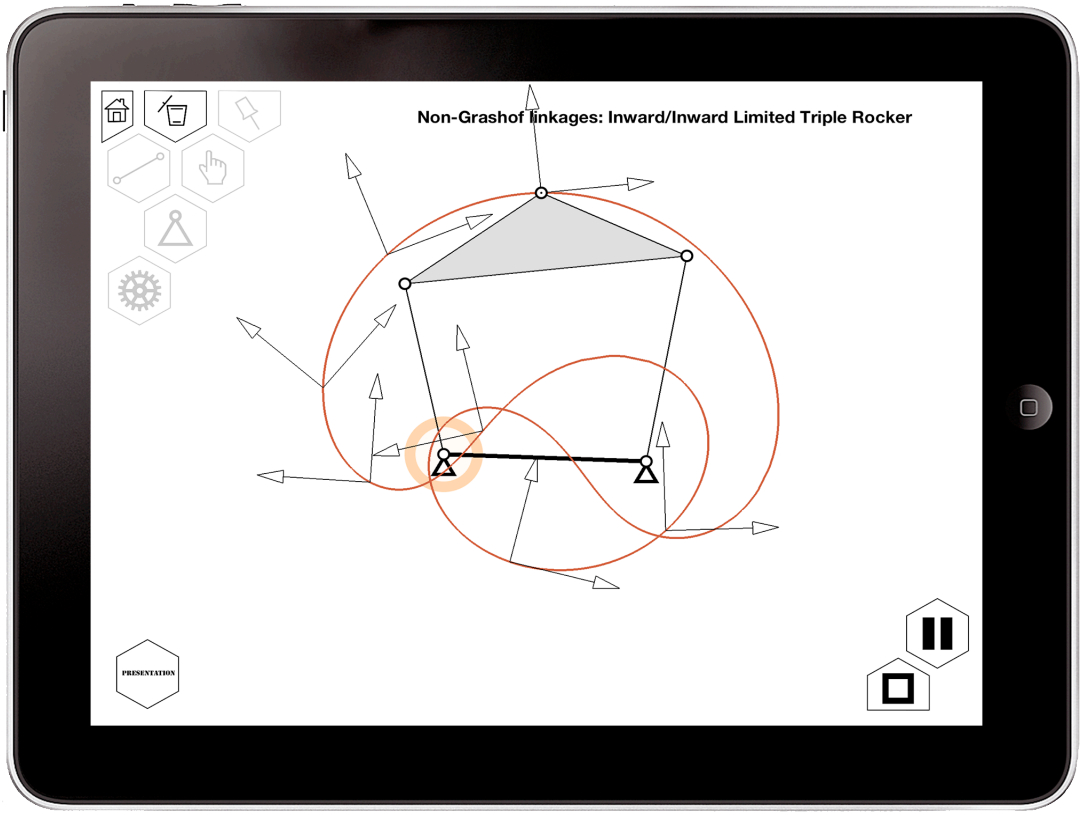
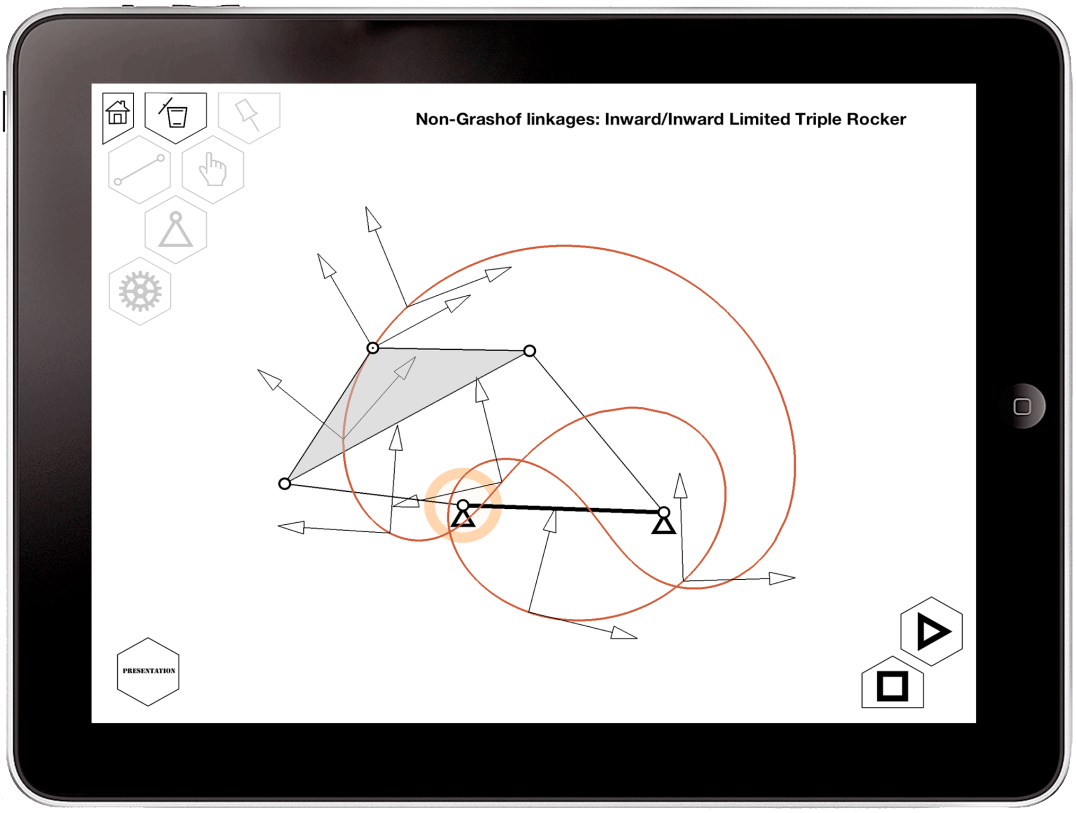
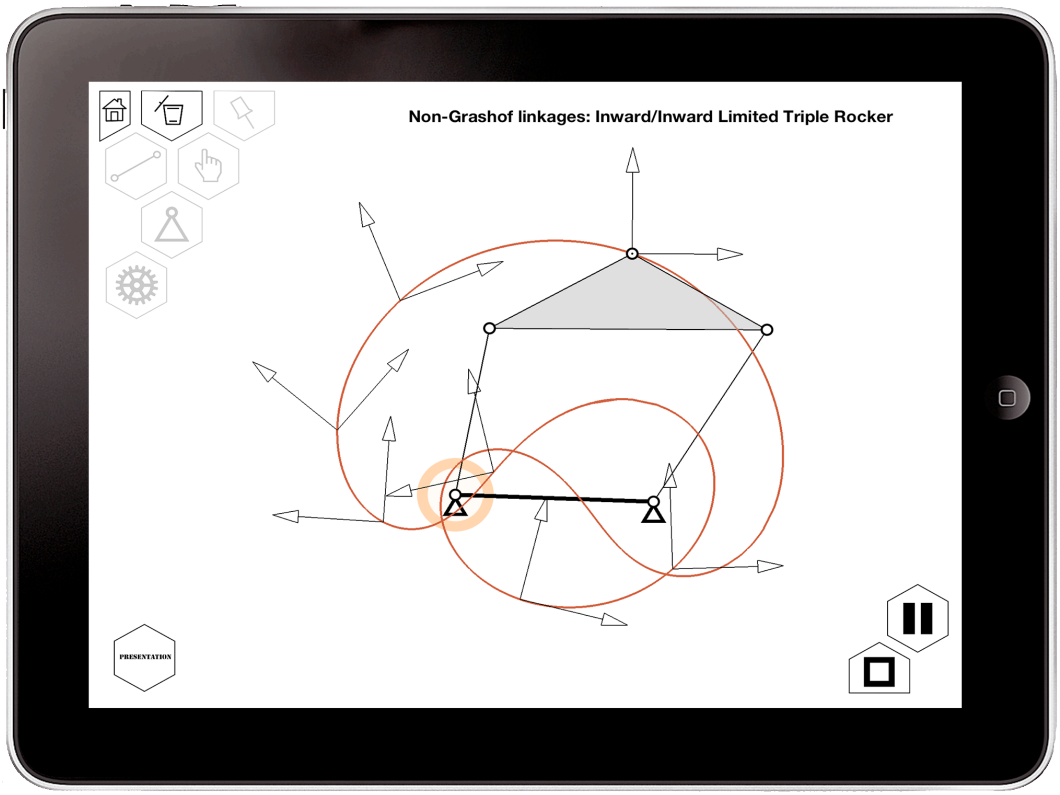


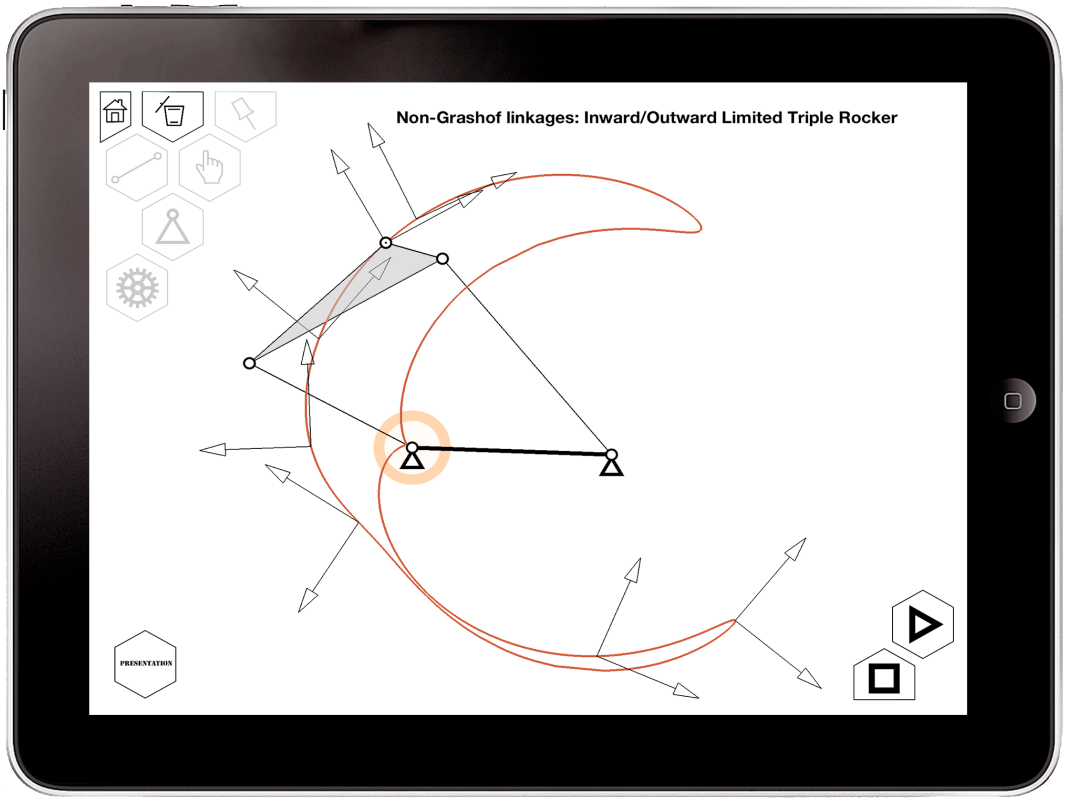
Figure 3.3: Screenshots of Grashof linkages categories

Non-Grashof:


Non-Grashof linkages: Inward/Inward Limited Triple Rocker
PRESENTATION


Non-Grashof linkages: Inward/Inward Limited Triple Rocker
PRESENTATION


Non-Grashof linkages: Inward/Inward Limited Triple Rocker
PRESENTATION


Non-Grashof linkages: Inward/Outward Limited Triple Rocker
PRESENTATION

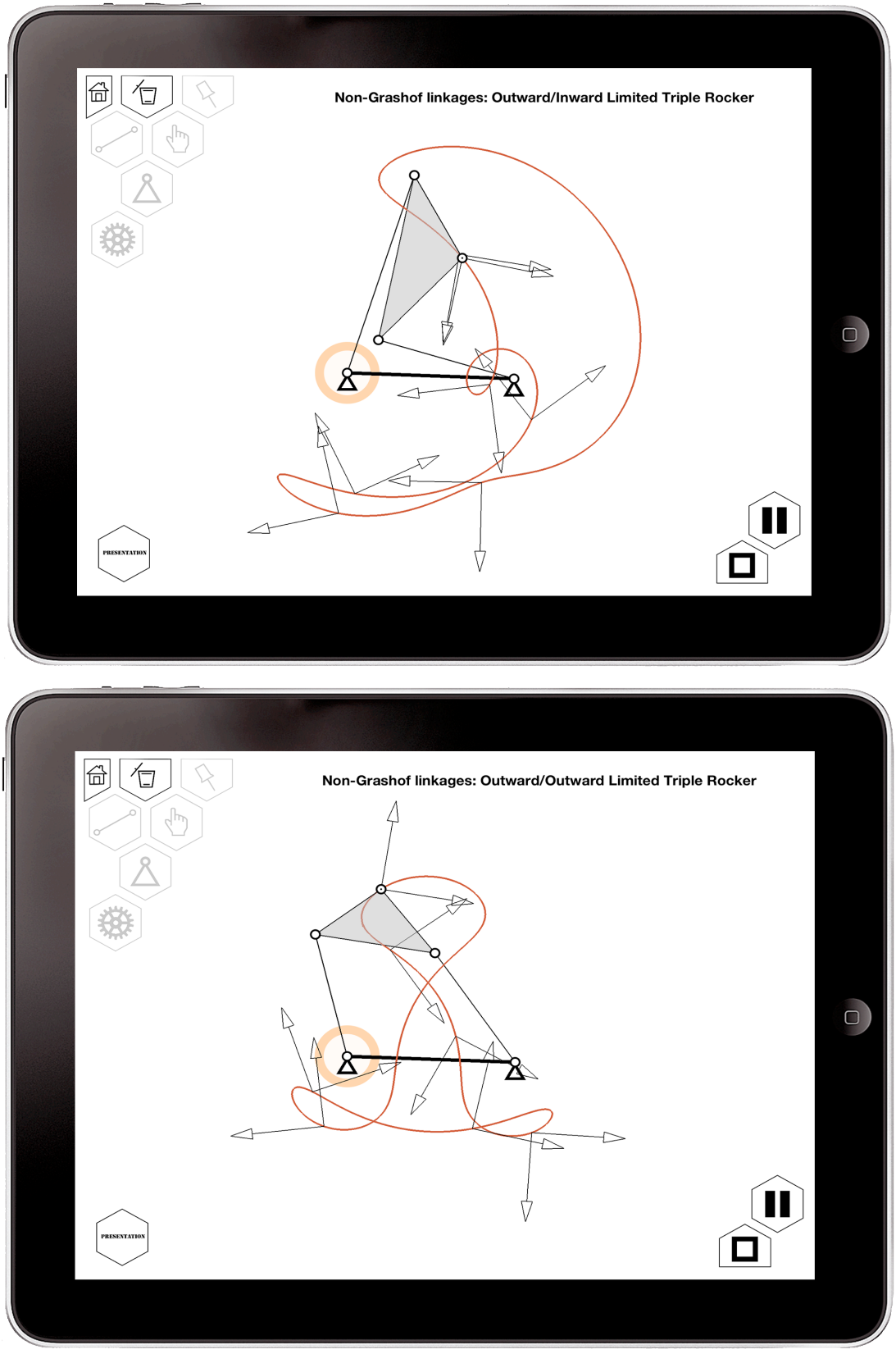


Figure 3.4: Screenshots of non-Grashof linkages categories

3.2.3. Algorithm

The following is the flowchart of the program in the simulation part.

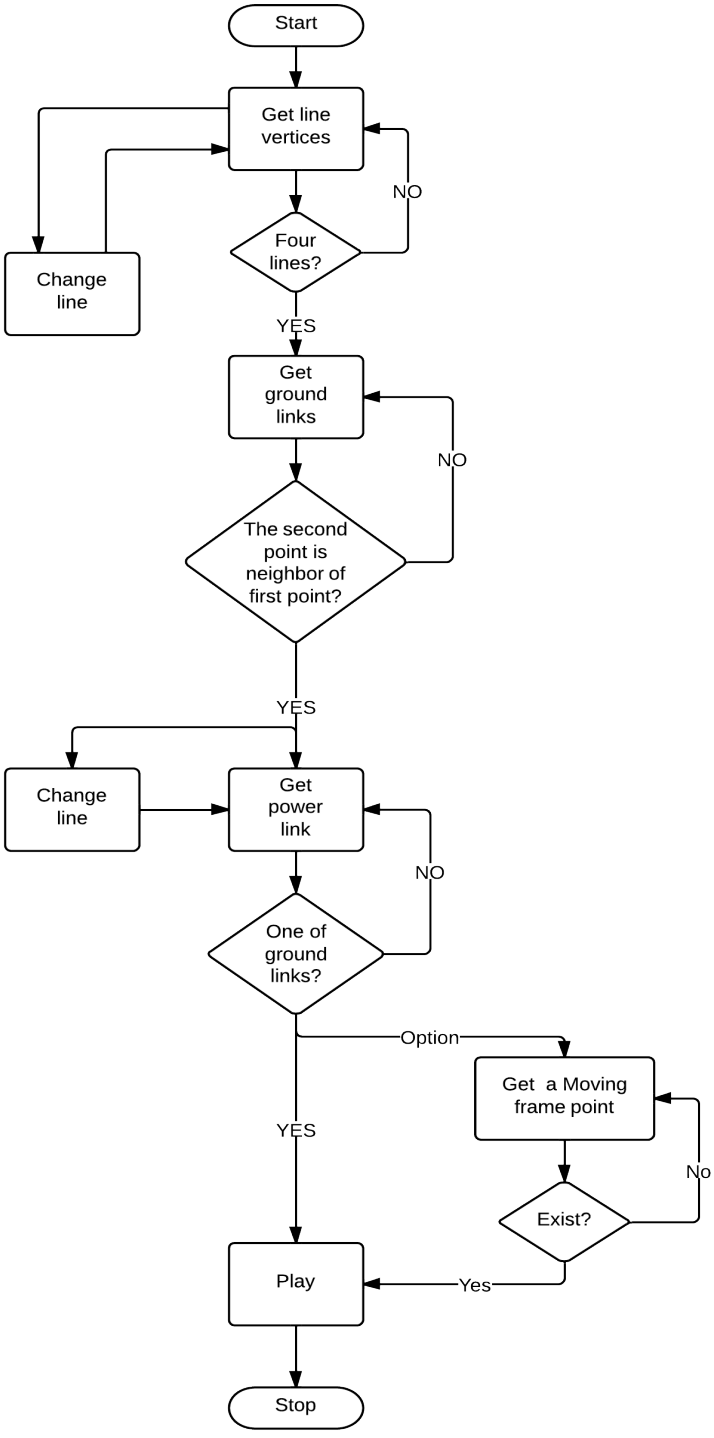


Figure 3.5: Flowchart of simulation

3.3 Synthesis

3.2.1 Overall construction

The aim of the synthesis is to allow users to give any five positions (more accepted in the future) and synthesize the four-bar mechanism that can go through these positions with both the coordinates and orientations matched.

Here is the screenshot of the synthesis screen.

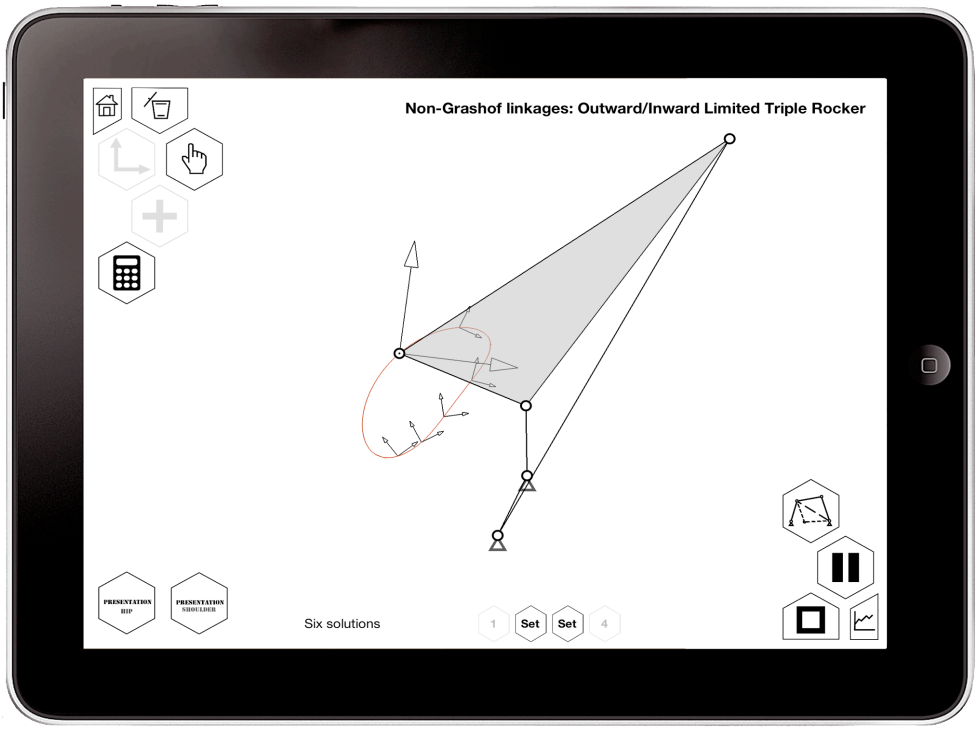


Figure 3.5: Screenshot of the synthesis part on iPad

Unlike the simulation, in synthesis users should provide the system the output positions only. With the latest theories, the system will calculate the given

information and shows the relative solutions, which would be six solutions, one solution or no solution. It can also give the necessary information of the mechanism on the screen.

3.2.2 UI design

Figure 3.5 shows a screenshot of the synthesis on iPad. The button design is the same as the one in simulation because of the consistency. The arrangement of the buttons is similar, although some buttons are changed and rearranged. A calculation button is added to pre-calculate the mechanism with the input given. A change circuit button is added to change the different circuit for some of the linkages have a circuit effect. A statistics button is added to show the users the relative length and coordinate information of the mechanism on the screen. A set of solution button is added to allow users to choose different solutions by pressing two of the four buttons for the condition that if there are six solutions. It also adds a label telling users how many solutions there are by the input they give.

The following is the specific functionalities of each of the button:

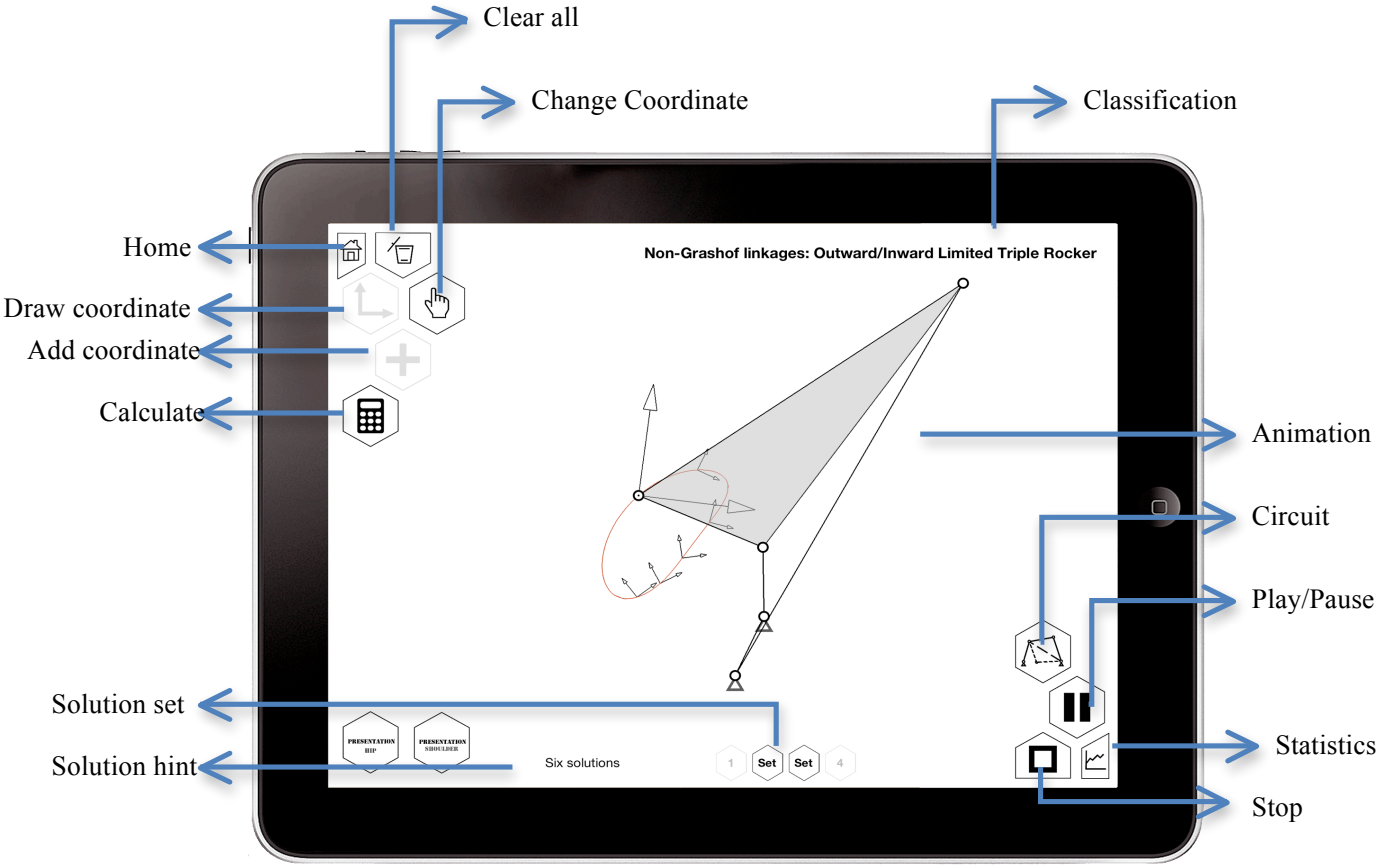


Figure 3.6: Buttons functionalities (synthesis)

When users press the statistics button on the right lower corner, a popover comes out. A popover is smaller and easier to have an overall view of the information shown than put them in a different screen. It saves both time and energy for the users.

The statistics screen shows the four bar synthesis data in relative units, including all the four bar length, fixed pivots coordinates and moving pivots position.

The next is the statistics screen:

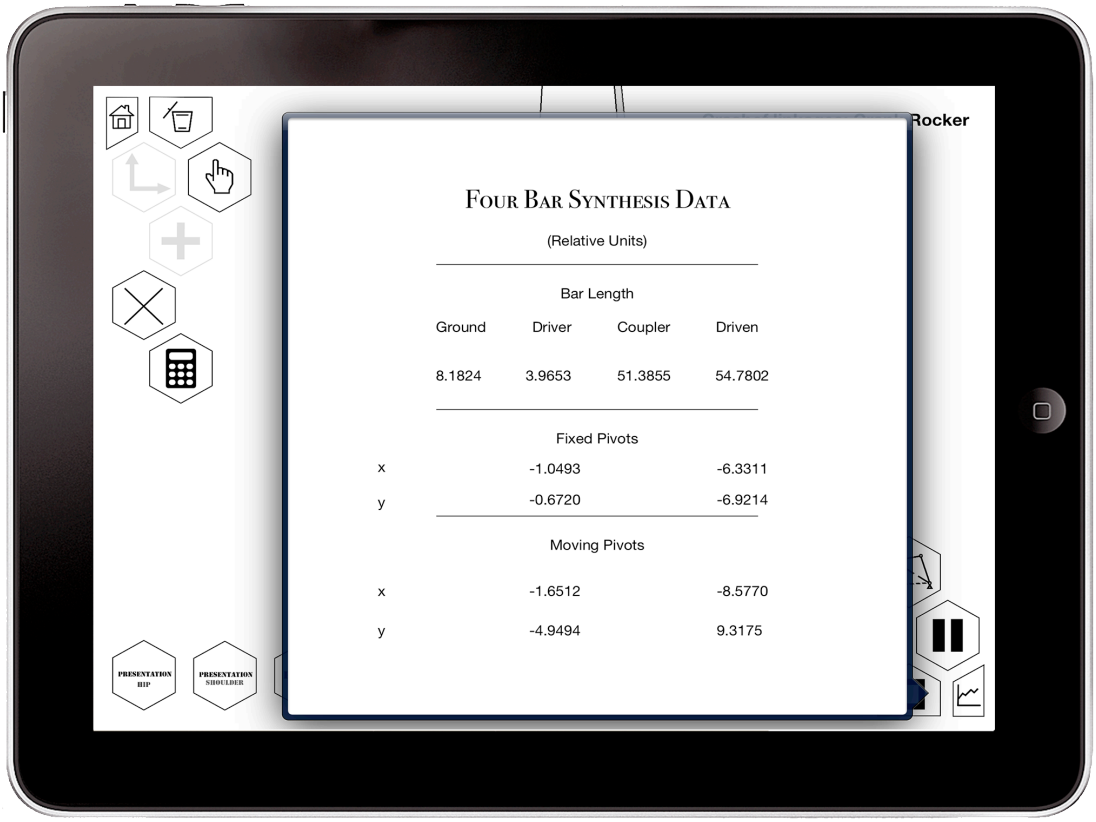
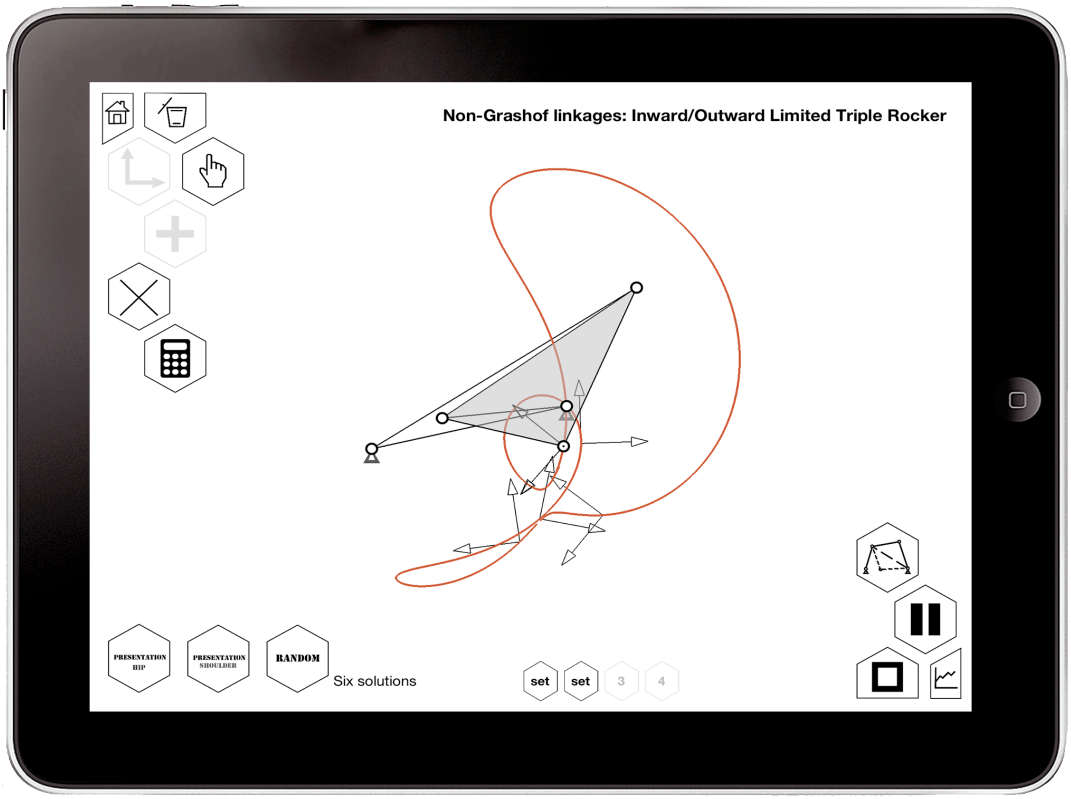
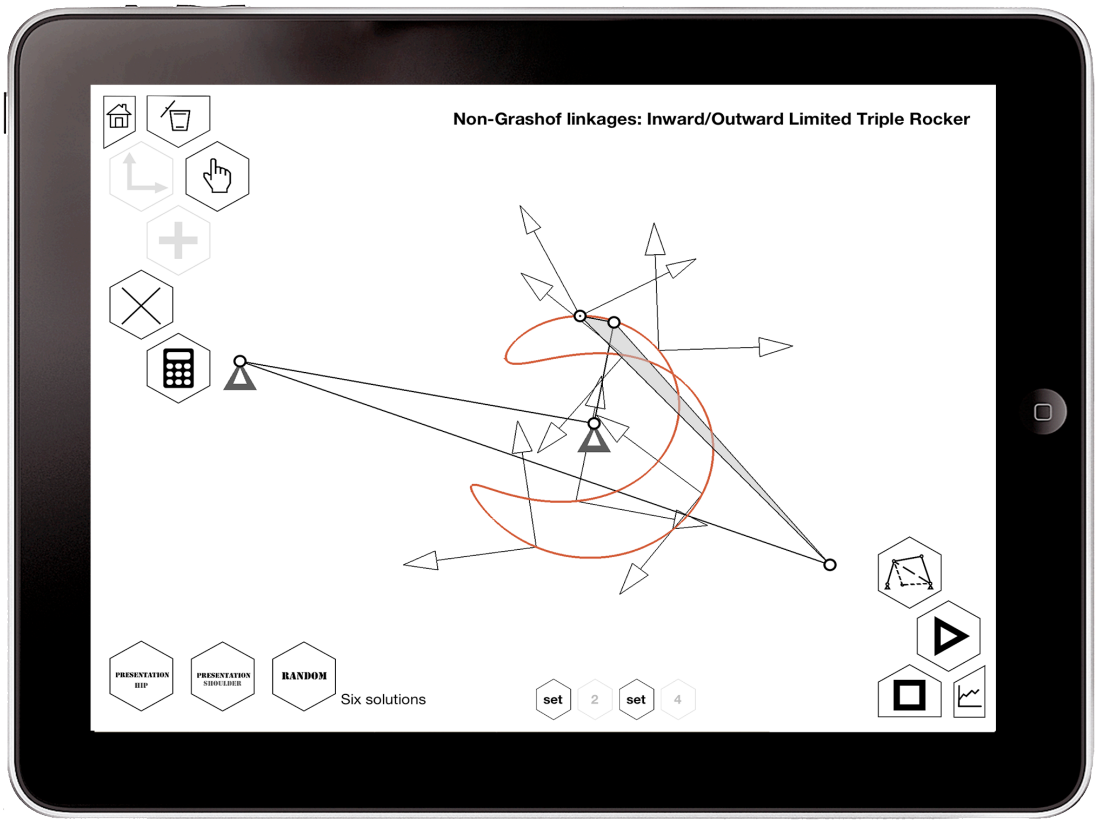
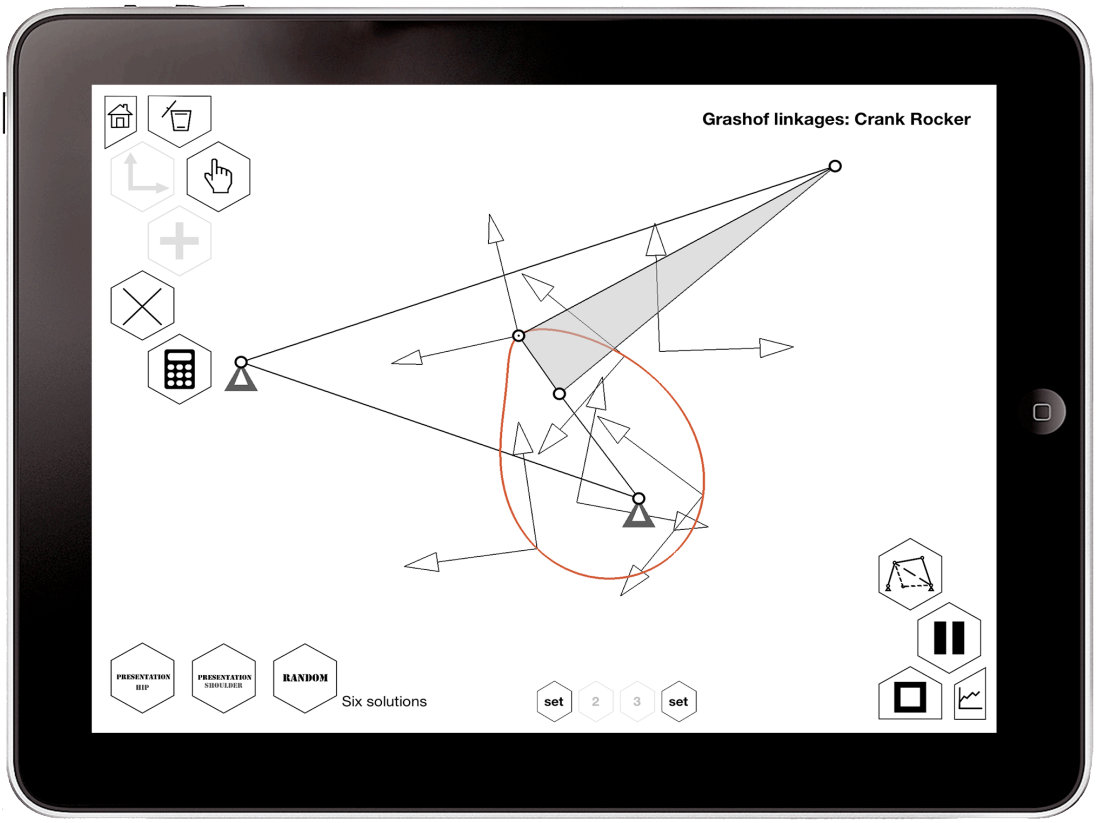


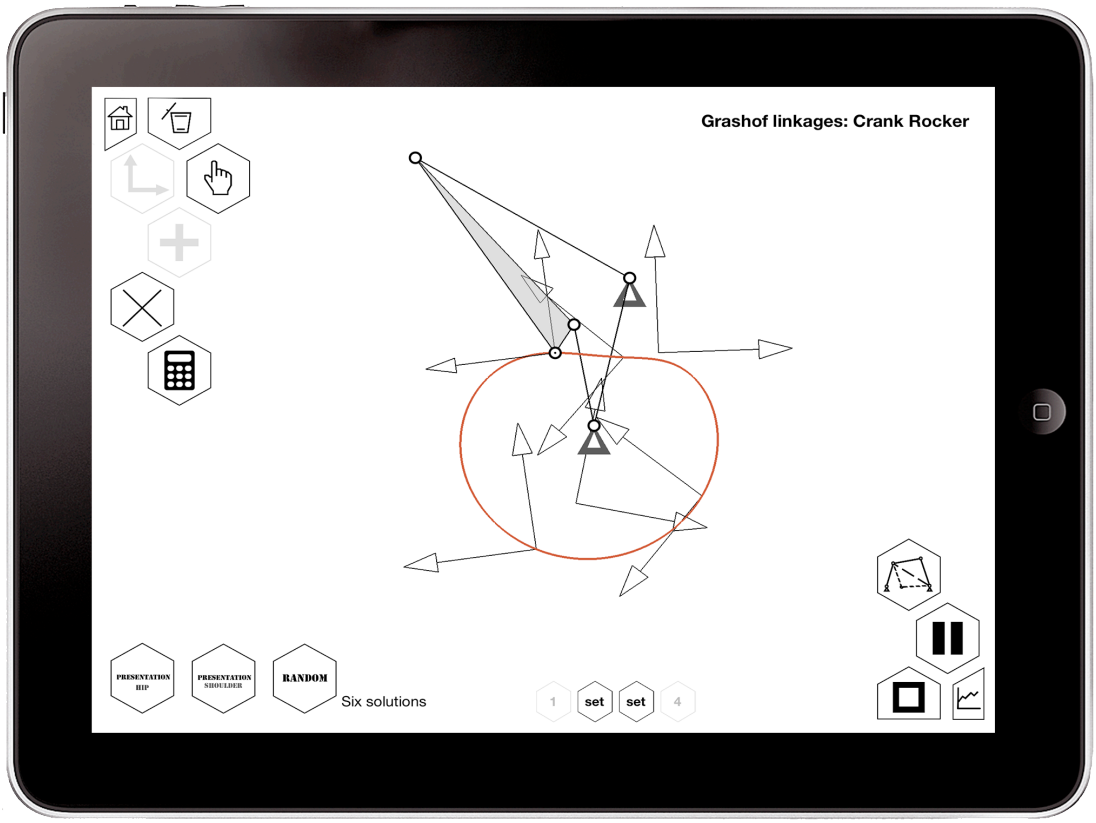
Figure 3.7: Statistics screen

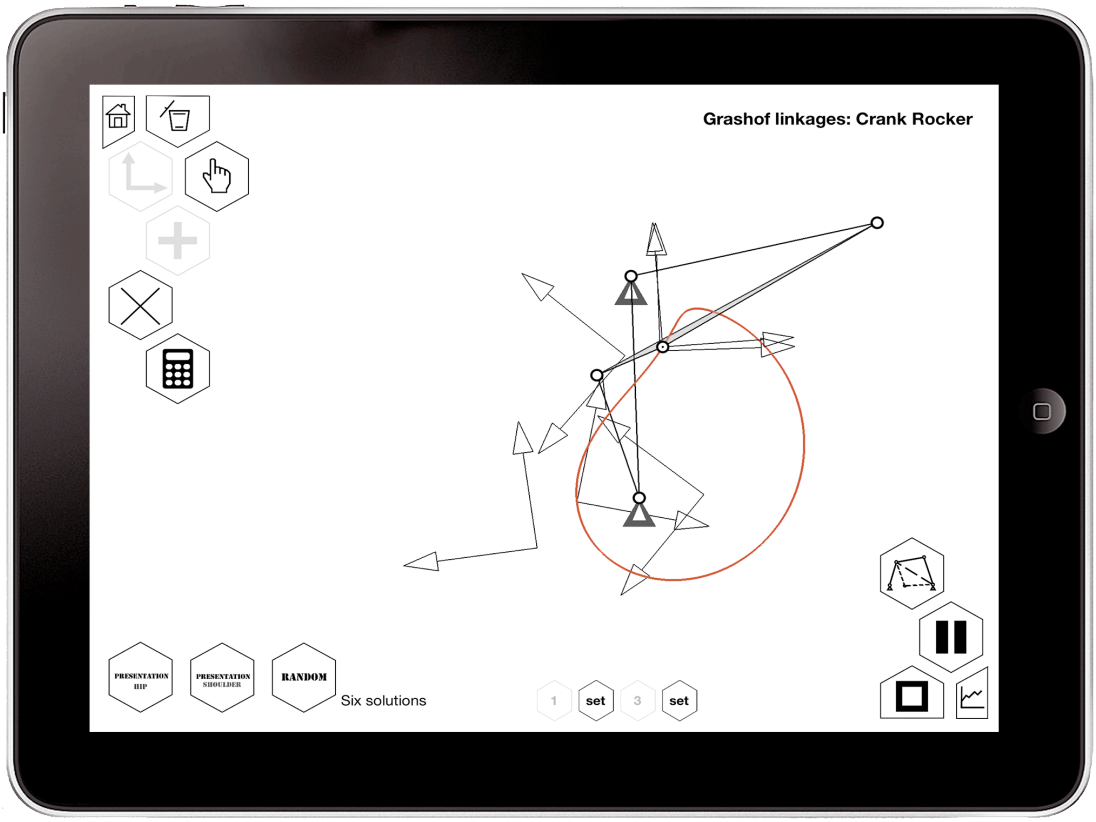
The solutions number can be six, one or zero. They are given as following:




Non-Grashof linkages: Inward/Outward Limited Triple Rocker
PRESENTATION HIP
PRESENTATION SHOULDER
RANDOM
Six solutions
set
2
set
4


Grashof linkages: Crank Rocker
PRESENTATION HIP
PRESENTATION SHOULDER
RANDOM
Six solutions
set
2
3
set


Grashof linkages: Crank Rocker
PRESENTATION HIP
PRESENTATION SHOULDER
RANDOM
Six solutions
1
set
set
4


Grashof linkages: Crank Rocker
PRESENTATION HIP
PRESENTATION SHOULDER
RANDOM
Six solutions
1
set
3
set

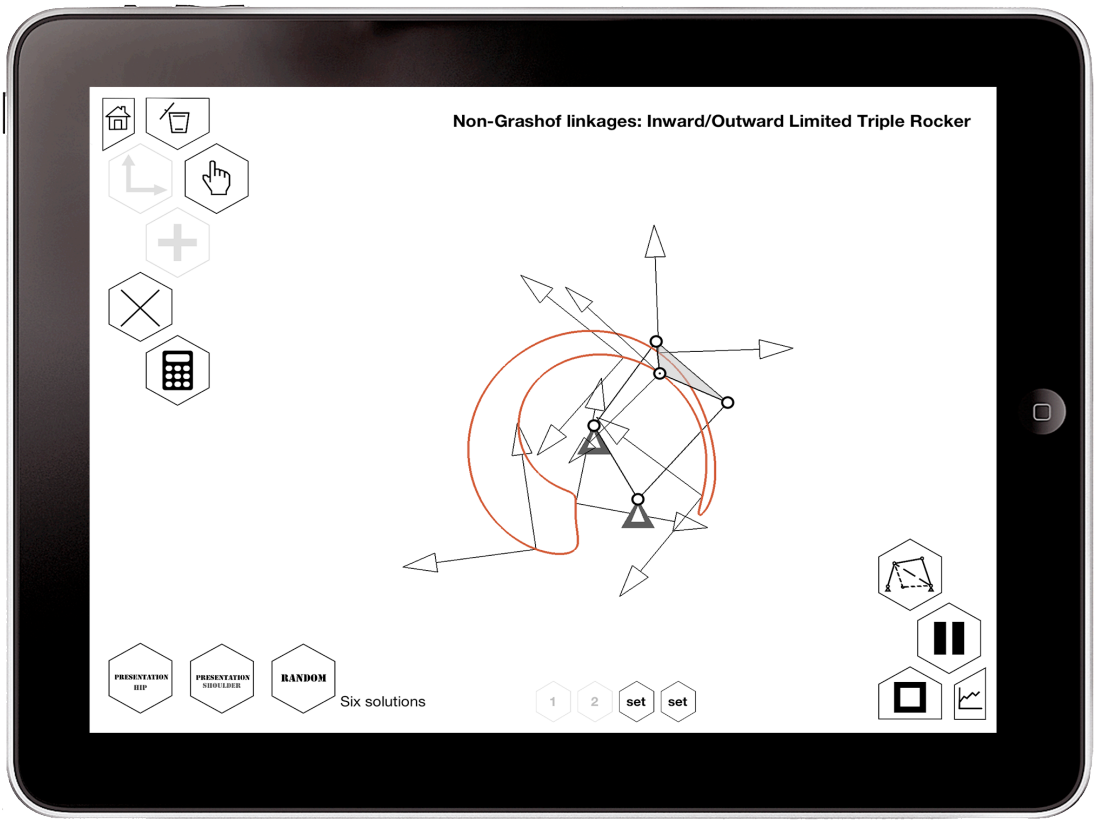


Figure 3.8: Screenshots of six solutions

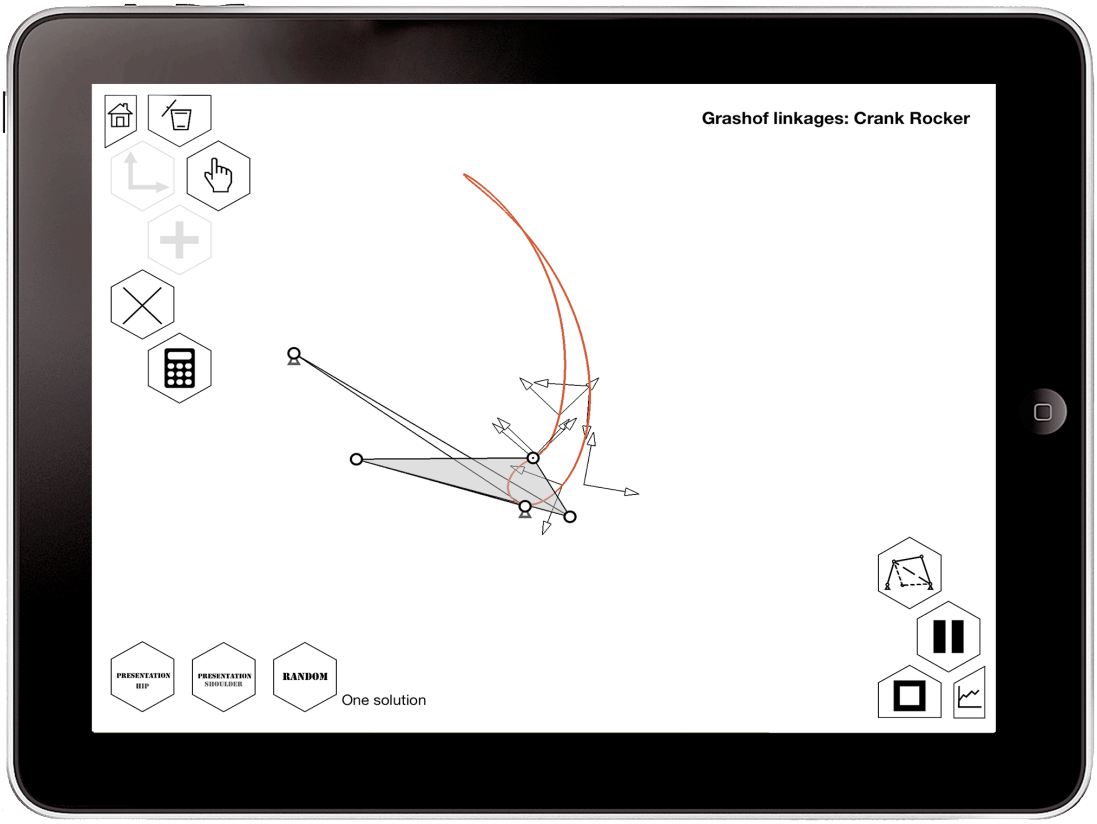


Figure 3.9: Screenshot of one solution

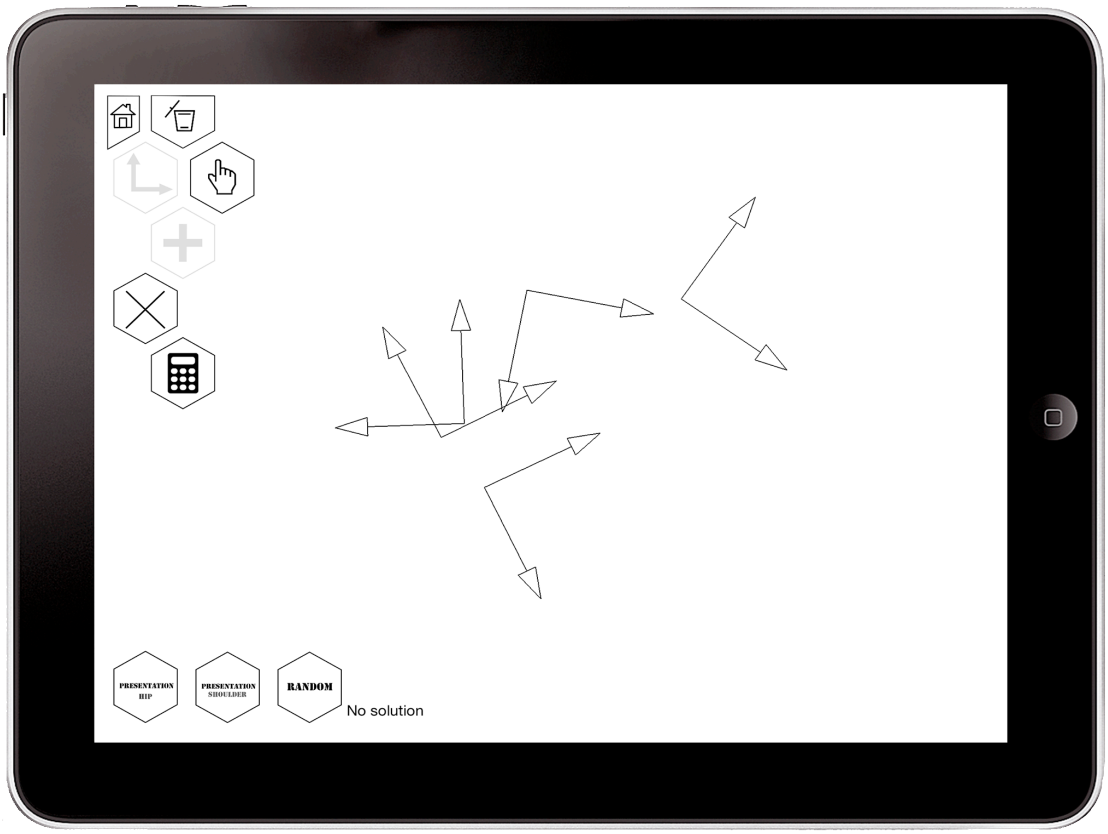


Figure 3.10: Screenshot of no solution

3.2.3 Algorithm

The following is the flowchart of the synthesis part of the program.

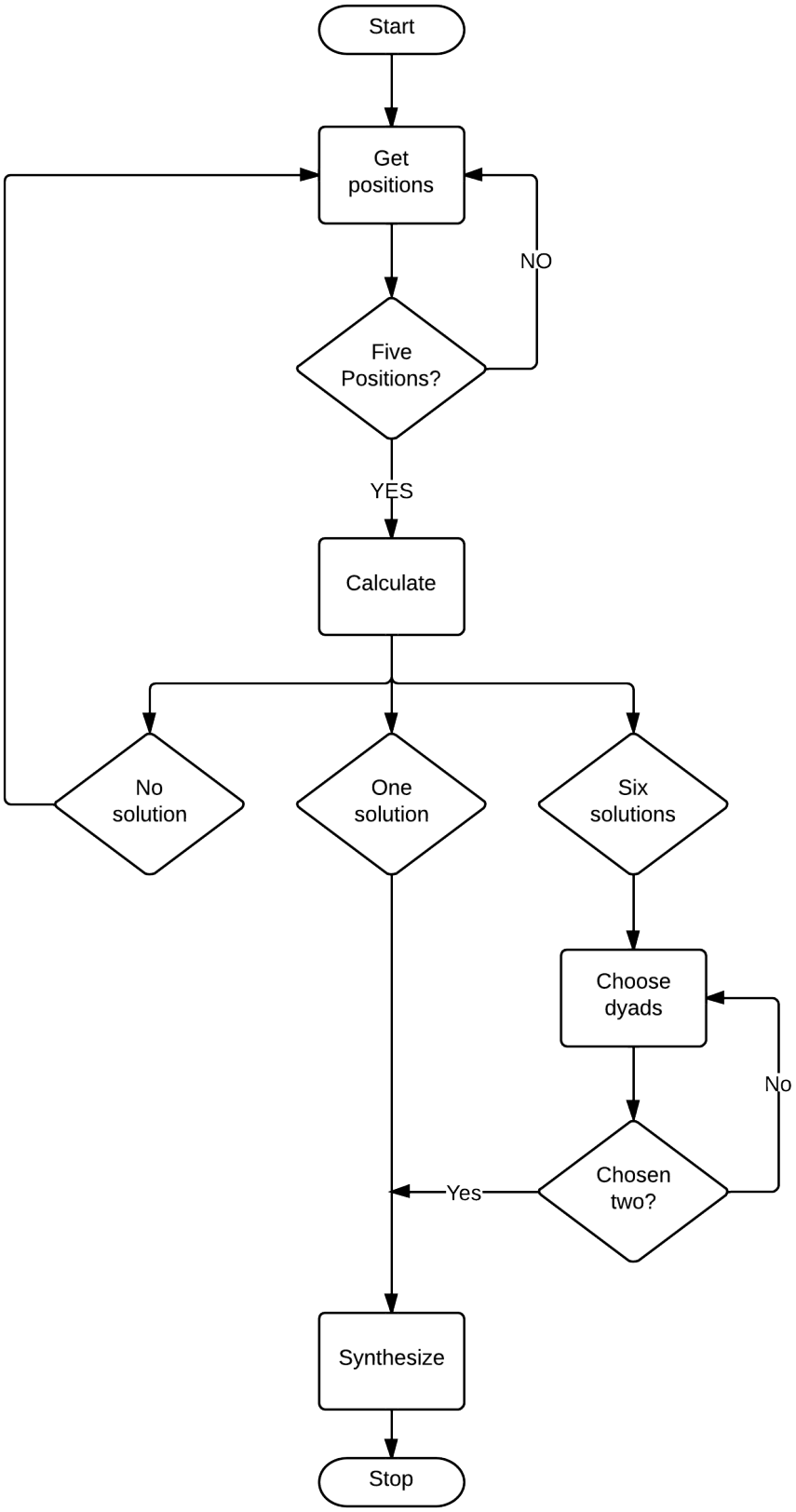


Figure 3.11: Flowchart of synthesis

Chapter 4

Problems Solved

4.1 Eigen Problem

In the main algorithm part of the program, it needs to calculate the eigenvectors of an 8 by 8 matrix. Xcode has an imbedded math library LAPACK (Linear Algebra PACKage) [17].

LAPACK was a Fortran 90 library and provided routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) can also be calculated, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all functions, they are suitable for real and complex matrices, in both single and double precision.

LAPACK is also extended to other machines with the names of ScaLAPACK, PLAPACK and CLAPACK.

An example can be shown as following of solving the eigenvector problem in Objective-C using LAPACK:

```
__CLPK_integer n = 8, lda = 8, info, lwork;
double wkopt;
double* work;
double w[8];
double A[8*8] = {
   2.3643, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
   4.5422, 5.5424, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
   0.5435, -5.4640, 3.4375, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
   4.3535,  -3.0982, 0.4387, 5.6655, 0.0000, 0.0000, 0.0000, 0.0000,
   -2.2223, 2.3937, 6.5943,  -3.4932, -9.0101, 0.0000, 0.0000, 0.0000,
   0.0011, -4.5443, 0.4446, 10.3445, 4.3322, 0.4456, 0.0000, 0.0000,
   9.0193, -8.4593, -9.4733, 2.3455, 8.3342, 4.5436, 9.3572, 0.0000,
   2.3822, 4.5443, 0.3532, -3.255, -3.4526, 4.5938, -3.3048, 4.5938
};
lwork = -1;
dsyev_( "Vectors", "Upper", &n, A, &lda, w, &wkopt, &lwork, &info );
lwork = (__CLPK_integer)wkopt;
work = (double*)malloc( lwork);
dsyev_( "Vectors", "Upper", &n, A, &lda, V, work, &lwork, &info );
```

dsyev_() is to calculate the eigenvalues and/or eigenvectors of a N by N

symmetric.

The problem occurred when the program is transferred from the Mac to the iPad. The eigenvectors are not the same when the eigenvalues are zeros. Several other solutions, such as solving SVD and using Jacobian method are tried. The results between Mac computer and the iPad mobile system are still not the same with eigenvalues of zero, even though it gives the same results with non-zero eigenvalues.

After checking the eigenvectors and calculating the fitting errors, it is found that the eigenvectors should be in the same vector space because the dot products of two of the eigenvectors are zero which means they are perpendicular to each other, the modulus of each is one meaning they are unit vectors. The results show that all of the eigenvectors are effective and give the right mechanism.

But in the future a better way of avoiding the zero value is on pursuit. One method is to transfer mathematically all the zero values into one to avoid floating point precision which needs to transfigure the original 8 by 8 matrix A.

4.2 Simulation Method

In the simulation part of the program, three algorithms had been used.

The first version was purely calculating the situations in each of the categories and depended on the position of the driver's bar. It extends to as many as almost 32 different conditions, which is quite complex and easy to make a mistake.

The second method is getting the intersecting points of the two circles path of

driver bar and driven bar go along with. This method asks for choosing the right intersecting point and ignoring another. On the extreme points of the moving paths, the positions of the two points are the same. When the mechanism moves backwards, the system can't choose another point as to make the coupler bar move a whole circle. The mechanism always goes through on a half path instead of a closed loop. Another problem of this method is also the complexity of the categorization.

The final algorithm in the program that is being used now is the Fourier Method which was mentioned before in the theoretical part of this thesis. This method uses the general four-bar linkages model so that the result can be used in all of the situations with only "one" calculation, although it does need to set the maximum and minimum angles in each category. The latter is very easy to get and can cause much less mistakes.

4.3 Multi-touch functionalities

iOS allows single finger and multi-finger touch screen functionalities. Users can use their fingers to touch, swipe, pinch, rotate and press in order to reach certain features. In this app, users need to use one finger to drag a line, to touch the screen to give the links and power positions, to give a moving frame coordinate. Users can also use two fingers to pinch to zoom in/out, two fingers to move to translate the canvas. In the future, it might be contains three finger touch or relevant functionalities.

It is not complex and works fine with a few multi-touch features. When the

touch-based functions become more complex and they interact with each other, the system may have a cognition problem. That is to say, when we have more than two multi-finger functionalities, they may interact with each other and give the wrong indication to the system.

For example, if we have a two finger touch functionality to realize the pinch to zoom in/out feature and we also have a two finger touch functionality to realize the move to translate the coordinate feature. When user uses two fingers to do the pinch, the system will sometimes judge it as a translation, vice versa. But this situation is not very obvious and it can be attributed to the sensitivity of the touch screen.

The program also has a way to avoid it or to make the use of it. One can add the following sentence to allow the simultaneous gesture recognition:

```
- (BOOL) gestureRecognizer: (UIGestureRecognizer *) gestureRecognizer
shouldRecognizeSimultaneouslyWithGestureRecognizer: (UIGestureRecognizer*)
otherGestureRecognizer {
    return YES;
}
```

The return can be NO in order to cancel the simultaneous gesture recognition.

Chapter 5

Summary

5.1 The extension of this app

This app can be used in class as educational software for presentation and instruction as a supplement of the textbook. The students can understand multiple four bar linkages visually and practically try by themselves. It can also extended into research assisting tool while users may use it to test their ideas efficiently, verify certain theories or just try to get some inspirations from it. Based on the applicability, the app may have a helpful role in the industry fields. It will help engineers visually understand and correct some design work benefited from our latest motion synthesis theories. Even for the common users, they can still gain much knowledge from our well-classified four bar linkages user interface.

5.2 The future development of this app

This app will include not only four bar linkages in the future. Focused on a big scale, it should be able to synthesize multi-bar linkages beyond four bars. Slider will be added in the future to complete all the situations in four bar linkages. It will be also able to synthesize positions more than five positions and less than five positions.

More features of the user interface will be added as the progress of the development goes. The graphics will be optimized. A spatial version of user interface might be coming in the future.

Reference

[1] Ravani, B., and Roth, B., 1983, "Motion Synthesis Using Kinematic Mappings," ASME J. Mech., Transm., Autom. Des., 105(3), pp.460-467.

[2] Ravani, B., and Roth, B., 1984, "Mappings of Spatial Kinematics," ASME J.Mech., Transm., Autom, Des., 106(3), pp. 341-347.

[3] Ge, Q.J. and Zhao, P. and Purwar, A. and Li, X. 2012. "A novel approach to algebraic fitting of a pencil of quadrics for planar 4R motion synthesis," ASME Journal of Computing and Information Science in Engineering, 12(4), p.041003.

[4] Li, X. and Zhao, P. and Ge, Q.J. 2012. "A fourier descriptor based approach to design space decomposition for planar motion approximation," ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference. Paper No. 2012-71264.

[5] McCarthy, J. M., 1990, *Introducation to Theoretical Kinematics*, MIT, Cambridge, MA.

[6] Analysts at Canalys. Available at http://www.canalys.com/newsroom/wintel-pc-market-share-set-fall-65-2013.

[7] Chitra Sethi, ASME.org. Available at http://www.asme.org/kb/news---articles/articles/technology-and-society/mobile-apps-for-engineers--what's-in-store-.

[8] Nancy Giges, ASME.org. Available at http://www.asme.org/kb/news---articles/articles/technology-and-society/engineering-design-apps--a-work-in-progress.

[9] AppleInsider staff. Available at http://appleinsider.com/articles/13/01/25/apples-iphone-grew-to-251-global-market-share-in-2012.

[10] TIOBE Programming Community Index for April 2013. Available at http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

[11] Ray, J., 2013, *Sams Teach Yourself iOS 6 Application Development in 24 Hours (4th Edition)*, Pearson Education, Inc.

[12] SoftIntegration Inc: Ch Mechanism Toolkit. Available at http://www.softintegration.com/chhtml/toolkit/mechanism/fourbar.

[13] Burmester Theory. Available at https://en.wikipedia.org/wiki/Burmester's_theory.

[14] Joshua Porter: "Principles of User Interface Design". Available at http://bokardo.com/principles-of-user-interface-design.

[15] Christopher Diggins: "The Principles of Good Programming". Available at http://www.artima.com/weblogs/viewpost.jsp?thread=331531.

[16] LAPACK. Available at http://www.netlib.org/lapack/.