An Integrated Tool Suite for Simulation and Programming of Palletizing Units

Mauro Argenti, Davide Buratti
OCME S.r.l., Parma, Italy

Dario Lodi Rizzini, Stefano Caselli
University of Parma, Italy

Abstract

In this paper, we describe an integrated software tool suite for simulation, programming and monitoring of palletizing machines. All the tools belonging to the suite share a common machine model in order to allow software reuse according to sound software design principles. The simulator emulates the machine behavior and, while abstracting unnecessary physical details, can be used to accurately assess machine throughput and correctness of control parameters. The programming tool assists non-expert human operators in visual programming of the palletizing task, including layers disposition, number of layers, and fractional pallets. Based on the user's high-level specification of the pallet format, the system generates optimal pallet layers together with the machine control parameters that meet the given specification. Finally, the monitoring tool supports diagnostic activities and allows efficient recovery when failure occurs.

1 Introduction

Even though palletizing units have represented an important part of manufacturing systems for several decades, the design of modern palletizing systems is facing new challenges dictated by market demands. These challenges include the need to cope with increasing production rates, the larger variability in product packaging and formats, as well as the thinner and more fragile primary containers and products to be palletized. In order to comply with these demands, palletizing units must ensure careful, faster and smarter product manipulation. They must also support immediate reconfiguration to adapt to the specific type of product and pallet format. In essence, in modern palletizing systems complexity has extended from the mechanical design to overall machine operation and control. This trend also challenges human operators, whose skills often do not match the new generation of machines.

Simulation and visual programming software tools can help to meet the requirements for modern palletizing units and assist both manufacturers and end users. Several software packages have been proposed for simulation of a machine component [1], for integrated simulation and parameters optimization of a manipulator [2], and for high-level simulation of manufacturing systems [3]. Few tools, however, integrate simulation, programming and monitoring of a complete palletizing line.

In this paper, we describe an integrated software tool suite for simulation, programming and monitoring of palletizing machines jointly developed by OCME S.r.l. and the University of Parma. The tool suite has been designed for a line of robot-based palletizer machines manufactured by OCME S.r.l., but the underlying concepts are suitable for a wide range of end-of-line machines.

The software tool suite allows simulation, programming and monitoring of the palletizing machines. Individual applications for simulation, programming and monitoring have been designed starting from a common machine model that describes the structure of the machine. The model handles all different machine configurations and its flexibility has been demonstrated by new configurations added after model design. Each tool controls and manipulates the common data structure to perform its specific task. Thus, the design of the tool suite is inspired by the *Model-View-Control* software design paradigm, which allows code reuse and ensures consistency among the behaviors achieved by the applications.

Simulation is used to verify the correctness of machine control parameters and to estimate its throughput together with the time required to complete specific production steps. Simulation is carried out at an adequate level of detail to support negotiation of technical specifications and cost-performance tradeoffs between customers and designers before a machine is actually built. The simulator is decomposed according to the activities performed by the components of the palletising machine on the product units. Since the product units are processed sequentially following the input order, the activities of the machines are modelled by queues. These queues are then assembled together to obtain the complete simulated system.

The integrated software environment provides tools for visual programming of the palletizing task, including layers

disposition, number of layers, fractional pallets, and more. Based on the user's high-level specification of the pallet format, the system generates optimal pallet layers together with the control parameters for the machine that meet the given specification. The programming tool can automatically generate the required parameters or graphically assist the editing of user-proposed solutions.

The tool periodically monitors the state of the palletizer machine by receiving the data from the control unit in order to support diagnostic activities. Indeed, when unstable products must be packaged at high speed (e.g., very thin bottles), or conveyors are not properly fed with products, failures are possible. When a failure occurs, the monitoring tool graphically displays the parts of the machine involved and the correct location for the product units. In this way, the human operator is assisted in restoring the state of the machine, yielding reduced down and recovery times. The real-time monitoring tool is complementary to the simulator, as the latter performs an offline verification of the program.

It should be mentioned that the software tool suite has been conceived to run on desktop engineering workstations for machine design as well as on the same industrial PC in charge of controlling the palletizing machine. Hence, it is characterized by rather limited computational and graphical requirements.

The paper is organized as follows. Section 2 illustrates the common model of the palletizing machine used by all tools. Section 3 describes the simulator. Section 4 presents the visual programming tool for the generation of control parameters. Section 5 illustrates the monitoring tool for diagnostic purposes. Finally, section 6 gives conclusion remarks.

2 Model of the Palletizing Machine

The palletizing machine model is the core of the whole tool suite. Indeed, the tools for simulation, monitoring and programming require a common software representation of the physical machine. In particular, a model should describe the elementary components, the dimensional parameters, and the state of the machine. Since the aim of the tool suite is to provide an user friendly interface for palletizer management, the graphical representation of each component is as important as the list of the attributes. Each tool uses the machine model according to its own purposes. The simulator modifies the state of the machine to emulate its behavior through state transitions. The monitoring tool represents the current state of the palletizer according to the messages received from the machine controller. The programmer reads and writes the parameters that allow formation of pallet layers. Thus, the model, the graphical representation and the behavior of the palletizer should be independent. The software design of the tool suite has adopted the *Model-View-Control* (MVC) paradigm that proposes such separation between the three aspects as a general design criterion. This section presents an overview of the machine model that corresponds to the part common to all the applications.

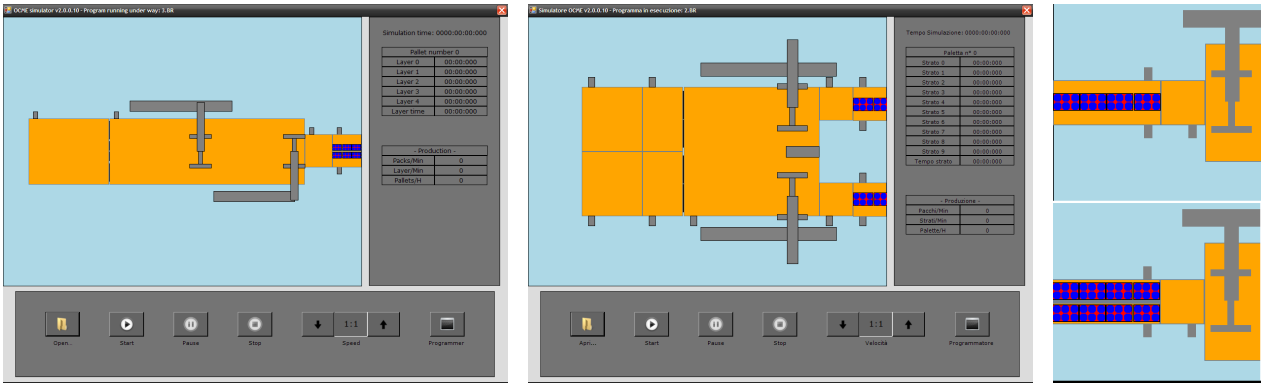
The model of the palletizing machine has been decomposed into basic components each corresponding to an abstract operation performed on the product units. Software modularity is motivated by the strong customization capability often required to palletizing machines. For example, palletizers may differ in the number of *channels*, of *input rows* per channel and of *manipulators*. The software model has been designed to support all the required variability. A channel corresponds to an elementary product path that is able to perform all the required operations on the units and that contributes to a section of each pallet layer. Typical configurations are single and double channel, but the model supports an open number of channels. Channels can handle single or double input rows. Double rows have been introduced to handle efficiently smaller product formats. The number and position of manipulators that operate on each channel is another highly configurable feature. Figure 1 summarizes the possible configurations.

The machine component set includes conveyor belts, moveable barriers, manipulator joints and end-effectors. Although product units are not strictly part of the machine, the concept of component has been conveniently extended also to units and group of units. A component is defined as the basic part of the system that contributes to the description of the system state. Components also have a graphical representation and their visualization is handled by proper routines. According to this definition, product units are proper components and share the same interface with machine parts, as shown in Figure 2. A machine configuration is defined through a configuration file in text format. During initialization, this file is parsed and used to create the palletizer model with the aggregation of the specified components. The design of the machine model has proven flexible and robust to the new features introduced during and after the development.

3 Simulator

Given a specific machine configuration and the programming parameters, the tool suite performs an accurate simulation of the machine and estimates its throughput and the time required to complete specific production steps. Simulation is essentially defined by the system state variables and the transition function that modifies the value of state. Furthermore, the updated state of the machine must be displayed periodically for the human operator. The user should be able to select a simulation time that is slower, equal, or faster than the real time in order to check the correctness of the simulation.

System state is defined by the values of component state variables and by the production step of each product unit. The state of the simulation is described both by continuous variables, e.g. the current position of the machine compo-



(a) (b) (c)

Figure 1: Configuration features of the palletizers: single (a) or double (b) channel with arbitrary number and position of manipulators, and single or double input row (c).

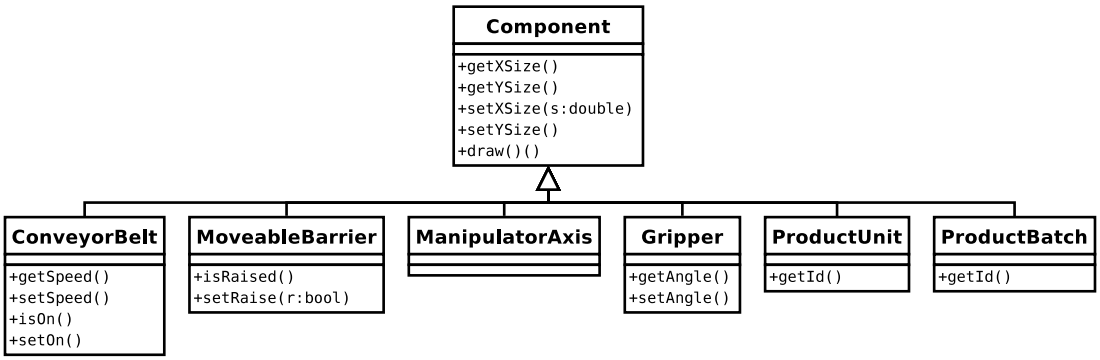


Figure 2: Simplified class diagram of classes inheriting `Component` interface.

nents and of product units, and by logic variables usually representing the state of a machine component or the current activity performed on a given product unit. Incoming product units are processed sequentially following the input order and the activities of the machines are modelled by *queues* or *FIFOs*. FIFOs contain *batches* of product units instead of the basic items. A batch corresponds to a group of units that are processed and manipulated together.

A FIFO corresponds to a specific operation or group of operations performed on the product units that are stored in the FIFO. Decomposition of palletizer activities into queues is consistent with the control architecture of the real machine in order to achieve an accurate simulation. In particular, the simulator consists of the following queues: the input conveyor belt queue, the manipulation queue, the barrier queue and the pallet merging queue.

Update of the system state is different in the case of continuous and logic variables. Continuous variables are periodically updated at given time steps. Such variables include product unit coordinates and manipulator joint values and are updated according to kinematic equations that provide a sufficiently accurate description of system evolution. On the other hand, transition from one logic state to another may occur at unpredictable time instants. Such transition events are handled at the beginning of a simulation time interval. A modification of logic state may affect the value of continuous variables since after a logic transition the kinematic laws may change, e.g. a product unit is dragged by a conveyor belt to another belt with a different rolling speed. Thus, the default simulation time step is adapted [4]. Algorithm 1 illustrates how simulation update iterations are executed.

Algorithm 1: Basic simulation step

Require: t_{sim}: maximum simulation time period

$t = 0.0$
while $t < t_{sim}$ **do**
  $\Delta t = t - t_{sim}$
  for each FIFO f **do**
    $\Delta t_{trans} = f$.computeMinTransTime()
    if $\Delta t < \Delta t_{trans}$ **then**
      $\Delta t = \Delta t_{trans}$
    end if
  end for
  for each FIFO f **do**
    component state update on f with time step Δt;
  end for
  $t = t + \Delta t$
end while

Simulation must be accurate enough to estimate the performance of the machine, but should avoid unnecessarily detailed physical description of system dynamics. In particular, simulation does not handle computationally expensive operations like 3D geometric body collisions or dynamics. When the machine is correctly programmed, logic and geometric properties hold like the absence of body compenetration. Simplified kinematic control and collision detection are adopted when necessary, e.g. for the accumulation of product units on moveable barriers. In the following, the activities performed by each queue are described.

3.1 Input Conveyor Belt Queue

The *input conveyor belt queue* controls the state of the first group of conveyor belts in each channel. This initial block handles the arrival of product items into the palletizer and their separation in batches. Product units are carried into the machine by a conveyor belt, which can be switched on or off, until they reach a second conveyor belt with higher rolling speed. The simulated controller splits the input row of items into batches by switching off the first belt for a proper time. In case of double row configuration (Figure 1-(c)), separation of batches requires a more complex procedure since there are two conveyor belts, one for each row. The input conveyor belt queue handles all the transitions between consecutive belts.

3.2 Manipulation Queue

The *manipulation queue* controls the manipulators that move product batches in the required position. Manipulator motion is difficult to predict due to the complexity of the finite state machine describing the manipulation (Figure 3).

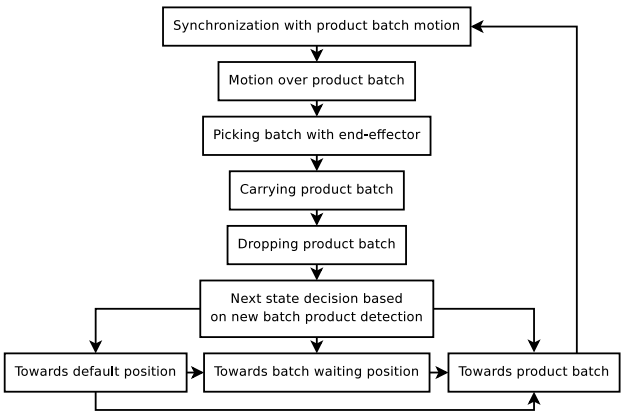


Figure 3: Finite state machine representing the manipulator behavior.

A product batch formed in the previous queue is carried on the manipulation conveyor belt and cannot be manipulated until it reaches a given position. Then, the product batch is picked by the end-effector of the robot arm in a position that depends on the current position of both the batch and the arm. When the item is placed in the target position, the manipulator moves towards the next batch or towards the waiting position, if none is available. The transitions handled by the manipulation queue are mainly related to the finite state machine of the manipulator. Other events include the transitions to the moveable barrier queue. Although the sequence of operations performed by manipulation queue is deterministic, evaluating machine throughput would be difficult without a simulator.

3.3 Moveable Barriers Queue

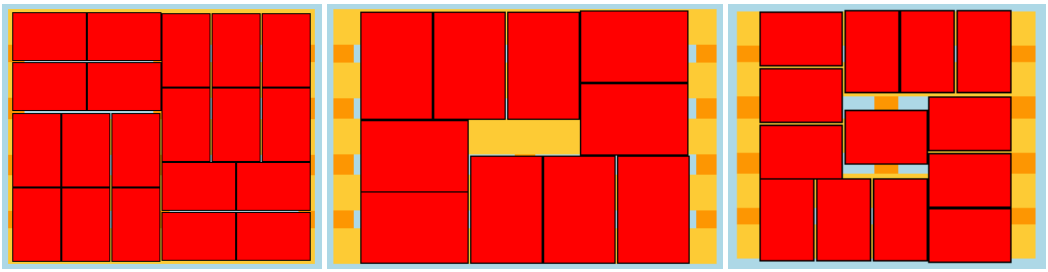
The *moveable barriers queue* controls the product batches piling on the moveable barriers located at the end of the manipulation conveyor belt. Barriers are raised in order to form a section of the pallet and are then lowered when the cumulated items are transferred to the merging conveyor belt. The raise and release of a moveable barrier are triggered by the position offset of specific product batches. The *moveable barriers queue* is the only part of the simulator that requires the management of collisions among product units. Management can be simplified by assuming that all the incoming product units are orthogonally oriented with respect to the barriers and that the torques generated by contacts are negligible independently from the contact surface. When the palletizing line is correctly programmed, such hypotheses hold. As a matter of fact, all the violations can be detected and used to check the correctness of the palletizer program. The accumulation has been handled using a *segment tree* [5]. This data structure can efficiently detect the overlapping of product units during simulation and prevent comprenetration. The transitions handled by moveable barriers queue are related to the state of moveable barrier and to the contacts between product units.

3.4 Final Conveyor Belt Queue

The *final conveyor belt queue* controls the last conveyor belt that transfers the formed pallet. Depending on the palletizer configuration, there may be a conveyor belt for each channel or a common one for all the channels. Pallet transfer depends on the configuration too and can be performed by a manipulator or by a mechanical pusher. The motion of the final conveyor belt is synchronized with the manipulator conveyor belt in the previous queue. The final conveyor belt queue is also responsible for the removal of the formed pallets.

4 Programmer

Machine programming for generation of each layer of a pallet consists of three steps: planning the layer layout, subdividing product units into batches suitable for common manipulation by the machine, and generating machine parameters. A correct layer layout is such that all



(a) (b) (c)

Figure 4: Three examples of layout patterns: *binary cut* (a), *single spiral* (b), and *double spiral* (c).

product units are fully contained within the assigned layer area. The developed systems encodes several assumptions typically made in industrial contexts, such as a purely 2D problem and orthogonal strategies for placing parts aligned with pallet sides.

4.1 Layout Generation

The computational counterpart of the physical layer planning problem is the well-known bin packing problem [6], which is usually transformed into an unidimensional knapsack problem. Unfortunately, bin packing algorithms assist in maximizing the number of product units per layer or pallet but do not consider other practical requirements such as regularity in products arrangement, layer stability, and presence of symmetries. Alternative computational approaches such as rectangular tessellation and rectangle tiling suffer from similar drawbacks and do not exploit special features of the problem such as the uniform product size.

For these reasons, the proposed layout generation system is based on a set of predefined patterns incorporating user expectations as well as typical industrial requirements. The input data for layout computation consist of product size, allowed product overflow or required margin from pallet borders, and pallet style (plain or hierarchical). Available patterns explored by the system include:

- *Binary Cut.* This pattern operates an axis-aligned subdivision of the layer in quarters labeled BL (Bottom Left), BR (Bottom Right), Top Left (TL), Top Right (TR). In BR and TL quarters items are arranged horizontally, whereas in BL and TR quarters items are arranged vertically.
- *Single Spiral.* Similar to Binary Cut but leaving empty space, if any, at the center of the layer.
- *Column.* All items are arranged along the same direction.
- *Double Spiral.* Items are first arranged along a same direction in a diagonal area of the layer. Additional items are then placed in the empty areas, arranged along the orthogonal direction.

Exploring these patterns (Figure 4), the layout generation algorithm computes a number of solutions, which are then proposed to the user ranked according to the number of products per layer. The user can also require a hierarchical pallet organization, where the pallet is composed by 2 or 4 subpallets. Layouts are then computed on a subpallet and replicated on the remaining ones. The user can graphically edit the found solutions and select the layouts to be used in each layer.

4.2 Batch Decomposition and Parameters Computation

The second step of palletizer programming is the computation of control parameters required to generate the given layout. The layout generated in the previous step consists of product units arranged inside the planar shape of the pallet. Since the manipulation of product units is oriented to batches, the decomposition of the layouts in batches is crucial for parameters computation. Batch formation must follow some rules about maximum number and proximity of product items. A batch corresponds to a group of units that are manipulated together. Thus, its weight and size are limited by the payload and the end-effector size of the manipulators. Furthermore, the batch should have single or double row rectangular shape depending on the machine configuration, since it is formed through commands to the input conveyor belts. Batch selection follows different rules. The sequence number of each batch must be consistent with the order of accumulation.

Batch feasibility is checked using a graph-like data structure that represents proximity relationships between units. The selection of batch items can be performed manually, semi-automatically or completely automatically. In manual mode, the user graphically selects the items and correctness is checked only after assignment. In semi-automatic mode, the user selects only among the allowed product units and suggestions are provided when requested. In automatic mode, the decomposition of pallet layer into batches is generated without user involvement. The first item of each batch is chosen from the non-occluded and unselected product units. The other items of the batch are added after an exhaustive depth-first explo-

ration of feasible batch formats built around the first item. Given the batch decomposition of the pallet layer, machine parameters are completely determined. The parameters related to a batch include the number of product items, the number of rows, the spacing from the previous batch, the offset positions for manipulation picking and release, the rotation flag, and the offset positions for barrier raise and release. Barrier offsets depend on the relative displacement of batches in the pallet layer. Thus, the programmer divides the batches in groups according to their alignment to a common front line. Each group corresponds to product batches simultaneously accumulated on the same barrier.

5 Monitor

A final tool available in the tool suite is conceived for real-time monitoring of palletizing machines during actual production. The tool periodically monitors the state of the palletizer by receiving data from the control unit in order to support diagnostic activities. When unstable products must be packaged at high speed (e.g., very thin bottles), or conveyors are not properly fed with products, failures are possible. When a failure occurs, the monitoring tool graphically displays the parts of the machine involved and the correct location for the product units. In this way, the human operator is assisted in restoring the state of the machine, yielding reduced down and recovery times. The real-time monitoring tool is complementary to the simulator, as the latter performs an offline verification of the program.

Figure 5: A screenshot of the monitor.

6 Conclusion

In this paper, we have described an integrated software tool suite for simulation, programming and monitoring of palletizing machines. The tool suite is designed by separating the management of data, graphical interface and logic control according to the Model-View-Control design paradigm. Hence, all the tools are based on a common model representing the structure of the machine, but each tool operates differently on the machine data. The simulation tool accurately reproduces the behavior of the palletizing machine avoiding unnecessary modelling of physical interaction and dynamics of bodies. The simulator is decomposed according to the activities performed by the components of the palletising machine on the product units and each activity is modelled as a queue. The programming tool assists non-expert human operators in visual programming of the palletizing task, and supports a variety of configuration options automatically planning feasible layers. Based on the user's high-level specification of the pallet format, the system generates optimal pallet layers together with the control parameters for the machine that meet the given specification. The monitoring tool supports diagnostic activities and allows efficient recovery when failure occurs. The tool suite has a low computational footprint and can be installed on the industrical PC controlling a palletizing machine. Moreover, it has been already successfully used by several operators for simulation, programming and monitoring of palletizing machines in working plants.

Acknowledgment

The contribution of the University of Parma in this research has been supported also by laboratory AER-TECH of Regione Emilia-Romagna, Italy.

References

[1] Dong, W.; Palmquist, F.; Lidholm, S.: *A Simple and Effective Emulation Tool Interface Development for Tricept Application*, in Proc. 33rd Int. Symp. on Robotics, Oct. 7-11, 2002

[2] Kazi, A.; Merck, G.; Otter, M.; Fan, H.: *Design optimization of industrial robots using the Modelica multi-physics modelling language*, in Proc. 33rd Int. Symp. on Robotics, Oct. 7-11, 2002.

[3] Inukai, T.; Hibino, H.; Fukuda, Y.: *Simulation Environment Synchronizing Real Equipment for Manufacturing Cell*, J. Advanced Mechanical Design, Systems, Manufacturing, pp. 238-249, Vol. 1, No. 2, 2007.

[4] Zeigler, B. P.; Praehofer, H.; Kim, T. G.: *Theory of Modeling and Simulation*, 2nd Edition, Academic Press, 2000.

[5] De Berg, M.; Cheong, O.; van Kreveld, M.; Overmars, M.: *Theory of Modeling and Simulation*, 3th Edition, Springer-Verlag, 2008.

[6] Lodi, A.; Martello, S.; Vigo, D.: *Recent advances on two-dimensional bin packing problems*, Discrete Appl. Math., Volume 123, p. 379-396, 2002.