OptiLift

Tietotekniikan Sovellusprojekti

Lauri Laasala
Olli Lukkarinen
Ville Räisänen
Vesa Tanhua-Tyrkkö

Sovellussuunnitelma
Versio 1.0
20.4.2004

Jyväskylän yliopisto
Tietotekniikan laitos

Tekijät: Laasala Lauri, lvlaasal@cc.jyu.fi, 044-3375831
Lukkarinen Olli, olliluk@cc.jyu.fi, 040-7668234
Räisänen Ville, vtraisan@cc.jyu.fi, 044-3060999
Tanhua-Tyrkkö Vesa, vttanhua@cc.jyu.fi, 050-3475670

Projektin tiedot: OptiLift-projekti
Projektitila: Agora C225.3, 014-260 4971
Kotisivu: http://sovellusprojektit.it.jyu.fi/optilift/

Työn nimi: OptiLift-projektin sovellussuunnitelma.

Työn kuvaus: Sovellussuunnitelma tietotekniikan Sovellusprojektiin.

Tilaaja: Kilpa- ja huippu-urheilun tutkimuskeskus KIHU.

Teettäjä: Jyväskylän yliopisto, tietotekniikan laitos.
Vastaava ohjaaja: Markus Inkeroinen.
Tekninen ohjaaja: Ville Tirronen.

Tiivistelmä: Tämä dokumentti on Jyväskylän yliopiston keväällä 2004 toteutettavan OptiLift-Sovellusprojektin sovellussuunnitelma. Dokumentissa kuvataan kuinka vaatimusmäärittelyssä esitetyt vaatimukset sovelluksessa toteutetaan. Käsiteltäviä asioita ovat arkkitehtuuri, rakenne ja toteutustavat.

Avainsanat: KIHU, kilpa- ja huippu-urheilun tutkimuskeskus, käyttöliittymä, luokkakaavio, olio, OptiLift, sovellusprojekti, sovellussuunnitelma.

Versiohistoria

Versio	Päiväys	Tehnyt	Muutokset
0.1	18.2.2004	Ville Räisänen, Vesa Tanhua-Tyrkkö	Ensimmäinen versio.
0.2	20.2.2004	Ville Räisänen, Vesa Tanhua-Tyrkkö	Dokumentin rungon tekeminen, ulkoasun muokkaaminen.
0.3	8.3.2004	Ville Räisänen	Lisätty kuva arkkitehtuurista ja luokkakuvauksia.
0.4	16.3.2004	Ville Räisänen, Vesa Tanhua-Tyrkkö	Tarkennettu luokkakuvauksia.
0.5	18.3.2004	Ville Räisänen, Vesa Tanhua-Tyrkkö	Lisätty kuvia käyttöliittymästä, lopullinen luokkakaavio ja luokka-kuvauksia. Säädetty hiukan ulkoasua.
0.6	22.3.2004	Ville Räisänen	Muokattu kuvia.
0.7	23.3.2004	Ville Räisänen, Vesa Tanhua-Tyrkkö	Ulkoasua säädetty ja korjattu käyttöliittymän kuvauksia.
0.8	24.3.2004	Ville Räisänen, Vesa Tanhua-Tyrkkö, Olli Lukkarinen	Lisätty ulkoisten komponenttien kuvaukset, analyysiosio ja kuvia käyttöliittymästä. Korjattu virheitä.
0.9	29.3.2004	Ville Räisänen, Vesa Tanhua-Tyrkkö, Olli Lukkarinen	Analyysiosaa muokattu, muutettu lukuihin jako 3-tasoiseksi.
0.10	31.3.2004	Ville Räisänen, Olli Lukkarinen	Korjattu kirjoitusvirheitä, muokattu ja lisätty kuvia, analyysiosaa tarkennettu ja korjattu, yleistä viimeistelyä.
1.0	14.4.2004	Ville Räisänen, Olli Lukkarinen	Viimeistelyä, kuvia ja analyysiosa korjattu.
1.0	15.4.2004	Ville Räisänen, Olli Lukkarinen	Korjattu löydetyt virheet.

Sisältö

1 Johdanto

OptiLift-projekti suunnittelee ja toteuttaa Kilpa- ja huippu-urheilun tutkimuskeskukselle painonnoston levytankoharjoitteluun nostotekniikan automaattisen mittaus- ja analysointisovelluksen. Järjestelmän avulla analysoidaan urheilijan levytankoharjoittelua, joka on olennainen osa monien eri urheilulajien voimaharjoittelua. Oikean tekniikan hallitseminen on tärkeää niin vammattoman kuin mahdollisimman tehokkaankin harjoittelun kannalta. Monilla valmentajilla ei kuitenkaan ole riittävästi tietoa oikeista nostotekniikoista, jotta mahdollisimman hyödyllisen palautteen antaminen olisi mahdollista [7].

Projekti toteutetaan Jyväskylän yliopiston tietotekniikan sovellusprojektina kevään 2004 aikana. Projektiin liittyvistä käytännöistä ja toimintatavoista sekä projektiin kuuluvista henkilöistä on kerrottu tarkemmin projektisuunnitelmassa [5].

Tässä dokumentissa kuvataan sitä, miten vaatimusmäärittelyssä [6] esitetyt vaatimukset toteutetaan. Vaatimuksista on huomioitu lähinnä ensimmäinen prioriteettiluokka, eli vaatimukset, jotka sovelluksen on toteutettava. Luvussa 2 selitetään sovelluksen toteutukseen liittyviä keskeisiä termejä. Luvussa 3 käydään läpi sovelluksen yleinen arkkitehtuuri. Luvussa 4 on kuvattu sovelluksen käyttöliittymän toimintaa erilaisissa käyttötapauksissa. Luvussa 5 kuvataan sovelluksen rakenne luokkakaavion ja luokkajakojen avulla. Luvussa 6 kerrotaan ohjelmakoodin kommentoinnista sekä muuttujien nimeämisestä, ja luvussa 7 sovelluksessa datan tallennukseen käytettävistä tiedostoista. Luvussa 8 on kuvattu muuttujien analysointia, ja luvussa 9 mainitaan sovelluksen testaamisesta.

2 Termistöä

Tässä luvussa selitetään yleisimmät sovelluksen toteutukseen liittyvät termit.

AVI	Microsoftin tiedostomuoto äänelle ja liikkuvalle kuvalle. Tiedostot ovat yhteensopivia sekä PC:lle että Applen Macintoshille. AVIin sisältyy CODEC-tiedosto, jota tarvitaan videotiedostoja käyttävissä ohjelmissa tiedon pakkaukseen ja purkuun.
Delphi	Ohjelmistotalo Borlandin olioperustaiselle Pascal-ohjelmointikielelle suunnittelema ohjelmistokehitin, jolla tuotetaan valmiiksi käännettyjä ohjelmia.
Dialogi	Keskusteluikkuna, valintaikkuna. Graafisissa käyttöliittymissä ikkuna, johon täydennetään ohjelman tarvitsemat tiedot.
Form(i)	Lomake, sivu, graafisessa ohjelmoinnissa visuaalinen tapa esittää yksi ohjelman ikkuna [8].
Frame	Videoleikkeen tai animaation yksi kuvaruutu.
Freeware	Ilmaisohjelmat. Tietokoneohjelmia, joita voi vapaasti jakaa. Oikeuden omistaja ei vaadi ohjelman käyttämisestä mitään korvausta. Freeware-ohjelmat ovat kuitenkin tekijänoikeudellisesti suojattuja, vaikka korvausta ei vaadita. Käyttöoikeuksia koskevat ehdot esitetään tavallisesti ohjelman käynnistyksen yhteydessä [4].
Koodekki	Ohjelma tai laite, jota käytetään informaation pakkaamiseen ja purkamiseen. (engl. *code & decode= codec*).
MDI	Suom. moniasiakirjaliittymä. Tekniikka sovelluksen ikkunoinnin ja asiakirjojen hallintaan [8]. (engl. *Multiple Document Interface*).
Olio	Luokan ilmentymä.
Säiliöluokka	Jonkin luokan olioiden tallettamiseen tarkoitettu luokka.

3 Yleisarkkitehtuuri

Kuvassa 3.1 esitetään järjestelmän yleinen rakenne. Järjestelmä koostuu tietokoneeseen asennetusta sovelluksesta ja sen käyttämistä tiedostoista sekä tietokoneeseen ja sovellukseen liitetystä digitaalisesta videokamerasta. Käyttäjä käyttää sovellusta käyttöliittymän kautta. Sisäinen toteutus ja käyttöliittymän toteutus on selkeästi erotettu toisistaan.

3.1 Käyttöliittymä

Käyttöliittymässä on erilaisia näkymiä käyttötapausten mukaan.

Kuvalistanäkymästä hallitaan suorituksesta tulostettavaa kuvalistaa.

Videonäkymässä on videokuvanäyttö, jolla näytetään videota ja liikeratakäyrää.

Analysointitulosnäkymässä näkyvät nostosuorituksen analysoidut muuttujat graafisina kuvaajina ja numeerisina arvoina taulukossa.

Kerrallaan voidaan näyttää kahta suoritusta, jolloin sekä videokuvanäyttöjä että graafisia kuvaajia on kaksi kappaletta.

Suoritusryhmänäkymästä hallitaan suoritusryhmää, käyttäjärekisteriä sekä nauhoituksen ja vertailun käynnistämistä ja lopettamista.

Käyttöliittymän toteutusta on kuvattu tarkemmin luvussa 4.

3.2 Sisäinen toteutus

Sisäisen toteutuksen loogisia osia ovat videon- ja kuvankaappaus, analysaattori ja käyttäjärekisteri.

Videon- ja kuvankaappausosan tehtäviä ovat videonpätkien ja yksittäisten kuvien kaappaaminen videokameran välittämästä kuvasta, liikeratakäyrän tunnistaminen sekä videoiden ja käyrän tallentaminen tiedostoon.

Analysaattori laskee analysoitavat biomekaaniset muuttujat liikeratakäyrän sekä noston ja nostajan parametrien perusteella.

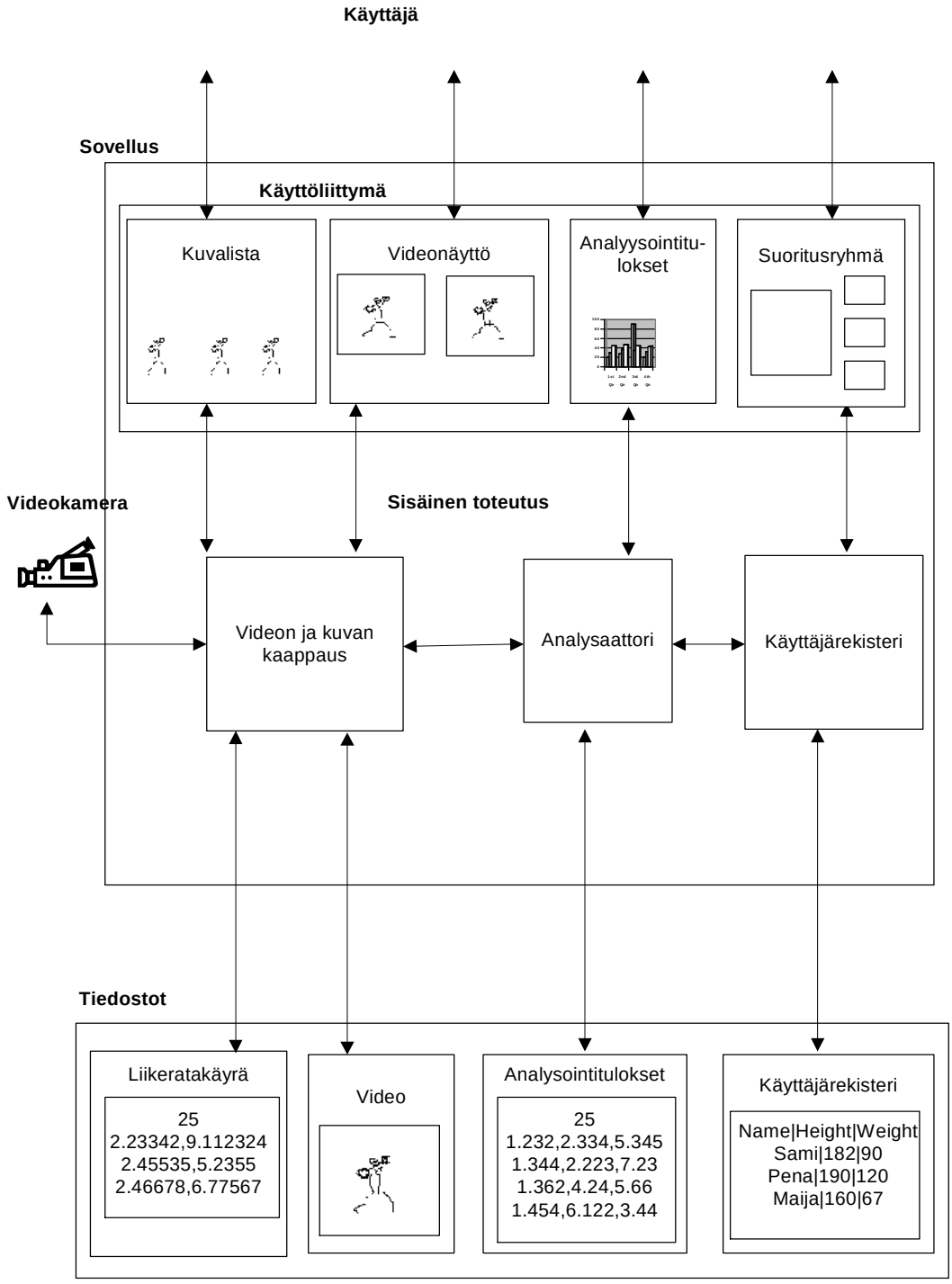
Käyttäjärekisteri lukee ja tallettaa käyttäjiä rekisteritiedostoihin sekä tallettaa ajonaikaisesti tiedot suoritusryhmästä.

Sovelluksen sisäiseen toteutukseen perehdytään tarkemmin luvussa 5

3.3 Tiedostot

Kuvan 3.1 alalaidassa on esitelty tiedostot, joita sovellus käyttää. Niitä ovat liikeratakäyrä, video, analysointitulokset ja käyttäjärekisteri.

Tiedostoista on kerrottu enemmän luvussa 7.



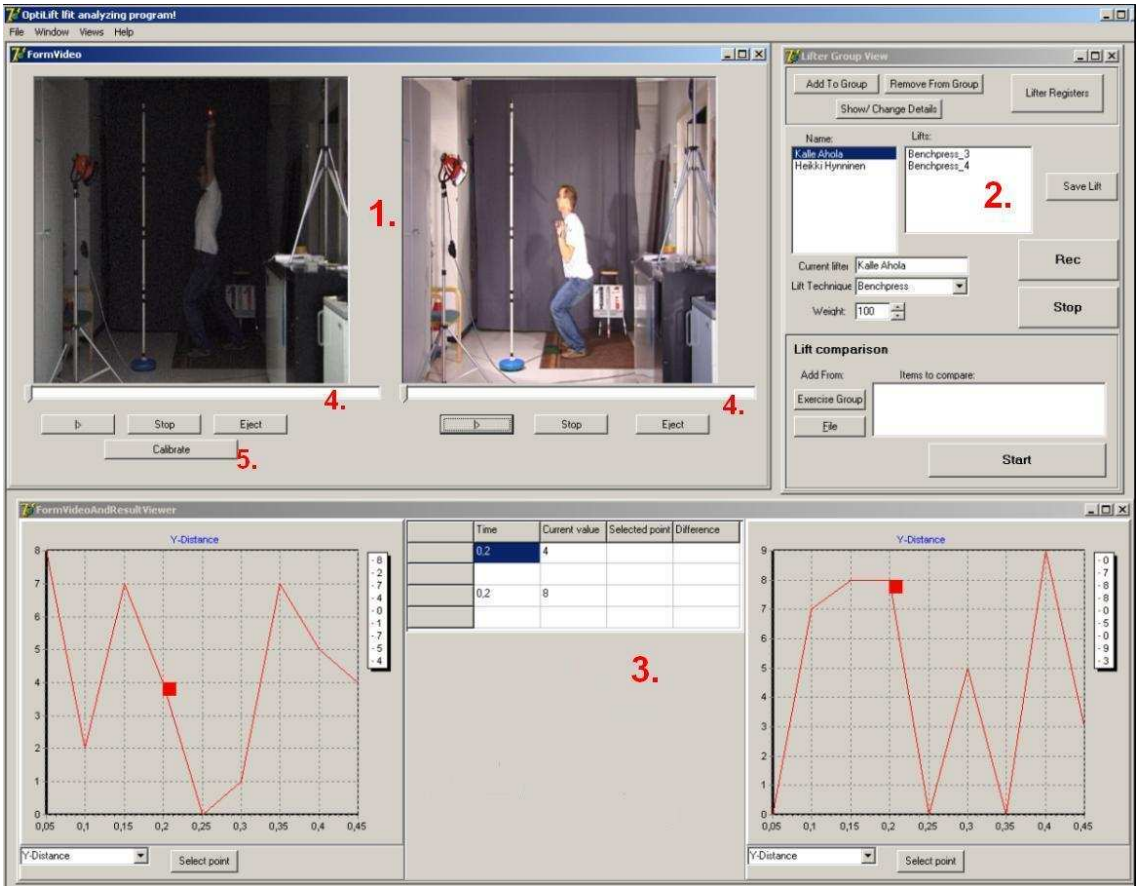
Kuva 3.1: Sovelluksen arkkitehtuuri.

4 Käyttöliittymäkuvaus

Tässä luvussa kuvataan sovelluksen käyttöliittymän toteutusta havainnollistavien kuvien ja niiden selitysten avulla. Kussakin aliluvussa on esitetty yhden käyttötapauksen toteuttaminen käyttöliittymän avulla. Käyttöliittymän suunnittelun lähtökohtana on helppokäyttöisyys ja havainnollisuus. Koska sovellus suunnitellaan erityisesti käytettäväksi kannettavissa tietokoneissa, pitää käyttöliittymän toteutuksessa varautua erilaisiin näytön resoluutioihin. Käyttöliittymä toteutetaan kokonaisuudessaan englanniksi.

Käyttöliittymä toteutetaan käyttäen MDI-tekniikkaa. Siinä yksi sovelluksen ikkunoista on pääikkuna, johon sijoitetaan kaikki menut ja joka toimii kehyksenä ali-ikkunoille. Ali-ikkunat ovat pääikkunalle alisteisia. Tämä tarkoittaa sitä, ettei niitä voi liikuttaa pääikkunan asettamien aluerajojen ulkopuolelle. Ikkunoita pystytään näin hallitsemaan kootummin ja niiden katoamisilta toisten ikkunoiden alle vältytään. MDI-tekniikkaa käytetään tavallisesti moniasiakirjasovelluksissa, joissa tarvitaan usean asiakirjan samanaikaista käsittelyä. Tätä ominaisuutta ei sovelluksessa tarvita, mutta MDI-tekniikka on käyttökelpoinen myös sovelluksen ikkunointiin.

Kuvassa 4.1 on esitetty alustava näkemys sovelluksen käyttöliittymästä. Sen osat ovat videonäyttö (1), suoritusryhmänäkymä (2) ja analyysitulosnäkymä (3).

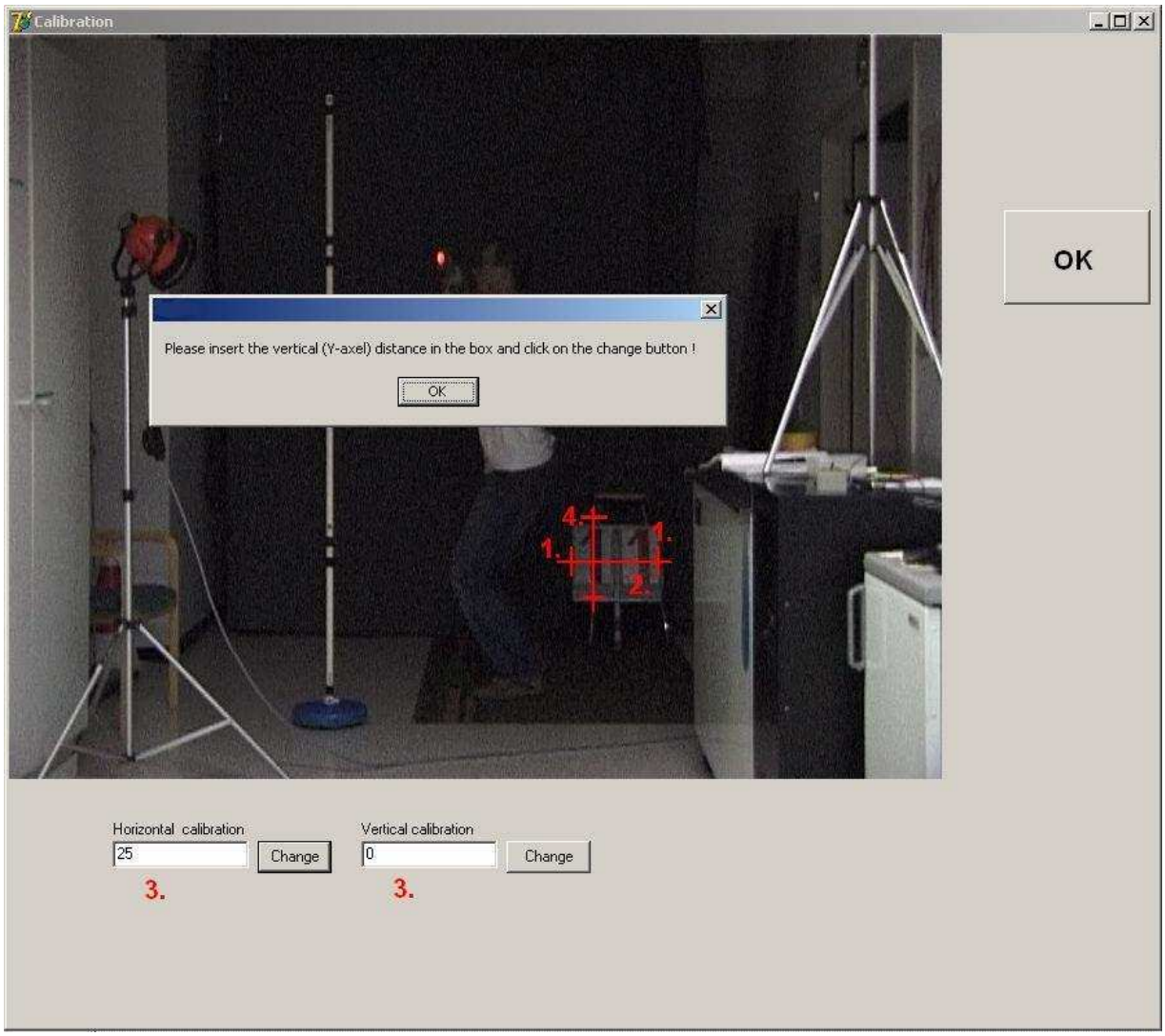


Kuva 4.1: Sovelluksen käyttöliittymä.

4.1 Kalibrointi

Kalibroinnissa käyttäjä asettaa ensin mitoiltaan tunnetun esineen kuvattavalle alueelle. Esine tulisi asettaa mahdollisimman keskelle kuvattavaa aluetta, jotta kalibrointi olisi tarkka. Tämän jälkeen käyttäjä valitsee videonäytöstä kalibroinnin (kuva 4.1 – (5)), jolloin kuvattava alue tulee näkyviin sovelluksen kalibrointi-ikkunassa, joka on esitetty kuvassa 4.2. Kalibrointi suoritetaan erikseen x- ja y-suunnassa. Ensin käyttäjä valitsee kuvasta esineen laidoilta vaakatasossa eli x-suunnassa kaksi pistettä (1). Sovellus piirtää kuvaan viivan pisteiden välille (2) ja antaa käyttäjälle mahdollisuuden hienosäätää pisteiden paikkaa. Kun pisteet on valittu käyttäjää tyydyttävällä tavalla, syöttää käyttäjä sovellukselle pisteiden välisen todellisen etäisyyden (3). Seuraavaksi sama suoritetaan

pystytasossa eli y-suunnassa (4). Näiden tietojen perusteella sovellus saa tietoonsa kuvan mittasuhteet, joita tarvitaan analysoitaessa suorituksista haluttuja muuttujia. Kalibroinnin yhteydessä sovellus selvittää myös ledin mittasuhteet ja ilmoittaa käyttäjälle suoritusten analysoinnin mittaustarkkuudet.

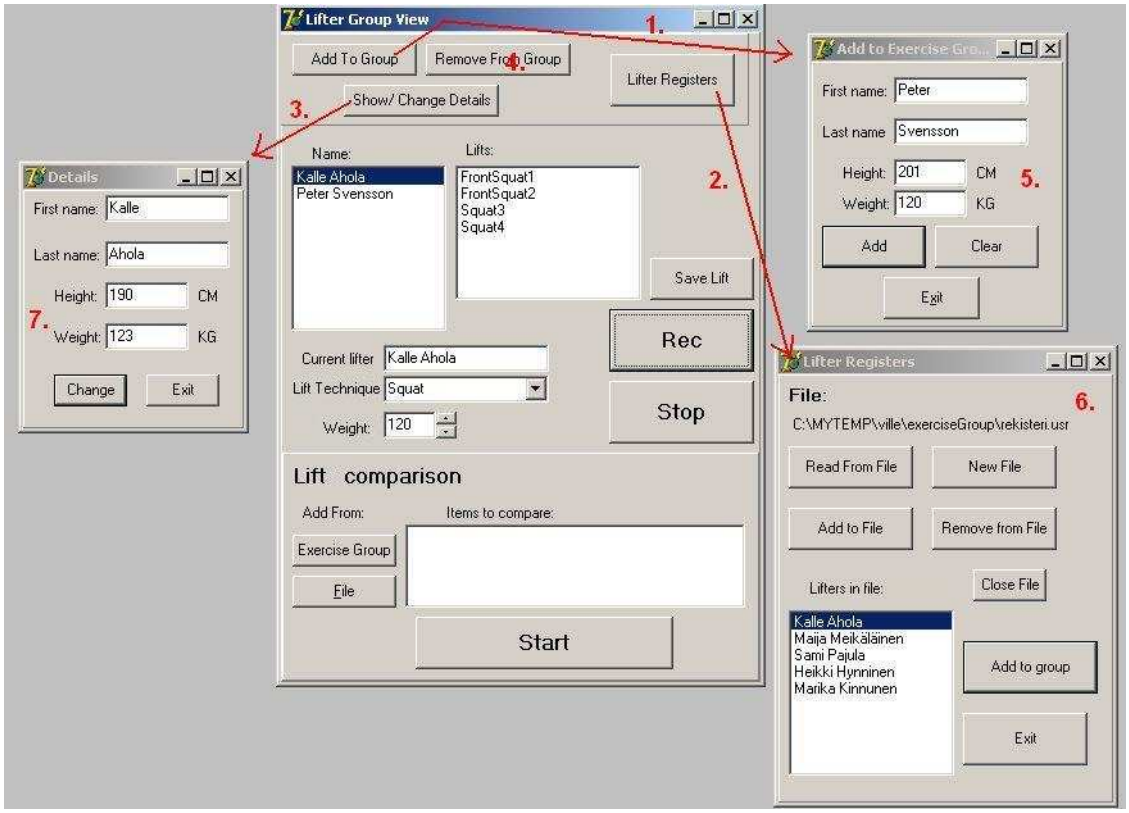


Kuva 4.2: Kalibrointi-ikkuna.

4.2 Suoritusryhmän luominen ja hallinta

Suoritusryhmää hallitaan suoritusryhmänäkymän ja sen ali-ikkunoiden avulla, jotka on esitetty kuvassa 4.3. Suoritusryhmää ei varsinaisesti luoda tai tuhota, koska sitä ei tallenneta tiedostoon kuten käyttäjärekisteri tallennetaan. Suoritusryhmä on siis ajonaikainen tietorakenne, johon voidaan lisätä uusia nostajia syöttämällä heidän tietonsa (1), tai lisäämällä heitä käyttäjärekisteritiedostosta (2). Suoritusryhmään lisättyjen nostajien tietoja voidaan muuttaa (3), tai heitä voidaan poistaa suoritusryhmästä (4). Suoritusryhmässä olevia nostajia voidaan lisätä käyttäjärekisteritiedostoon, jolloin nostajien tiedot voidaan säilyttää käyttökertojen välillä.

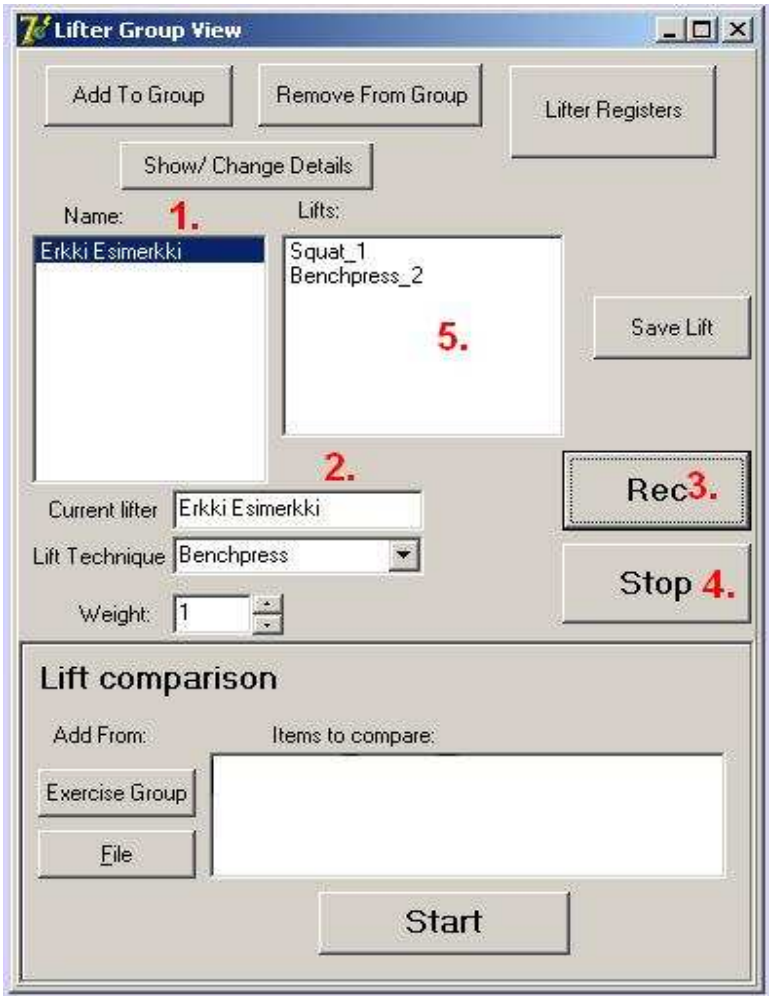
Ali-ikkunoita ovat uuden nostajan lisäämisikkuna (5), käyttäjärekisteritiedostojen hallintaikkuna (6) ja henkilötietoikkuna (7). Uuden nostajan lisäämisikkunassa annetaan henkilötiedot ja lisätään painikkeella nostaja suoritusryhmään. Käyttäjärekisteri-tiedostojen hallintaikkunassa voidaan lukea entisiä käyttäjärekisteritiedostoja, luoda uusia, lisätä nostajia rekisteritiedostoon ja poistaa heitä sieltä sekä lisätä nostajia rekisteritiedostosta suoritusryhmään. Henkilötietoikkunassa näkyvät valitun nostajan henkilötiedot, joita voidaan muuttaa.



Kuva 4.3: Näkymä suoritusryhmän hallinnasta.

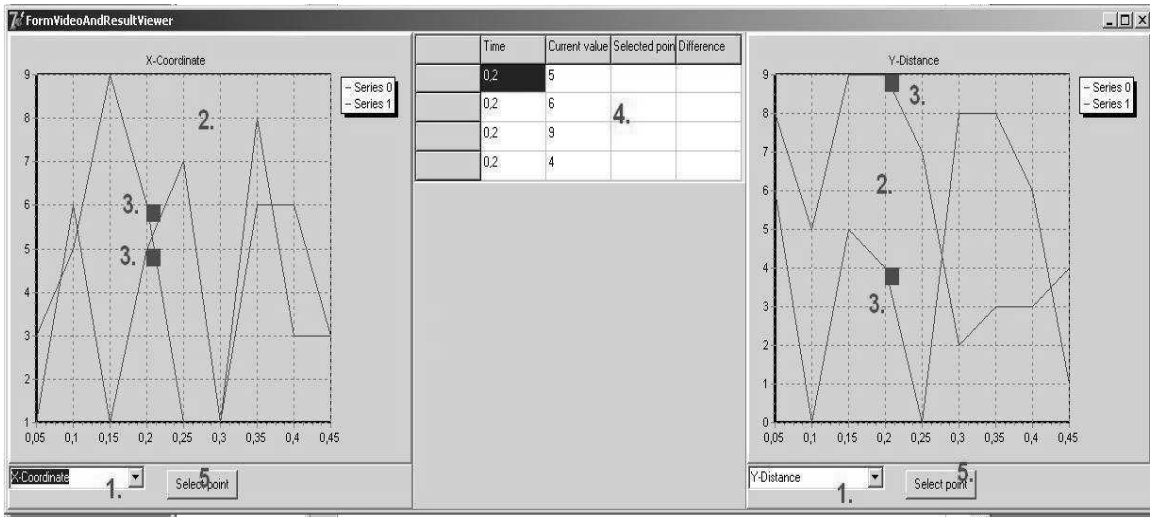
4.3 Suorituksen läpivienti

Nostosuorituksen nauhoittaminen aloitetaan samasta suoritusryhmänäkymästä, joka on esitetty tarkemmin kuvassa 4.4. Ensin valitaan suoritusryhmästä nostaja (1), sitten suoritusparametrit, eli nostotekniikka ja painomäärä (2), ja lopuksi painetaan nauhoituspainiketta (3), jolloin videonkaappaus alkaa välittömästi tai määritellyn ajan kuluttua. On myös mahdollista, että sovellus nauhoittaa nostoa esim. sekunnin ennen *Rec*-painikkeen painamista, ja käyttäjän määrittelemän ajan sen painamisen jälkeen. Kaikissa tapauksissa nauhoittaminen voidaan katkaista painamalla *Stop*-painiketta (4). Nauhoitettu suoritus lisätään nostajan nostojen listaan (5).



Kuva 4.4: Suoritusryhmänäkymä.

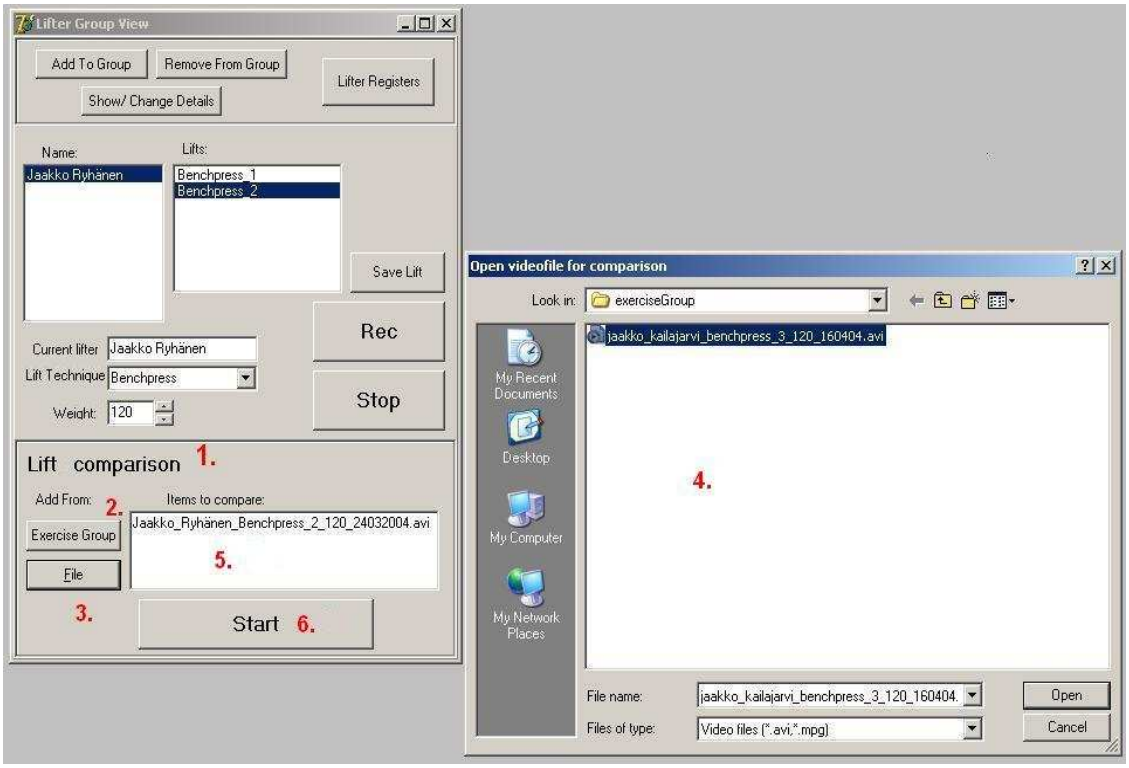
4.4 Analyysitulosten esittäminen



Kuva 4.5: Analyysitulosnäkymä.

Suorituksesta saadut analyysitulokset näytetään kuvan 4.5 kaltaisessa ikkunassa. Käyttäjä voi valita kuvan alareunassa näkyvistä alasvetolistoista (1), mitä muuttujaa halutaan seurata. Tällöin vastaavassa koordinaatistossa (2) näytetään valittua muuttujaa käsiteltävänä olevista suorituksista. Jos esimerkiksi on valittuna kaksi suoritusta vertailua varten, näkyy molemmista suorituksista sama muuttuja yhdessä koordinaatistossa. Tällöin suoritusten väliset erot kunkin muuttujan osalta tulevat selkeästi esille. Kursori (3) näyttää käyrältä vastaavan kohdan, kun videokuvaa vieritetään videonäytön palkilla (kuva 4.1 - (4)). Graafisen kuvaajan lisäksi tiedot näytetään numeroina näytön keskellä (4). Käyttäjä voi valita painikkeella (5) kultakin käyrältä haluamansa vertailukohdan. Tällöin valitun kohdan ja nykyisen kursorin osoittaman kohdan erotus tulee näkyviin numerotaulukossa (4). Numerotaulukossa näytettäviä tietoja ovat aika, muuttujan nykyinen arvo, maksimiarvo, valitut arvo, valitun ja nykyisen arvon erotus tai valittujen arvojen erotus ja mahdollisesti muuttujan keskiarvo koko suorituksen aikana.

4.5 Suoritusten vertailu



Kuva 4.6: Suoritusten vertailun aloittaminen.

Suoritusten vertailu aloitetaan kuvassa 4.6 näkyvästä suoritusryhmänäkymästä. Näkymän alalaidassa on suoritusten vertailuosa (1). Suoritus voidaan lisätä vertailuun joko harjoituksessa suoritetuista valitsemalla haluttu suoritus ja painamalla painiketta (2), tai tiedostosta painikkeella (3), jolloin avautuu tiedostonavausdialogi (4). Vertailuun valitut suoritukset lisätään listaan (5), ja vertailun voi aloittaa aloituspainikkeella (6).

Kun vertailu on aloitettu, lisätään videonäyttöön toinen videokuvanäyttö, ja analyysinäkymään toinen koordinaatisto kuvaajaa varten.

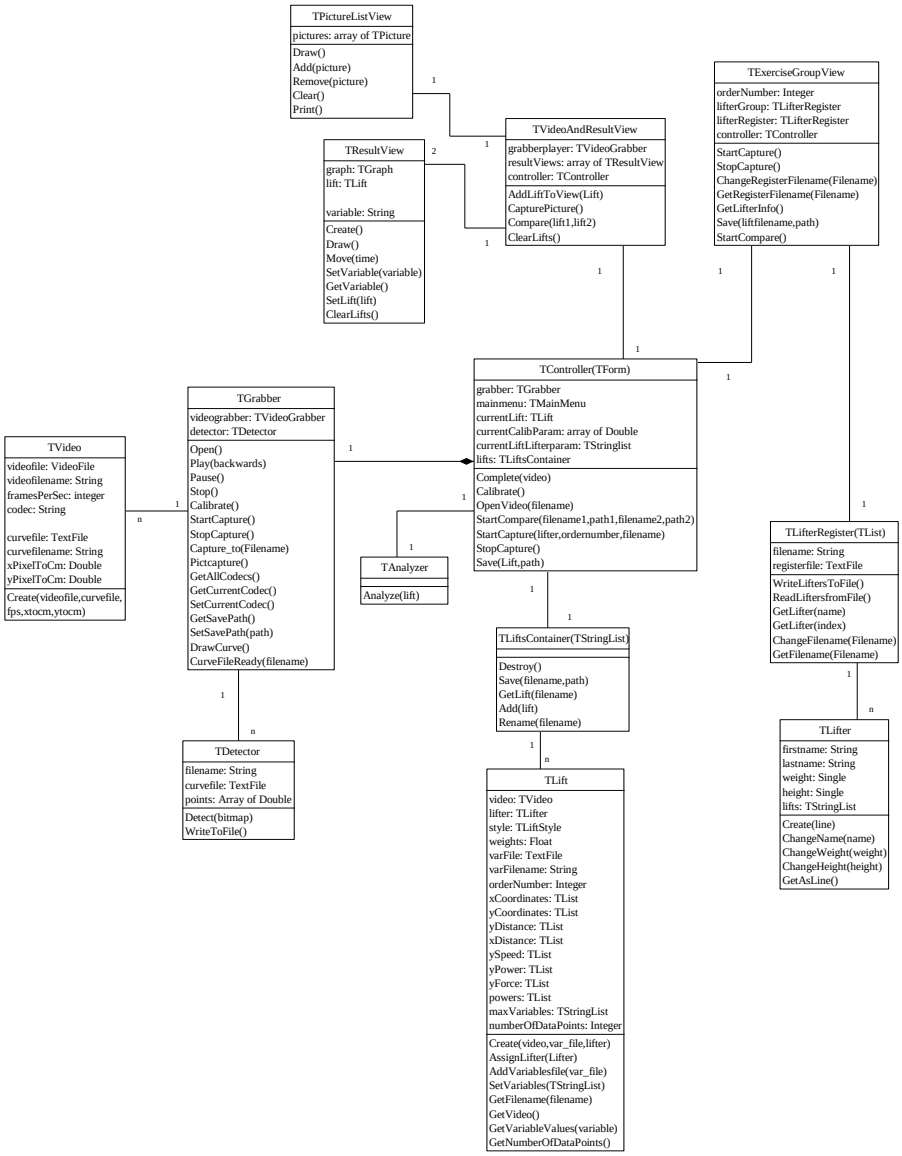
5 Sovelluksen rakenne

Tässä luvussa kuvataan sovelluksen sisäistä rakennetta luokkakaavion ja luokkakuvausten avulla. Kustakin sovelluksen luokasta on kuvattu tärkeimmät attribuutit ja metodien toiminta.

5.1 Sovelluksen luokat

Luokkakaaviosta näkyvät sovelluksen luokkien keskinäiset suhteet, niiden attribuutit ja metodit sekä luokkien väliset kardinaalisuudet, eli luokkien olioiden esiintymien määrät. Sovelluksen luokkakaavio on esitetty kuvassa 5.1.

Luvuissa 5.2 ja 5.3 esitellään sovelluksen luokat luokkakuvausten avulla. Sovelluksen luokat nimetään hyvän Delphi-ohjelmointitavan mukaisesti T-alkuisina, esim. `TController`. Luokat on eroteltu käyttöliittymän luokkiin, luku 5.2, ja sisäisen toteutuksen luokkiin, luku 5.3. Luvussa 5.4 kerrotaan sovelluksessa käytettävistä ulkoisista komponenteista.



Kuva 5.1: Sovelluksen luokkakaavio.

5.2 Käyttöliittymän luokat

5.2.1 TResultView

`TResultView` on yhden noston muuttujien näyttämiseen tarkoitettu visuaalinen komponentti. Niitä voi olla sovelluksessa kaksi yhtäaikaisesti. Se sisältää käyrien piirtämiseen tarkoitetun `TChart`-komponentin. `TResultView` käyttää hyväkseen `TLiftin` muuttujataulukoita joten sen ei tarvitse lukea muuttujatiedostoa.

	Nimi:	**Kuvaus:**
Attribuutti	`graph: TChart`	Muuttujan käyrän piirtämiseen käytetty komponentti.
	`lift: TLift`	Nosto, jota halutaan analysoida.
	`variable: String`	Kertoo, mitä muuttujaa halutaan analysoida.
Metodi	`Draw()`	Piirtää muuttujan käyrän `TChart`-komponenttiin.
	`Move(time)`	Päivittää `TChart`-komponenttia, kun video etenee.
	`SetVariable(variable)`	Asettaa muuttujan, jota analysoidaan.
	`GetVariable()`	Palauttaa muuttujan, jota ollaan analysoimassa.
	`SetLift(lift)`	Asettaa `lift`:in analyysitulokset katseluun.
	`ClearLifts()`	Poistaa kaikki `TLift:t` analyysistä.

Taulukko 5.1: `TResultView`.

5.2.2 TVideoAndResultView

`TVideoAndResultView` ohjaa `TResultView`-luokan olioita, jotka näyttävät analysointituloksia. Tämä luokka sisältää toisen videonpätkien näyttämiseen tarkoitetun `TGrabber`-komponentin. `TVideoGrabberia` ei voida käyttää suoraan, koska pelkän videon näyttämisen lisäksi piirretään liikeratakäyrää. `TResultView` päivittää lisäksi taulukkoa, joka näyttää muuttujien arvot numeerisina, `TResultView`-luokan olioiden välittämillä arvoilla.

	Nimi:	**Kuvaus:**
Attribuutti	`grabberplayer: TGrabber`	Osoitin videon näyttimeen.
	`ResultViews: array of TResultView`	Osoittimet `TResultView`-komponentteihin.
	`controller: TController`	Osoitin ohjausluokkaan.
Metodi	`AddLiftToView(lift)`	Asettaa videon katselua varten.
	`CapturePicture()`	Kaappaa näytettävästä videosta yhden kuvan.
	`Compare(lift1,lift2)`	Asettaa kaksi videota vertailuun.
	`ClearLifts()`	Poistaa näytettävät suoritukset.

Taulukko 5.2: `TVideoAndResultView`.

5.2.3 TUserGroupView

`TUserGroupView` on näkymä, jolla hallitaan suoritusryhmää ja käyttäjärekisteriä. Sen kautta aloitetaan ja lopetetaan suoritusten nauhoitukset sekä myös käynnistetään vertailu.

	Nimi:	**Kuvaus:**
Attribuutti	`orderNumber: integer`	Suorituksen järjestysnumero harjoituksessa.
	`lifterGroup: TLifterRegister`	Käytetään suoritusryhmän tallentamiseen ja hallitsemiseen ajonaikaisesti.
	`lifterRegister: TLifterRegister`	Lukee ja tallentaa käyttäjärekisteritiedostoja, joista suoritusryhmään voidaan lisätä nostajia.
Metodi	`StartCapture()`	Välittää `TControllerille` videon kaappauksen aloituspyynnön.
	`StopCapture()`	Välittää `TControllerille` videon kaappauksen lopetuspyynnön.
	`Compare(lift1,lift2)`	Asettaa kaksi videota vertailuun.
	`GetLifterInfo()`	Palauttaa `TControllerille` nostajan tiedot, joita tarvitaan `TVideon` muokkaamiseksi `TLiftiksi`.
	`Save(liftfilename,path)`	Tallennuspyyntö `liftfilename`:n osoittamasta videonpätkästä `pathin` osoittamaan paikkaan.
	`StartCompare()`	Välittää tiedot käyttäjän näkymässä valitsemista vertailtavista nostoista `TControllerille.`

Taulukko 5.3: `TUserGroupView`.

5.3 Sovelluksen sisäiset luokat

5.3.1 TDetector

`TDetector`-luokka saa `TGrabber`-luokalta jokaisen videon kuvan eli framen parametrina, etsii siitä ledin paikan ja tallentaa tämän tiedon ensin muistiin ja lopuksi kirjoittaa pisteet liikeratatiedostoon.

	Nimi:	**Kuvaus:**
Attribuutti	`filename: String`	Tiedoston nimi, johon liikeratakäyrä tallennetaan.
	`Points: Array of Double`	Taulukko liikeradan pisteille.
Metodi	`Detect(bitmap)`	Etsii parametrina olevasta kuvasta ledin paikan, ja tallentaa tiedon `points`-taulukkoon.
	`WriteToFile()`	Tallentaa liikeratakäyrän pisteet `points`-taulukosta tiedostoon.

Taulukko 5.4: `TDetector`.

5.3.2 TAnalyzer

`TAnalyzer`-luokka analysoi nostosuorituksen biomekaaniset muuttujat tangon liikeradan ja käyttäjän antamien tietojen perusteella. Se ei siis käsittele varsinaista videota. Metodi `Analyze()` analysoi aina kaikki muuttujat, koska laskenta ei ole kovin aikaa vievää ja raskasta.

	Nimi:	**Kuvaus:**
Attribuutti		
Metodi	`Analyze(lift)`	Analysoi parametrina annetun noston muuttujat ja tallettaa ne tiedostoon, jonka tiedostonnimi saadaan `liftin` parametrista, sekä `liftiin`. Maksimiarvot tallennetaan suoraan `liftin` parametritaulukoihin.

Taulukko 5.5: `TAnalyzer`.

5.3.3 TController

`TController` on ohjausluokka, joka toimii muiden luokkien välissä viestien välittäjänä. Käytännössä se on sovelluksen pääformi. Kaikkia sovelluksen sisäisen toteutuksen luokkia ei ole järkevää sijoittaa `TControllerin` alle, jolloin kaikki viestit kulkisivat sen kautta.

	Nimi:	**Kuvaus:**
Attribuutti	`currentLift: TLift`	Käsiteltävä nosto.
	`grabber: TGrabber`	Videonkaappausolio.
	`currentCalibParam: array of Double`	Kalibrointiparametrit.
Metodi	`Complete(video)`	Tekee `TVideosta TLiftin` liittämällä siihen nostajan tiedot, painomäärän, noston tekniikan ja päivittämällä tiedostonnimet. Lisää valmiin `TLiftin TLiftsContaineriin`.
	`Calibrate()`	Välittää kalibrointipyynnön `TGrabberille`.
	`OpenVideo(filename)`	Avaa videonpätkän tiedostosta, ja tekee siitä `TLiftin`, joka voidaan avata vertailua / katselua varten.
	`Compare(filename1, from1,filename2,from2)`	Hakee kaksi nostosuoritusta vertailtavaksi. Parametrit `from1` ja `from2` ilmoittavat, löytyvätkö suoritukset `TLiftsContainerista` vai tiedostosta. Välittää haetut nostot `TVideoAndResultView:`lle niiden näyttämistä varten.
	`StartCapture(lifter, ordernumber, filename)`	Välittää videonkaappauksen aloituspyynnön `TGrabberille`.
	`StopCapture()`	Välittää videonkaappauksen lopetuspyynnön `TGrabberille`.
	`Save(lift,path)`	Välittää tallennuspyynnön `TLiftsContainerille`.

Taulukko 5.6: `TController`.

5.3.4 TLifterRegister

`TLifterRegister`-luokka lukee ja tallettaa käyttäjärekisterin tiedostoon. Se on peritty `TList`-luokasta, ja sitä voidaan käyttää myös suoritusryhmän tallentamiseen ajonaikaisesti. Objektien lisäämiseen ja poistamiseen tarvittavat metodit tulevat suoraan perittyinä.

Attribuutti	`filename: String`	Tiedostonnimi, josta käyttäjärekisteri luetaan / johon se tallennetaan.
Metodi	`WriteLiftersToFile()`	Tallettaa käyttäjät tiedostoon.
	`ReadLiftersFromFile()`	Lukee käyttäjien tiedot rekisteritiedostosta.

Taulukko 5.7: `TLifterRegister`.

5.3.5 TLifter

`TLifter` sisältää sovelluksen yhden käyttäjän eli nostajan tiedot.

	Nimi:	**Kuvaus:**
Attribuutti	`firstname: String`	Nostajan etunimi.
	`lastname: String`	Nostajan sukunimi.
	`height: Float`	Nostajan pituus.
	`weight: Float`	Nostajan paino.
	`lifts: TStringList`	Ajonaikainen tieto suorituksista, jotka nostaja on tehnyt. Suoritukset tallennetaan nostovideon tiedostonnimen perusteella.
Metodi	`Create(line)`	Luo uuden nostajan yhden tiedoston rivin tietojen perusteella.
	`GetAsLine(): String`	Palauttaa nostajan tiedot merkkijonona, joka voidaan suoraan kirjoittaa tiedostoon.

Taulukko 5.8: `TLifter`.

5.3.6 TVideo

`TVideo`-luokka sisältää videonpätkän, liikeratatiedoston sekä tiedot videon framemäärästä ja käytetystä koodekista.

	Nimi:	**Kuvaus:**
Attribuutti	`videofilename: String`	Videotiedoston tiedostonnimi.
	`codec: String`	Videossa käytetty koodekki.
	`framesPerSec: Integer`	Videon kuvataajuus.
	`xPixelToCm: Double`	Kalibrointitieto pikselin koosta sentteinä x-akselin suunnassa.
	`yPixelToCm: Double`	Kalibrointitieto pikselin koosta sentteinä y-akselin suunnassa.
Metodi	`Create(videofile, curvefile, fps, xtocm, ytocm)`	Luo uuden `TVideon` annetuista parametreista.

Taulukko 5.9: `TVideo`.

5.3.7 TLiftsContainer

`TStringList`-luokasta peritty säiliöluokka, joka tallettaa nostovideoita ajonaikaisesti. Metodeja videoiden lisäämiseksi ja poistamiseksi säiliöstä ei tarvitse kirjoittaa uudelleen tähän luokkaan, sillä perityt metodit ovat käyttökelpoisia. `TLiftsContainer` hoitaa myös videoiden lopulliset tallennukset kiintolevylle.

	Nimi:	**Kuvaus:**
Attribuutti		
Metodi	`Destroy()`	Destruktori täytyy ylikirjoittaa, jotta tuhottaessa säiliöluokka myös kaikki sen sisältämät objektit tuhotaan.
	`Save(filename,path)`	Tallettaa `filenamen` osoittaman suorituksen lopullisesti levylle `pathin` osoittamaan paikkaan.
	`Rename(filename)`	Nimeää uudelleen jo tallennetun `TLift`:in.
	`Add(lift)`	Lisää uuden noston säiliöön.
	`GetLift(filename)`	Palauttaa `filenamen` mukaisen noston.

Taulukko 5.10: `TLiftsContainer.`

5.3.8 TLift

`TLift`-luokka kapseloi videopätkän sekä sen liikerata- ja käyttäjätiedot nostoksi. Kullekin analysoidulle muuttujalle ja mittauspisteiden koordinaateille on omat `TList`-tyyppiset attribuuttinsa, jolloin ne tarvitsee lukea tiedostosta vain, jos nosto haetaan levyltä.

	Nimi:	**Kuvaus:**
Attribuutti	`video: TVideo`	Videon tiedot.
	`lifter: TLifter`	Nostajan tiedot, jonka nostosta on kyse.
	`style: TliftStyle`	Nostotekniikka.
	`weights: Single`	Noston, josta video nauhoitettu, painomäärä.
	`varFilename: String`	Noston muuttujatiedoston tiedostonimi.
	`xCoordinates: TList`	Liikeratakäyrän x-koordinaatit mittauspisteissä.
	`yCoordinates: TList`	Liikeratakäyrän y-koordinaatit mittauspisteissä.
	`xDistance: TList`	Matka x-akselin suunnassa sivupoikkeaman laskemiseen, eli kahden mittauspisteen etäisyys toisistaan.
	`yDistance: TList`	Matka y-akselin suunnassa tehon laskemiseen.
	`ySpeed: TList`	Nopeus y-akselin suunnassa.
	`yPower: TList`	Teho y-akselin suunnassa.
	`yForce: TList`	Voima y-akselin suunnassa
	`maxVariables: TList`	Tiedot kunkin analysoidun muuttujan maksimiarvoista.
	`NumberOfDataPoints: integer`	Mittauspisteiden, joissa nostosta mitattu muuttujien arvoja, määrä.
Metodi	`Create(video,varFilename,lifter)`	Luo uuden `TLiftin` parametrien arvojen perusteella. Lukee samalla muuttujien arvot analysointitulostiedostosta.
	`GetVariableValues(variable)`	Palauttaa valitun muuttujan arvot sisältävän `TListin`.

Taulukko 5.11: `TLift`.

5.3.9 TGrabber

`TGrabber` luokka sisältää `TVideoGrabber`-komponentin ja sen kalibroinnin. `TGrabber`-luokan parametrina olevaa `TVideoGrabberia` käytetään videonpätkän kaappaamiseen, ja myös videon pyörittämiseen suorituksia vertailtaessa, sekä kaappauksen aikana. Normaalisti jo kaapatut videot pyöritetään `TvideoAndResultView`-luokan `TVideoGrabber`-komponentilla. Videokamera on siis kytkettynä juuri tämän luokan olioon.

	Nimi:	**Kuvaus:**
Attribuutti	`videograbber: TVideoGrabber`	Kuvankaappauskomponentti.
Metodi	`Calibrate()`	Suorittaa sovelluksen kalibroinnin.

Taulukko 5.12: `TGrabber`.

5.4 Ulkoiset komponentit

Tässä luvussa esitellään sovelluksen toteutukseen käytettäviä ulkoisia komponentteja, jotka eivät sisälly Delphin mukana tuleviin vakiokomponentteihin. Sovelluksessa saatetaan käyttää muitakin ulkoisia komponentteja, mutta niiden käytöstä tulee sopia erikseen KIHUn kanssa. Käytön edellytyksenä on, että ne ovat freewarea eli vapaasti käytettävissä. Myös KIHUn ATK-suunnittelija Risto Toivosen tekemiä komponentteja saatetaan käyttää hieman muokattuina, tai niistä otetaan mallia sovelluksen toteutukseen. Toivonen on antanut tähän hyväksyntänsä.

5.4.1 VideoGrabber

`VideoGrabber`-komponenttia käytetään videon kaappaamiseen ulkoiselta videolähteeltä eli digitaaliselta videokameralta. Sovelluksessa sitä käytetään myös videoiden näyttämiseen. `TVideoGrabber`-luokka on peritty Delphin `TCustomPanelista`. Se on tavallinen graafinen komponentti, jonka voi asentaa Delphin komponenttipalettiin ja lisätä lomakeikkunaan hiirellä vetämällä. Sen attribuuttien arvoja voi muuttaa graafisesti Delphin ominaisuuseditorin avulla. `VideoGrabberissa` ei ole videoiden näyttämisen hallintaan tarvittavaa painikeriviä valmiina, vaan se täytyy tehdä itse. KIHUlla on kyseiseen kaupalliseen komponenttiin lisenssi, joka on annettu projektiryhmän käyttöön. Lisätietoa komponentista löytyy sen tekijän verkkosivuilta [3].

5.4.2 ResizerPanel

`ResizerPanel` on Delphin vakiokomponenttia `Panelia` toiminnaltaan ja `TPanel`-luokkaa rajapinnaltaan vastaava komponentti. `ResizerPanel` säilyttää sillä olevien näkyvien komponenttien mittasuhteet, kun ikkunan kokoa tai resoluutiota muutetaan. Tämä helpottaa sovelluksen suunnittelua käytettäväksi erilaisilla resoluutioilla. Myös `ResizerPanel` voidaan asentaa Delphin komponenttipalettiin.

Paneeleita käytetään graafisissa sovelluksissa ryhmittelemään muita kontrolleja sekä niiden keskitettyyn hallitsemiseen. Komponentti on freewarea, ja tekijä antaa komponentilleen täyden käyttöoikeuden sovelluksissa. Komponentin ja lisätietoa siitä saa tekijän verkkosivuilta [1].

5.4.3 ExtendedListBox

Projektiryhmän jäsenen Vesa Tanhua-Tyrkön yliopiston kurssilla tekemä komponentti, joka lisää Delphin tavalliseen `ListBox`-valintalistakomponenttiin lisäominaisuuksia. Rajapinta on samanlainen kuin `TListBox`-luokassa, täydennettynä esimerkiksi tapahtumalla, jota kutsutaan kun valintalistan valinta muuttuu. Tekijä luovuttaa komponenttiinsa tilaajalle täydet rinnakkaiset muuntelu-, käyttö- ja edelleenluovutusoikeudet allekirjoittamalla oikeuksiensiirtosopimuksen. Komponentista voi kysyä lisätietoja Vesa Tanhua-Tyrköltä sähköpostiosoitteesta `vttanhua@cc.jyu.fi`.

6 Koodi

6.1 Muuttujien nimeäminen

Muuttujat nimetään englanniksi mahdollisimman havainnollisesti. Muuttujan nimen pituus ei ole este sen havainnollisuudelle.

6.2 Kommentointi

Ohjelmakoodin kommentit kirjoitetaan englanniksi. Kooditiedostojen alussa on otsikkorivit, joilla kerrotaan tiedoston sisällöstä, tekijöistä ja sen sisältämistä luokista. Luokista on selitetty tärkeimmät attribuutit ja metodit. Mahdollisten vapaiden aliohjelmien toiminta on selitetty. Kooditiedostoihin tehtävät muutokset kirjataan otsikkoriveille, ja niistä tulee käydä ilmi muutoksen tekijä. Metodien toteutuksen yhteydessä on vielä tiivistelmä otsikkorivin kuvauksesta, eli kerrotaan lyhyesti metodin käyttötarkoitus ja toiminta.

Alla esimerkki ohjelmakoodin kommentoinnista ja kooditiedostojen otsikkoriveistä.

```
unit example;

(*********************************************************

 (c) OptiLift Project, spring 2004
     University of Jyväskylä, Department of Mathematical
     Information Technology

Lauri Laasala(LL), Olli Lukkarinen(OL), Ville Räisänen(VR),
Vesa Tanhua-Tyrkkö (VTT)


   Unit Name: example.pas
   Purpose  : a simple example unit to illustrate code style
              and commenting
   Author   : OptiLift
   Date     : 8.3.2004 /VR
   Changed  : 8.3.2004 /VR
              + added commenting

  Requires: -

   Classes: TCodeExample, ...


---------------------------------------------------------
Classes:
========

 TCodeExample

 attributes:
  * text: String           - stores the text

 methods and functions:
  * GetText: String        - returns the value of text
                             attribute
  * SetText(itext:String)  - inserts the value of itext in
                             text attribute


*********************************************************)

interface
```

```
type
  TCodeExample = class
  private
    { Private declarations }
    text: String;                 // used to store the text
  public
    { Public declarations }
    function GetText: string;
    procedure SetText(itext: String);
  end;


implementation

//------------------------------------------------------------
//  GetText - returns the value of text
//------------------------------------------------------------

function TCodeExample.GetText: string;
begin
  GetText := self.text;
end;

//------------------------------------------------------------
// SetText - inserts the value of itext to text
//------------------------------------------------------------

procedure TCodeExample.SetText(itext: String);
begin
  if ( itext = '')
    then text := 'Hello World!' // if itext empty, then
                                //'Hello World!' put
  else text := itext;
end;

end.
```

Esimerkki 6.1: Ohjelmakoodin kommentointi.

7 Tiedostot

Sovelluksessa tarvitaan tiedostoja datan tallennukseen. Liikeratakäyrän pisteet sekä analysointitulokset tallennetaan tiedostoihin, josta ne voidaan lukea uudelleenpiirtämistä tai suoritusten vertailua varten. Käyttäjärekisterit tallennetaan tiedostoihin, jolloin käyttäjien tietoja ei tarvitse joka kerta syöttää uudestaan. Videopätkät tallennetaan suoraan tiedostoihin, sillä ne ovat liian suuria suoraan keskusmuistissa käsiteltäväksi.

7.1 Tiedostojen formaatit

Liikeratakäyrät ja käyttäjien tiedot tallennetaan tavallisiin tekstitiedostoihin. Videotiedostot tallennetaan joko `AVI:`na tai `MPG:`nä. Huomattavaa on, että pelkkä tiedostopääte ei videotiedostojen kohdalla kerro kaikkea, vaan niihin liittyy aina jokin koodekki. Tallennettujen videoiden katseleminen ei onnistu ilman asianomaista koodekkia. Siksi videoita tallennettaessa pitää ilmaista, mitä koodekkia on käytetty ja mistä sen voi ladata.

Kaikki nostoon liittyvät tiedostot tallennetaan saman nimisinä erilaisin tiedostopäättein. Nimeämiskäytäntö tehdään yhdenmukaiseksi jo aiemmin KIHUn käytössä olevien järjestelmien kanssa. Tiedostojen nimi on muotoa `v1_v2_v3_v4.BAL`, missä `v1` kuvaa valintaa 1, `v2` valintaa 2 jne. ja `BAL` tiedostopäätettä. Eri valintoja voi olla korkeintaan seitsemän. Tiedoston nimi voi olla siis esimerkiksi:

```
etunimi_sukunimi_tekniikka_suorituksenro_painot_pvm.tiedosto
pääte
```

```
jaakko_kailajarvi_squat_4_150_160304.BAL.
```

Esimerkki 7.1: Tiedoston nimi.

7.1.1 Käyttäjärekisteritiedosto

Käyttäjärekisteri tallennetaan tavalliseen tekstitiedostoon, jonka tiedostonimen päätteenä on `.usr`. Käyttäjistä tallennettavat kentät erotellaan '`|`'-merkin avulla. Ensimmäinen rivi on otsikkorivi, jossa selitetään kenttien sisällöt. Alla on esimerkki käyttäjärekisteritiedoston sisällöstä.

```
First name|Last name|Height|Weight
Kalle|Ahola|190|123
Maija|Meikäläinen|167|50
Sami|Pajula|190|100
Heikki|Hynninen|190|87
Marika|Kinnunen|150|62
```

Esimerkki 7.2: Käyttäjärekisteritiedoston sisältö.

7.1.2 Liikeratatiedosto

Liikeratatiedoston tiedostopääte on `.BAL`. Ensimmäisellä rivillä on tieto käytetystä kuvataajuudesta ja virhemarginaalista pilkulla eroteltuina. Kenttinä, jotka erotellaan pilkuin, x-koordinaatti, y-koordinaatti ja aika millisekunteina. Desimaalierottimena, kuten myös analysointitulostiedostossa, on piste.

```
25, 0.22
0,0,39.9999199
25,11,80.1111111
25,24,119.9999199
50,50,160.0011010
61,100,200.101001
```

Esimerkki 7.3: Liikeratatiedoston sisältö.

7.1.3 Analysointitulostiedosto

Analysointitulostiedoston tiedostopääte on `.CSV`. Ensimmäisellä rivillä on tieto käytetystä kuvataajuudesta. Muilla riveillä ensimmäinen kenttä on mittausaika. Sen jälkeen on puolipiste sekä muuttujan virhemarginaali ja muuttuja pilkulla eroteltuina. Seuraavaksi on taas puolipiste, jonka jälkeen samalla tavalla virhemarginaali ja muuttuja pilkulla eroteltuina. Muuttujia ja niiden virhemarginaaleja voi olla ennalta määrittelemätön määrä. Niiden laskemisesta on kerrottu enemmän luvussa 8.

```
25
39.9999;0.12,5.1404;0.12,3.2122;0.21,2.4249;0.22,2.780
80.9599;0.02,6.2566;0.27,2.2541;0.51,2.5349;0.52,2.681
119.929;0.30,5.6357;0.45,3.7322;0.16,2.5821;0.52,2.630
160.029;0.42,7.1522;0.22,5.2222;0.31,2.2456;0.02,2.546
199.999;0.12,5.3213;0.76,2.6642;0.12,2.1111;0.12,2.280
```

Esimerkki 7.4: Analysointitulostiedoston sisältö.

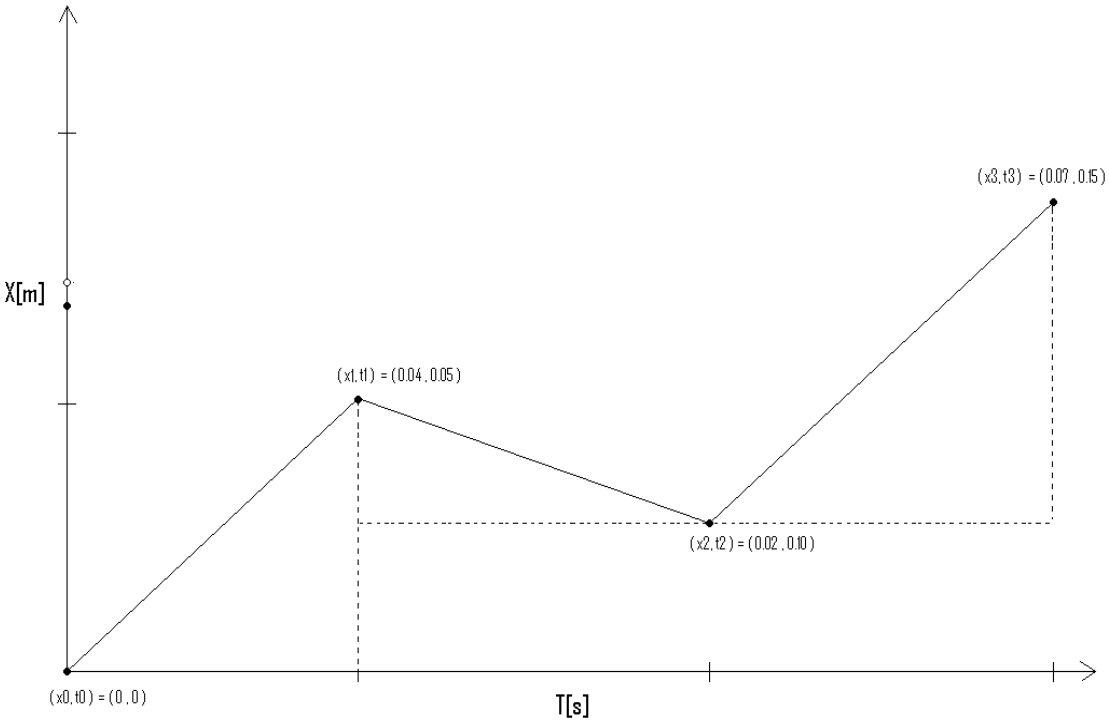
7.2 Videokoodekit

Videon pakkaamiseen valitun koodekin valinta on erityisen tärkeää. Koodekkien välillä on suuria eroja niin kuvanlaadussa, vaaditussa tallennustilassa kuin nopeudessakin. Lähtökohta on, että koodekin valinta on käyttäjän tehtävissä. Sovellus näyttää kaikki käyttäjän koneella käytettävissä olevat koodekit, joista käyttäjä valitsee haluamansa. Kaikkien koodekkien käyttö ei videonkaappaukseen tarkoitetun komponentin asettamista rajoitteista johtuen ole mahdollista. Käyttäjää täytyy informoida siitä, mitkä koodekeista ovat soveltuvia käytettäväksi sovelluksessa. Video pakataan valittua koodekkia käyttäen lennossa.

8 Analyysi

Tässä luvussa kerrotaan sovelluksessa biomekaanisten muuttujien laskemiseen käytettävistä kaavoista.

Kuvan 8.1 vaaka-akselin ajanhetket t_0 - t_3 vastaavat nauhoitetun videon joitakin neljää peräkkäistä framea. Näistä frameista mitattujen ledin paikkojen avulla saadaan laskettua halutut biomekaaniset muuttujat.



Kuva 8.1: Esimerkki mittaussarjasta, jonka tuloksena on saatu levytangon paikka ajan funktiona

Kaikki virhekaavat on johdettu käyttäen yleistä virheen etenemiskaavaa [9]

$$\delta f = \sqrt{\left(\frac{\partial f}{\partial q}\delta q\right)^2 + \left(\frac{\partial f}{\partial r}\delta r\right)^2 + \ldots + \left(\frac{\partial f}{\partial w}\delta w\right)^2}, \qquad (8.1)$$

missä $f = f(q, r, \ldots, w)$.

Laskenta suoritetaan erikseen pysty- ja vaaka-akselilla. Kaikki vektorisuureet on esitetty skalaarimuotoisina. Laskenta on jäljempänä mainituin poikkeuksin samanlainen kummallakin akselilla, ts. kaavoissa (8.2)-(8.10) esiintyvien koordinaattien x_n tilalle voidaan tarvittaessa vaihtaa koordinaatit y_n.

Kuljettu matka:

$$s_n = x_n - x_{n-1}, \tag{8.2}$$

missä paikka x_n on jokin tuntematon funktio $x_n = f(t_n)$ ja vastaavasti $x_{n-1} = f(t_{n-1})$.

Matkan virhe:

$$\delta s_n = \sqrt{(\delta x_n)^2 + (\delta x_{n-1})^2}\,. \tag{8.3}$$

Virheitä δx_n ja δx_{n-1} ei voida johtaa virheen etenemiskaavalla, vaan ne saadaan videokameran kuvan ja käytetyn led-seuranta-algoritmin tarkkuuksista.

Matkaan s_n käytetty aika ja sen virhe:

$$\Delta t_n = t_n - t_{n-1}\,, \tag{8.4}$$

$$\delta \Delta t_n = \sqrt{(\delta t_n)^2 + (\delta t_{n-1})^2}\,. \tag{8.5}$$

Käytetään hetkellisen nopeuden $v(t_n) = \frac{dx_n}{dt} = \frac{df(t_n)}{dt} = f'(t_n)$ numeeriseen approksimointiin kolmipistemenetelmää [2], jonka mukaan

$$f'(t_n) = \frac{1}{2h}\left[f(t_n + h) - f(t_n - h)\right] + \frac{h^2}{6} f'''(\zeta_1)\,, \tag{8.6}$$

missä $t_n - h < \zeta_1 < t_n + h$.

Koska funktiota $f(x_n)$ ei tunneta, virhetermin suuruutta ei voida luotettavasti arvioida. Sen sijaan käytetään yleisestä virheen etenemiskaavasta (8.1) saatavaa maksimivirheen arvoa.

Kolmipistemenetelmällä ei voida arvioida nopeutta mittausalueen päätepisteissä, ja nämä pisteet jätetään laskennasta pois (arvojen tuntemisesta päätepisteissä ei ole erityistä hyötyä).

Hetkellinen nopeus ja sen virhe:

$$v(t_n) = f'(t_n) = \frac{1}{t_{n+1} - t_{n-1}}\left[f(t_{n+1}) - f(t_{n-1})\right] = \frac{x_{n+1} - x_{n-1}}{t_{n+1} - t_{n-1}}, \tag{8.7}$$

$$\delta v(t_n) = \sqrt{\left(\frac{\delta x_{n+1}}{t_{n+1} - t_{n-1}}\right)^2 + \left(\frac{\delta x_{n-1}}{t_{n+1} - t_{n-1}}\right)^2 + \left(\frac{x_{n+1} - x_{n-1}}{(t_{n+1} - t_{n-1})^2}\delta t_{n+1}\right)^2 + \left(\frac{-x_{n+1} + x_{n-1}}{(t_{n+1} - t_{n-1})^2}\delta t_{n-1}\right)^2} \tag{8.8}$$

Toinen derivaatta(kiihtyvyys) ja sen virhe saadaan vastaavasti:

$$a(t_n) = v'(t_n) = f''(t_n) = \frac{1}{t_{n+1} - t_{n-1}}\left[f'(t_{n+1}) - f'(t_{n-1})\right] = \frac{v(t_{n+1}) - v(t_{n-1})}{t_{n+1} - t_{n-1}}, \tag{8.9}$$

$$\delta a(t_n) = \sqrt{\left(\frac{\delta v(t_{n+1})}{t_{n+1} - t_{n-1}}\right)^2 + \left(\frac{\delta v(t_{n-1})}{t_{n+1} - t_{n-1}}\right)^2 + \left(\frac{v(t_{n+1}) - v(t_{n-1})}{(t_{n+1} - t_{n-1})^2}\delta t_{n+1}\right)^2 + \left(\frac{-v(t_{n+1}) + v(t_{n-1})}{(t_{n+1} - t_{n-1})^2}\delta t_{n-1}\right)^2}. \tag{8.10}$$

Hetkellinen voima (vaaka- tai pystyakselilla):

$$F(t_n) = m_t a(t_n), \tag{8.11}$$

missä m_t on käytetyn levytangon massa.

Hetkellisen voiman virhe:

$$\delta F(t_n) = \sqrt{\left(a(t_n)\delta m_t\right)^2 + \left(m_t \delta a(t_n)\right)^2} \tag{8.12}$$

Painonnostajan aikaansaama voima F_p^x on ainoa levytankoon vaakasuunnassa (merkitsevästi) vaikuttava voima, ja se siis vastaa mittauspisteissä kaavalla (8.11) vaaka-akselin suunnassa laskettua voimaa: $F_p^x(t_n) = F^x(t_n)$. Vastaavasti levytangon kaavalla (8.9) vaaka-akselin suunnassa laskettu kiihtyvyys on sama kuin nostajan aikaansaama kiihtyvyys a_p^x, eli $a_p^x(t_n) = a^x(t_n)$.

Pystysuunnassa levytankoon vaikuttava kokonaisvoima on painonnostajan aikaansaaman voiman F_p^y ja painovoiman F_g summa. Ts. mittauspisteissä kaavalla (8.9) pystysuunnassa laskettu kiihtyvyys $a^y(t_n) = a_p^y(t_n) + a_g$. Kun levytangon pystysuuntainen kokonaiskiihtyvyys eroaa vapaata pudotusta vastaavasta arvosta $a_g = -9{,}81\frac{m}{s^2}$ (etumerkki viittaa kiihtyvyyden suuntaan alaspäin), erotus on nostajan aikaansaama (poislukien tietysti tilanne, jolloin tanko on maassa).

Painonnostajan (hetkellisesti) pystysuunnassa levytankoon kohdistama voima F_p^y on siis:

$$F_p^y(t_n) = m_t(a^y(t_n) - a_g), \tag{8.13}$$

missä $a^y(t_n)$ on kaavalla (8.9) pystysuunnassa laskettu kiihtyvyys.

Voiman $F_p^y(t_n)$ virhe:

$$\delta F_p^y(t_n) = \sqrt{\left((a^y(t_n) - a_g)\delta m_t\right)^2 + \left(m_t \delta a^y(t_n)\right)^2 + \left(m_t \delta a_g\right)^2} \tag{8.14}$$

Painonnostajan(pystysuunnassa) tekemän työn hetkellinen teho:

$$P_p^y(t_n) = F_p^y(t_n) \cdot v^y(t_n), \tag{8.15}$$

missä $v^y(t_n)$ on kaavalla (8.7) pystysuunnassa laskettu nopeus.

Tehon $P_p^y(t_n)$ virhe:

$$\delta P_p^y(t_n) = \sqrt{\left(v^y(t_n)\delta F_p^y(t_n)\right)^2 + \left(F_p^y(t_n)\delta v^y(t_n)\right)^2} \quad (8.16)$$

Kun levytanko, jonka massa on m_t kohotetaan korkeudelle $h = y_n - y_0$ ajassa $t = t_n - t_0$, tehdään painovoimaa vastaan työ

$$W_g = F_g h = m_t a_g h, \quad (8.17)$$

jonka keskiteho on

$$P_{avg} = \frac{W_g}{t} = \frac{m_t a_g h}{t} . \quad (8.18)$$

Keskitehon virhe:

$$\delta P_{avg} = \sqrt{\left(\frac{a_g h}{t}\delta m_t\right)^2 + \left(\frac{m_t h}{t}\delta a_g\right)^2 + \left(\frac{m_t a_g}{t}\delta h\right)^2 + \left(\frac{m_t a_g h}{t^2}\delta t\right)^2} \quad (8.19)$$

9 Testaus

Sovelluksen testauksessa käytettävistä menetelmistä ja käytännöistä kerrotaan testaussuunnitelmassa. Kyseisessä dokumentissa käydään läpi testausympäristö ja laitteet, joilla testausta suoritetaan, testauksen yleiset periaatteet ja testitapaukset.

10 Yhteenveto

Tässä OptiLift-projektin sovellussuunnitelmassa kerrotaan kevään 2004 OptiLift-Sovellusprojektin toteuttamasta sovelluksesta. Kyseessä on Kilpa- ja huippu-urheilun tutkimuskeskukselle toteutettava painonnoston levytankoharjoittelun nostotekniikan automaattinen mittaus- ja analysointisovellus. Tämä dokumentti käsittelee sovelluksen arkkitehtuuria, rakennetta ja toteutustapoja vaatimusmäärittelyssä esitettyjen vaatimusten pohjalta.

11 Lähteet

[1] Barbora Carlos, ”Carlos Barbora’s homepage’, saatavilla WWW-muodossa <URL: `http://www.carlosb.com/>`, *viitattu 30.3.2004*.

[2] Burden Richard, Faires Douglas, “Numerical Analysis”, Fifth Edition, PWS Publishing Company, Boston, 1993.

[3] Datastead software, ”Datastead software homepage”, saatavilla WWW-muodossa <URL: `http://www.datastead.com/vidgrab/index.htm` >, 2004.

[4] Jaakohuhta, Hannu, IT Ensyklopedia, Sanasto, Edita Oyj, Helsinki, 2001.

[5] Laasala Lauri, Lukkarinen Olli, Räisänen Ville ja Tanhua-Tyrkkö Vesa, OptiLift-Sovellusprojektin projektisuunnitelma, Jyväskylän yliopisto, Tietotekniikan laitos, 2004.

[6] Laasala Lauri, Lukkarinen Olli, Räisänen Ville ja Tanhua-Tyrkkö Vesa, OptiLift-Sovellusprojektin vaatimusmäärittely, Jyväskylän yliopisto, Tietotekniikan laitos, 2004.

[7] Keränen Tapani, Viitasalo Jukka ym., Nostotekniikan automaattinen mittaus- ja analysointijärjestelmä-esittelymoniste, KIHU, Jyväskylä, 2003.

[8] Swan, Tom, ”Delphi 4”, Teknolit Oy, Porvoo, 1999.

[9] Taylor John, “Introduction to Error Analysis”, Second Edition, University Science Books, 1997.